Red Hat OpenShift Container Storage 4.8

Deploying and managing OpenShift Container Storage using Google Cloud

How to install and manage
Red Hat OpenShift Container Storage 4.8 Deploying and managing OpenShift Container Storage using Google Cloud

How to install and manage
Abstract

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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preface</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Deploying OpenShift Container Storage on Google Cloud</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Verifying OpenShift Container Storage Deployment</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Uninstalling OpenShift Container Storage</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Storage Classes and Storage Pools</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Configure Storage for OpenShift Container Platform Services</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>Backing OpenShift Container Platform Applications with OpenShift Container Storage</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>How to Use Dedicated Worker Nodes for Red Hat OpenShift Container Storage</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>Scaling Storage Nodes</td>
<td>48</td>
</tr>
</tbody>
</table>

## Preface

### Making Open Source More Inclusive

### Providing Feedback on Red Hat Documentation

### Chapter 1. Preparing to Deploy OpenShift Container Storage

#### 1.1. Enabling Key Value Backend Path and Policy in Vault

### Chapter 2. Deploying OpenShift Container Storage on Google Cloud

#### 2.1. Installing Red Hat OpenShift Container Storage Operator

#### 2.2. Creating an OpenShift Container Storage Cluster Service in Internal Mode

### Chapter 3. Verifying OpenShift Container Storage Deployment

#### 3.1. Verifying the State of the Pods

#### 3.2. Verifying the OpenShift Container Storage Cluster is Healthy

#### 3.3. Verifying the Multicloud Object Gateway is Healthy

#### 3.4. Verifying That the OpenShift Container Storage Specific Storage Classes Exist

### Chapter 4. Uninstalling OpenShift Container Storage

#### 4.1. Uninstalling OpenShift Container Storage in Internal Mode

#### 4.2. Removing Monitoring Stack from OpenShift Container Storage

#### 4.3. Removing OpenShift Container Platform Registry from OpenShift Container Storage

#### 4.4. Removing the Cluster Logging Operator from OpenShift Container Storage

### Chapter 5. Storage Classes and Storage Pools

#### 5.1. Creating Storage Classes and Pools

#### 5.2. Creating a Storage Class for Persistent Volume Encryption

### Chapter 6. Configure Storage for OpenShift Container Platform Services

#### 6.1. Configuring Image Registry to Use OpenShift Container Storage

#### 6.2. Configuring Monitoring to Use OpenShift Container Storage

#### 6.3. Cluster Logging for OpenShift Container Storage

#### 6.3.1. Configuring persistent storage

#### 6.3.2. Configuring cluster logging to use OpenShift Container Storage

### Chapter 7. Backing OpenShift Container Platform Applications with OpenShift Container Storage

### Chapter 8. How to Use Dedicated Worker Nodes for Red Hat OpenShift Container Storage

#### 8.1. Anatomy of an Infrastructure Node

#### 8.2. Machine Sets for Creating Infrastructure Nodes

#### 8.3. Manual Creation of Infrastructure Nodes

### Chapter 9. Scaling Storage Nodes

#### 9.1. Requirements for Scaling Storage Nodes

#### 9.2. Scaling Up Storage by Adding Capacity to Your OpenShift Container Storage Nodes on Google Cloud Infrastructure

#### 9.3. Scaling Out Storage Capacity by Adding New Nodes

#### 9.3.1. Adding a node on Google Cloud installer-provisioned infrastructure

#### 9.3.2. Verifying the addition of a new node

#### 9.3.3. Scaling up storage capacity
CHAPTER 10. MULTICLOUD OBJECT GATEWAY

10.1. ABOUT THE MULTICLOUD OBJECT GATEWAY

10.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS

10.2.1. Accessing the Multicloud Object Gateway from the terminal

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

10.3. ALLOWING USER ACCESS TO THE MULTICLOUD OBJECT GATEWAY CONSOLE

10.4. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD

10.4.1. Creating a new backing store

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

10.4.2.1. Creating an AWS-backed backingstore

10.4.2.2. Creating an IBM COS-backed backingstore

10.4.2.3. Creating an Azure-backed backingstore

10.4.2.4. Creating a GCP-backed backingstore

10.4.2.5. Creating a local Persistent Volume-backed backingstore

10.4.3. Creating an s3 compatible Multicloud Object Gateway backingstore

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

10.4.5. Creating a new bucket class

10.4.6. Editing a bucket class

10.4.7. Editing backing stores for bucket class

10.5. MANAGING NAMESPACE BUCKETS

10.5.1. Adding provider connections to the Multicloud Object Gateway

10.5.2. Adding namespace resources using the Multicloud Object Gateway

10.5.3. Adding resources to namespace buckets using the Multicloud Object Gateway

10.5.4. Amazon S3 API endpoints for objects in namespace buckets

10.5.5. Adding a namespace bucket using the Multicloud Object Gateway CLI and YAML

10.5.5.1. Adding an AWS S3 namespace bucket using YAML

10.5.5.2. Adding an IBM COS namespace bucket using YAML

10.5.5.3. Adding an AWS S3 namespace bucket using the Multicloud Object Gateway CLI

10.5.5.4. Adding an IBM COS namespace bucket using the Multicloud Object Gateway CLI

10.6. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS

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

10.6.2. Creating bucket classes to mirror data using a YAML

10.6.3. Configuring buckets to mirror data using the user interface

10.7. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY

10.7.1. About bucket policies

10.7.2. Using bucket policies

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

10.8. OBJECT BUCKET CLAIM

10.8.1. Dynamic Object Bucket Claim

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

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

10.8.4. Attaching an Object Bucket Claim to a deployment

10.8.5. Viewing object buckets using the OpenShift Web Console

10.8.6. Deleting Object Bucket Claims

10.9. CACHING POLICY FOR OBJECT BUCKETS

10.9.1. Creating an AWS cache bucket

10.9.2. Creating an IBM COS cache bucket

10.10. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS

10.10.1. S3 endpoints in the Multicloud Object Gateway

