Red Hat OpenShift Container Storage 4.3

Managing OpenShift Container Storage

Instructions for cluster and storage administrators

Last Updated: 2020-12-08

Red Hat OpenShift Container Storage 4.3 Managing OpenShift Container Storage

Instructions for cluster and storage administrators

Abstract

This document covers instructions for managing an OpenShift Container Storage cluster.

Table of Contents

CHAPTER 1. OVERVIEW 4

CHAPTER 2. CONFIGURE STORAGE FOR OPENSHIFT CONTAINER PLATFORM SERVICES 5
2.1. CONFIGURING IMAGE REGISTRY TO USE OPENSHIFT CONTAINER STORAGE 5
2.2. CONFIGURING MONITORING TO USE OPENSHIFT CONTAINER STORAGE 7
2.3. CLUSTER LOGGING FOR OPENSHIFT CONTAINER STORAGE 9
2.3.1. Configuring persistent storage 10
2.3.2. Configuring cluster logging to use OpenShift Container Storage 11

CHAPTER 3. BACKING OPENSHIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSHIFT CONTAINER STORAGE 14

CHAPTER 4. SCALING STORAGE NODES 16
4.1. REQUIREMENTS FOR SCALING STORAGE NODES 16
4.1.1. Supported Deployments for Red Hat OpenShift Container Storage 16
4.2. SCALING UP STORAGE CAPACITY 16
4.2.1. Scaling up storage by adding capacity to your OpenShift Container Storage nodes on AWS or VMware infrastructure 17
4.2.2. Scaling up storage by adding capacity to your OpenShift Container Storage nodes using local storage devices 19
4.3. SCALING OUT STORAGE CAPACITY 22
4.3.1. Adding a node 22
4.3.1.1. Adding a node on an AWS installer-provisioned infrastructure 23
4.3.1.2. Adding a node on an AWS or a VMware user-provisioned infrastructure 23
4.3.1.3. Adding a node using a local storage device 24
4.3.2. Verifying the addition of a new node 26
4.3.3. Scaling up storage capacity 26

CHAPTER 5. MANAGING PERSISTENT VOLUME CLAIMS 27
5.1. CONFIGURING APPLICATION PODS TO USE OPENSHIFT CONTAINER STORAGE 27
5.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS 28
5.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS 29
5.4. DYNAMIC PROVISIONING 29
5.4.1. About dynamic provisioning 30
5.4.2. Dynamic provisioning in OpenShift Container Storage 30
5.4.3. Available dynamic provisioning plug-ins 30

CHAPTER 6. MANAGING CONTAINER STORAGE INTERFACE (CSI) COMPONENT PLACEMENTS 33

CHAPTER 7. MULTICLOUD OBJECT GATEWAY 34
7.1. ABOUT THE MULTICLOUD OBJECT GATEWAY 34
7.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS 34
7.2.1. Accessing the Multicloud Object Gateway from the terminal 34
7.2.2. Accessing the Multicloud Object Gateway from the MCG command-line interface 36
7.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD 39
7.3.1. Creating a new backing store 39
7.3.2. Adding storage resources for hybrid or Multicloud using the MCG command line interface 40
7.3.2.1. Creating an AWS-backed backingstore 41
7.3.2.2. Creating an IBM COS-backed backingstore 42
7.3.3. Creating an S3 compatible Multicloud Object Gateway backingstore 44
7.3.4. Adding storage resources for hybrid and Multicloud using the user interface 45
7.3.5. Creating a new bucket class 47
7.4. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS 50

7.4.1. Creating bucket classes to mirror data using the MCG command-line-interface 50
7.4.2. Creating bucket classes to mirror data using a YAML 50
7.4.3. Configuring buckets to mirror data using the user interface 51
7.5. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY 53
7.5.1. About bucket policies 53
7.5.2. Using bucket policies 53
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway 54
7.6. OBJECT BUCKET CLAIM 57
7.6.1. Dynamic Object Bucket Claim 57
7.6.2. Creating an Object Bucket Claim using the command line interface 59
7.6.3. Creating an Object Bucket Claim using the OpenShift Web Console 62
7.6.3.1. Delete an Object Bucket Claim 64
7.6.3.2. Viewing object buckets using the Multicloud Object Gateway user interface 65
7.7. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS [TECHNOLOGY PREVIEW] 66
7.7.1. S3 endpoints in the Multicloud Object Gateway [Technology Preview] 66
7.7.2. Scaling with storage nodes [Technology Preview] 66
7.7.3. Stretching buckets across two different clusters [Technology Preview] 69
7.7.3.1. Create an external NooBaa Daemon pool on Cluster B 69
7.7.3.2. Adding Cluster B as a backing store to Cluster A 72
7.7.3.3. Create a federated bucket on Cluster A using Cluster B’s NooBaa system 73
7.7.3.4. Assign regions for the locality optimization 75
7.7.3.5. Managing the clusters 76

CHAPTER 8. REPLACING STORAGE NODES FOR OPENSHIFT CONTAINER STORAGE 77
8.1. OPENSHIFT CONTAINER STORAGE DEPLOYED ON AWS 77
8.1.1. Replacing an operational AWS node on user-provisioned infrastructure 77
8.1.2. Replacing an operational AWS node on installer-provisioned infrastructure 78
8.1.3. Replacing a failed AWS node on user-provisioned infrastructure 80
8.1.4. Replacing a failed AWS node on installer-provisioned infrastructure 81
8.2. OPENSHIFT CONTAINER STORAGE DEPLOYED ON VMWARE 82
8.2.1. Replacing an operational VMware node on user-provisioned infrastructure 82
8.2.2. Replacing a failed VMware node on user-provisioned infrastructure 84

CHAPTER 9. UPDATING OPENSHIFT CONTAINER STORAGE 86
9.1. ENABLING AUTOMATIC UPDATES FOR OPENSHIFT CONTAINER STORAGE OPERATOR 86
9.2. MANUALLY UPDATING OPENSHIFT CONTAINER STORAGE OPERATOR 87

CHAPTER 1. OVERVIEW

Managing OpenShift Container Storage is written to help administrators understand how to manage and administer their Red Hat OpenShift Container Storage cluster.

Most management tasks focus on a single resource. This document is divided into chapters based on the resource that an administrator is trying to modify:

- Chapter 2, *Configure storage for OpenShift Container Platform services* shows you how to use OpenShift Container Storage for core OpenShift Container Platform services.
- Chapter 3, *Backing OpenShift Container Platform applications with OpenShift Container Storage* provides information about how to configure OpenShift Container Platform applications to use OpenShift Container Storage.
- Chapter 4, *Scaling storage nodes* provides information about scaling storage capacity of OpenShift Container Storage nodes.
- Chapter 5, *Managing persistent volume claims* provides information about managing Persistent Volume Claim requests, and automating the fulfillment of those requests.
- Chapter 6, *Managing container storage interface (CSI) component placements* provides information about setting tolerations to bring up container storage interface component on the nodes.
- Chapter 7, *Multicloud Object Gateway* provides information about the Multicloud Object Gateway.
- Chapter 8, *Replacing storage nodes for OpenShift Container Storage* shows you how to replace an operational or failed node on AWS UPI, AWS IPI, and VMware UPI for OpenShift Container Storage.
- Chapter 9, *Updating OpenShift Container Storage* provides instructions for upgrading your OpenShift Container Storage cluster.

CHAPTER 2. CONFIGURE STORAGE FOR OPENSHIFT CONTAINER PLATFORM SERVICES

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

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

WARNING

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

Red Hat recommends configuring shorter curation and retention intervals for these services. See Configuring Curator and Modifying retention time for Prometheus metrics data in the OpenShift Container Platform documentation for details.

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

2.1. CONFIGURING IMAGE REGISTRY TO USE OPENSHIFT CONTAINER STORAGE

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

Follow the instructions in this section to configure OpenShift Container Storage as storage for the Container Image Registry. On AWS, it is not required to change the storage for the registry. However, it is recommended to change the storage to OpenShift Container Storage persistent volume for vSphere platform.

WARNING

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

Prerequisites

- You have administrative access to OpenShift Web Console.

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

- Image Registry Operator is installed and running in the **openshift-image-registry** namespace. In OpenShift Web Console, click **Administration → Cluster Settings → Cluster Operators** to view cluster operators.

- The **ocs-storagecluster-cephfs** storage class is available. In OpenShift Web Console, click **Storage → Storage Classes** to view available storage classes.

Procedure

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

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

For example:

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

    ii. Click **Save**.

3. **Verify that the new configuration is being used.**
    a. Click **Workloads** → **Pods**.
    b. Set the **Project** to **openshift-image-registry**.
    c. Verify that the new **image-registry-*** pod appears with a status of **Running**, and that the previous **image-registry-*** pod terminates.
    d. Click the new **image-registry-*** pod to view pod details.
    e. Scroll down to **Volumes** and verify that the **registry-storage** volume has a **Type** that matches your new Persistent Volume Claim, for example, **ocs4registry**.

2.2. CONFIGURING MONITORING TO USE OPENSHIFT CONTAINER STORAGE

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

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

IMPORTANT

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

Red Hat recommends configuring a short retention intervals for this service. See the *Modifying retention time for Prometheus metrics data* sub section of Configuring persistent storage in the OpenShift Container Platform documentation for details.

Prerequisites

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

Procedure

1. In OpenShift Web Console, go to **Workloads** → **Config Maps**.
2. Set the **Project** dropdown to **openshift-monitoring**.

3. Click **Create Config Map**.

4. Define a new **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**.

   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

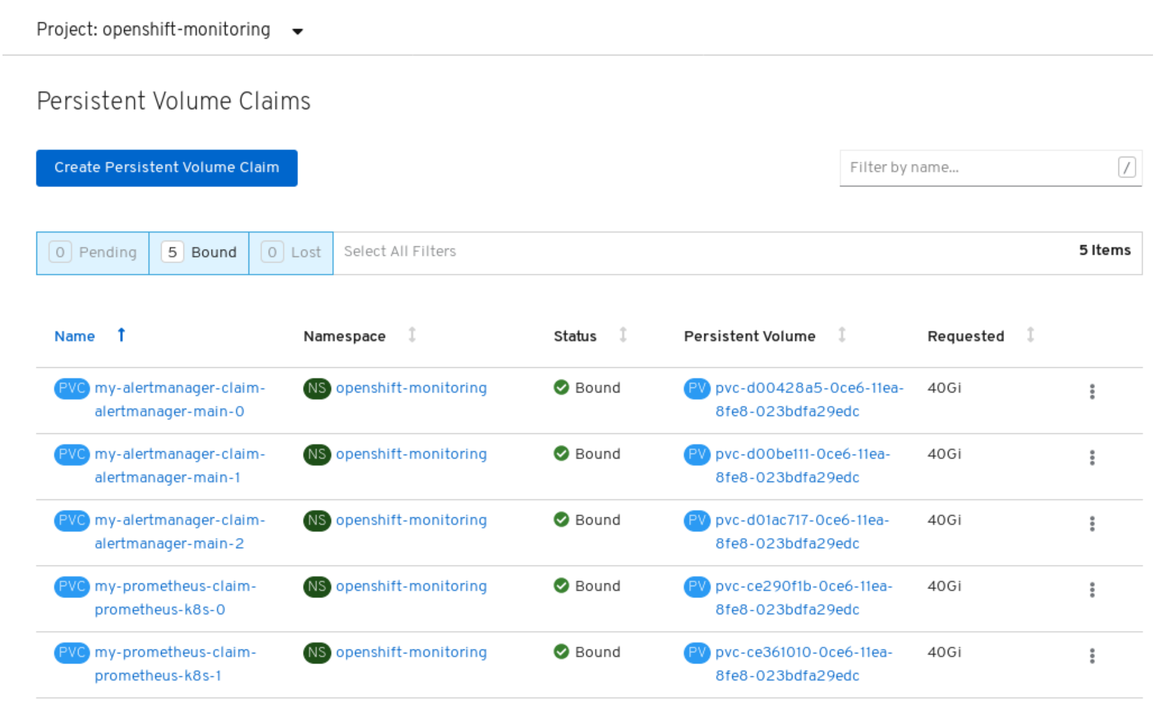
1. Verify that the Persistent Volume claims are bound to the pods.

   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

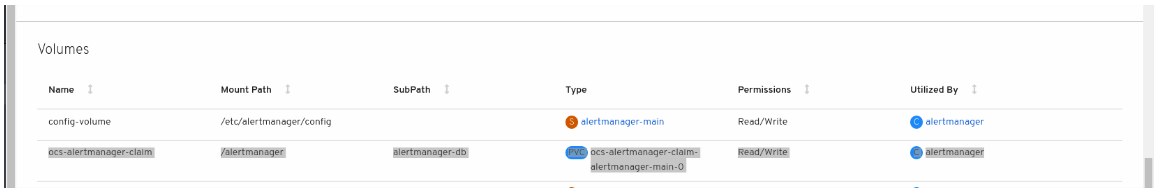
Name	Namespace	Status	Persistent Volume	Requested
PVC my-alertmanager-claim-alertmanager-main-0	NS openshift-monitoring	Bound	PV pvc-d00428a5-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC my-alertmanager-claim-alertmanager-main-1	NS openshift-monitoring	Bound	PV pvc-d00be111-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC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

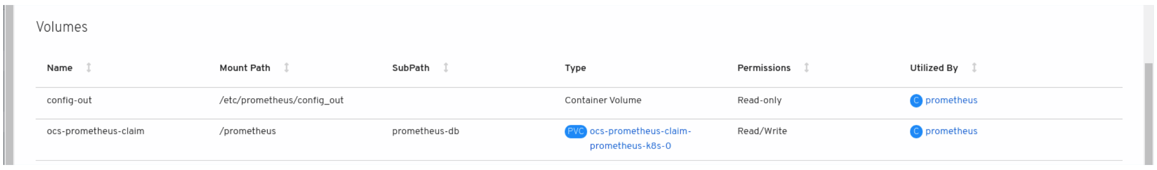
Name	Mount Path	SubPath	Type	Permissions	Utilized By
config-volume	/etc/alertmanager/config		S alertmanager-main	Read/Write	C alertmanager
ocs-alertmanager-claim	/alertmanager	alertmanager-db	PVC ocs-alertmanager-claim-alertmanager-main-0	Read/Write	C alertmanager

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

2.3. CLUSTER LOGGING FOR OPENSHIFT CONTAINER STORAGE

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

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

IMPORTANT

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

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

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

2.3.1. 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.

2.3.2. Configuring cluster logging to use OpenShift Container Storage

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

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

Prerequisites

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

Procedure

1. Click **Administration → Custom Resource Definitions**from the left pane of the OpenShift Web Console.
2. On the Custom Resource Definitions page, click **ClusterLogging**.
3. On the Custom Resource Definition Overview page, select **View Instances** from the Actions menu or click the **Instances** Tab.
4. On the Cluster Logging page, click **Create Cluster Logging**.
   You might have to refresh the page to load the data.
5. In the YAML, replace the code with the following:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: ocs-storagecluster-ceph-rbd
        size: 200G
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

6. Click **Save**.

Verification steps

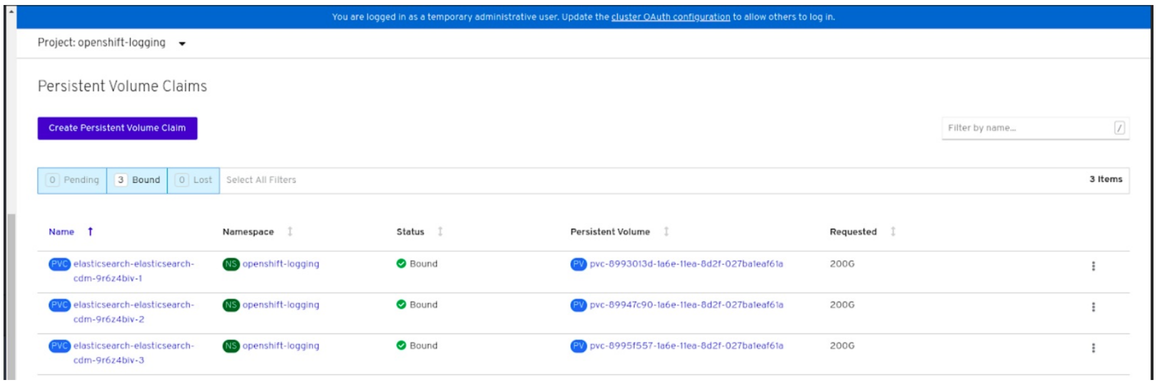
1. Verify that the Persistent Volume Claims are bound to the **elasticsearch** pods.

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

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

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

   Figure 2.1. 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

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

CHAPTER 3. BACKING OPENSHIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSHIFT CONTAINER STORAGE

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

Prerequisites

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

Procedure

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

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

    > **NOTE**
    >
    > ReadOnlyMany (ROX) is deactivated as it is not supported.

    d. Select the size of the desired storage capacity.

    > **NOTE**
    >
    > You cannot resize the storage capacity after the creation of Persistent Volume Claim.

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

Verification steps

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

CHAPTER 4. 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 Red Hat OpenShift Container Storage worker nodes
- **Scale out storage nodes** - Add new worker nodes containing storage capacity

4.1. 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:

- Supported Infrastructure and Platforms
- Supported configurations
  - For dynamically created storage
  - For local storage devices

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.

4.1.1. Supported Deployments for Red Hat OpenShift Container Storage

- User-provisioned infrastructure:
  - Amazon Web Services (AWS)
  - VMware
  - Bare metal (Technology Preview)
- Installer-provisioned infrastructure:
  - Amazon Web Services (AWS)

4.2. SCALING UP STORAGE CAPACITY

Depending on the type of your deployment, you can choose one of the following procedures to scale up storage capacity.

- For AWS or VMware infrastructures using dynamic or automated provisioning of storage devices, see Section 4.2.1, “Scaling up storage by adding capacity to your OpenShift Container Storage nodes on AWS or VMware infrastructure”
- For bare metal, Amazon EC2 (storage optimized - i3en.2xlarge), or VMware infrastructures using local sotrage devices, see Section 4.2.2, “Scaling up storage by adding capacity to your OpenShift Container Storage nodes using local storage devices”

4.2.1. Scaling up storage by adding capacity to your OpenShift Container Storage nodes on AWS or VMware infrastructure

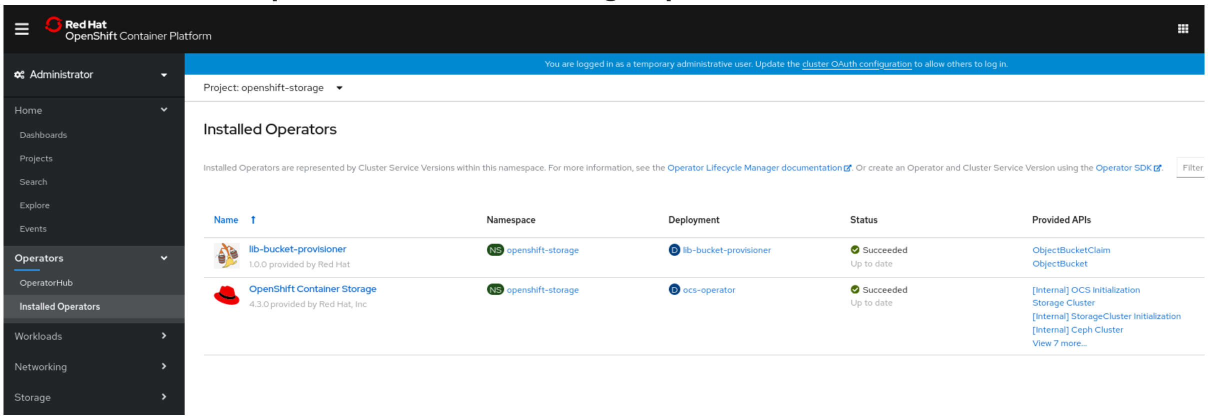
Use this procedure to add storage capacity and performance to your configured Red Hat OpenShift Container Storage worker nodes.

Prerequisites

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

Procedure

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



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

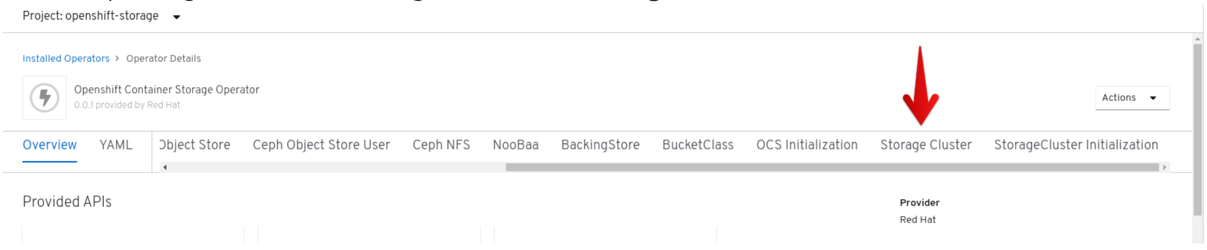
Project: openshift-storage

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. Filter

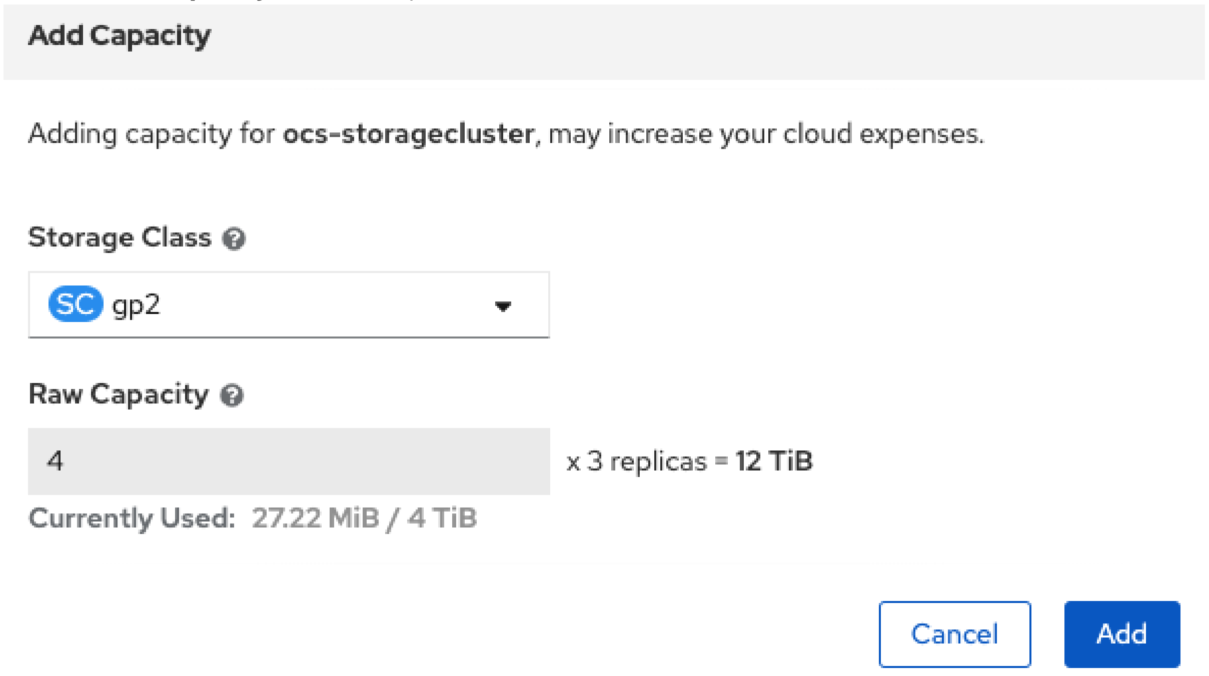
Name	Namespace	Deployment	Status	Provided APIs
lib-bucket-provisioner 1.0.0 provided by Red Hat	NS openshift-storage	D lib-bucket-provisioner	Succeeded Up to date	ObjectBucketClaim ObjectBucket
OpenShift Container Storage 4.3.0 provided by Red Hat, Inc	NS openshift-storage	D ocs-operator	Succeeded Up to date	[Internal] OCS Initialization Storage Cluster [Internal] StorageCluster Initialization [Internal] Ceph Cluster View 7 more...