10.10.2. Scaling with storage nodes

10.11. AUTOMATIC SCALING OF MULTICLOUD OBJECT GATEWAY ENDPOINTS

CHAPTER 11. MANAGING PERSISTENT VOLUME CLAIMS

11.1. Dynamic Persistent Volume Claim

11.2. Creating a Persistent Volume Claim using the command line interface

11.3. Creating a Persistent Volume Claim using the OpenShift Web Console

11.4. Attaching a Persistent Volume Claim to a deployment

11.5. Viewing Persistent Volume Claims using the OpenShift Web Console

11.6. Deleting Persistent Volume Claims

11.7. CACHING POLICY FOR Persistent VOLUME CLAIMS

11.8. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS

11.9. AUTOMATIC SCALING OF MULTICLOUD OBJECT GATEWAY ENDPOINTS
11.1. CONFIGURING APPLICATION PODS TO USE OPENSIFT CONTAINER STORAGE 115
11.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS 116
11.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS 117
11.4. DYNAMIC PROVISIONING 117
  11.4.1. About dynamic provisioning 117
  11.4.2. Dynamic provisioning in OpenShift Container Storage 118
  11.4.3. Available dynamic provisioning plug-ins 118

CHAPTER 12. VOLUME SNAPSHOTSF 120
  12.1. CREATING VOLUME SNAPSHOTS 120
  12.2. RESTORING VOLUME SNAPSHOTS 121
  12.3. DELETING VOLUME SNAPSHOTS 123

CHAPTER 13. VOLUME CLONING 125
  13.1. CREATING A CLONE 125

CHAPTER 14. REPLACING STORAGE NODES 126
  14.1. REPLACING OPERATIONAL NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE 126
  14.2. REPLACING FAILED NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE 127

CHAPTER 15. REPLACING STORAGE DEVICES 130
  15.1. REPLACING OPERATIONAL OR FAILED STORAGE DEVICES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE 130

CHAPTER 16. UPDATING OPENSIFT CONTAINER STORAGE 131
  16.1. OVERVIEW OF THE OPENSIFT CONTAINER STORAGE UPDATE PROCESS 131
  16.2. PREPARING TO UPDATE IN A DISCONNECTED ENVIRONMENT 131
    16.2.1. Adding mirror registry authentication details 132
    16.2.2. Building and mirroring the Red Hat operator catalog 133
    16.2.3. Creating Operator imageContentSourcePolicy 134
    16.2.4. Updating redhat-operator CatalogSource 134
    16.2.5. Continue to update 135
  16.3. UPDATING OPENSIFT CONTAINER STORAGE IN INTERNAL MODE 136
    16.3.1. Enabling automatic updates for OpenShift Container Storage operator in internal mode 136
    16.3.2. Manually updating OpenShift Container Storage operator in internal mode 138
Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Do let us know how we can make it better. To give feedback:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the **Bugzilla** website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.
Red Hat OpenShift Container Storage 4.8 supports deployment on existing Red Hat OpenShift Container Platform (RHOCP) Google Cloud clusters.

**NOTE**

Only internal Openshift Container Storage clusters are supported on Google Cloud. See Planning your deployment for more information about deployment requirements.

To deploy OpenShift Container Storage in internal mode, start with the requirements in Preparing to deploy OpenShift Container Storage chapter and then follow the deployment process Deploying OpenShift Container Storage on Google Cloud.
CHAPTER 1. PREPARING TO DEPLOY OPENSShift CONTAINER STORAGE

Deploying OpenShift Container Storage on OpenShift Container Platform using dynamic storage devices provides you with the option to create internal cluster resources. This will result in the internal provisioning of the base services, which helps to make additional storage classes available to applications.

Before you begin the deployment of Red Hat OpenShift Container Storage, follow these steps:

1. Optional: If you want to enable cluster-wide encryption using an external Key Management System (KMS):
   - Ensure that a policy with a token exists and the key value backend path in Vault is enabled. See Enabling the key value backend path and policy in Vault.
   - Ensure that you are using signed certificates on your Vault servers.

2. Minimum starting node requirements [Technology Preview]
   An OpenShift Container Storage cluster will be deployed with minimum configuration when the standard deployment resource requirement is not met. See Resource requirements section in Planning guide.

1.1. ENABLING KEY VALUE BACKEND PATH AND POLICY IN VAULT

Prerequisites

- Administrator access to Vault.
- Choose a unique path name as the backend path that follows the naming convention since it cannot be changed later.

Procedure

1. Enable the Key/Value (KV) backend path in Vault.
   For Vault KV secret engine API, version 1:
   
   ```bash
   $ vault secrets enable -path=ocs kv
   ```
   
   For Vault KV secret engine API, version 2:
   
   ```bash
   $ vault secrets enable -path=ocs kv-v2
   ```

2. Create a policy to restrict users to perform a write or delete operation on the secret using the following commands:

   ```bash
   echo '
   path "ocs/*" {
   capabilities = ["create", "read", "update", "delete", "list"]
   }
   path "sys/mounts" {
   capabilities = ["read"]
   }
   ' | vault policy write ocs -
   ```
3. Create a token matching the above policy:

$ vault token create -policy=ocs -format json
CHAPTER 2. DEPLOYING OPENSIFT CONTAINER STORAGE ON GOOGLE CLOUD

Deploying OpenShift Container Storage on OpenShift Container Platform using dynamic storage devices provided by Google Cloud installer-provisioned infrastructure (IPI) enables you to create internal cluster resources. This results in internal provisioning of the base services, which helps to make additional storage classes available to applications.

NOTE

Only internal OpenShift Container Storage clusters are supported on Google Cloud. See Planning your deployment for more information about deployment requirements.

Ensure that you have addressed the requirements in Preparing to deploy OpenShift Container Storage chapter before proceeding with the below steps for deploying using dynamic storage devices:

1. Install the Red Hat OpenShift Container Storage Operator.
2. Create the OpenShift Container Storage Cluster Service

2.1. INSTALLING RED HAT OPENSIFT CONTAINER STORAGE OPERATOR


Prerequisites

- Access to an OpenShift Container Platform cluster using an account with cluster-admin and operator installation permissions.
- You have at least three worker nodes in the Red Hat OpenShift Container Platform cluster.
- You have satisfied any additional requirements required. For more information, see Planning your deployment.