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


Project: openshift-storage
Installed Operators > Operator Details
Openshift Container Storage Operator
0.0.1 provided by Red Hat
Actions
Overview
YAML
Object Store
Ceph Object Store User
Ceph NFS
NooBaa
BackingStore
BucketClass
OCS Initialization
Storage Cluster
StorageCluster Initialization
Provided APIs
Provider
Red Hat

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

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



Add Capacity

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

Storage Class: gp2

Raw Capacity: 4 x 3 replicas = 12 TiB

Currently Used: 27.22 MiB / 4 TiB

From this dialog box, you can set the requested additional capacity and the storage class. **Add capacity** will show the capacity selected at the time of installation and will allow to add the capacity only in this increment. On AWS, the storage class should be set to **gp2**. On VMWare, the storage class should be set to **thin**.

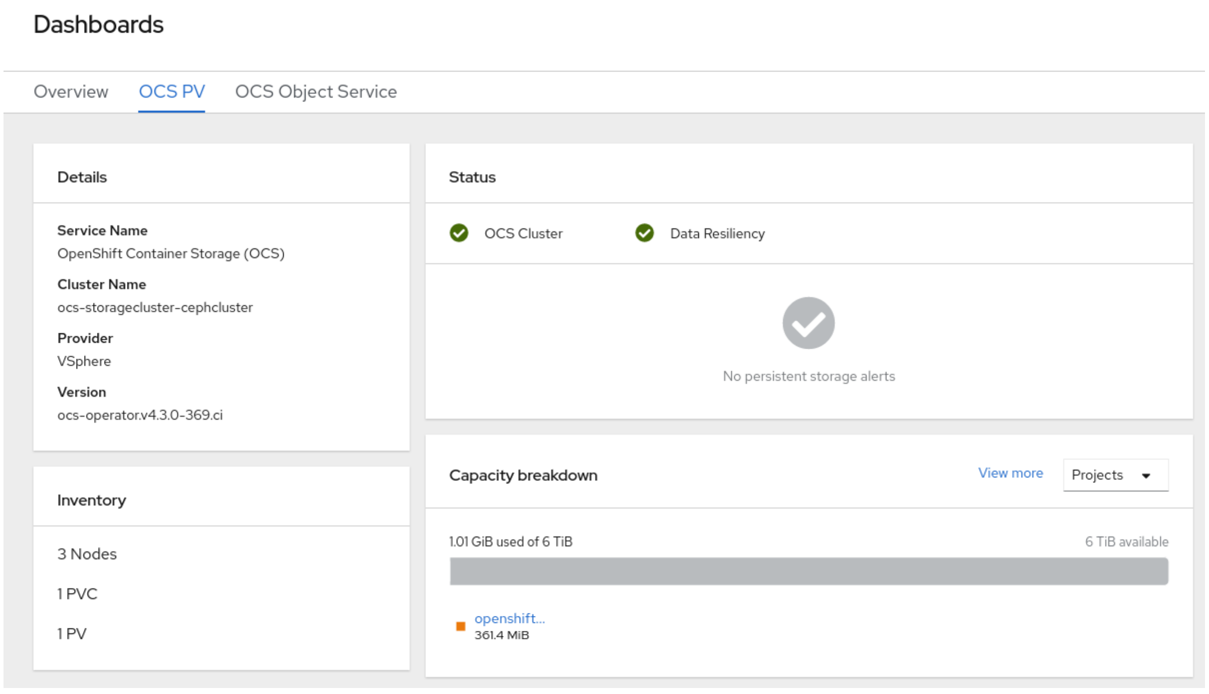
NOTE

The effectively provisioned capacity will be three times as much as you put into the **Requested Capacity** field because OpenShift Container Storage uses a replica count of 3.

8. Once you are done with your setting, click **Add**. You will not see the status of the storage cluster until it reaches **Ready**. You might need to wait a couple of minutes after you see the **Ready** state.

Verification steps

1. Navigate to **Dashboards → OCS PV** tab, then check the **Capacity breakdown** card.


Dashboards

Overview | OCS PV | OCS Object Service

Details

Service Name
OpenShift Container Storage (OCS)
Cluster Name
ocs-storagecluster-cephcluster
Provider
VSphere
Version
ocs-operator.v4.3.0-369.ci

Inventory

3 Nodes
1 PVC
1 PV

Status

OCS Cluster
Data Resiliency

No persistent storage alerts

Capacity breakdown
View more
Projects

1.01 GiB used of 6 TiB
6 TiB available

openshift...
361.4 MiB

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

IMPORTANT

As of OpenShift Container Storage 4.2, cluster reduction, whether by reducing OSDs or nodes, is not supported.

4.2.2. 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 bare metal, Amazon EC2 (storage optimized - i3en.2xlarge), and VMware infrastructures.

IMPORTANT

Scaling up storage on bare metal, Amazon EC2 (storage optimized - i3en.2xlarge), and VMware 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.

NOTE

For Amazon EC2 (storage optimized - i3en.2xlarge) infrastructure, adding nodes is the only option for adding capacity, as deployment is done using both the available NVMe devices.

Prerequisites

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

- You must have installed local storage operator. For information, see Installing 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 find the unique **by-id** identifier for available devices that you want to add, that is, a minimum of one device per worker node. For more details, see Finding available storage devices.

   NOTE

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

   Example of **LocalVolume** CR **local-storage-block-expand.yaml** using OpenShift Storage Container label as node selector and **by-id** device identifier:

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: local-storage
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-
Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS60382E5D7441494EC   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4   # <-- modify this line
```

   You can see that in this CR new devices using **by-id** have been added. Each device maps to **nvme1n1** on one of the three worker nodes.

- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS60382E5D7441494EC**
- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9**
- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4**

2. Create the **LocalVolume** CR.

```
$ oc apply -f local-storage-block-expand.yaml
```

Example output:

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

3. Display PVs with **localblock**.

```
$ oc get pv
```

Example output:

```
local-pv-1d63db9e   1769Gi   RWO   Delete   Available   localblock
33s
local-pv-1eb9da0a   1769Gi   RWO   Delete   Available   localblock
25s
local-pv-31021a83   1769Gi   RWO   Delete   Available   localblock
48s
...
```

There are three more available PVs that you need to add to the **StorageCluster**.

4. To expand storage capacity, modify the **count** for **StorageDeviceSets** from **1** to **2**. Example of **StorageCluster** CR **cluster-service-AWS-expand.yaml** using **monDataDirHostPath** and **localblock** StorageClass:

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  manageNodes: false
  monDataDirHostPath: /var/lib/rook
  storageDeviceSets:
  - count: 2   # <-- modify count to 2
    dataPVCTemplate:
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 1
        storageClassName: localblock
        volumeMode: Block
    name: ocs-deviceset
    placement: {}
    portable: false
    replica: 3
    resources: {}
```

> **IMPORTANT**
>
> To ensure that the OSDs have a guaranteed size across the nodes, the storage size for **storageDeviceSets** must be specified as less than or equal to the size of the desired PVs created on the nodes.

5. Create the **StorageCluster** CR.

```
$ oc apply -f cluster-service-AWS-expand.yaml
```

Example output:

```
storagecluster.ocs.openshift.io/ocs-storagecluster configured
```

6. Display pods. You must now have three more OSD pods (osd-3, osd-4, and osd-5) and three more **osd-prepare** pods.

```
$ oc get pods -n openshift-storage | grep 'ceph-osd'
```

Example output:

```
rook-ceph-osd-3-568d8797b6-j5xqx                          1/1     Running     0          14m
rook-ceph-osd-4-cc4747fdf-5glgl                        1/1     Running     0          14m
rook-ceph-osd-5-94c46bbcc-tb7pw                          1/1     Running     0          14m
...
rook-ceph-osd-prepare-ocs-deviceset-0-1-mcmlv-qmn4r              0/1     Completed   0
14m
rook-ceph-osd-prepare-ocs-deviceset-1-1-tjh2d-fl5zc              0/1     Completed   0          14m
rook-ceph-osd-prepare-ocs-deviceset-2-1-nqlkg-x9wdn              0/1     Completed   0
14m
```

Verification steps

See Verifying your OpenShift Container Storage installation .

4.3. SCALING OUT STORAGE CAPACITY

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

- Add a new node
- Verify that the new node is added successfully
- Scale up the storage capacity

4.3.1. Adding a node

You can add nodes to increase the storage capacity when existing worker nodes are already running at their maximum supported OSDs, which is increment of 3 OSDs of the capacity selected during initial configuration.

Depending on the type of your deployment, you can choose one of the following procedures to add a storage node:

- For AWS installer-provisioned infrastructure, see Section 4.3.1.1, “Adding a node on an AWS installer-provisioned infrastructure”
- For AWS or VMware user-provisioned infrastructure, see Section 4.3.1.2, “Adding a node on an AWS or a VMware user-provisioned infrastructure”
- For bare metal, Amazon EC2, or VMware infrastructures, see Section 4.3.1.3, “Adding a node using a local storage device”

4.3.1.1. Adding a node on an AWS installer-provisioned infrastructure

Prerequisites

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

Procedure

1. Navigate to **Compute → Machine Sets**.
2. On the machine set where you want to add nodes, select **Edit Count**.
3. Add the amount of nodes, and click **Save**.
4. Click **Compute → Nodes** and confirm if the new node is in **Ready** state.
5. Apply the OpenShift Container Storage label to the new node.
   a. For the new node, **Action menu (⋮) → Edit Labels**.
   b. Add **cluster.ocs.openshift.io/openshift-storage=''** and click **Save**.

NOTE

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

Verification steps

To verify that the new node is added, see Section 4.3.2, “Verifying the addition of a new node” .

4.3.1.2. Adding a node on an AWS or a VMware user-provisioned infrastructure

Prerequisites

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

Procedure

1. Depending on whether you are adding a node on an AWS user provisioned infrastructure or a VMware user-provisioned infrastructure, perform the following steps:
   - For AWS
     a. Create a new AWS machine instance with the required infrastructure. See Supported Infrastructure and Platforms.
     b. Create a new OpenShift Container Platform node using the new AWS machine instance.
   - For VMware:
     a. Create a new VM on vSphere with the required infrastructure. See Supported Infrastructure and Platforms.
     b. Create a new OpenShift Container Platform worker node using the new VM.
2. Check for certificate signing requests (CSRs) related to OpenShift Container Storage that are in **Pending** state:

```
$ oc get csr
```

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

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

4. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
5. 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 4.3.2, “Verifying the addition of a new node” .

4.3.1.3. Adding a node using a local storage device

Use this procedure to add a node on bare metal, Amazon EC2 (storage optimized - **i3en.2xlarge**), and VMware infrastructures.

IMPORTANT

Scaling storage nodes for bare metal, Amazon EC2, and VMware 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.

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. Depending on whether you are adding a node on bare metal, Amazon EC2 (storage optimized - i3en.2xlarge), or VMware infrastructure, perform the following steps:
   - For AWS
     a. Create a new AWS machine instance with the required infrastructure. See Supported Infrastructure and Platforms.
     b. Create a new OpenShift Container Platform node using the new AWS machine instance.
   - For VMware:
     a. Create a new VM on vSphere with the required infrastructure. See Supported Infrastructure and Platforms.
     b. Create a new OpenShift Container Platform worker node using the new VM.
   - For bare metal:
     a. Get a new bare metal machine with the required infrastructure. See Supported Infrastructure and Platforms.
     b. Create a new OpenShift Container Platform node using the new bare metal machine.
2. Check for certificate signing requests (CSRs) related to OpenShift Container Storage that are in **Pending** state:

```
$ oc get csr
```

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

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

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

5. 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 4.3.2, “Verifying the addition of a new node” .

4.3.2. 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-***

3. Verify the health of OpenShift Container Storage cluster using the persistent storage dashboard:

   a. Click **Home → Dashboards** from the left pane of the OpenShift Web Console and click the **OCS PV** tab.

   b. In the **Status** card, verify that the cluster health displays a green tick icon.

4.3.3. Scaling up storage capacity

To scale up storage capacity, see Scaling up storage by adding capacity .

CHAPTER 5. MANAGING PERSISTENT VOLUME CLAIMS

IMPORTANT

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

5.1. 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**.

5.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS

WARNING

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

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

Prerequisites

- Administrator access to OpenShift Container Storage.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Storage** → **Persistent Volume Claims**
3. Search for the required PVC name by using the **Filter** textbox.
4. Check the **Status** column corresponding to the required PVC.
5. Click the required **Name** to view the PVC details.

5.3. 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** → **Dashboards** → **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.

5.4. DYNAMIC PROVISIONING

5.4.1. About dynamic provisioning

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

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

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

5.4.2. Dynamic provisioning in OpenShift Container Storage

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

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

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

Version 4.3 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). For more information about data federation, please see Section 7.7.3, “Stretching buckets across two different clusters [Technology Preview]”.

In OpenShift Container Storage 4.3, 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.

5.4.3. Available dynamic provisioning plug-ins

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

Storage type	Provisioner plug-in name	Notes
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	For dynamic provisioning when using multiple clusters in different zones, tag each node with **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** where **<cluster_name>** and **<cluster_id>** are unique per cluster.
AWS Elastic File System (EFS)		Dynamic provisioning is accomplished through the EFS provisioner pod and not through a provisioner plug-in.
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	The **persistent-volume-binder** ServiceAccount requires permissions to create and get Secrets to store the Azure storage account and keys.
Ceph File System (POSIX Compliant filesystem)	**openshift-storage.cephfs.csi.ceph.com**	Provisions a volume for ReadWriteMany (RWX) or ReadWriteOnce (RWO) access modes using the Ceph Filesytem configured in a Ceph cluster.
Ceph RBD (Block Device)	**openshift-storage.rbd.csi.ceph.com**	Provisions a volume for RWO access mode for Ceph RBD, RWO and RWX access mode for block PVC, and RWO access mode for Filesystem PVC.
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	In multi-zone configurations, it is advisable to run one OpenShift Container Platform cluster per GCE project to avoid PVs from being created in zones where no node in the current cluster exists.

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

IMPORTANT

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

CHAPTER 6. MANAGING CONTAINER STORAGE INTERFACE (CSI) COMPONENT PLACEMENTS

Each cluster consists of a number of dedicated nodes such as **infra** and **storage** nodes. However, a **infra** node will not be able to use OpenShift Container Storage persistent volume claims (PVCs) on the node. So, if you want to use such nodes, you can set tolerations to bring up **csi-plugins** on the nodes. For more information, see https://access.redhat.com/solutions/4827161.

Procedure

1. Create a **configmap rook-ceph-operator-config**.

```
$ oc create -f rook-ceph-operator-config.yaml
configmap/rook-ceph-operator-config created
```

2. Display **configmap**.

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

Example output of **configmap** with **key** set to **nodetype** and **value** set to **infra**:

```
apiVersion: v1
data:
  CSI_PLUGIN_TOLERATIONS: |
    - effect: NoSchedule
      key: nodetype
      operator: Equal
      value: infra
    - effect: NoSchedule
      key: node.ocs.openshift.io/storage
      operator: Exists
kind: ConfigMap
metadata:
  creationTimestamp: "2020-03-23T11:49:27Z"
  name: rook-ceph-operator-config
  namespace: openshift-storage
  resourceVersion: "114879"
  selfLink: /api/v1/namespaces/openshift-storage/configmaps/rook-ceph-operator-config
  uid: ac22e63a-8df1-4650-a57f-89bf7a2ce06a
```

3. Restart the **rook-ceph-operator**.

Verification step

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

CHAPTER 7. MULTICLOUD OBJECT GATEWAY

7.1. ABOUT THE MULTICLOUD OBJECT GATEWAY

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

7.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS

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

Prerequisites

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

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

- Alternatively, you can install the **mcg** package from the OpenShift Container Storage RPMs found here https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages

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

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

7.2.1. Accessing the Multicloud Object Gateway from the terminal

Procedure

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

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

The output will look similar to the following:

```
Name:         noobaa
Namespace:    openshift-storage
Labels:       <none>
Annotations:  <none>
API Version:  noobaa.io/v1alpha1
Kind:         NooBaa
Metadata:
```

```
  Creation Timestamp:  2019-07-29T16:22:06Z
  Generation:          1
  Resource Version:    6718822
  Self Link:           /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                 019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret Ref:
        Name:           noobaa-admin
        Namespace:      openshift-storage
  Actual Image:         noobaa/noobaa-core:4.0
  Observed Generation:  1
  Phase:                Ready
  Readme:

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

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

    Lets get started:

    1. Connect to Management console:

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

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

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

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

    2. Test S3 client:

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


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

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

NOTE

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

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

Prerequisites

- Download the MCG command-line interface:

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

Procedure

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

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

The output will look similar to the following:

```
INFO[0000] Namespace: openshift-storage
INFO[0000]
INFO[0000] CRD Status:
INFO[0003]   Exists: CustomResourceDefinition "noobaas.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "backingstores.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "bucketclasses.noobaa.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbucketclaims.objectbucket.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbuckets.objectbucket.io"
INFO[0004]
INFO[0004] Operator Status:
INFO[0004]   Exists: Namespace "openshift-storage"
INFO[0004]   Exists: ServiceAccount "noobaa"
INFO[0005]   Exists: Role "ocs-operator.v0.0.271-6g45f"
INFO[0005]   Exists: RoleBinding "ocs-operator.v0.0.271-6g45f-noobaa-f9vpj"
INFO[0006]   Exists: ClusterRole "ocs-operator.v0.0.271-fjhgh"
INFO[0006]   Exists: ClusterRoleBinding "ocs-operator.v0.0.271-fjhgh-noobaa-pdxn5"
INFO[0006]   Exists: Deployment "noobaa-operator"
INFO[0006]
INFO[0006] System Status:
INFO[0007]   Exists: NooBaa "noobaa"
INFO[0007]   Exists: StatefulSet "noobaa-core"
INFO[0007]   Exists: Service "noobaa-mgmt"
INFO[0008]   Exists: Service "s3"
INFO[0008]   Exists: Secret "noobaa-server"
INFO[0008]   Exists: Secret "noobaa-operator"
INFO[0008]   Exists: Secret "noobaa-admin"
INFO[0009]   Exists: StorageClass "openshift-storage.noobaa.io"
INFO[0009]   Exists: BucketClass "noobaa-default-bucket-class"
INFO[0009]   (Optional) Exists: BackingStore "noobaa-default-backing-store"
INFO[0010]   (Optional) Exists: CredentialsRequest "noobaa-cloud-creds"
INFO[0010]   (Optional) Exists: PrometheusRule "noobaa-prometheus-rules"
INFO[0010]   (Optional) Exists: ServiceMonitor "noobaa-service-monitor"
INFO[0011]   (Optional) Exists: Route "noobaa-mgmt"
INFO[0011]   (Optional) Exists: Route "s3"
INFO[0011]   Exists: PersistentVolumeClaim "db-noobaa-core-0"
INFO[0011]   System Phase is "Ready"
INFO[0011]   Exists:  "noobaa-admin"

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

1
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 -#
#----------------#

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. Creating a new backing store

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

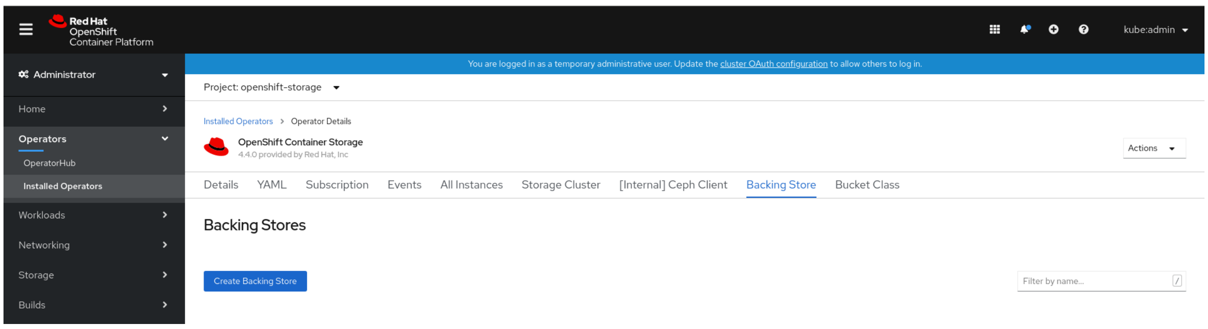
Prerequisites

- 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. On the Installed Operator page, select **openshift-storage** from the Project drop down list to switch to the **openshift-storage** project.
3. Click **OpenShift Container Storage** Operator.
4. On the OpenShift Container Storage Operator page, scroll right and click the **Backing Store** tab.