NOTE

- When you need to override the cluster-wide default node selector for OpenShift Container Storage, you can use the following command to specify a blank node selector for the openshift-storage namespace (create openshift-storage namespace in this case):

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

- Taint a node as infra to ensure only Red Hat OpenShift Container Storage resources are scheduled on that node. This helps you save on subscription costs. For more information, see How to use dedicated worker nodes for Red Hat OpenShift Container Storage chapter in Managing and Allocating Storage Resources guide.

Procedure
1. Log in to OpenShift Web Console.
2. Click Operators → OperatorHub.
3. Search for OpenShift Container Storage from the list of operators and click on it.
4. Click Install.
5. Set the following options on the Install Operator page:
   a. Channel as stable-4.8.
   b. Installation Mode as A specific namespace on the cluster
   c. Installed Namespace as Operator recommended namespace openshift-storage. If Namespace openshift-storage does not exist, it will be created during the operator installation.
   d. Approval Strategy as Automatic or Manual.
   e. Click Install.
      If you select Automatic updates, the Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your operator without any intervention.
      If you select Manual updates, the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to have the operator updated to the new version.

Verification step
   • Verify that the OpenShift Container Storage Operator shows a green tick indicating successful installation.

2.2. CREATING AN OPENSHIFT CONTAINER STORAGE CLUSTER SERVICE IN INTERNAL MODE

Use this procedure to create an OpenShift Container Storage Cluster Service after you install the OpenShift Container Storage operator.

Prerequisites
   • The OpenShift Container Storage operator must be installed from the Operator Hub. For more information, see Installing OpenShift Container Storage Operator using the Operator Hub.
   • Be aware that the default storage class of Google Cloud uses hard disk drive (HDD). To use solid state drive (SSD) based disks for better performance, you need to create a storage class, using pd-ssd as shown in the following ssd-storageclass.yaml example:

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: faster
provisioner: kubernetes.io/gce-pd
parameters:
```
Procedure

1. Log into the OpenShift Web Console.

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

3. Click OpenShift Container Storage > Create Instance link of Storage Cluster.

4. Select Mode is set to Internal by default.

5. Select Capacity and nodes
   a. Select Storage Class. By default, it is set to standard. However, if you created a storage class to use SSD based disks for better performance, you need to select that storage class.
   b. Select Requested Capacity from the drop down list. It is set to 2 TiB by default. You can use the drop down to modify the capacity value.

   NOTE
   Once you select the initial storage capacity, cluster expansion is performed only using the selected usable capacity (3 times of raw storage).

   c. In the Select Nodes section, select at least three available nodes. For cloud platforms with multiple availability zones, ensure that the Nodes are spread across different Locations/availability zones.

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

6. (Optional) Set Security and network configuration
   a. Select the Enable encryption checkbox to encrypt block and file storage.

   b. Choose any one or both Encryption level
      - Cluster-wide encryption to encrypt the entire cluster (block and file).
      - Storage class encryption to create encrypted persistent volume (block only) using encryption enabled storage class.

   c. Select the Connect to an external key management service checkbox. This is optional for cluster-wide encryption.
      - Key Management Service Provider is set to Vault by default.
      - Enter Vault Service Name, host Address of Vault server ('https://<hostname or ip>'), Port number and Token.
iii. Expand **Advanced Settings** to enter additional settings and certificate details based on your Vault configuration:

A. Enter the Key Value secret path in **Backend Path** that is dedicated and unique to OpenShift Container Storage.

B. (Optional) Enter **TLS Server Name** and **Vault Enterprise Namespace**

C. Provide **CA Certificate**, **Client Certificate** and **Client Private Key** by uploading the respective PEM encoded certificate file.

D. Click **Save**.

7. Select **Default** (SDN) if you are using a single network or **Custom** (Multus) Network if you plan on using multiple network interfaces.

   a. Select a **Public Network Interface** from drop down.

   b. Select a **Cluster Network Interface** from drop down.

   **NOTE**
   
   If only using one additional network interface select the single NetworkAttachmentDefinition (i.e. ocs-public-cluster) for the Public Network Interface and leave the Cluster Network Interface blank.

8. Click **Next**.

9. Review the configuration details. To modify any configuration settings, click **Back** to go back to the previous configuration page.

10. Click **Create**.

11. Edit the configmap if Vault Key/Value (KV) secret engine API, version 2 is used for cluster-wide encryption with Key Management System (KMS).

   a. On the OpenShift Web Console, navigate to **Workloads → ConfigMaps**

   b. To view the KMS connection details, click **ocs-kms-connection-details**

   c. Edit the configmap.

   i. Click **Action menu ( ⋮ ) → Edit ConfigMap**

   ii. Set the **VAULT_BACKEND** parameter to **v2**.

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: ocs-kms-connection-details
[...]
data:
  KMS_PROVIDER: vault
  KMS_SERVICE_NAME: vault
[...]
  VAULT_BACKEND: v2
[...]
```
iii. Click **Save**.

**Verification steps**

1. On the storage cluster details page, the storage cluster name displays a green tick next to it to indicate that the cluster was created successfully.

2. Verify that the final **Status** of the installed storage cluster shows as **Phase: Ready** with a green tick mark.
   - Click **Operators → Installed Operators → Storage Cluster** link to view the storage cluster installation status.
   - Alternatively, when you are on the Operator **Details** tab, you can click on the **Storage Cluster** tab to view the status.

3. To verify that all components for OpenShift Container Storage are successfully installed, see **Verifying your OpenShift Container Storage installation**.
CHAPTER 3. VERIFYING OPENSIFT CONTAINER STORAGE DEPLOYMENT

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

3.1. VERIFYING THE STATE OF THE PODS

To verify that the pods of OpenShift Containers Storage are in running state, follow the below procedure:

Procedure

1. Log in to OpenShift Web Console.
2. Click Workloads → Pods from the left pane of the OpenShift Web Console.
3. Select openshift-storage from the Project drop down list. For more information on the expected number of pods for each component and how it varies depending on the number of nodes, see Table 3.1, "Pods corresponding to OpenShift Container storage cluster".
4. Click on the Running and Completed tabs to verify that the pods are running and in a completed state:

Table 3.1. Pods corresponding to OpenShift Container storage cluster

<table>
<thead>
<tr>
<th>Component</th>
<th>Corresponding pods</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Container Storage Operator</td>
<td>ocs-operator-* (1 pod on any worker node)</td>
</tr>
<tr>
<td></td>
<td>ocs-metrics-exporter-*</td>
</tr>
<tr>
<td>Rook-ceph Operator</td>
<td>rook-ceph-operator-*</td>
</tr>
<tr>
<td></td>
<td>(1 pod on any worker node)</td>
</tr>
<tr>
<td>Multicloud Object Gateway</td>
<td>noobaa-operator-* (1 pod on any worker node)</td>
</tr>
<tr>
<td></td>
<td>noobaa-core-* (1 pod on any storage node)</td>
</tr>
<tr>
<td></td>
<td>noobaa-db-pg-* (1 pod on any storage node)</td>
</tr>
<tr>
<td></td>
<td>noobaa-endpoint-* (1 pod on any storage node)</td>
</tr>
<tr>
<td>MON</td>
<td>rook-ceph-mon-*</td>
</tr>
<tr>
<td></td>
<td>(3 pods distributed across storage nodes)</td>
</tr>
<tr>
<td>Component</td>
<td>Corresponding pods</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| MGR                | `rook-ceph-mgr-*`  
(1 pod on any storage node) |
| MDS                | `rook-ceph-mds-ocs-storagecluster-cephfilesystem-*`  
(2 pods distributed across storage nodes) |
| CSI                |  
- `cephfs`  
  - `csi-cephfsplugin-*` (1 pod on each worker node)  
  - `csi-cephfsplugin-provisioner-*` (2 pods distributed across worker nodes)  
- `rbd`  
  - `csi-rbdplugin-*` (1 pod on each worker node)  
  - `csi-rbdplugin-provisioner-*` (2 pods distributed across worker nodes)  |
| rook-ceph-crashcollector | `rook-ceph-crashcollector-*`  
(1 pod on each storage node) |
| OSD                |  
- `rook-ceph-osd-*` (1 pod for each device)  
- `rook-ceph-osd-prepare-ocs-deviceset-*` (1 pod for each device) |

### 3.2. VERIFYING THE OPENSIFT CONTAINER STORAGE CLUSTER IS HEALTHY

To verify that the cluster of OpenShift Container Storage is healthy, follow the steps in the procedure.

**Procedure**

1. Click **Storage → Overview** and click the **Block and File** tab.
2. In the **Status card**, verify that **Storage Cluster** and **Data Resiliency** has a green tick mark.
3. In the **Details card**, verify that the cluster information is displayed.

For more information on the health of the OpenShift Container Storage clusters using the Block and File dashboard, see [Monitoring OpenShift Container Storage](#).
3.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY

To verify that the OpenShift Container Storage Multicloud Object Gateway is healthy, follow the steps in the procedure.

Procedure

1. Click Storage → Overview from the OpenShift Web Console and click the Object tab.

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

3. In the Details card, verify that the Multicloud Object Gateway information is displayed.

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

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

To verify the storage classes exists in the cluster, follow the steps in the procedure.

Procedure

1. Click Storage → Storage Classes from the OpenShift Web Console.

2. Verify that the following storage classes are created with the OpenShift Container Storage cluster creation:

   • ocs-storagecluster-ceph-rbd
   • ocs-storagecluster-cephfs
   • openshift-storage.noobaa.io
CHAPTER 4. UNINSTALLING OPENSSHIFT CONTAINER STORAGE

4.1. UNINSTALLING OPENShift CONTAINER STORAGE IN INTERNAL MODE

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

Uninstall Annotations

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

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

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

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

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

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

- Ensure that the OpenShift Container Storage cluster is in a healthy state. The uninstall process can fail when some of the pods are not terminated successfully due to insufficient resources or nodes. In case the cluster is in an unhealthy state, contact Red Hat Customer Support before uninstalling OpenShift Container Storage.

- Ensure that applications are not consuming persistent volume claims (PVCs) or object bucket claims (OBCs) using the storage classes provided by OpenShift Container Storage.

- If any custom resources (such as custom storage classes, cephblockpools) were created by the admin, they must be deleted by the admin after removing the resources which consumed them.

Procedure

1. Delete the volume snapshots that are using OpenShift Container Storage.
   a. List the volume snapshots from all the namespaces.
      
      $ oc get volumesnapshot --all-namespaces
   
   b. From the output of the previous command, identify and delete the volume snapshots that are using OpenShift Container Storage.
      
      $ oc delete volumesnapshot <VOLUME-SNAPSHOT-NAME> -n <NAMESPACE>

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

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

      For more information, see Section 4.2, “Removing monitoring stack from OpenShift Container Storage”.

   b. Delete OpenShift Container Platform Registry PVCs using OpenShift Container Storage.
      For more information, see Section 4.3, “Removing OpenShift Container Platform registry from OpenShift Container Storage”.

   c. Delete OpenShift Container Platform logging PVCs using OpenShift Container Storage.
      For more information, see Section 4.4, “Removing the cluster logging operator from OpenShift Container Storage”.

   d. Delete other PVCs and OBCs provisioned using OpenShift Container Storage.

   - Following script is sample script to identify the PVCs and OBCs provisioned using OpenShift Container Storage. The script ignores the PVCs that are used internally by OpenShift Container Storage.

```bash
$ oc get pods -n openshift-storage | grep `grep -l ocs-internal /etc/ceph/ceph.conf | awk -F/ '{print $7}'`
```

Following script is sample script to identify the PVCs and OBCs provisioned using OpenShift Container Storage. The script ignores the PVCs that are used internally by OpenShift Container Storage.
Openshift Container Storage.

#!/bin/bash

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

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

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

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

NOTE

Omit RGW_PROVISIONER for cloud platforms.

- Delete the OBCs.
  
  $ oc delete obc <obc name> -n <project name>

- Delete the PVCs.
  
  $ oc delete pvc <pvc name> -n <project-name>

NOTE

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

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

  $ oc delete -n openshift-storage storagecluster --all --wait=true
4. Check for cleanup pods if the `uninstall.ocs.openshift.io/cleanup-policy` was set to `delete` (default) and ensure that their status is **Completed**.

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

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

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

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

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

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

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

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

   c. Remove the mapped device.

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

**NOTE**

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

- Press **CTRL+Z** to exit the above command.

```bash
$ ps -ef | grep crypt
```

- Find PID of the process which was stuck.