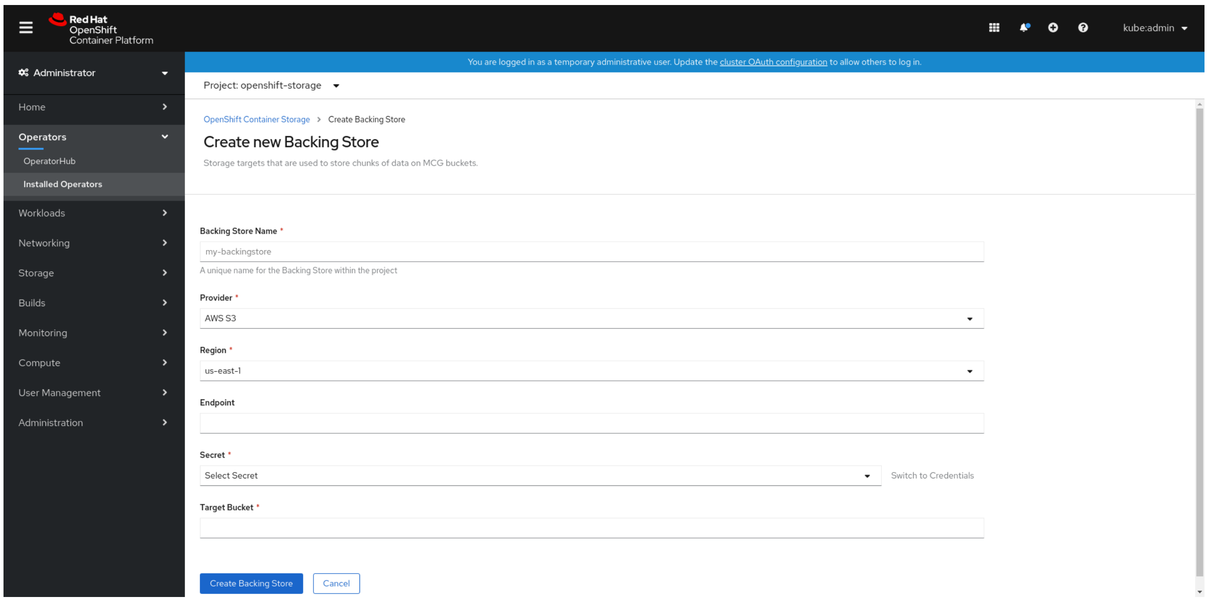
Figure 7.1. OpenShift Container Storage Operator page with backing store tab



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

5. Click **Create Backing Store**.

Figure 7.2. Create Backing Store page



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

Project: openshift-storage

OpenShift Container Storage > Create Backing Store

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
Switch to Credentials

Target Bucket *

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

   a. Make sure the Namespace is set to **openshift-storage**.

   b. Enter a **Backing Store Name**.

   c. Select **Provider**.

   d. Select **Region**.

   e. Enter **Endpoint**. This is optional.

   f. Select **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.

   g. 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 NooBaa that it can use this bucket for the system.

7. Click **Create Backing Store**.

Verification steps

1. Click **Operators → Installed Operators**.

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

3. Search for the new backing store.

7.3.2. 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.

You must add a backing storage that can be used by the MCG.

Depending on the type of your deployment, you can choose one of the following procedures to create a backing storage:

- For creating a AWS-backed backingstore, see Section 7.3.2.1, "Creating an AWS-backed backingstore"
- For creating a IBM COS-backed backingstore, see Section 7.3.2.2, "Creating an IBM COS-backed backingstore"

For VMWare deployments, skip to Section 7.3.3, "Creating an S3 compatible Multicloud Object Gateway backingstore" for further instructions.

7.3.2.1. Creating an AWS-backed backingstore

Prerequisites

- Download the Multicloud Object Gateway (MCG) command-line interface:

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

- Alternatively, you can install the **mcg** package from the OpenShift Container Storage RPMs found here https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages

Procedure

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

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

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

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

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

   ```
   INFO[0001]   Exists: NooBaa "noobaa"
   INFO[0002]   Created: BackingStore "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: openshift-storage
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: noobaa
    targetBucket: <bucket-name>
  type: aws-s3
```

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

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

7.3.2.2. Creating an IBM COS-backed backingstore

Prerequisites

- Download the Multicloud Object Gateway (MCG) command-line interface:

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

- Alternatively, you can install the **mcg** package from the OpenShift Container Storage RPMs found here https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages

Procedure

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

```
noobaa backingstore create ibm-cos <backingstore_name> --access-key=<IBM ACCESS
KEY> --secret-key=<IBM SECRET ACCESS KEY> --endpoint=<IBM COS ENDPOINT> --
target-bucket <bucket-name>
```

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

   b. Replace **<IBM ACCESS KEY>**, **<IBM SECRET ACCESS KEY>**, **<IBM COS ENDPOINT>** with an IBM access key ID, secret access key and the appropriate regional endpoint that corresponds to the location of the existing IBM bucket.
   To generate the above keys on IBM cloud, you must include HMAC credentials while creating the service credentials for your target bucket.

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

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "ibm-resource"
INFO[0002]   Created: Secret "backing-store-secret-ibm-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:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

   a. You must supply and encode your own IBM COS access key ID and secret access key using Base64, and use the results in place of **<IBM COS ACCESS KEY ID ENCODED IN BASE64>** and **<IBM COS 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: openshift-storage
spec:
  ibmCos:
    endpoint: <endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: ibm-cos
```

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

b. Replace **<endpoint>** with a regional endpoint that corresponds to the location of the existing IBM bucket name. This argument tells Multicloud Object Gateway which endpoint to use for its backing store, and subsequently, data storage and administration.

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

7.3.3. Creating an S3 compatible Multicloud Object Gateway backingstore

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

Procedure

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

```
noobaa backingstore create s3-compatible rgw-resource --access-key=<RGW ACCESS
KEY> --secret-key=<RGW SECRET KEY> --target-bucket=<bucket-name> --
endpoint=http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
storage.svc.cluster.local:80
```

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

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

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

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

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

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

You can also create the backingstore using a YAML:

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

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

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

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

   ```
   apiVersion: noobaa.io/v1alpha1
   kind: BackingStore
   metadata:
     finalizers:
     - noobaa.io/finalizer
     labels:
       app: noobaa
     name: <backingstore-name>
     namespace: openshift-storage
   spec:
     s3Compatible:
       endpoint: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
   storage.svc.cluster.local:80
       secret:
         name: <backingstore-secret-name>
         namespace: openshift-storage
       signatureVersion: v4
       targetBucket: <RGW-bucket-name>
     type: s3-compatible
   ```

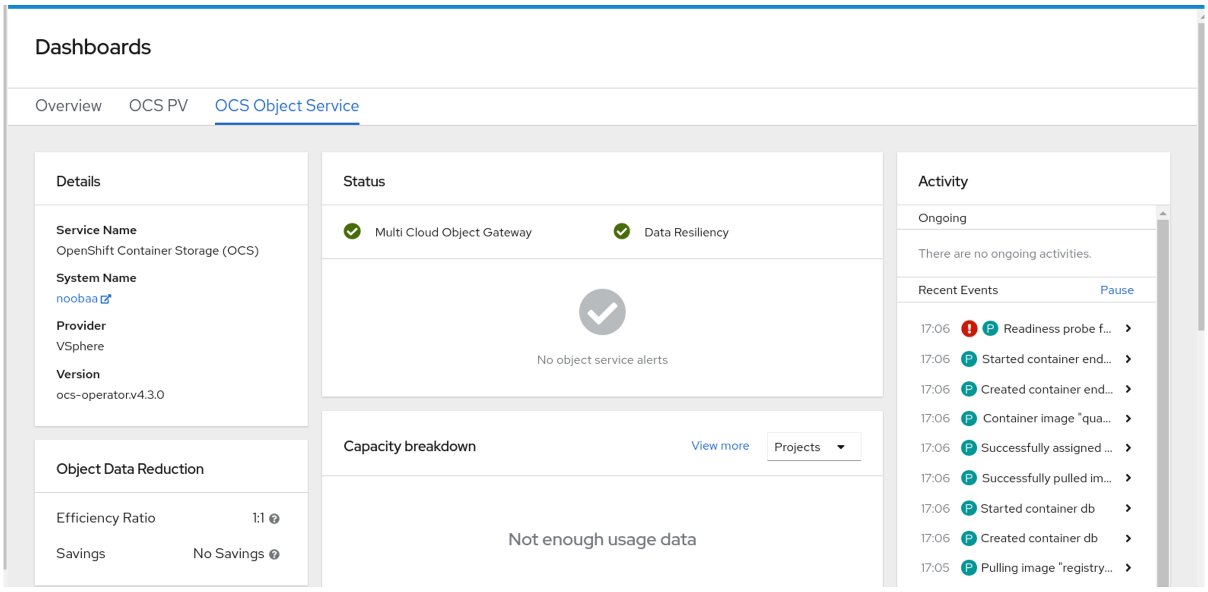
   a. Replace **<backingstore-secret-name>** with the name of the secret that was created with **CephObjectStore** in the previous step.

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

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

Procedure

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



Details

Service Name
OpenShift Container Storage (OCS)

System Name
noobaa

Provider
VSphere

Version
ocs-operator.v4.3.0

Object Data Reduction

Efficiency Ratio 1:1

Savings No Savings

Status

Multi Cloud Object Gateway ✓ Data Resiliency ✓

No object service alerts

Capacity breakdown — View more — Projects

Not enough usage data

Activity

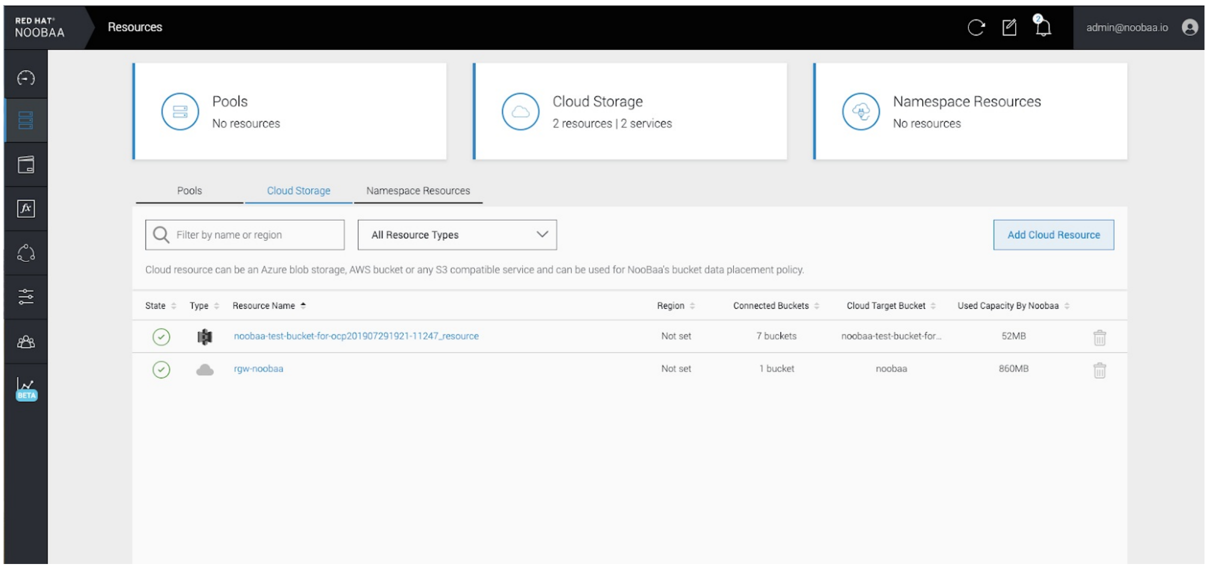
Ongoing

There are no ongoing activities.

Recent Events — Pause

17:06	Readiness probe f...
17:06	Started container end...
17:06	Created container end...
17:06	Container image "qua...
17:06	Successfully assigned ...
17:06	Successfully pulled im...
17:06	Started container db
17:06	Created container db
17:05	Pulling image "registry...

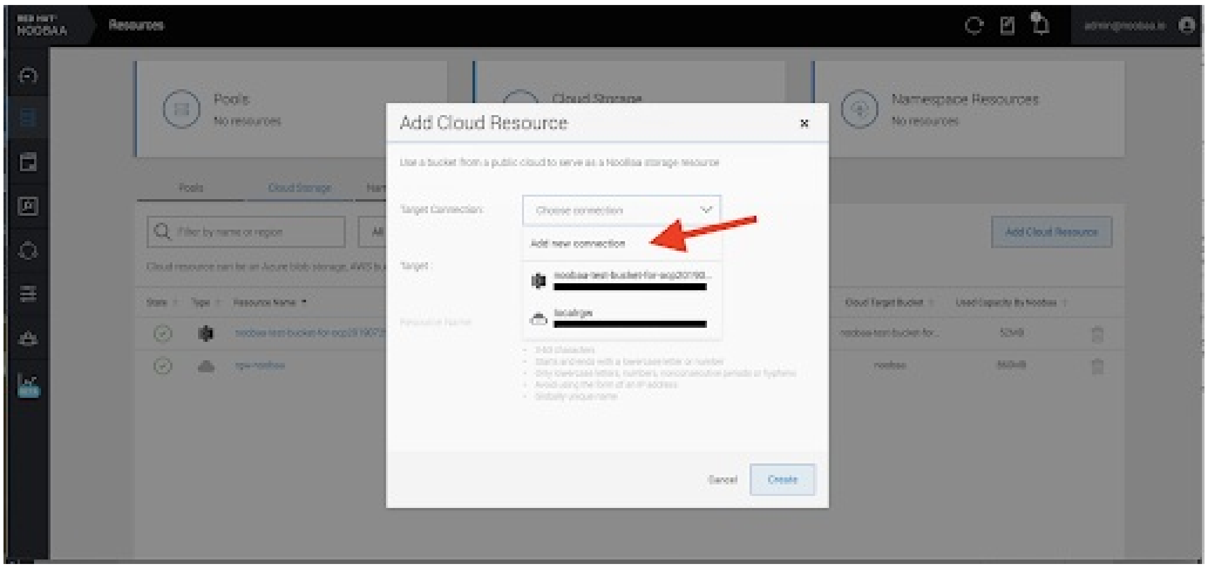
2. Select the Resources tab in the left, highlighted below. From the list that populates, select **Add Cloud Resource**:



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.

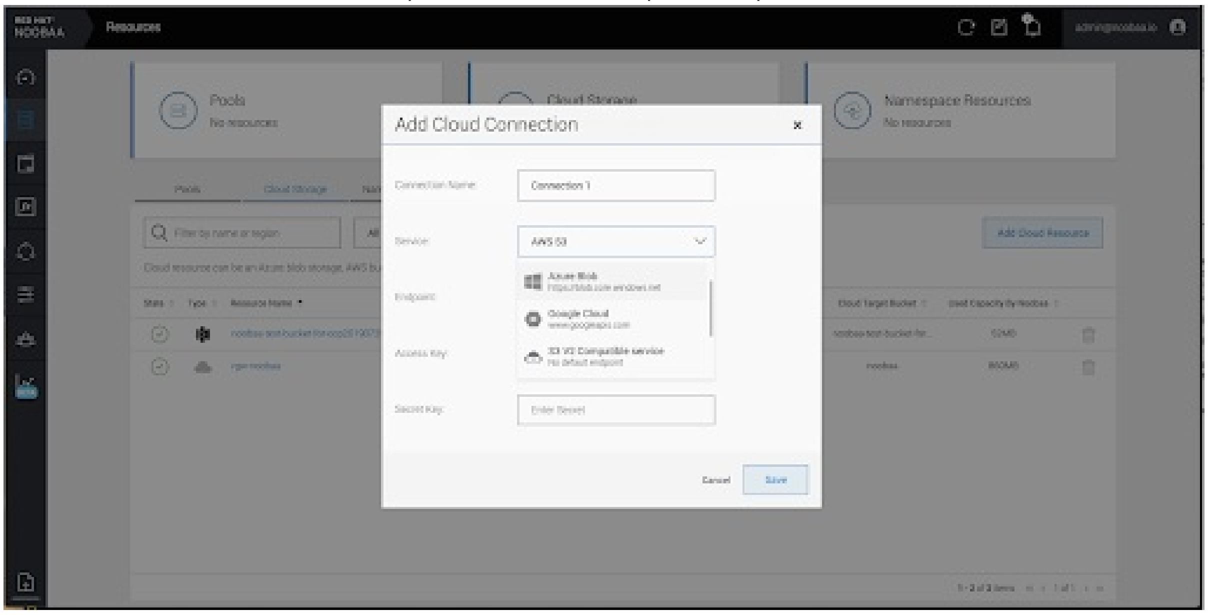
State	Type	Resource Name	Region	Connected Buckets	Cloud Target Bucket	Used Capacity By Noobaa
✓		noobaa-test-bucket-for-ocp201907291921-11247_resource	Not set	7 buckets	noobaa-test-bucket-for...	52MB
✓		rgw-noobaa	Not set	1 bucket	noobaa	860MB

3. Select **Add new connection**:

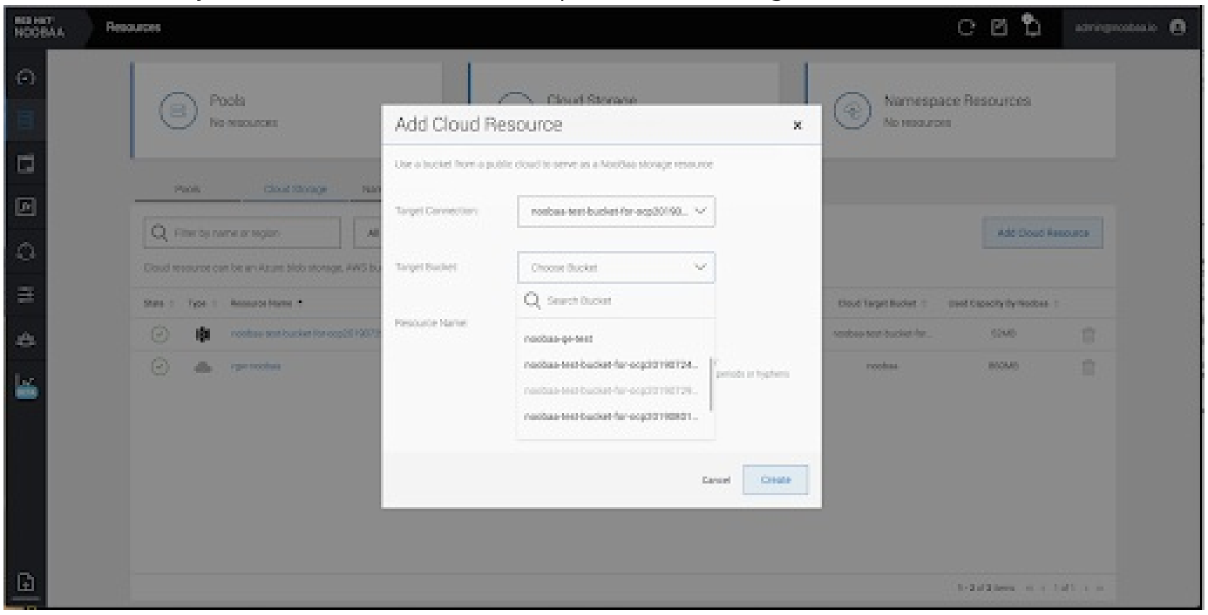


noobaa-test-bucket-for-ocp201907...
[redacted]
localrgw
[redacted]

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



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



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

NOTE

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

7.3.5. 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.

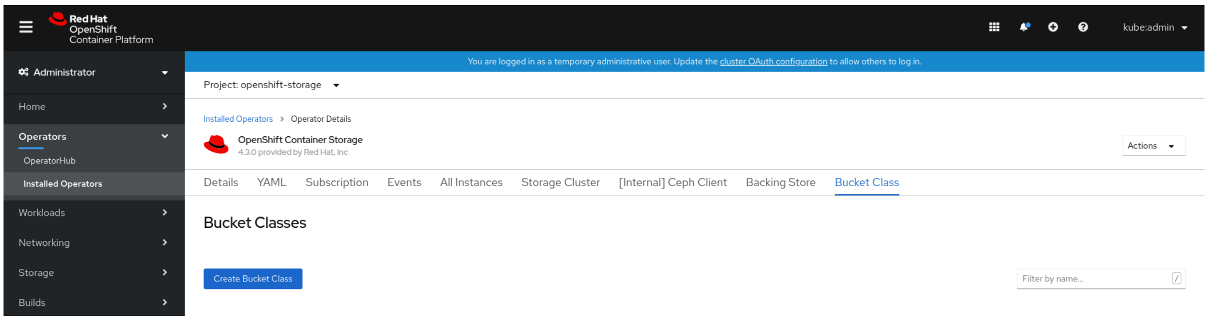
Procedure

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 7.3. OpenShift Container Storage Operator page with Bucket Class tab

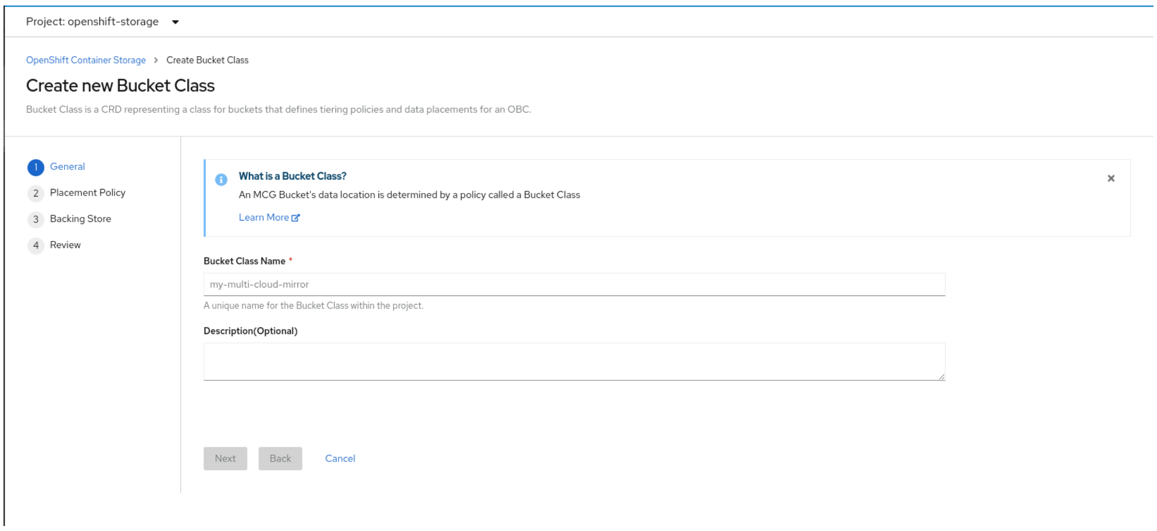


4. Click **Create Bucket Class**.

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

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

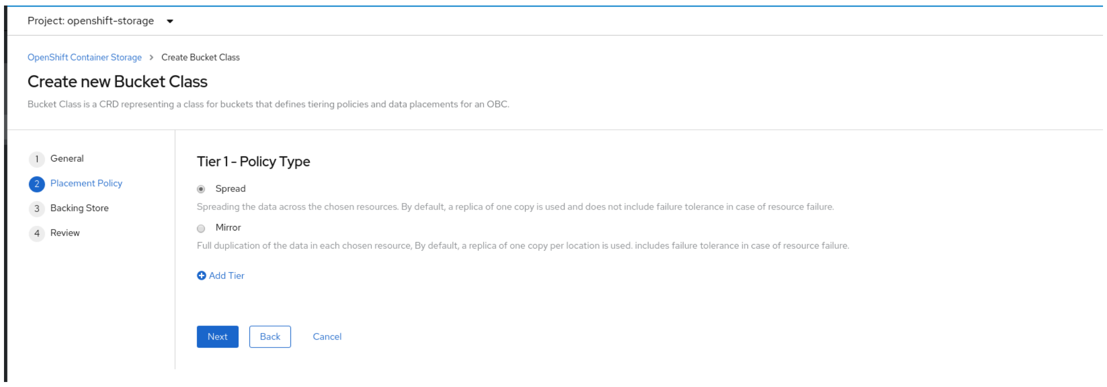
   Figure 7.4. Create Bucket Class page


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 7.5. 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

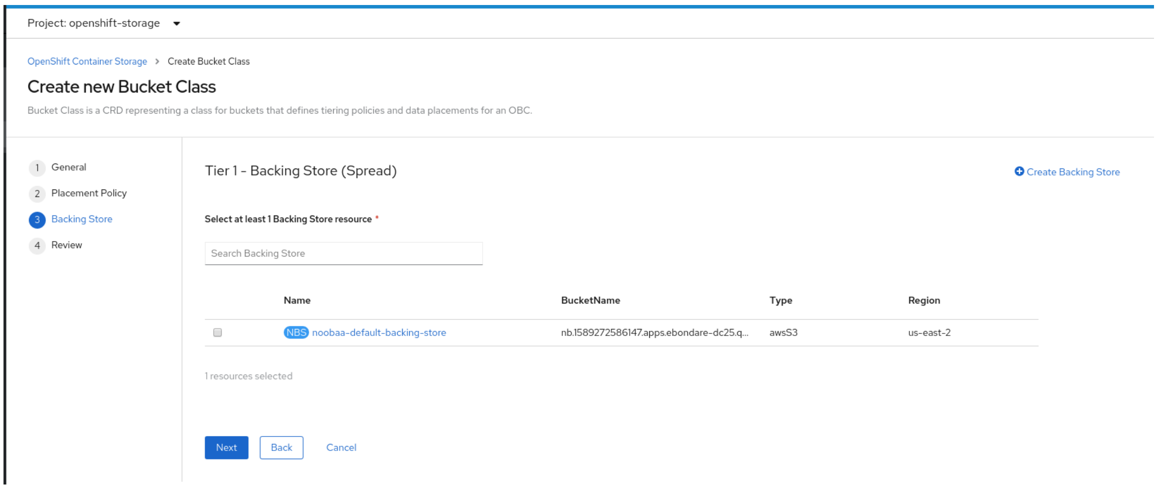
Full duplication of the data in each chosen resource, By default, a replica of one copy per location is used. includes failure tolerance in case of resource failure.

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 7.6. 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

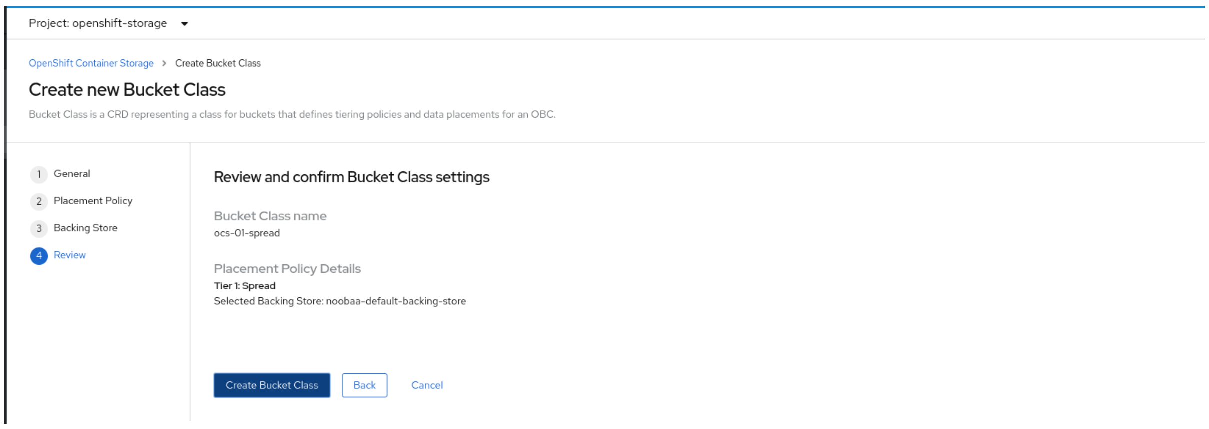
Next Back Cancel

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 7.7. 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

Create Bucket Class Back Cancel

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.

7.4. 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
```