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

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

- Verify that the device name is removed.

```bash
$ dmsetup ls
```
7. Delete the namespace and wait till the deletion is complete. You need to switch to another project if `openshift-storage` is the active project. For example:

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

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

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

**NOTE**

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

8. Unlabel the storage nodes.

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

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

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

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

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

11. Delete the Multicloud Object Gateway storageclass.

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

12. Remove **CustomResourceDefinitions**.

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

13. Optional: To ensure that the vault keys are deleted permanently you need to manually delete the metadata associated with the vault key.
NOTE

Execute this step only if Vault Key/Value (KV) secret engine API, version 2 is used for cluster-wide encryption with Key Management System (KMS) since the vault keys are marked as deleted and not permanently deleted during the uninstallation of OpenShift Container Storage. You can always restore it later if required.

a. List the keys in the vault.

   $ vault kv list <backend_path>

   <backend_path>
   Is the path in the vault where the encryption keys are stored.
   For example:

   $ vault kv list kv-v2

   Example output:

   Keys
   -----
   NOOBAA_ROOT_SECRET_PATH/
   rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8
   rook-ceph-osd-encryption-key-ocs-deviceset-thin-1-data-0sq227
   rook-ceph-osd-encryption-key-ocs-deviceset-thin-2-data-0xzszb

b. List the metadata associated with the vault key.

   $ vault kv get kv-v2/<key>

   For the Multicloud Object Gateway (MCG) key:

   $ vault kv get kv-v2/NOOBAA_ROOT_SECRET_PATH/<key>

   <key>
   Is the encryption key.
   For Example:

   $ vault kv get kv-v2/rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8

   Example output:

   ====== Metadata ======
   Key      Value
   ----     -----    
   created_time  2021-06-23T10:06:30.650103555Z
   deletion_time 2021-06-23T11:46:35.045328495Z
   destroyed    false
   version       1
c. Delete the metadata.

   $ vault kv metadata delete kv-v2/<key>

For the MCG key:

   $ vault kv metadata delete kv-v2/NOOBAA_ROOT_SECRET_PATH/<key>

   <key>
   
   Is the encryption key.
   
   For Example:

   $ vault kv metadata delete kv-v2/rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8

   Example output:

   Success! Data deleted (if it existed) at: kv-v2/metadata/rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8

d. Repeat these steps to delete the metadata associated with all the vault keys.

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

   a. Click Storage.

   b. Verify that Overview no longer appears under Storage.

### 4.2. REMOVING MONITORING STACK FROM OPENSIFT CONTAINER STORAGE

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

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

**Prerequisites**

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

**Procedure**

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

   $ oc get pod,pvc -n openshift-monitoring

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pod/alertmanager-main-0</td>
<td>3/3</td>
<td>Running 0</td>
<td>8d</td>
<td></td>
</tr>
<tr>
<td>pod/alertmanager-main-1</td>
<td>3/3</td>
<td>Running 0</td>
<td>8d</td>
<td></td>
</tr>
<tr>
<td>pod/alertmanager-main-2</td>
<td>3/3</td>
<td>Running 0</td>
<td>8d</td>
<td></td>
</tr>
<tr>
<td>pod/cluster-monitoring-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Edit the monitoring **configmap**.

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

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

   **Before editing**
```yaml
apiVersion: v1
data:  
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        metadata:
          name: my-alertmanager-claim
        spec:
          resources:
            requests:
              storage: 40Gi
        storageClassName: ocs-storagecluster-ceph-rbd

    prometheusK8s:
      volumeClaimTemplate:
        metadata:
          name: my-prometheus-claim
        spec:
          resources:
            requests:
              storage: 40Gi
        storageClassName: ocs-storagecluster-ceph-rbd

kind: ConfigMap
metadata:  
  creationTimestamp: "2019-12-02T07:47:29Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "22110"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: fd6d988b-14d7-11ea-84ff-066035b9efa8
```

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

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

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

4.3. REMOVING OPENSHIFT CONTAINER PLATFORM REGISTRY FROM OPENSHIFT CONTAINER STORAGE

To clean the OpenShift Container Platform registry from OpenShift Container Storage, follow the steps in the procedure.

If you want to configure an alternative storage, see image registry

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

Prerequisites

- The image registry must be configured to use an OpenShift Container Storage PVC.

Procedure

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

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

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

2. Delete the PVC.
   
   ```bash
   $ oc delete pvc <pvc-name> -n openshift-image-registry --wait=true --timeout=5m
   ```

4.4. REMOVING THE CLUSTER LOGGING OPERATOR FROM OPENSHIFT CONTAINER STORAGE

To clean the cluster logging operator from the OpenShift Container Storage, follow the steps in the procedure.

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

Prerequisites

- The cluster logging instance must be configured to use OpenShift Container Storage PVCs.

Procedure

1. Remove the `ClusterLogging` instance in the namespace.
   
   ```bash
   $ oc delete clusterlogging instance -n openshift-logging --wait=true --timeout=5m
   ```

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

2. Delete PVCs.
$ oc delete pvc <pvc-name> -n openshift-logging --wait=true --timeout=5m
CHAPTER 5. STORAGE CLASSES AND STORAGE POOLS

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

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

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

NOTE

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

NOTE

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

5.1. CREATING STORAGE CLASSES AND POOLS

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

Prerequisites

- Ensure that you are logged into the OpenShift Container Platform web console and OpenShift Container Storage cluster is in Ready state.

Procedure

1. Click Storage → Storage Classes.
2. Click Create Storage Class
3. Enter the storage class Name and Description.
4. Select either Delete or Retain for the Reclaim Policy. By default, Delete is selected.
5. Select RBD Provisioner which is the plugin used for provisioning the persistent volumes.
6. Select an existing Storage Pool from the list or create a new pool.

Create new pool

a. Click Create New Pool
b. Enter Pool name.
c. Choose 2-way-Replication or 3-way-Replication as the Data Protection Policy.
d. Select **Enable compression** if you need to compress the data. Enabling compression can impact application performance and might prove ineffective when data to be written is already compressed or encrypted. Data written before enabling compression will not be compressed.

e. Click **Create** to create the new storage pool.

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

7. (Optional) Select **Enable Encryption** checkbox.