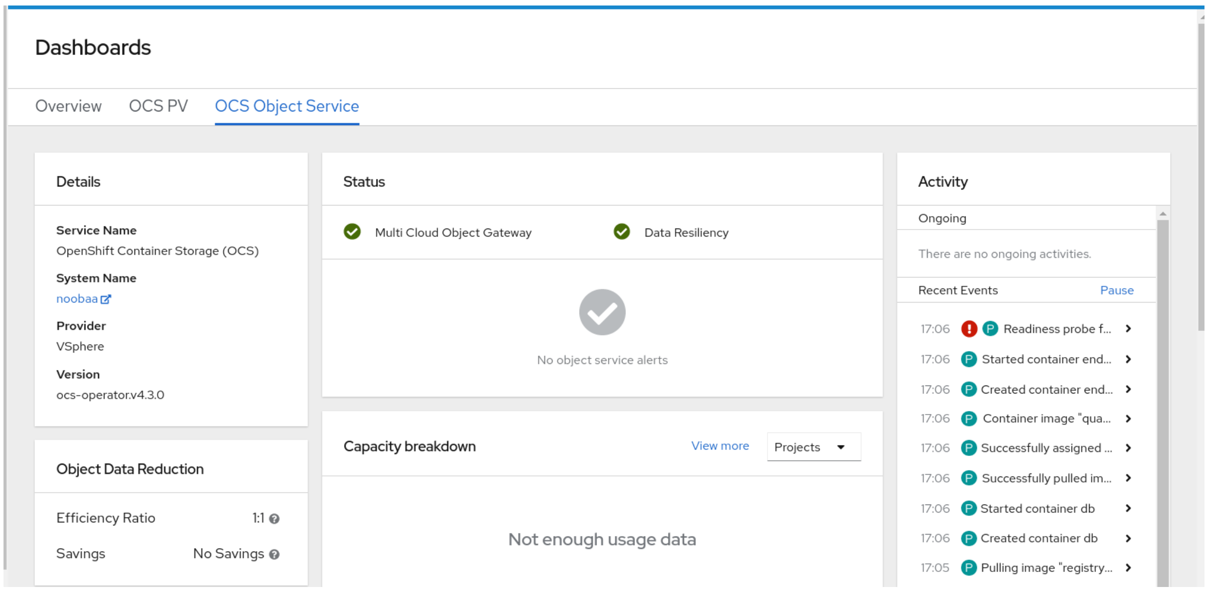
2. Add the following lines to your standard Object Bucket Claim (OBC):

```
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 **Dashboards** → **OCS Object Service** → select the **noobaa** link:

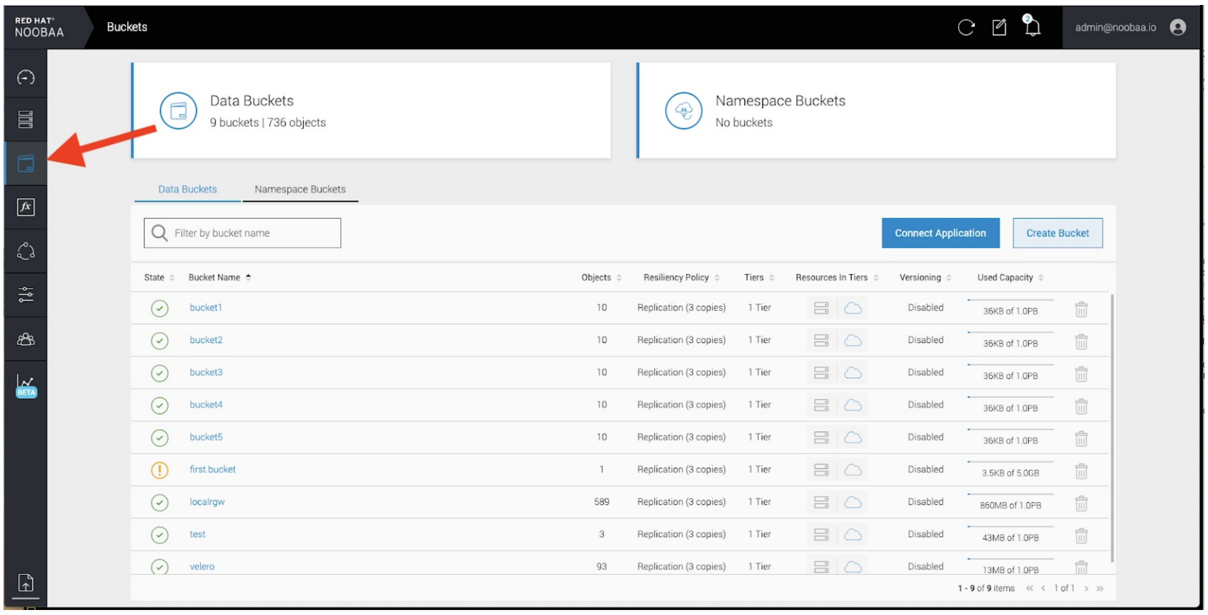


Service Name
OpenShift Container Storage (OCS)
System Name
noobaa
Provider
VSphere
Version
ocs-operator.v4.3.0

Efficiency Ratio	1:1
Savings	No Savings

17:06	Readiness probe f...
17:06	Started container end...
17:06	Created container end...
17:06	Container image "qua...
17:06	Successfully assigned ...
17:06	Successfully pulled im...
17:06	Started container db
17:06	Created container db
17:05	Pulling image "registry...

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

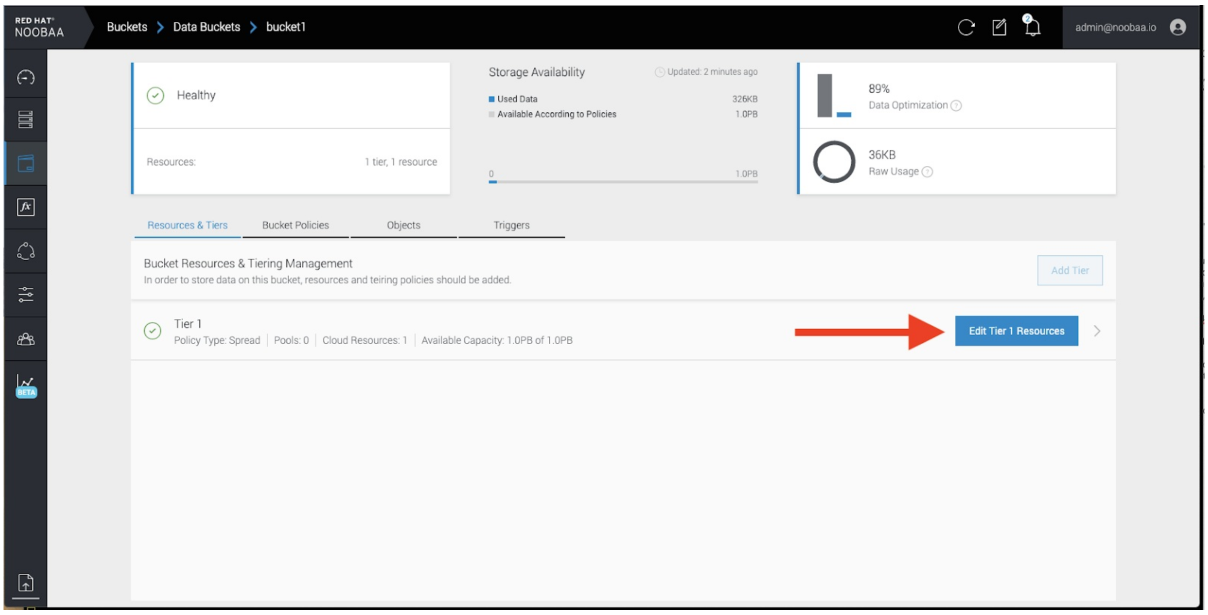


State	Bucket Name	Objects	Resiliency Policy	Tiers	Resources In Tiers	Versioning	Used Capacity
✓	bucket1	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket2	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket3	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket4	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

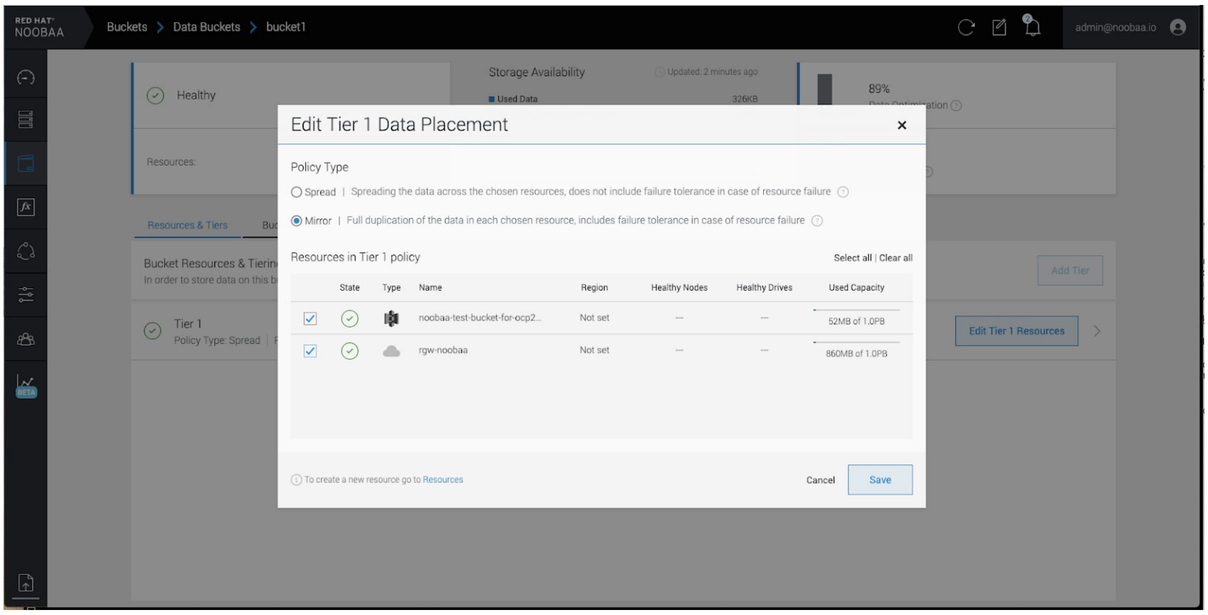
1 - 9 of 9 items 1 of 1

3. Click the bucket you want to update.

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


RED HAT NOOBAA
Buckets > Data Buckets > bucket1
admin@noobaa.io
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
Resources & Tiers
Bucket Policies
Objects
Triggers
Bucket Resources & Tiering Management
In order to store data on this bucket, resources and teiring policies should be added.
Add Tier
Tier 1
Policy Type: Spread | Pools: 0 | Cloud Resources: 1 | Available Capacity: 1.0PB of 1.0PB
Edit Tier 1 Resources

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



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 s3api put-bucket-policy --bucket MyBucket --policy BucketPolicy
```