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

### 5.2. CREATING A STORAGE CLASS FOR PERSISTENT VOLUME ENCRYPTION

Use the following procedure to create an encryption enabled storage class using an external key management system (KMS) for persistent volume encryption. Persistent volume encryption is only available for RBD PVs.

**Prerequisites**

- The OpenShift Container Storage cluster is in **Ready** state.
- On the external key management system (KMS),
  - Ensure that a policy with a token exists and the key value backend path in Vault is enabled. See [Enabling key value and policy in Vault](#).
  - Ensure that you are using signed certificates on your Vault servers.
- Create a secret in the tenant’s namespace as follows:
  - On the OpenShift Container Platform web console, navigate to **Workloads → Secrets**
  - Click **Create → Key/value secret**
  - Enter **Secret Name** as **ceph-csi-kms-token**.
  - Enter **Key** as **token**.
  - Enter **Value**. It is the token from Vault. You can either click **Browse** to select and upload the file containing the token or enter the token directly in the text box.
  - Click **Create**.

**NOTE**

The token can be deleted only after all the encrypted PVCs using the **ceph-csi-kms-token** have been deleted.

**Procedure**

1. Navigate to **Storage → Storage Classes**.
2. Click **Create Storage Class**

3. Enter the storage class **Name** and **Description**.

4. Select either Delete or Retain for the **Reclaim Policy**. By default, Delete is selected.

5. Select **RBD Provisioner openshift-storage.rbd.csi.ceph.com** which is the plugin used for provisioning the persistent volumes.

6. Select **Storage Pool** where the volume data will be stored from the list or create a new pool.

7. Select **Enable Encryption** checkbox.
   
   a. **Key Management Service Provider** is set to Vault by default.
   
   b. Enter Vault **Service Name**, host **Address** of Vault server ('https://<hostname or ip>'), and **Port number**.
   
   c. Expand **Advanced Settings** to enter additional settings and certificate details based on your Vault configuration.
      
      i. Enter the key value secret path in **Backend Path** that is dedicated and unique to OpenShift Container Storage.
      
      ii. (Optional) Enter **TLS Server Name** and **Vault Enterprise Namespace**
      
      iii. Provide **CA Certificate**, **Client Certificate** and **Client Private Key** by uploading the respective PEM encoded certificate file.
      
      iv. Click **Save**.
   
   d. Click **Connect**.

8. Review external key management service Connection details. To modify the information, click **Change connection details** and edit the fields.

9. Click **Create**.

10. Edit the configmap to add the **VAULT_BACKEND** parameter if the Hashicorp Vault setup does not allow automatic detection of the Key/Value (KV) secret engine API version used by the backend path.

    **NOTE**

    **VAULT_BACKEND** is an optional parameter that is added to the configmap to specify the version of the KV secret engine API associated with the backend path. Ensure that the value matches the KV secret engine API version that is set for the backend path, otherwise it might result in a failure during persistent volume claim (PVC) creation.

    a. Identify the **encryptionKMSID** being used by the newly created storage class.
       
       i. On the OpenShift Web Console, navigate to **Storage → Storage Classes**
      
       ii. Click the **Storage class** name → YAML tab.
      
      iii. Capture the **encryptionKMSID** being used by the storage class.
Example:

```
encryptionKMSID: 1-vault
```

b. On the OpenShift Web Console, navigate to **Workloads → ConfigMaps**

c. To view the KMS connection details, click **csi-kms-connection-details**.

d. Edit the configmap.

i. Click Action menu (⋮) → **Edit ConfigMap**.

ii. Add the **VAULT_BACKEND** parameter depending on the backend that is configured for the previously identified **encryptionKMSID**.

   You can assign **kv** for KV secret engine API, version 1 and **kv-v2** for KV secret engine API, version 2 as the **VAULT_BACKEND** parameter.

   Example:

   ```yaml
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: csi-kms-connection-details
   [...]
   data:
     1-vault: >-
       
       
       "KMS_PROVIDER": "vaulttokens",
       "KMS_SERVICE_NAME": "vault",
       [...]
       "VAULT_BACKEND": "kv-v2"
   
   ```

iii. Click **Save**.

**IMPORTANT**

Red Hat works with the technology partners to provide this documentation as a service to the customers. However, Red Hat does not provide support for the Hashicorp product. For technical assistance with this product, contact Hashicorp.

**Next steps**

- The storage class can be used to create encrypted persistent volumes. For more information, see **managing persistent volume claims**.
CHAPTER 6. CONFIGURE STORAGE FOR OPENSHEFT CONTAINER PLATFORM SERVICES

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

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

**WARNING**

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

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

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

6.1. CONFIGURING IMAGE REGISTRY TO USE OPENSHEFT 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 Google Cloud, it is not required to change the storage for the registry.

**WARNING**

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

**Prerequisites**

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

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

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

Procedure

1. Create a Persistent Volume Claim for the Image Registry to use.
   a. In the OpenShift Web Console, click Storage → Persistent Volume Claims.
   b. Set the Project to openshift-image-registry.
   c. Click Create Persistent Volume Claim
      i. From the list of available storage classes retrieved above, specify the Storage Class with the provisioner openshift-storage.cephfs.csi.ceph.com.
      ii. Specify the Persistent Volume Claim Name, for example, ocs4registry.
      iii. Specify an Access Mode of Shared Access (RWX).
      iv. Specify a Size of at least 100 GB.
      v. Click Create.
         Wait until the status of the new Persistent Volume Claim is listed as Bound.

2. Configure the cluster’s Image Registry to use the new Persistent Volume Claim.
   a. Click Administration → Custom Resource Definitions
   b. Click the Config custom resource definition associated with the imageregistry.operator.openshift.io group.
   c. Click the Instances tab.
   d. Beside the cluster instance, click the Action Menu (⋮) → Edit Config.
   e. Add the new Persistent Volume Claim as persistent storage for the Image Registry.
      i. Add the following under spec:, replacing the existing storage: section if necessary.

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

For example:

```
storage:
pvc:
  claim: ocs4registry
```
ii. Click **Save**.

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

### 6.2. Configuring Monitoring to Use OpenShift Container Storage

OpenShift Container Storage provides a monitoring stack that comprises of Prometheus and Alert Manager.

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

**IMPORTANT**

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

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

**Prerequisites**

- You have administrative access to OpenShift Web Console.