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

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

For example:

```
# aws 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

Only Noobaa account principals are supported.

NOTE

Bucket policy conditions are not supported.

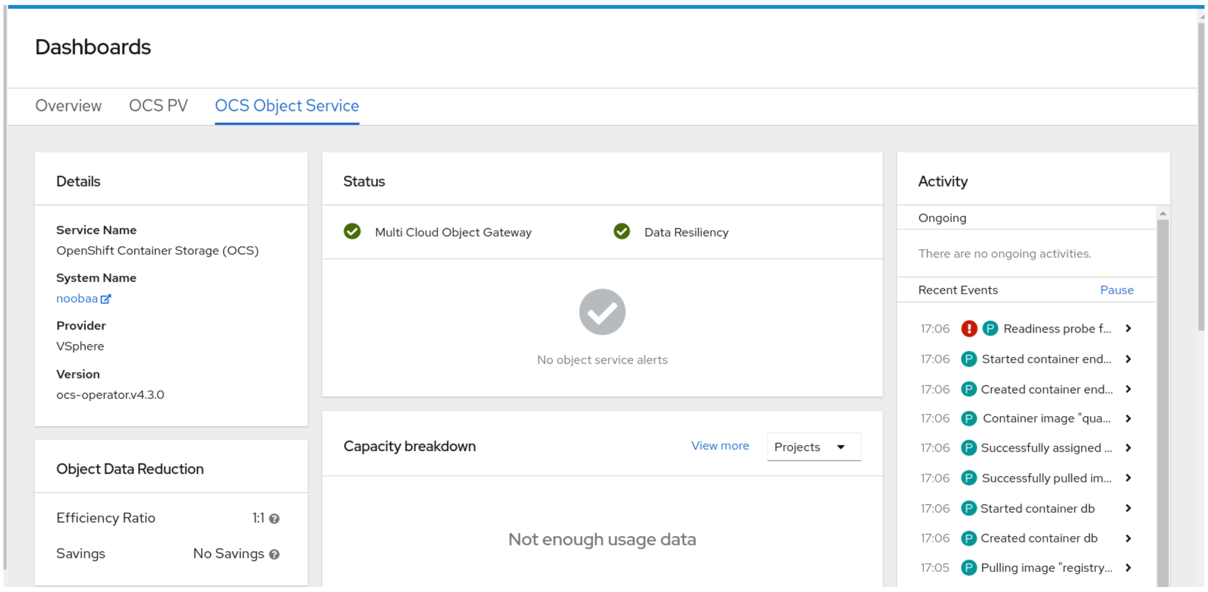
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway

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 **Dashboards** → **OCS Object Service** → select the **noobaa** link:



Dashboards

Overview | OCS PV | OCS Object Service

Details

Service Name
OpenShift Container Storage (OCS)

System Name
noobaa

Provider
VSphere

Version
ocs-operator.v4.3.0

Status

Multi Cloud Object Gateway | Data Resiliency

No object service alerts

Activity

Ongoing

There are no ongoing activities.

Recent Events | Pause

17:06	P	Readiness probe f...
17:06	P	Started container end...
17:06	P	Created container end...
17:06	P	Container image "qua...
17:06	P	Successfully assigned ...
17:06	P	Successfully pulled im...
17:06	P	Started container db
17:06	P	Created container db
17:05	P	Pulling image "registry...

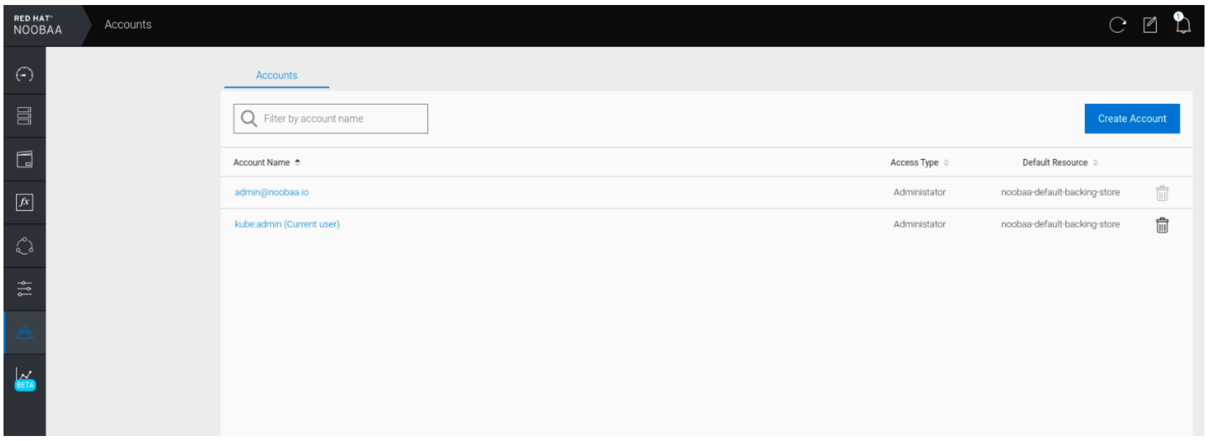
Object Data Reduction

Efficiency Ratio	1:1
Savings	No Savings

Capacity breakdown | View more | Projects

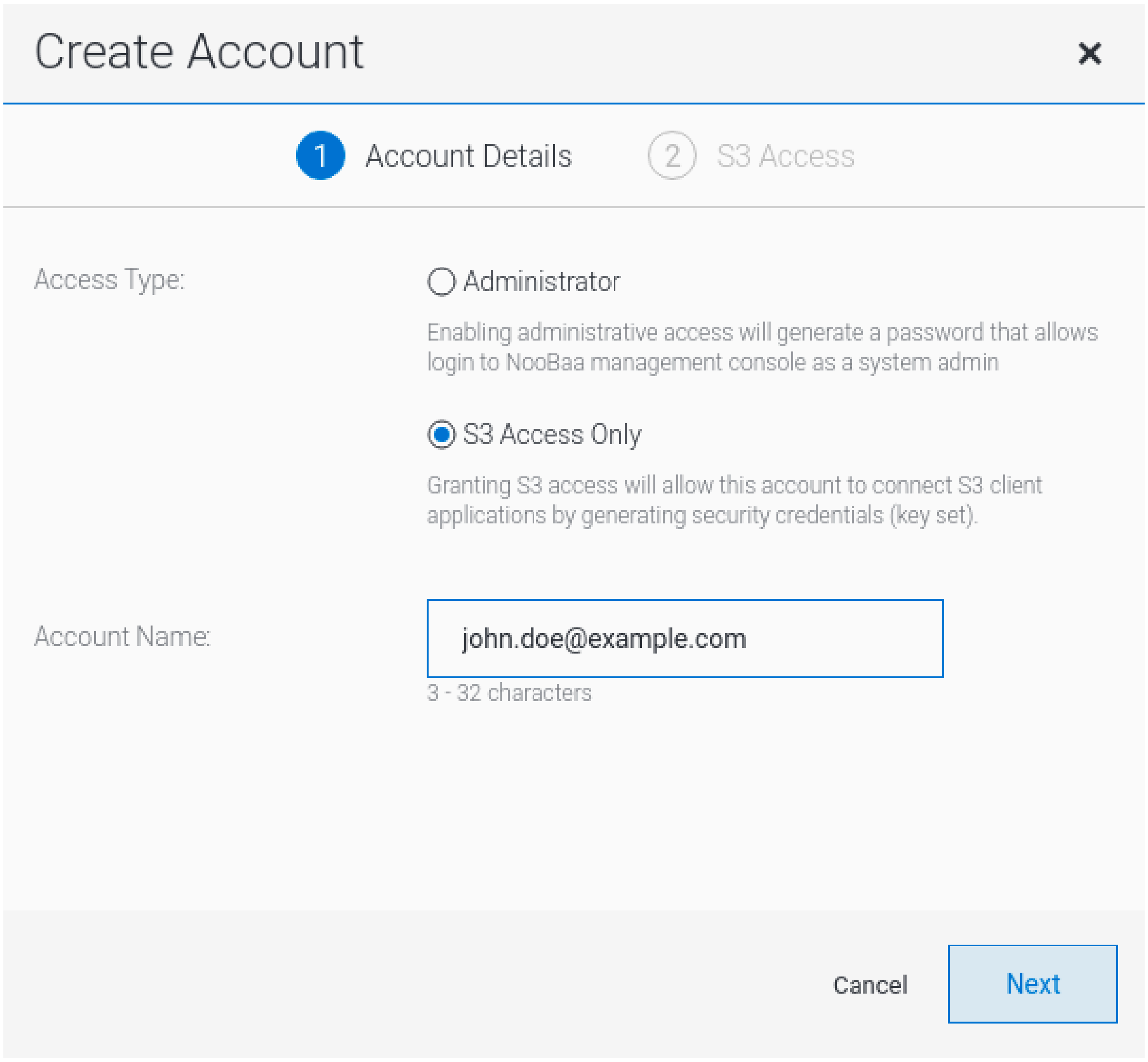
Not enough usage data

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

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

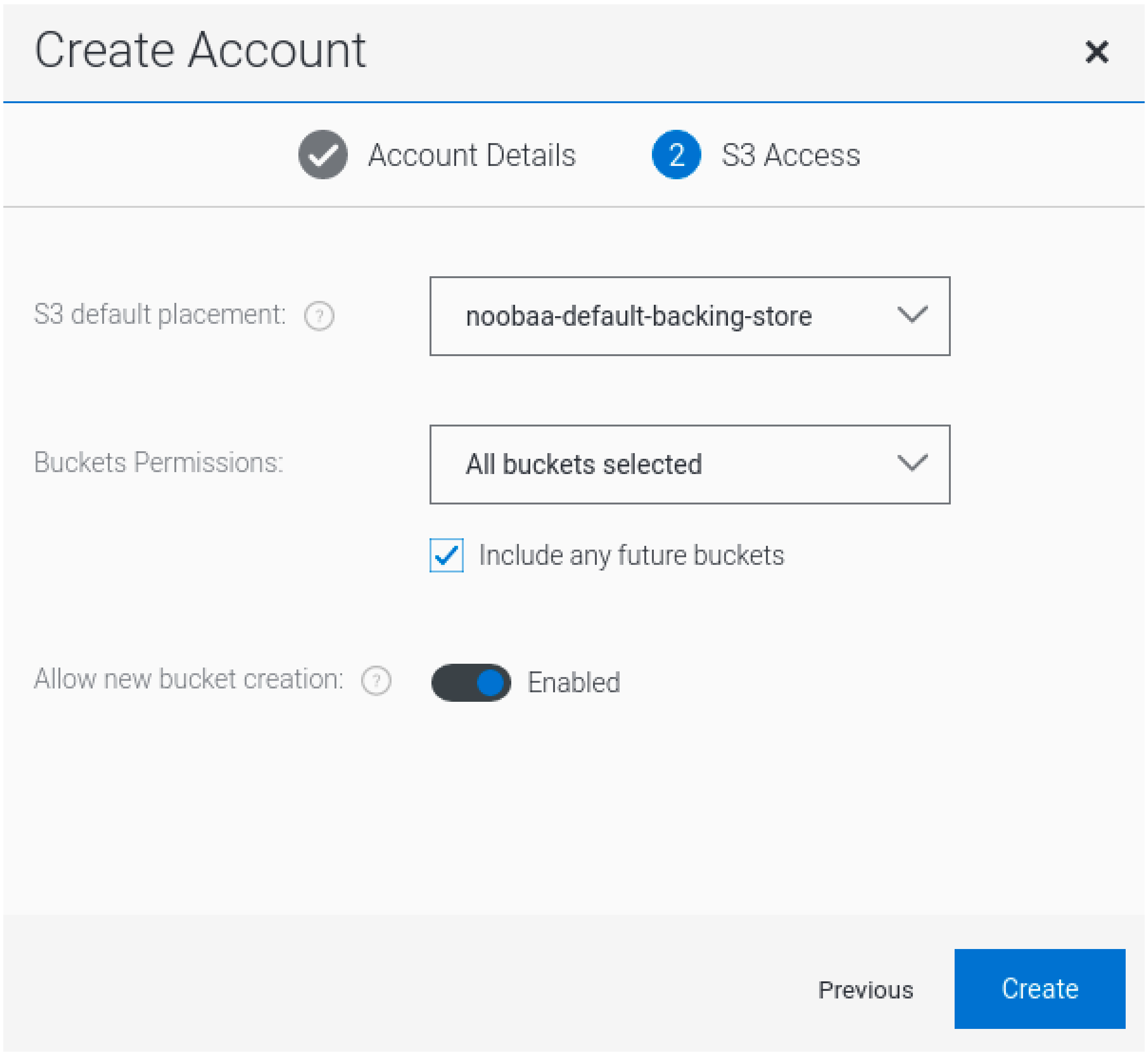
(●) S3 Access Only

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

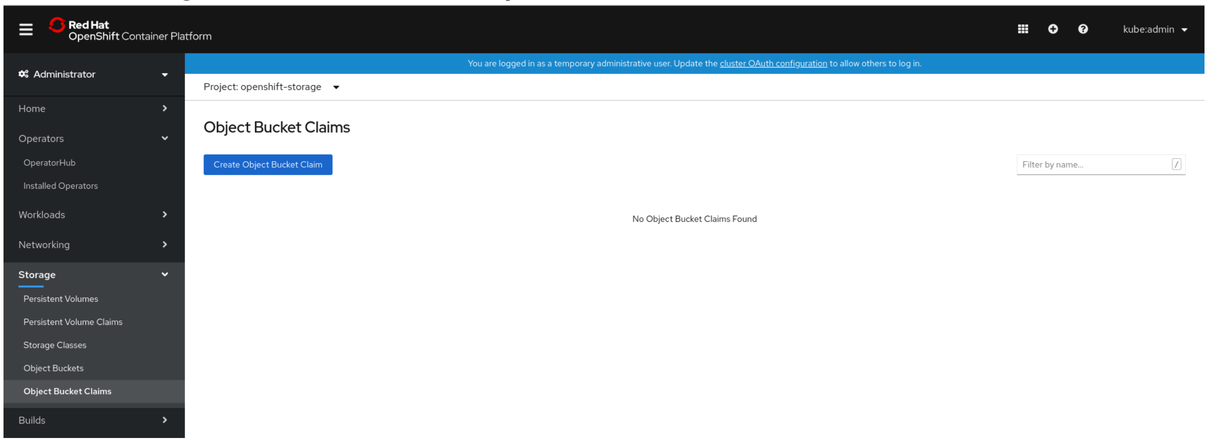
- Administrative access to the OpenShift Web Console.

Procedure

1. Log into the OpenShift Web Console.

2. On the left navigation bar, click **Storage → Object Bucket Claims**.

3. In the following window, click **Create Object Bucket Claim**



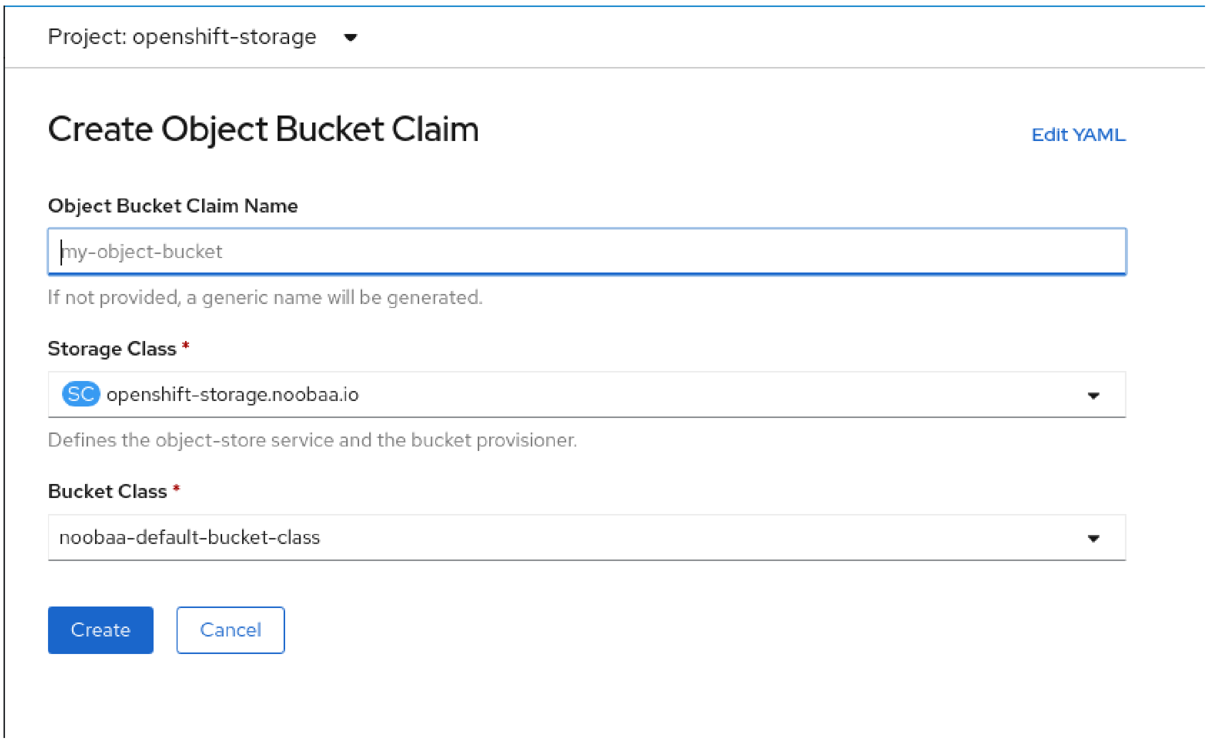
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-storage

Object Bucket Claims

No Object Bucket Claims Found

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



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 *

SC openshift-storage.noobaa.io

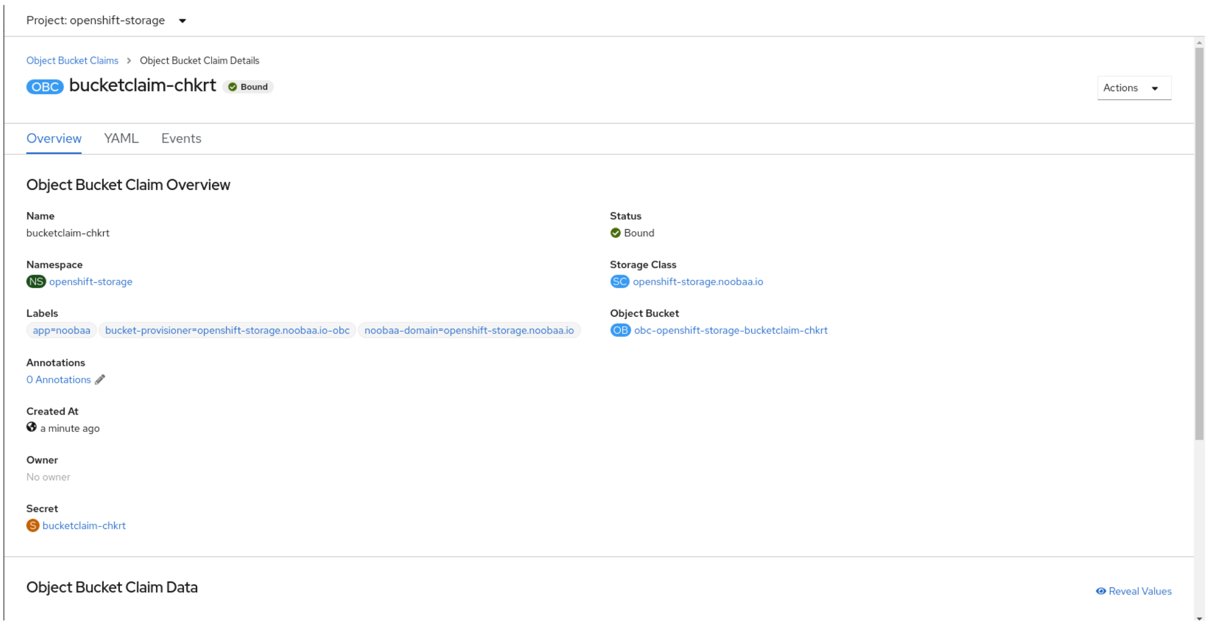
Defines the object-store service and the bucket provisioner.

Bucket Class *

noobaa-default-bucket-class

Create Cancel

5. Click **Create**.
   Once the OBC is created, you will be 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

Namespace
NS openshift-storage

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

Annotations
0 Annotations

Created At
a minute ago

Owner
No owner

Secret
S bucketclaim-chkrt

Status
Bound

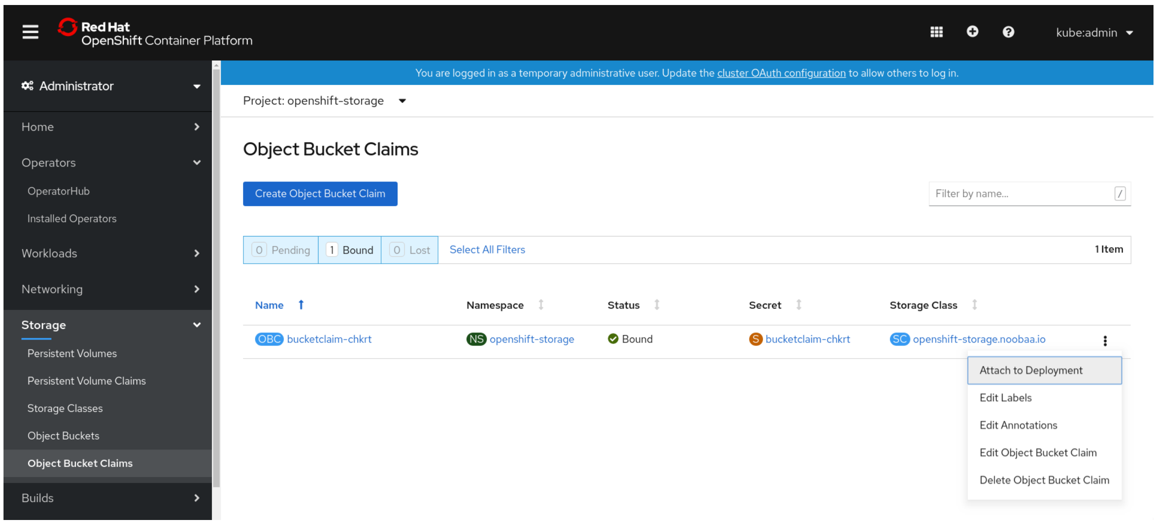
Storage Class
SC openshift-storage.noobaa.io

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

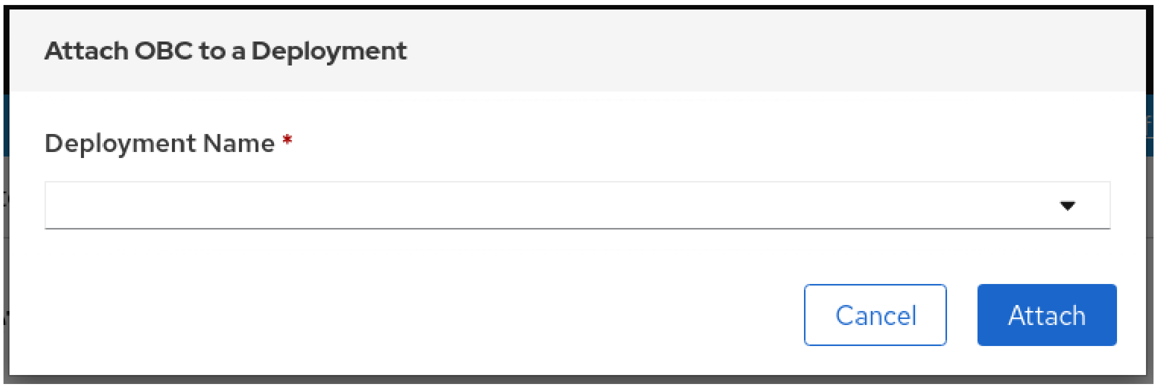
Object Bucket Claim Data

Reveal Values

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


Red Hat OpenShift Container Platform
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Persistent Volumes
Persistent Volume Claims
Storage Classes
Object Buckets
Object Bucket Claims
Builds
Project: openshift-storage
Object Bucket Claims
Create Object Bucket Claim
Filter by name...
0 Pending 1 Bound 0 Lost Select All Filters 1 Item
Name Namespace Status Secret Storage Class
OBC bucketclaim-chkrt NS openshift-storage Bound S bucketclaim-chkrt SC openshift-storage.noobaa.io
Attach to Deployment
Edit Labels
Edit Annotations
Edit Object Bucket Claim
Delete Object Bucket Claim

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

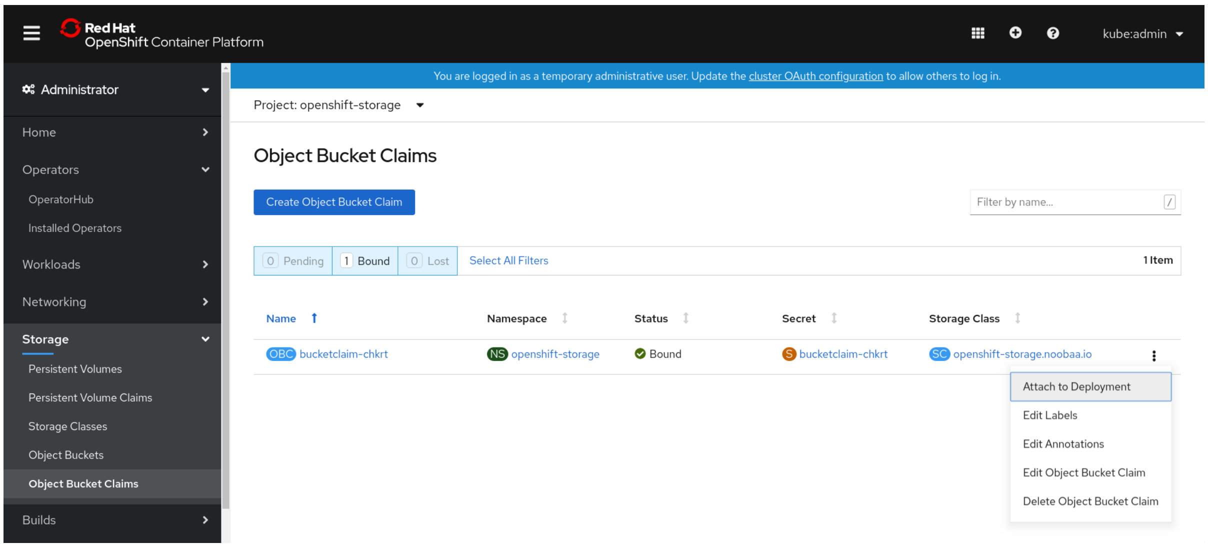


NOTE

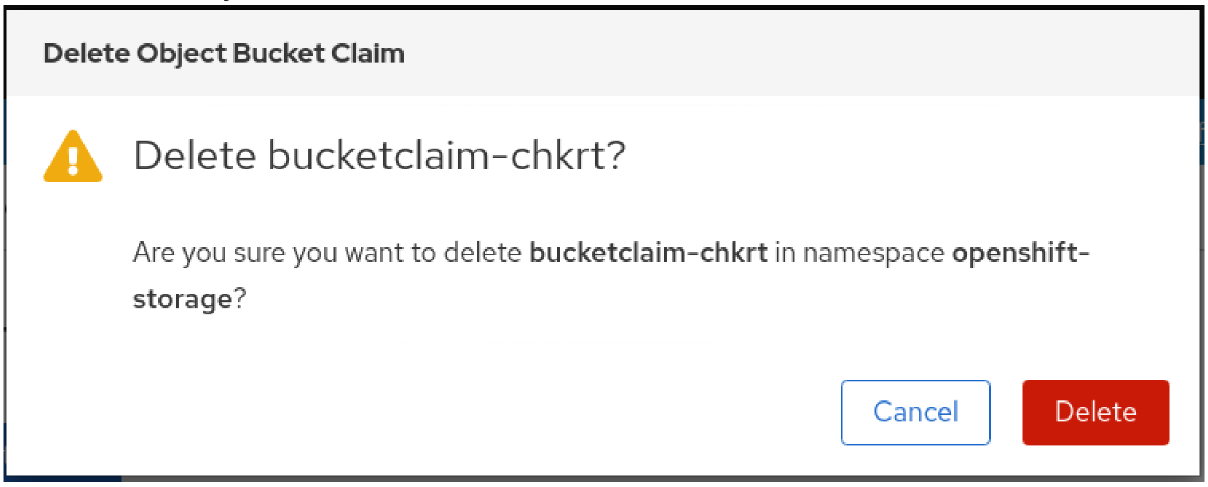
In order for your applications to communicate with the OBC, you need to use the configmap and secret. For more information about this, see Section 7.6.1, "Dynamic Object Bucket Claim".

7.6.3.1. Delete an Object Bucket Claim

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


Red Hat OpenShift Container Platform
kube:admin
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Persistent Volumes
Persistent Volume Claims
Storage Classes
Object Buckets
Object Bucket Claims
Builds
Project: openshift-storage
Object Bucket Claims
Create Object Bucket Claim
Filter by name...
0 Pending 1 Bound 0 Lost Select All Filters 1 Item
Name Namespace Status Secret Storage Class
OBC bucketclaim-chkrt NS openshift-storage Bound S bucketclaim-chkrt SC openshift-storage.noobaa.io
Attach to Deployment
Edit Labels
Edit Annotations
Edit Object Bucket Claim
Delete Object Bucket Claim

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



Delete Object Bucket Claim

Delete bucketclaim-chkrt?

Are you sure you want to delete **bucketclaim-chkrt** in namespace **openshift-storage**?

3. Click **Delete**.

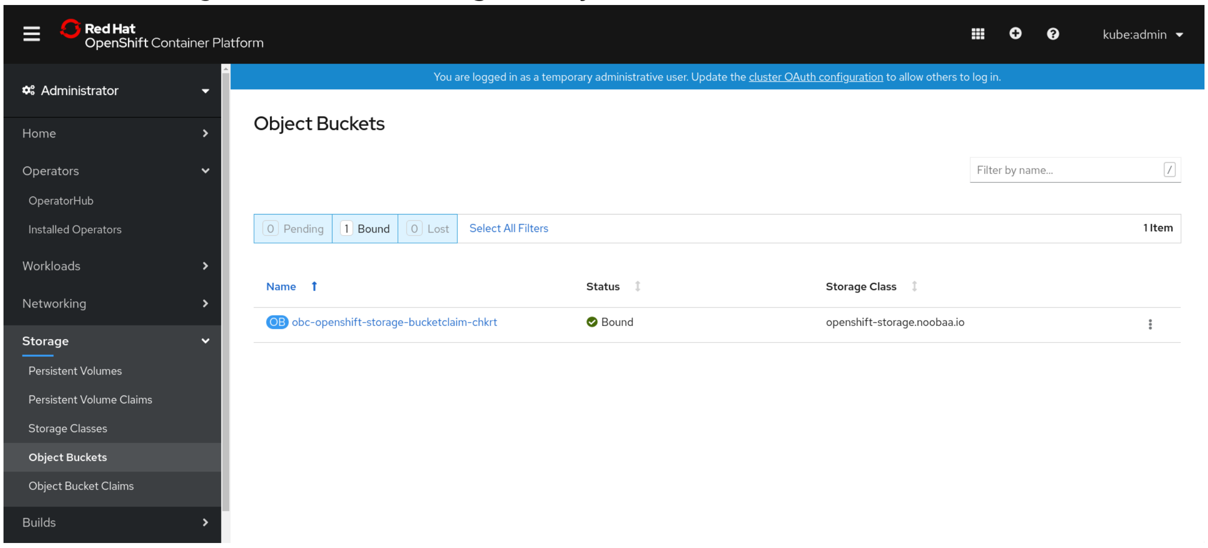
7.6.3.2. Viewing object buckets using the Multicloud Object Gateway user interface

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

Procedure

To view the object bucket details:

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



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

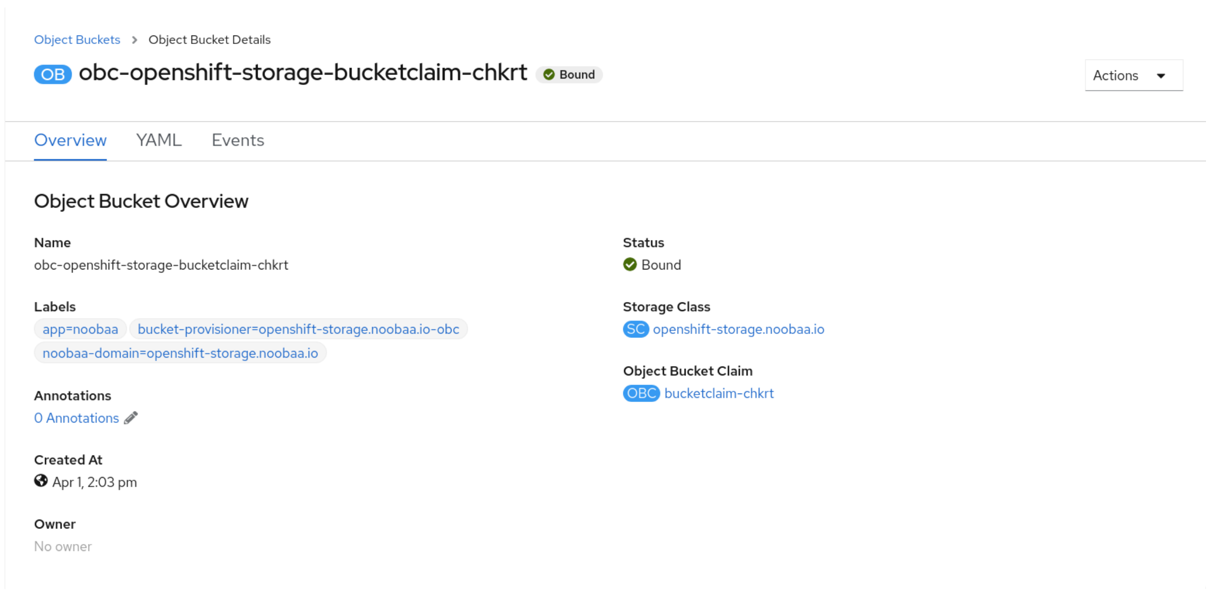
Object Buckets

0 Pending | 1 Bound | 0 Lost | Select All Filters | 1 Item

Name	Status	Storage Class
OB obc-openshift-storage-bucketclaim-chkrt	Bound	openshift-storage.noobaa.io

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

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


Object Buckets > Object Bucket Details
OB obc-openshift-storage-bucketclaim-chkrt Bound
Actions
Overview
YAML
Events

Object Bucket Overview

Name
obc-openshift-storage-bucketclaim-chkrt

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

Annotations
0 Annotations

Created At
Apr 1, 2:03 pm

Owner
No owner

Status
Bound

Storage Class
SC openshift-storage.noobaa.io

Object Bucket Claim
OBC bucketclaim-chkrt

7.7. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS [TECHNOLOGY PREVIEW]

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 [Technology Preview]

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 [Technology Preview]

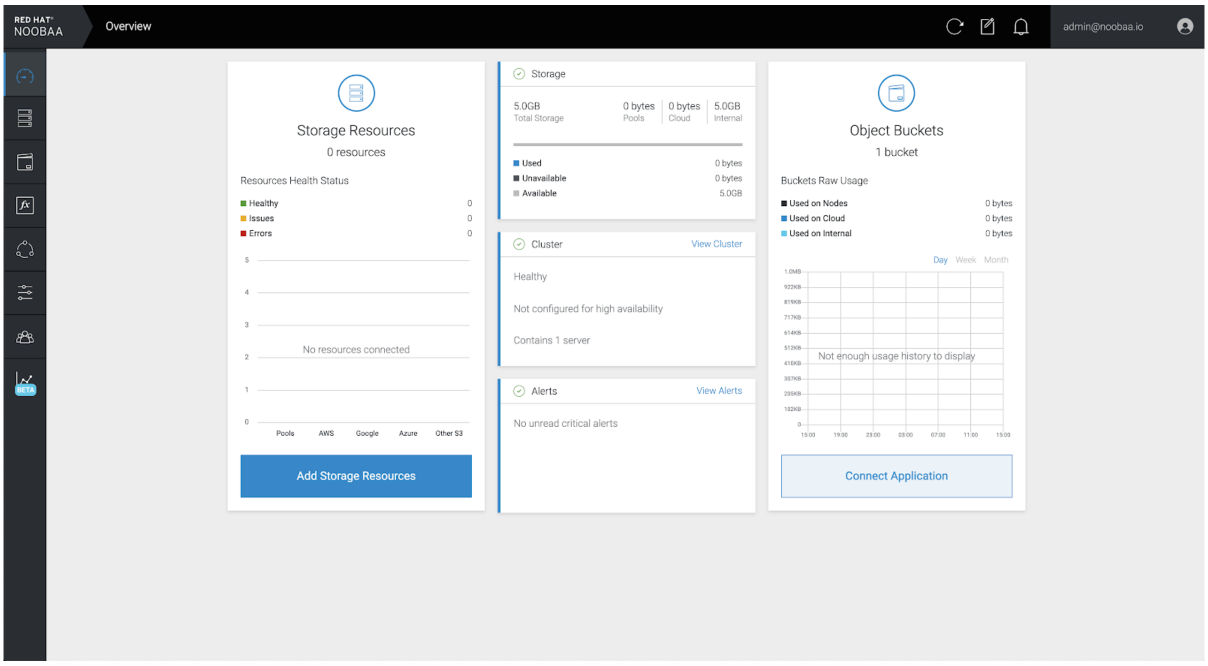
Prerequisites

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

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

Procedure

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



Storage Resources
0 resources

Resources Health Status

🟩 Healthy	0
🟨 Issues	0
🟥 Errors	0

No resources connected

Pools AWS Google Azure Other S3

Storage

5.0GB	0 bytes	0 bytes	5.0GB
Total Storage	Pools	Cloud	Internal

🟦 Used	0 bytes
⬛ Unavailable	0 bytes
⬜ Available	5.0GB

Cluster

Healthy

Not configured for high availability

Contains 1 server

Alerts

No unread critical alerts

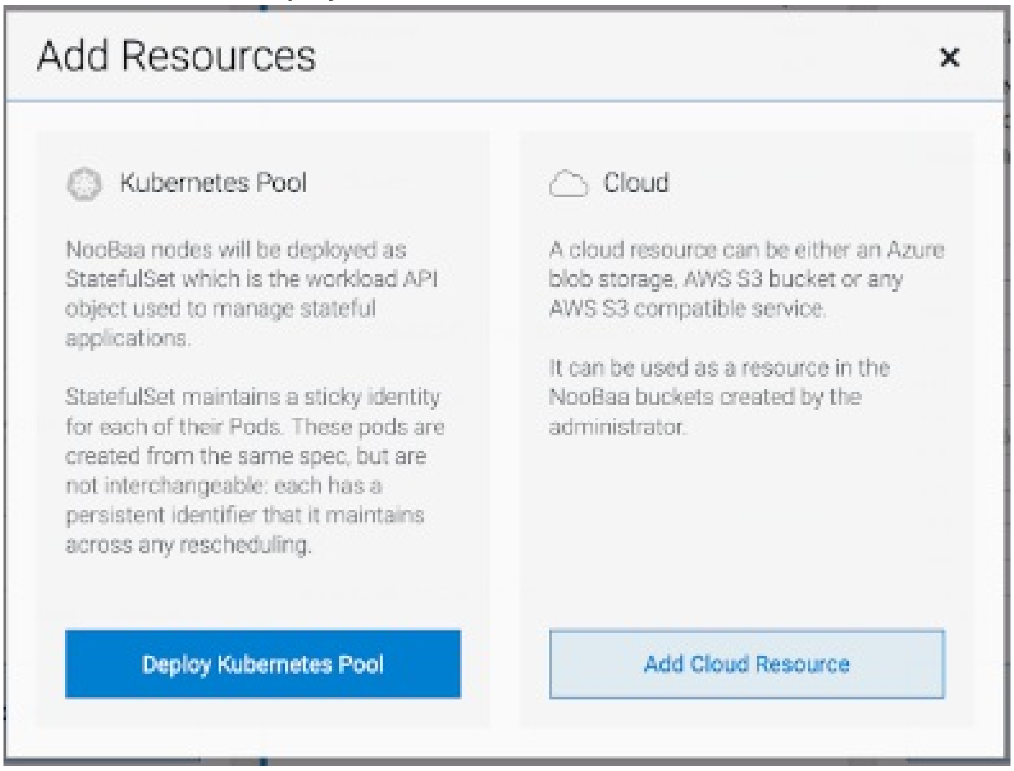
Object Buckets
1 bucket

Buckets Raw Usage

⬛ Used on Nodes	0 bytes
🟦 Used on Cloud	0 bytes
🟦 Used on Internal	0 bytes

Not enough usage history to display

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.

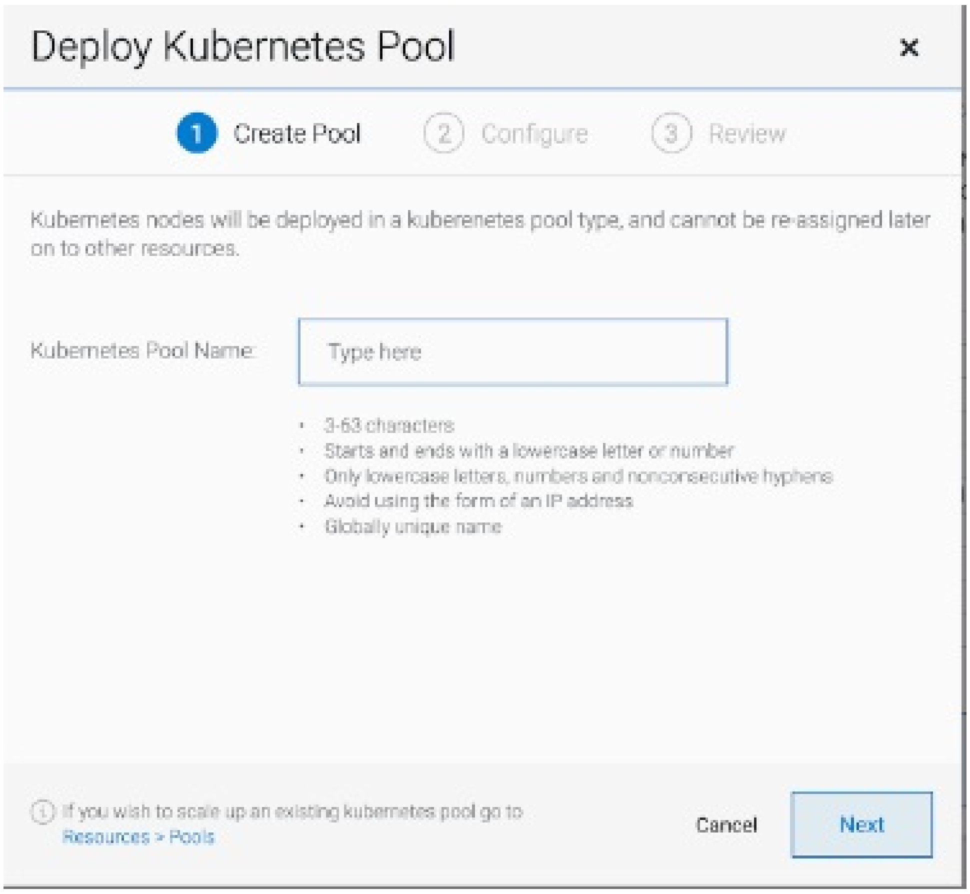
StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.

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

1 Create Pool 2 Configure 3 Review

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

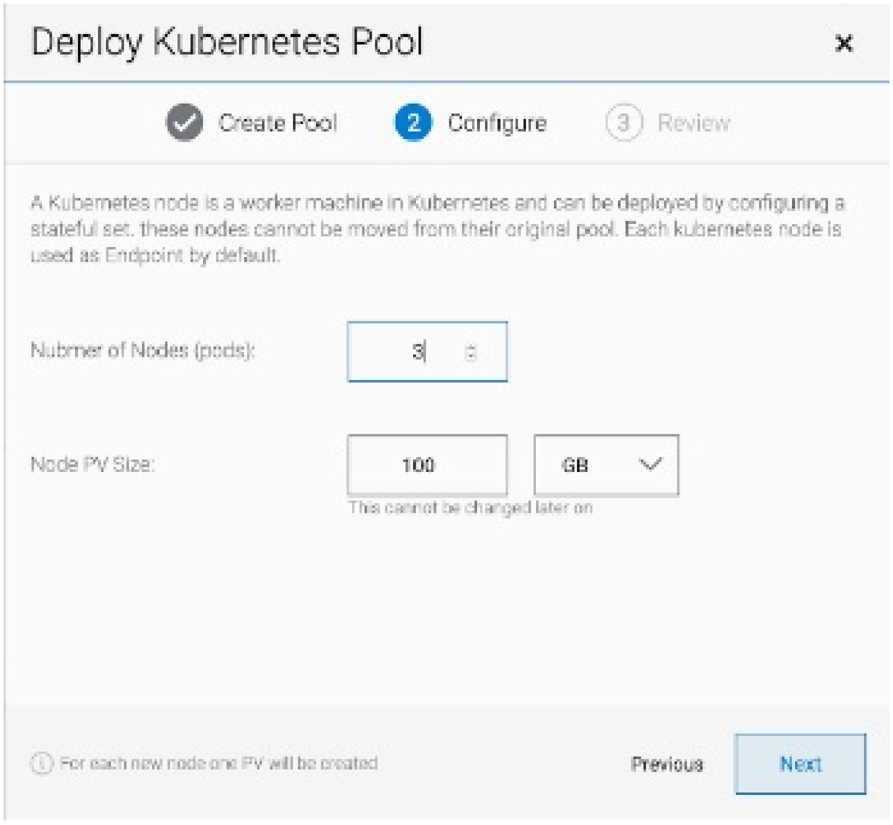
Kubernetes Pool Name: Type here

- 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

Cancel Next

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.



Deploy Kubernetes Pool

Create Pool 2 Configure 3 Review

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

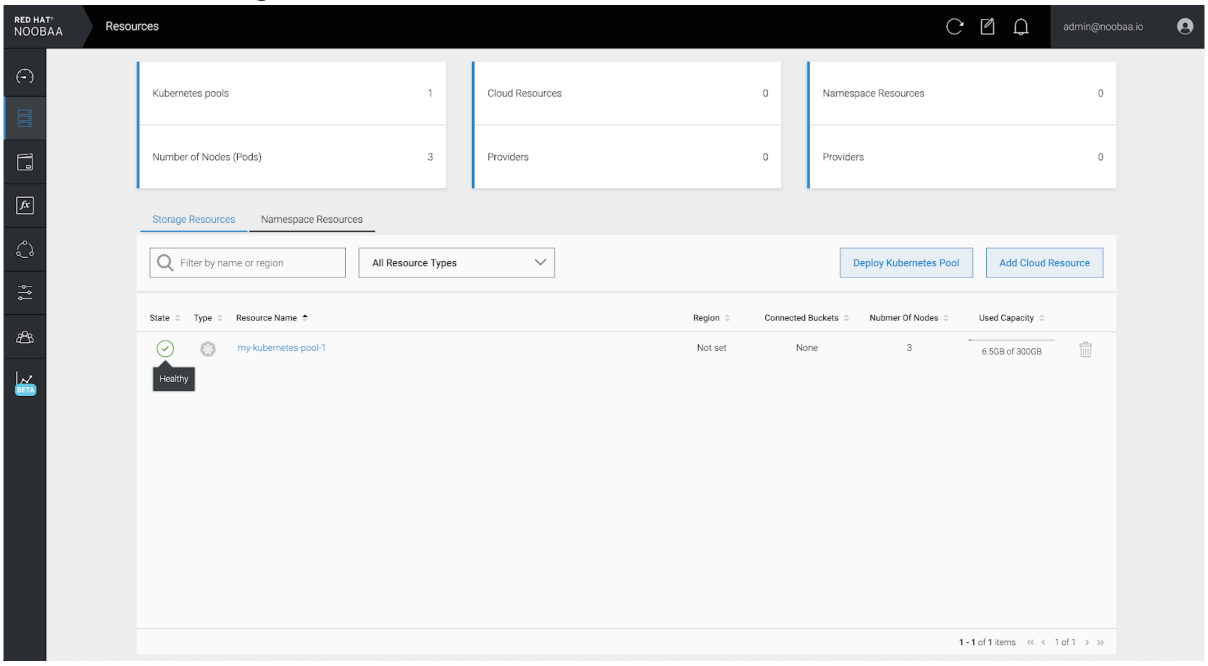
This cannot be changed later on

For each new node one PV will be created

Previous Next

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

7.7.3. Stretching buckets across two different clusters [Technology Preview]

Prerequisites

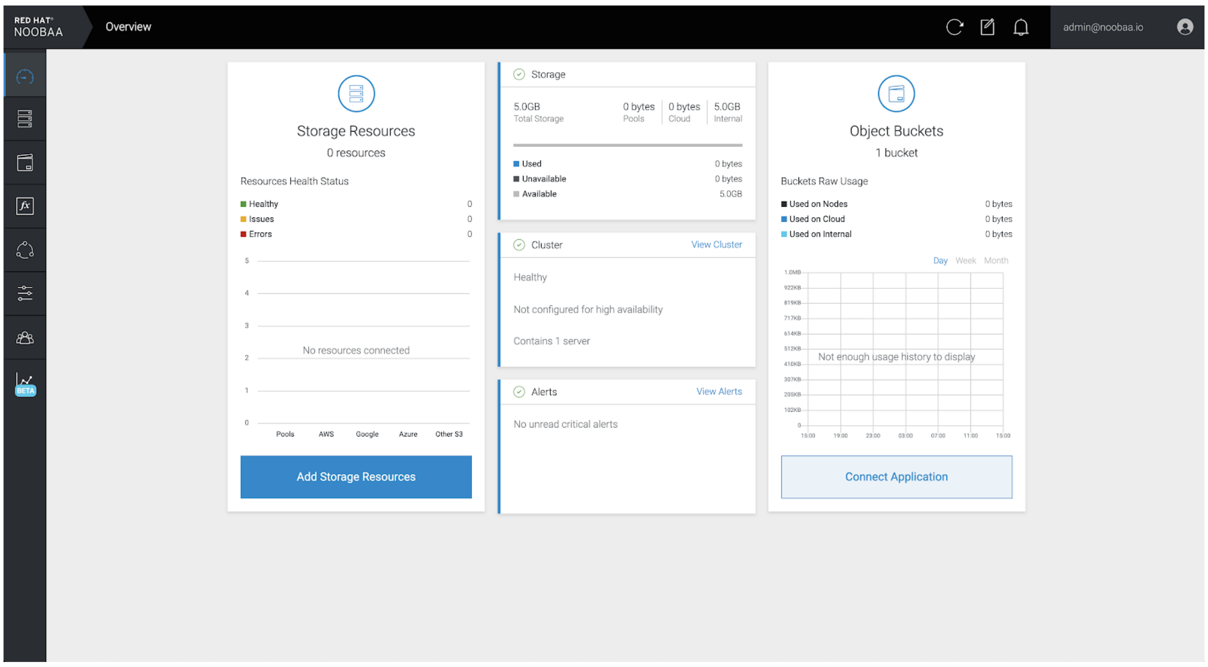
- A running OpenShift Container Storage Platform with access to the Multicloud Object Gateway
- An additional running OpenShift Container Platform that has network access to, and can communicate with the Multicloud Object Gateway on the other cluster

The following instructions will guide you to stretching a bucket across two different clusters: Cluster A and Cluster B. Stretching a bucket across two different clusters involves the following:

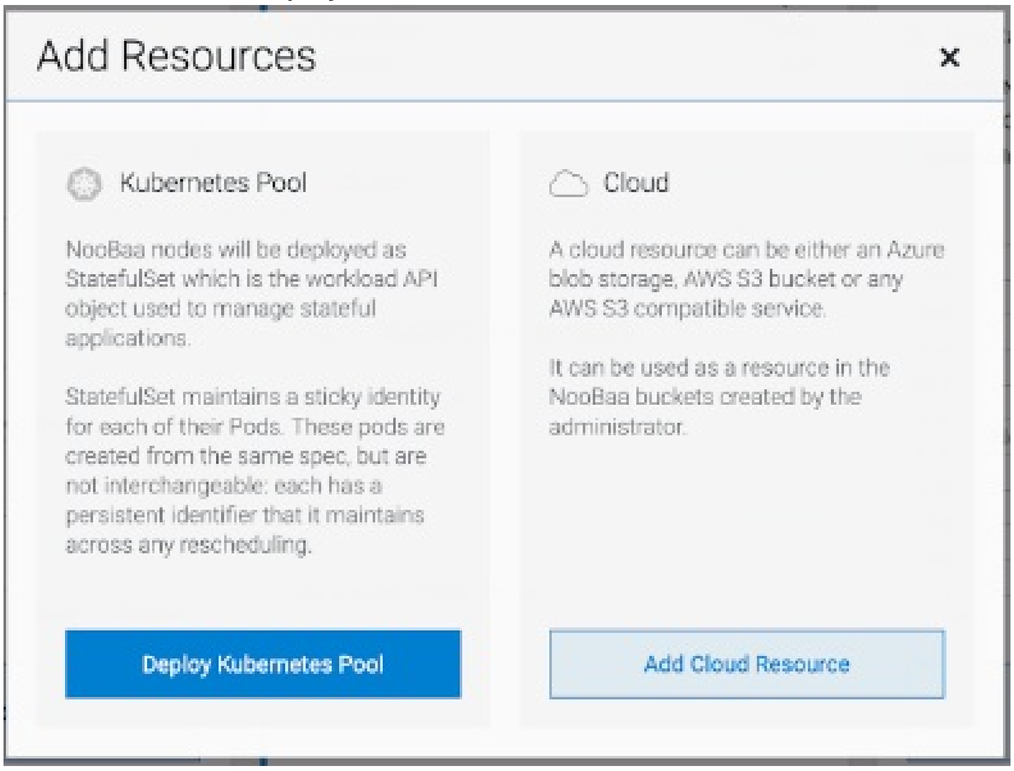
1. Section 7.7.3.1, “Create an external NooBaa Daemon pool on Cluster B”
2. Section 7.7.3.2, “Adding Cluster B as a backing store to Cluster A”
3. Section 7.7.3.3, “Create a federated bucket on Cluster A using Cluster B’s NooBaa system”
4. Section 7.7.3.4, “Assign regions for the locality optimization”
5. Section 7.7.3.5, “Managing the clusters”

7.7.3.1. Create an external NooBaa Daemon pool on Cluster B

1. In the Mult-Cloud Object Gateway user interface for cluster A, 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.

Deploy Kubernetes Pool

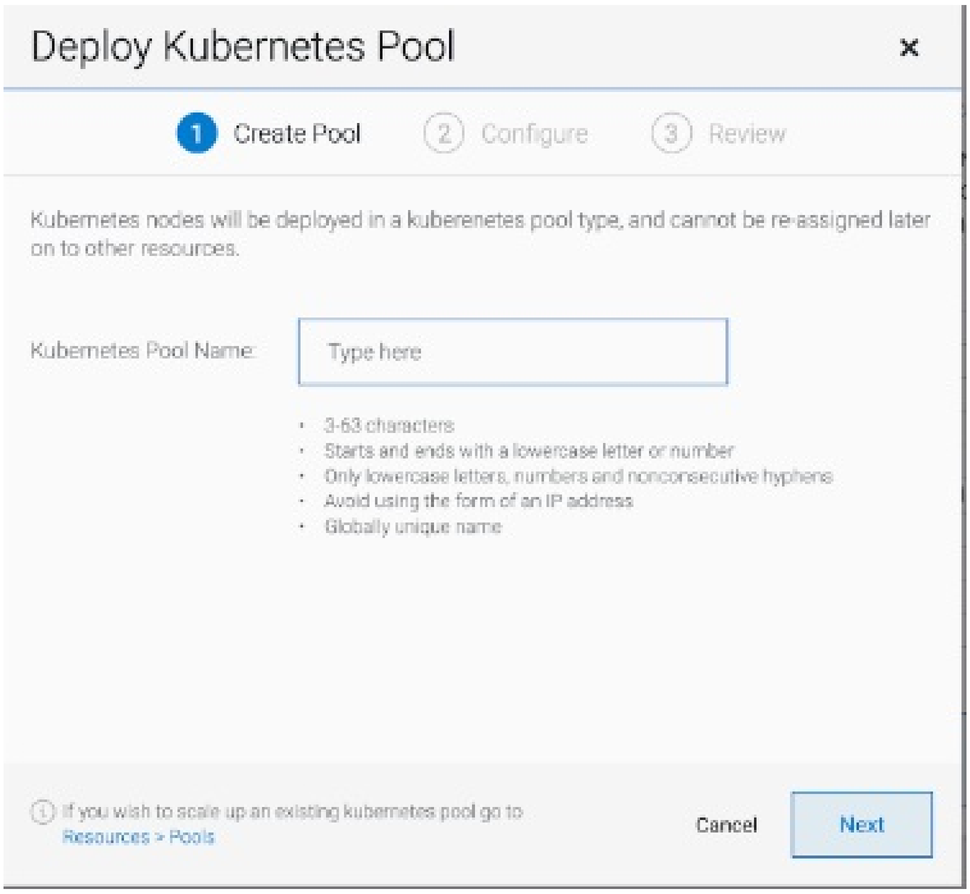
Cloud

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.

Add Cloud Resource

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



Deploy Kubernetes Pool

1 Create Pool 2 Configure 3 Review

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

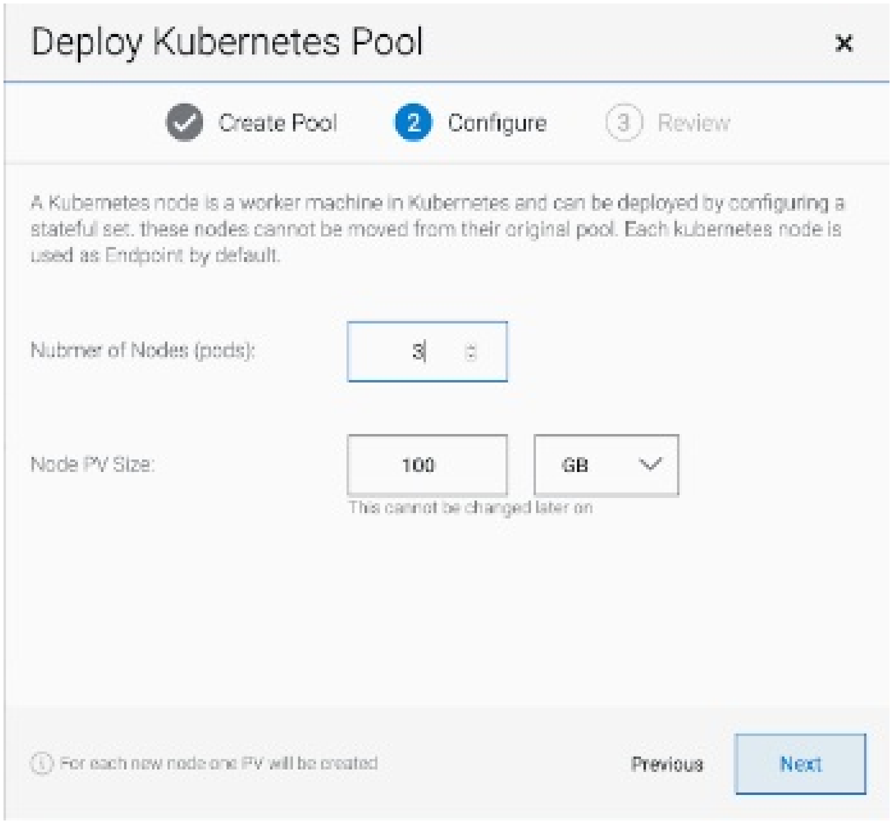
Kubernetes Pool Name: Type here

- 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

Cancel Next

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.



Deploy Kubernetes Pool

Create Pool 2 Configure 3 Review

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

Previous Next

5. In the **Review** step, select **Deploy external to the Kubernetes cluster**and click on **Download YAML**.

6. Connect to Cluster B and run the following command to deploy the NooBaa daemon pods on Cluster A.