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

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

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

**Procedure**

1. In the OpenShift Web Console, go to **Workloads → Config Maps**.

2. Set the **Project** dropdown to **openshift-monitoring**.
3. Click Create Config Map.

4. Define a new `cluster-monitoring-config` Config Map using the following example. Replace the content in angle brackets (<, >) with your own values, for example, **retention: 24h** or **storage: 40Gi**.

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

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

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

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

**Verification steps**

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

   **Monitoring storage created and bound**
2. Verify that the new `alertmanager-main-*` pods appear with a state of Running.
   a. Go to Workloads → Pods.
   b. Click the new `alertmanager-main-*` pods to view the pod details.
   c. Scroll down to Volumes and verify that the volume has a Type, `ocs-alertmanager-claim` that matches one of your new Persistent Volume Claims, for example, `ocs-alertmanager-claim-alertmanager-main-0`.

   Persistent Volume Claims attached to `alertmanager-main-*` pod

![Persistent Volume Claims](image)

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

![Persistent Volume Claims](image)

6.3. CLUSTER LOGGING FOR OPENSFOFT CONTAINER STORAGE
You can deploy cluster logging to aggregate logs for a range of OpenShift Container Platform services. For information about how to deploy cluster logging, see Deploying cluster logging.

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

**IMPORTANT**

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

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

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

### 6.3.1. 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:

```yaml
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:

```yaml
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage: {}
```
For more information, see Configuring cluster logging.

6.3.2. Configuring cluster logging to use OpenShift Container Storage

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

**NOTE**

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

**Prerequisites**

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

**Procedure**

1. Click **Administration → Custom Resource Definitions** from the left pane of the OpenShift Web Console.

2. On the Custom Resource Definitions page, click **ClusterLogging**.

3. On the Custom Resource Definition Overview page, select **View Instances** from the Actions menu or click the **Instances** Tab.

4. On the Cluster Logging page, click **Create Cluster Logging**. You might have to refresh the page to load the data.

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

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

2. Verify that the new cluster logging is being used.
   a. Click Workload → Pods
b. Set the Project to `openshift-logging`.

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

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

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

f. Click the Persistent Volume Claim name and verify the storage class name in the PersistentVolumeClaim 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.

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

For more details, see [Curation of Elasticsearch Data](#).

**NOTE**

To uninstall the cluster logging backed by Persistent Volume Claim, use the procedure removing the cluster logging operator from OpenShift Container Storage in the uninstall chapter of the respective deployment guide.
CHAPTER 7. 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.
       1. Click **Create Deployment** to create a new deployment.
       2. Edit the **YAML** based on your requirement to create a deployment.
       3. Click **Create**.
       4. 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.
         1. Click **Create Deployment Config** to create a new deployment.
         2. Edit the **YAML** based on your requirement to create a deployment.
         3. Click **Create**.
         4. Select **Add Storage** from the **Actions** drop down menu on the top right of the page.

2. In the Add Storage page, you can choose one of the following options:

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

  **NOTE**
  
  ReadOnlyMany (ROX) is deactivated as it is not supported.
  
  d. Select the size of the desired storage capacity.

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

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

4. Click Save.

**Verification steps**

1. Depending on your configuration, perform one of the following:
   
   - Click Workloads → Deployments.
   - Click Workloads → Deployment Configs

2. Set the Project as required.

3. Click the deployment for which you added storage to display the deployment details.

4. Scroll down to Volumes and verify that your deployment has a Type that matches the Persistent Volume Claim that you assigned.

5. Click the Persistent Volume Claim name and verify the storage class name in the Persistent Volume Claim Overview page.
CHAPTER 8. HOW TO USE DEDICATED WORKER NODES FOR RED HAT OPENSHIFT CONTAINER STORAGE

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

It is important to maintain consistency across environments with or without Machine API support. Because of this, it is highly recommended in all cases to have a special category of nodes labeled as either worker or infra or have both roles. See the Section 8.3, “Manual creation of infrastructure nodes” section for more information.

8.1. ANATOMY OF AN INFRASTRUCTURE NODE

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

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

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

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

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

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

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

8.2. MACHINE SETS FOR CREATING INFRASTRUCTURE NODES

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

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

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

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

8.3. MANUAL CREATION OF INFRASTRUCTURE NODES

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

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

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

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

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

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

If already removed, it should be added again to each `infra` node. Adding node-role `node-role.kubernetes.io/infra=""` and OpenShift Container Storage taint is sufficient to conform to entitlement exemption requirements.
CHAPTER 9. SCALING STORAGE NODES

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

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

9.1. REQUIREMENTS FOR SCALING STORAGE NODES

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

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

**WARNING**

Always ensure that you have plenty of storage capacity.

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

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

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

9.2. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSIFT CONTAINER STORAGE NODES ON GOOGLE CLOUD 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.
● To scale using a storage class other than the one provisioned during deployment, first define an additional storage class. See Creating a storage class for details.

Procedure

1. Log in to the OpenShift Web Console.

2. Click on Operators → Installed Operators.

3. Click OpenShift Container Storage Operator.

4. Click Storage Cluster tab.

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

6. Select Add Capacity from the options menu.

7. Select the Storage Class
   Set the storage class to standard if you are using the default storage class that uses HDD. However, if you created a storage class to use SSD based disks for better performance, you need to select that storage class.

   The Raw Capacity field shows the size set during storage class creation. The total amount of storage consumed is three times this amount, because OpenShift Container Storage uses a replica count of 3.

8. Click Add and wait for the cluster state to change to Ready.

Verification steps

● Navigate to Overview → Block and File tab, then check the Raw Capacity breakdown card. Note that the capacity increases based on your selections.

   NOTE

   The raw capacity does not take replication into account and shows the full capacity.

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

   ○ To view the state of the newly created OSDs:
     a. Click Workloads → Pods from the OpenShift Web Console.
     b. Select openshift-storage from the Project drop-down list.

   ○ To view the state of the PVCs:
     a. Click Storage → Persistent Volume Claims from the OpenShift Web Console.
     b. Select openshift-storage from the Project drop-down list.

● (Optional) If cluster-wide encryption is enabled on the cluster, verify that the new OSD devices are encrypted.

   a. Identify the node(s) where the new OSD pod(s) are running.
For each of the nodes identified in previous step, do the following:

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

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

ii. Run "lsblk" and check for the "crypt" keyword beside the `ocs-deviceset` name(s)

$ lsblk

**IMPORTANT**
Cluster reduction is not currently supported, regardless of whether reduction would be done by removing nodes or OSDs.

### 9.3. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES

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

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

**9.3.1. Adding a node on Google Cloud installer-provisioned infrastructure**

**Prerequisites**
- You must be logged into OpenShift Container Platform (RHOCP) cluster.

**Procedure**

1. Navigate to **Compute → Machine Sets**.
2. On the machine set where you want to add nodes, select **Edit Machine 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, one each in different zones. You must add 3 nodes and perform this procedure for all of them.

Verification steps

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

### 9.3.2. Verifying the addition of a new node

1. Execute the following command and verify that the new node is present in the output:
   
   ```bash
   $ 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-*`