```
oc create namespace external
oc apply -f <YAML file> -n external
```

Replace **<YAML file>** with the YAML file from the previous step.

7.7.3.2. Adding Cluster B as a backing store to Cluster A

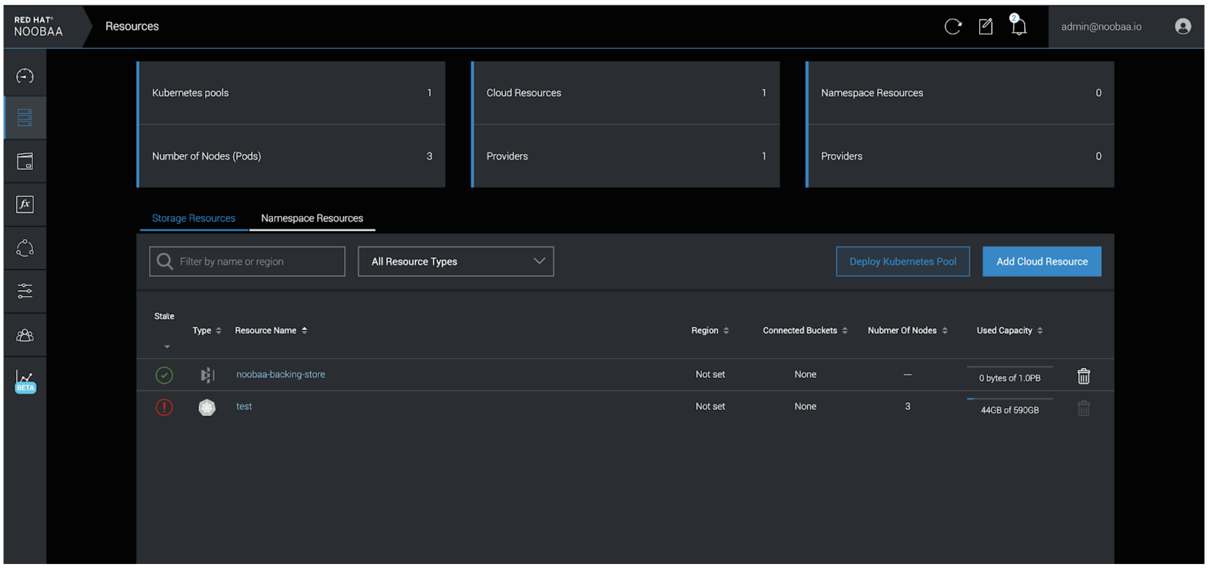
1. From Cluster B, run the following command:

```
noobaa status
```

2. Scroll down and look for the S3 information: **AWS_ACCESS_KEY_ID** and **AWS_SECRET_ACCESS_KEY**. Keep this information for the next step

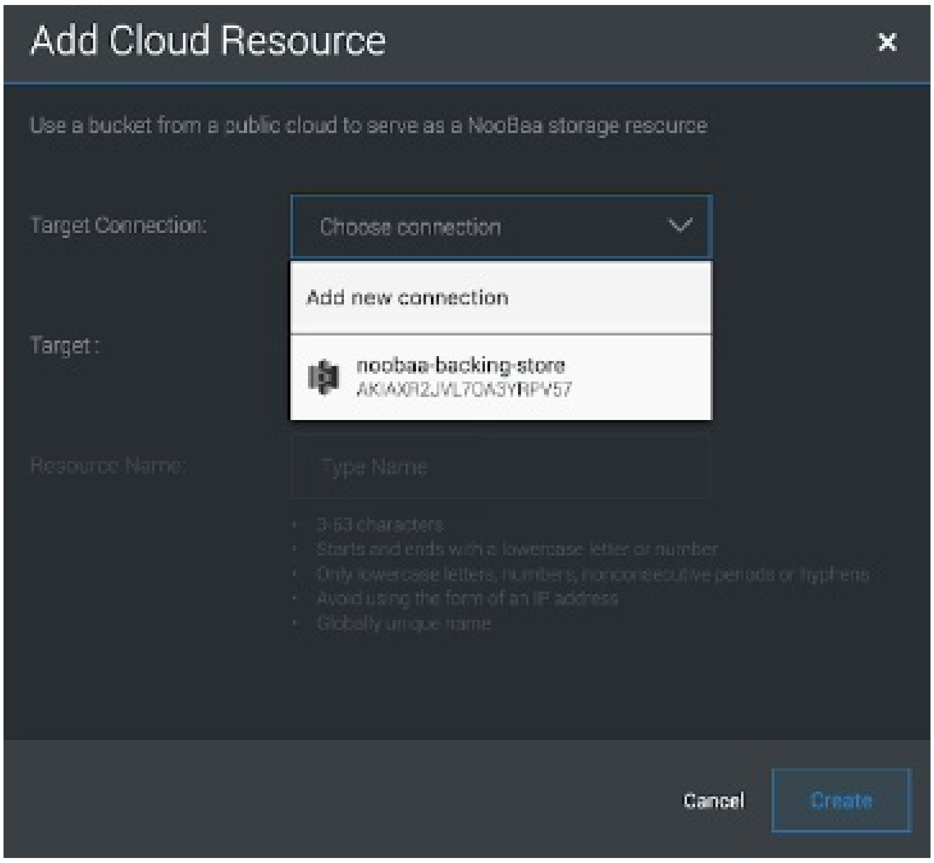
```
INFO[0000] #----------------#
INFO[0000] #- S3 Addresses -#
INFO[0000] #----------------#
INFO[0000]
INFO[0000] ExternalDNS : [s3-noobaa.apps.aa.example.open.com]
INFO[0000] ExternalIP  : [https://172.29.225.78:443 https://172.29.225.78:443]
INFO[0000] NodePorts   : [https://192.168.0.11:32530]
INFO[0000] InternalDNS : [https://s3.noobaa:443]
INFO[0000] InternalIP  : [https://172.30.147.221:443]
INFO[0000] PodPorts    : [https://10.1.4.21:6443]
INFO[0000]
INFO[0000] #------------------#
INFO[0000] #- S3 Credentials -#
INFO[0000] #------------------#
INFO[0000]
INFO[0000] AWS_ACCESS_KEY_ID: aTKel9GdBEGs2kBmJszM
INFO[0000] AWS_SECRET_ACCESS_KEY:
3JEzHc7pCZVOAAnTTTY5bZIG12T1I2a+lHkWlhQ8
INFO[0000]
```

3. On Cluster A, navigate to the **Resources** screen and click on **Add Cloud Resource**



State	Type	Resource Name	Region	Connected Buckets	Nubmer Of Nodes	Used Capacity
✓		noobaa-backing-store	Not set	None	---	0 bytes of 1.0PB
!		test	Not set	None	3	44GB of 590GB

4. Select **Add new connection**:



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

Target Connection: Choose connection

Add new connection

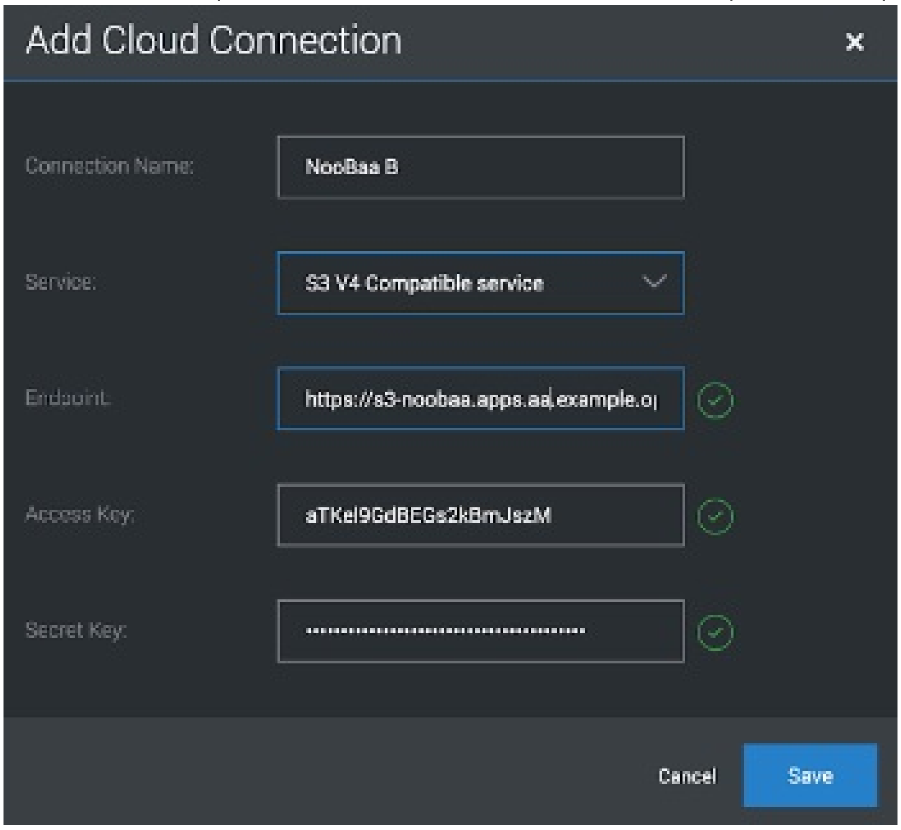
noobaa-backing-store
AKIAXR2JVL7OA3YRPV57

Target :

Resource Name: Type Name

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

5. Enter the S3 endpoint information for Cluster B from The previous step:



Connection Name: NooBaa B

Service: S3 V4 Compatible service

Endpoint: https://s3-noobaa.apps.aa.example.o

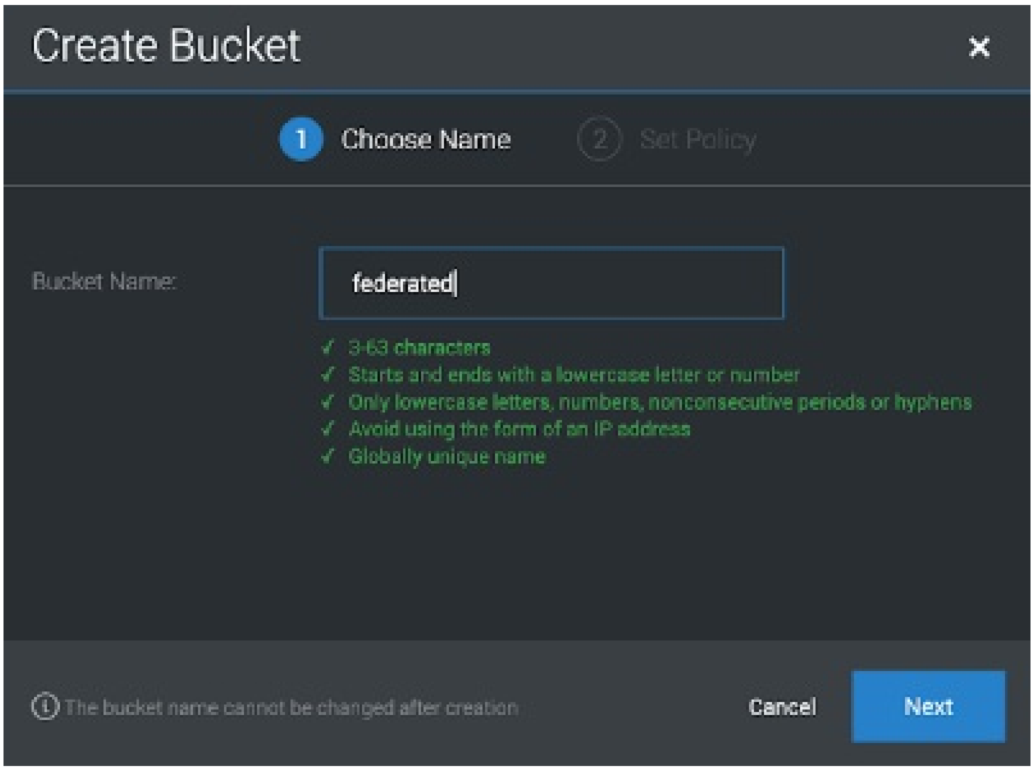
Access Key: aTKel9GdBEGs2kBmJszM

Secret Key: ••

6. Follow the steps in the window to finish creating the cloud resource.

7.7.3.3. Create a federated bucket on Cluster A using Cluster B's NooBaa system

1. Navigate to **Buckets** and click **Create Bucket**, then set the bucket name:

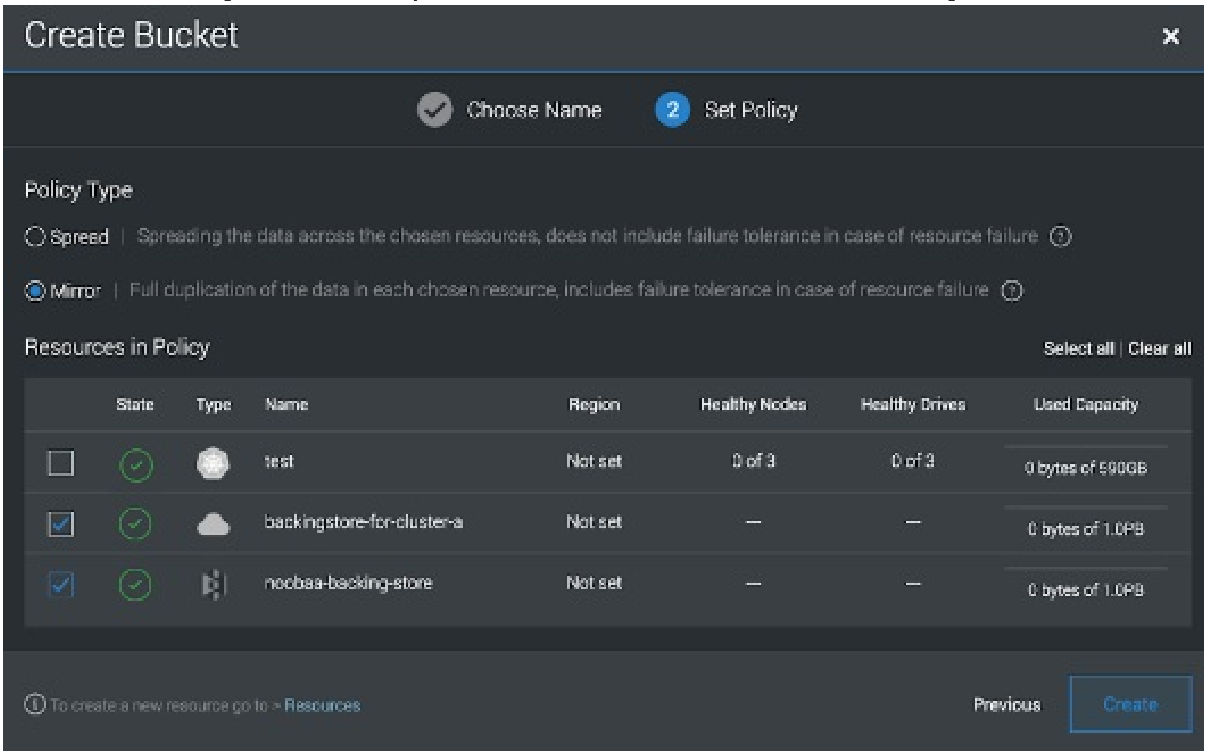


Bucket Name: federated

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

The bucket name cannot be changed after creation

2. Select the backing store that we just created with a mirror to the local storage of Cluster A:



Policy Type

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

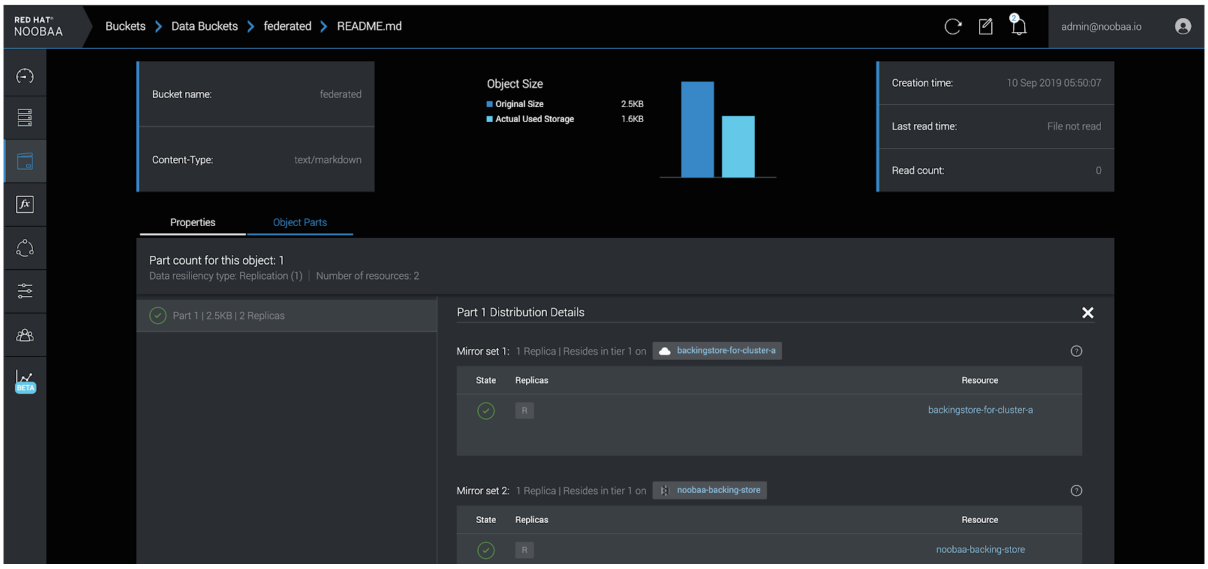
[✓ Mirror] | Full duplication of the data in each chosen resource, includes failure tolerance in case of resource failure

Resources in Policy

	State	Type	Name	Region	Healthy Nodes	Healthy Drives	Used Capacity
☐	✓		test	Not set	0 of 3	0 of 3	0 bytes of 590GB
☑	✓		backingstore-for-cluster-a	Not set	—	—	0 bytes of 1.0PB
☑	✓		noobaa-backing-store	Not set	—	—	0 bytes of 1.0PB

To create a new resource go to > Resources

3. Now any object uploaded to the federated bucket will be mirrored between Cluster A and Cluster B:



Buckets > Data Buckets > federated > README.md

Bucket name:	federated
Content-Type:	text/markdown

Object Size

Original Size	2.5KB
Actual Used Storage	1.6KB

Creation time:	10 Sep 2019 05:50:07
Last read time:	File not read
Read count:	0

Properties | Object Parts

Part count for this object: 1
Data resiliency type: Replication (1) | Number of resources: 2

Part 1 | 2.5KB | 2 Replicas

Part 1 Distribution Details

Mirror set 1: 1 Replica | Resides in tier 1 on backingstore-for-cluster-a

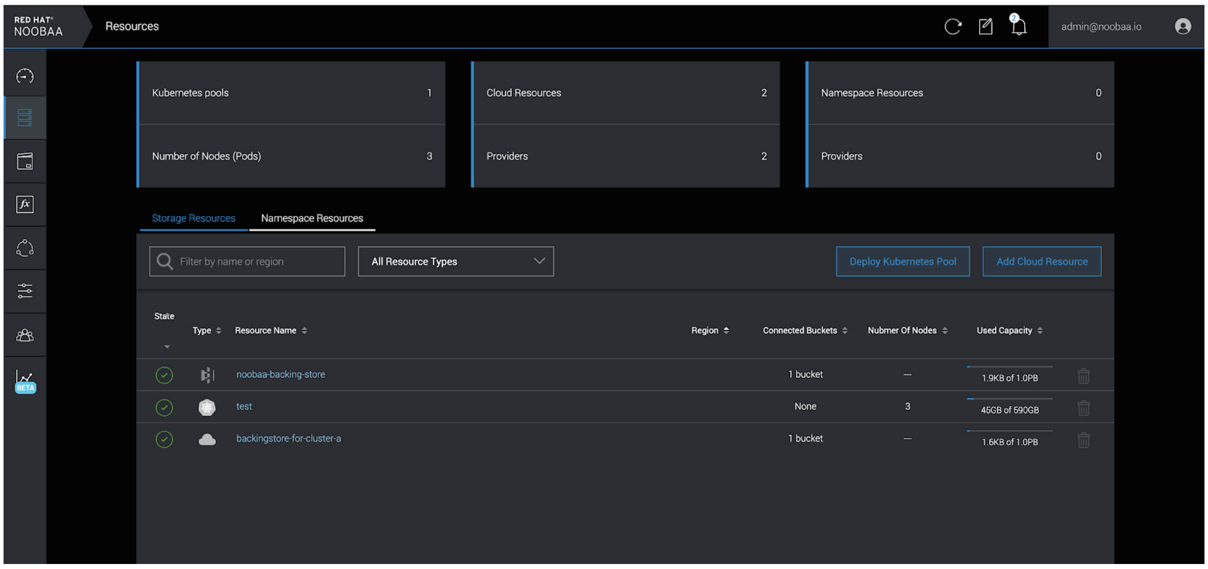
State	Replicas	Resource
✓	R	backingstore-for-cluster-a

Mirror set 2: 1 Replica | Resides in tier 1 on noobaa-backing-store

State	Replicas	Resource
✓	R	noobaa-backing-store

7.7.3.4. Assign regions for the locality optimization

1. On Cluster A, navigate to **Resources**:



Resources

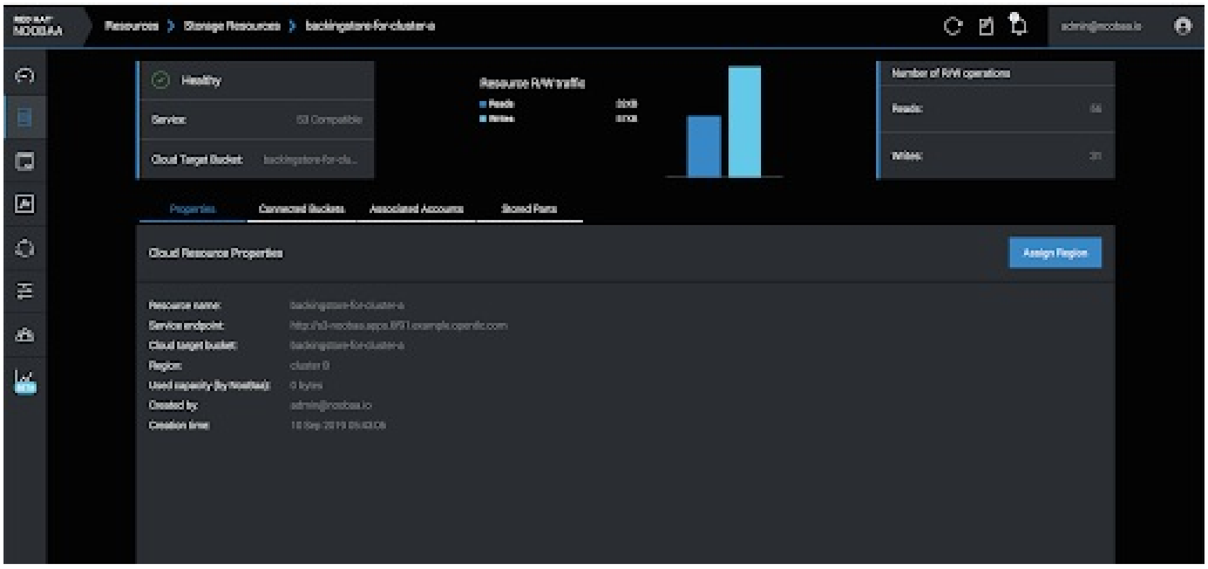
Kubernetes pools	1
Number of Nodes (Pods)	3
Cloud Resources	2
Providers	2
Namespace Resources	0
Providers	0

Storage Resources | Namespace Resources

Filter by name or region | All Resource Types | Deploy Kubernetes Pool | Add Cloud Resource

State	Type	Resource Name	Region	Connected Buckets	Nubmer Of Nodes	Used Capacity
✓		noobaa-backing-store		1 bucket	—	1.9KB of 1.0PB
✓		test		None	3	45GB of 590GB
✓		backingstore-for-cluster-a		1 bucket	—	1.6KB of 1.0PB

2. Click on each resource and assign a region:



Resources > Storage Resources > backingstore-for-cluster-a

Healthy

Service:	S3 Compatible
Cloud Target Bucket:	backingstore-for-clu...

Resource R/W traffic

Reads	[illegible]
Writes	[illegible]

Number of R/W operations

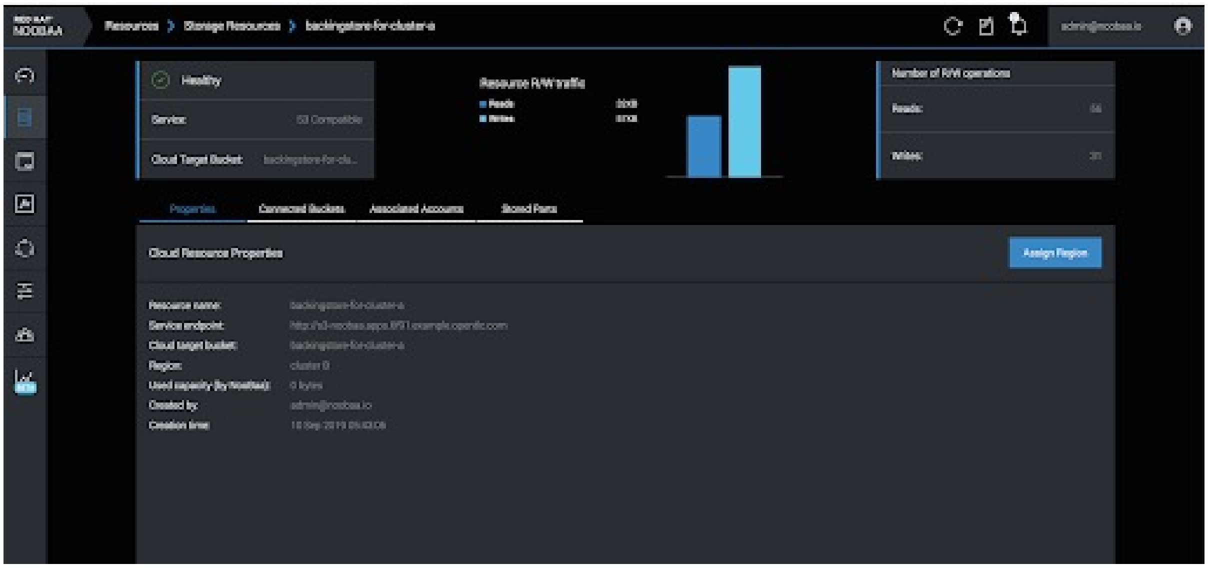
Reads:	[illegible]
Writes:	[illegible]

Properties | Connected Buckets | Associated Accounts | Stored Parts

Cloud Resource Properties | Assign Region

Resource name:	backingstore-for-cluster-a
Service endpoint:	http://s3-noobaa.apps.ff91.example.openfic.com
Cloud target bucket:	backingstore-for-cluster-a
Region:	cluster B
Used capacity (by Noobaa):	0 bytes
Created by:	admin@noobaa.io
Creation time:	10 Sep 2019 05:43:06

3. Once completed, it should look like the following image. The external endpoints and the NooBaa system on Cluster B are in one region, while the local backing store is in another region:



Service: S3 Compatible
Cloud Target Bucket: backingstore-for-clu...
Reads:
Writes:

Cloud Resource Properties

Resource name: backingstore-for-cluster-a
Service endpoint: http://s3-noobaa.apps.8f91.example.opentlc.com
Cloud target bucket: backingstore-for-cluster-a
Region: cluster B
Used capacity (by NooBaa): 0 bytes
Created by: admin@noobaa.io
Creation time: 18 Sep 2019 05:43:06

7.7.3.5. Managing the clusters

Applications on Cluster A will use the federated bucket by using the local S3 endpoint:

```
noobaa status
```

```
INFO[0000] #----------------#
INFO[0000] #- S3 Addresses -#
INFO[0000] #----------------#
INFO[0000]
INFO[0000] ExternalDNS : [s3-noobaa.apps.aa.example.open.com]
INFO[0000] ExternalIP  : [https://172.29.225.78:443 https://172.29.225.78:443]
INFO[0000] NodePorts   : [https://192.168.0.11:32530]
INFO[0000] InternalDNS : [https://s3.noobaa:443]
INFO[0000] InternalIP  : [https://172.30.147.221:443]
INFO[0000] PodPorts    : [https://10.1.4.21:6443]
INFO[0000]
INFO[0000] #------------------#
INFO[0000] #- S3 Credentials -#
INFO[0000] #------------------#
INFO[0000]
INFO[0000] AWS_ACCESS_KEY_ID: aTKel9GdBEGs2kBmJszM
INFO[0000] AWS_SECRET_ACCESS_KEY: 3JEzHc7pCZVOAAnTTTY5bZIG12T1I2a+lHkWlhQ8
INFO[0000]
```

Applications on cluster B, should use the same credentials, but the endpoint URL should be local S3 service

```
oc get service -n external
```

```
NAME     TYPE          CLUSTER-IP      EXTERNAL-IP                              PORT(S)
AGE
s3       LoadBalancer   172.30.18.172   s3-external.apps.aa.example.opentlc.com
80:32741/TCP,443:30949/TCP   4m
```

CHAPTER 8. REPLACING STORAGE NODES FOR OPENSHIFT CONTAINER STORAGE

For OpenShift Container Storage, node replacement can be performed proactively for an operational node and reactively for a failed node for the following deployments:

- For Amazon Web Services (AWS)
  - User-provisioned infrastructure
  - Installer-provisioned infrastructure
- For VMware
  - User-provisioned infrastructure

8.1. OPENSHIFT CONTAINER STORAGE DEPLOYED ON AWS

8.1.1. Replacing an operational AWS node on user-provisioned infrastructure

Perform this procedure to replace an operational node on AWS user-provisioned infrastructure.

Procedure

1. Identify the node that needs to be replaced.
2. Mark the node as unschedulable using the following command:

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

3. Drain the node using the following command:

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

   IMPORTANT

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

4. Delete the node using the following command:

   ```
   $ oc delete nodes <node_name>
   ```

5. Create a new AWS machine instance with the required infrastructure. See Supported Infrastructure and Platforms.
6. Create a new OpenShift Container Platform node using the new AWS machine instance.
7. Check for certificate signing requests (CSRs) related to OpenShift Container Platform that are in **Pending** state:

```
$ oc get csr
```

8. Approve all required OpenShift Container Platform CSRs for the new node:

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

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

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

    From User interface

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

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

    From Command line interface

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

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

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

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

Verification steps

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

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

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

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

3. Verify that all other required OpenShift Container Storage pods are in **Running** state.

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

8.1.2. Replacing an operational AWS node on installer-provisioned infrastructure

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

Procedure

1. Log in to OpenShift Web Console and click **Compute → Nodes**.

2. Identify the node that needs to be replaced. Take a note of its **Machine Name**.

3. Mark the node as unschedulable using the following command:

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

4. Drain the node using the following command:

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

IMPORTANT

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

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

IMPORTANT

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

9. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
10. Apply the OpenShift Container Storage label to the new node using any one of the following:

    From User interface

    a. For the new node, click **Action Menu (⋮) → Edit Labels**
    b. Add **cluster.ocs.openshift.io/openshift-storage** and click **Save**.

    From Command line interface

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

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

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

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

Verification steps

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

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

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:
    - **csi-cephfsplugin-***
    - **csi-rbdplugin-***
3. Verify that all other required OpenShift Container Storage pods are in **Running** state.
4. If verification steps fail, kindly contact Red Hat Support .

8.1.3. Replacing a failed AWS node on user-provisioned infrastructure

Perform this procedure to replace a failed node which is not operational on AWS user-provisioned infrastructure (UPI) for OpenShift Container Storage.

Procedure

1. Identify the AWS machine instance of the node that needs to be replaced.
2. Log in to AWS and terminate the identified AWS machine instance.
3. Create a new AWS machine instance with the required infrastructure. See Supported Infrastructure and Platforms.
4. Create a new OpenShift Container Platform node using the new AWS machine instance.
5. Check for certificate signing requests (CSRs) related to OpenShift Container Platform that are in **Pending** state:

    ```
    $ oc get csr
    ```

6. Approve all required OpenShift Container Platform CSRs for the new node:

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

7. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
8. Apply the OpenShift Container Storage label to the new node using any one of the following:

    From User interface

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

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

    From Command line interface

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

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

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

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

Verification steps

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

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

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:
   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***
3. Verify that all other required OpenShift Container Storage pods are in **Running** state.
4. If verification steps fail, kindly contact Red Hat Support .

8.1.4. Replacing a failed AWS node on installer-provisioned infrastructure

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

Procedure

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

   IMPORTANT

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

7. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
8. Apply the OpenShift Container Storage label to the new node using any one of the following:

   From User interface

   a. For the new node, click **Action Menu (⋮) → Edit Labels**
   b. Add **cluster.ocs.openshift.io/openshift-storage** and click **Save**.

   From Command line interface

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

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

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

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

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

Verification steps

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

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

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

3. Verify that all other required OpenShift Container Storage pods are in **Running** state.

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

8.2. OPENSHIFT CONTAINER STORAGE DEPLOYED ON VMWARE

8.2.1. Replacing an operational VMware node on user-provisioned infrastructure

Perform this procedure to replace an operational node on VMware user-provisioned infrastructure (UPI).

Procedure

1. Identify the node and its VM that needs to be replaced.

2. Mark the node as unschedulable using the following command:

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

3. Drain the node using the following command:

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

IMPORTANT

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

4. Delete the node using the following command:

```
$ oc delete nodes <node_name>
```

5. Log in to vSphere and terminate the identified VM.

IMPORTANT

VM should be deleted only from the inventory and not from the disk.

6. Create a new VM on vSphere with the required infrastructure. See Supported Infrastructure and Platforms.

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

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

```
$ oc get csr
```

9. Approve all required OpenShift Container Platform CSRs for the new node:

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

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

11. 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=""
```

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

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

Verification steps

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

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

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

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

3. Verify that all other required OpenShift Container Storage pods are in **Running** state.

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

8.2.2. Replacing a failed VMware node on user-provisioned infrastructure

Perform this procedure to replace a failed node on VMware user-provisioned infrastructure (UPI).

Procedure

1. Identify the node and its VM that needs to be replaced.

2. Delete the node using the following command:

```
$ oc delete nodes <node_name>
```

3. Log in to vSphere and terminate the identified VM.

IMPORTANT

VM should be deleted only from the inventory and not from the disk.

4. Create a new VM on vSphere with the required infrastructure. See Supported Infrastructure and Platforms.

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

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

```
$ oc get csr
```

7. Approve all required OpenShift Container Platform CSRs for the new node:

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

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

9. 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=""
```

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

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

Verification steps

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

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

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:
   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***
3. Verify that all other required OpenShift Container Storage pods are in **Running** state.
4. If verification steps fail, kindly contact Red Hat Support .

CHAPTER 9. UPDATING OPENSHIFT CONTAINER STORAGE

9.1. ENABLING AUTOMATIC UPDATES FOR OPENSHIFT CONTAINER STORAGE OPERATOR

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

Prerequisites

- Update the OpenShift Container Platform cluster to the latest stable release of version 4.3, see Updating Clusters.
- If you were already operating on OpenShift Container Platform version 4.3, and you are updating OpenShift Container Storage from version 4.2 to version 4.3, switch the channel from **stable-4.2** to **stable-4.3**. For details about channels, see OpenShift Container Platform upgrade channels and releases.

  NOTE

  You are required to switch channels only when you are updating minor versions (for example, updating from 4.2 to 4.3) and not when updating between batch updates of 4.3 (for example, updating from 4.3.0 to 4.3.1).

- Ensure that all OpenShift Container Storage nodes are in **Ready** status.
- Under **Persistent Storage** in **Health** card, confirm that the Ceph cluster is healthy and data is resilient.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Operators → Installed Operators**
3. Select the **openshift-storage** project.
4. Click on the OpenShift Container Storage operator name.
5. Click **Subscription** tab and click the link under **Approval**.
6. Select **Automatic (default)** and click **Save**.
7. Perform one of the following depending on the **Upgrade Status**:
   - **Upgrade Status** *shows* **requires approval**.
     a. Click on the **Install Plan** link.
     b. On the **InstallPlan Details** page, click **Preview Install Plan**.
     c. Review the install plan and click **Approve**.
     d. Wait for the **Status** to change from **Unknown** to **Created**.

e. Click **Operators → Installed Operators**

f. Select the **openshift-storage** project.

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

- **Upgrade Status** *does not show* **requires approval**:

  a. Wait for the update to initiate. This may take up to 20 minutes.

  b. Click **Operators → Installed Operators**

  c. Select the **openshift-storage** project.

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

Verification steps

1. Click **Dashboard → OCS PV** tab and in **Status** card confirm that the Ceph cluster has a green tick mark indicating it is healthy.

2. Click **Operators → Installed Operators → OpenShift Container Storage Operator**.

3. Under **Storage Cluster**, verify that the cluster service status in **Ready**.

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

9.2. MANUALLY UPDATING OPENSHIFT CONTAINER STORAGE OPERATOR

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

Prerequisites

- Update the OpenShift Container Platform cluster to the latest stable release of version 4.3, see Updating Clusters.

- If you were already operating on OpenShift Container Platform version 4.3, and you are updating OpenShift Container Storage from version 4.2 to version 4.3, switch the channel from **stable-4.2** to **stable-4.3**. For details about channels, see OpenShift Container Platform upgrade channels and releases.

  NOTE

  You are required to switch channels only when you are updating minor versions (for example, updating from 4.2 to 4.3) and not when updating between batch updates of 4.3 (for example, updating from 4.3.0 to 4.3.1).

- Ensure that all OpenShift Container Storage nodes are in **Ready** status.

- Under **Persistent Storage** in **Health** card, confirm that the Ceph cluster is healthy and data is resilient.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Operators → Installed Operators**
3. Select the **openshift-storage** project.
4. Click **Subscription** tab and click the link under **Approval**.
5. Select **Manual** and click **Save**.
6. Wait for the **Upgrade Status** to change to **Upgrading**.
7. If the **Upgrade Status** shows **requires approval**, click on **requires approval**.
8. On the **InstallPlan Details** page, click **Preview Install Plan**.
9. Review the install plan and click **Approve**.
10. Wait for the **Status** to change from **Unknown** to **Created**.
11. Click **Operators → Installed Operators**
12. Select the **openshift-storage** project.
13. Wait for the **Status** to change to **Up to date**

Verification steps

1. Click **Dashboard → OCS PV** tab and in **Status** card confirm that the Ceph cluster has a green tick mark indicating it is healthy.
2. Click **Operators → Installed Operators → OpenShift Container Storage Operator**.
3. Under **Storage Cluster**, verify that the cluster service status in **Ready**.
4. If verification steps fail, kindly contact Red Hat Support .