### 9.3.3. Scaling up storage capacity

After you add a new node to OpenShift Container Storage, you must scale up the storage capacity as described in Scaling up storage by adding capacity.
CHAPTER 10. MULTICLOUD OBJECT GATEWAY

10.1. 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.

10.2. 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:

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

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:

  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
  ```

- For IBM Z infrastructure, use the following command:

  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

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

**NOTE**

Choose the correct Product Variant according to your architecture.

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

- Section 10.2.1, “Accessing the Multicloud Object Gateway from the terminal”
- Section 10.2.2, “Accessing the Multicloud Object Gateway from the MCG command-line interface”
Accessing the MCG bucket(s) using the virtual-hosted style

Example 10.1. Example

If the client application tries to access https://<bucket-name>.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com

where <bucket-name> is the name of the MCG bucket

For example, https://mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com

A DNS entry is needed for mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com to point to the S3 Service.

IMPORTANT

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

10.2.1. 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):

```bash
# 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:
  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:

Let's 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://a340f4e1315be11eeaa3b70683061451e-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.

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

**Prerequisites**

- Download the MCG command-line interface:

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

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:

```bash
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- For IBM Z infrastructure, use the following command:

```bash
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

**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[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]
#------------------#
#- Mgmt Addresses -#
#------------------#

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

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

email: admin@noobaa.io
password: HKLbH1rSuVU0I/soulkSiA==
#--------#
#- S3 Addresses -#
#--------#

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

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

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

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

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

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

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

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

No OBC's found.

1 endpoint
2 access key
3 secret access key

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

Example 10.2. Example
If AWS S3 CLI is the application, the following command will list buckets in OpenShift Container Storage:

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

10.3. ALLOWING USER ACCESS TO THE MULTICLOUD OBJECT GATEWAY CONSOLE

To allow access to the Multicloud Object Gateway Console to a user, ensure that the user meets the following conditions:

- User is in `cluster-admins` group.
- User is in `system:cluster-admins` virtual group.

**Prerequisites**

- A running OpenShift Container Storage Platform.

**Procedure**

1. Enable access to the Multicloud Object Gateway console.
   Perform the following steps once on the cluster:
   
   a. Create a `cluster-admins` group.
      
      ```
      # oc adm groups new cluster-admins
      ```
   b. Bind the group to the `cluster-admin` role.
      
      ```
      # oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
      ```

2. Add or remove users from the `cluster-admins` group to control access to the Multicloud Object Gateway console.
   
   - To add a set of users to the `cluster-admins` group:
     
     ```
     # oc adm groups add-users cluster-admins <user-name> <user-name> <user-name>...
     ```
   
   where `<user-name>` is the name of the user to be added.

   **NOTE**
   
   If you are adding a set of users to the `cluster-admins` group, you do not need to bind the newly added users to the cluster-admin role to allow access to the OpenShift Container Storage dashboard.

   - To remove a set of users from the `cluster-admins` group:
# oc adm groups remove-users cluster-admins <user-name> <user-name> <user-name>...

where `<user-name>` is the name of the user to be removed.

Verification steps

1. On the OpenShift Web Console, login as a user with access permission to Multicloud Object Gateway Console.
2. Navigate to Storage → Overview → Object tab → select the Multicloud Object Gateway link.
3. On the Multicloud Object Gateway Console, login as the same user with access permission.
4. Click Allow selected permissions.

10.4. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD

10.4.1. 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. Click OpenShift Container Storage Operator.
3. On the OpenShift Container Storage Operator page, scroll right and click the Backing Store tab.
4. Click Create Backing Store.
5. On the Create New Backing Store page, perform the following:

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

   b. Select a **Provider**.

   c. Select a **Region**.

   d. Enter an **Endpoint**. This is optional.

   e. Select a **Secret** from drop down list, or create your own secret. Optionally, you can **Switch to Credentials** view which lets you fill in the required secrets. For more information on creating an OCP secret, see the section **Creating the secret** in the Openshift Container Platform documentation.

   Each backingstore requires a different secret. For more information on creating the secret for a particular backingstore, see the Section 10.4.2, “Adding storage resources for hybrid or Multicloud using the MCG command line interface” and follow the procedure for the addition of storage resources using a YAML.

   **NOTE**

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

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

6. Click **Create Backing Store**

**Verification steps**

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

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

3. Search for the new backing store or click **Backing Store** tab to view all the backing stores.
10.4.2. 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 an AWS-backed backingstore, see Section 10.4.2.1, “Creating an AWS-backed backingstore”
- For creating an IBM COS-backed backingstore, see Section 10.4.2.2, “Creating an IBM COS-backed backingstore”
- For creating an Azure-backed backingstore, see Section 10.4.2.3, “Creating an Azure-backed backingstore”
- For creating a GCP-backed backingstore, see Section 10.4.2.4, “Creating a GCP-backed backingstore”
- For creating a local Persistent Volume-backed backingstore, see Section 10.4.2.5, “Creating a local Persistent Volume-backed backingstore”

For VMware deployments, skip to Section 10.4.3, “Creating an s3 compatible Multicloud Object Gateway backingstore” for further instructions.

10.4.2.1. 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
  ```

  **NOTE**

  Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

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

  **NOTE**

  Choose the correct Product Variant according to your architecture.
Procedure

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

   ```
   noobaa backingstore create aws-s3 <backingstore_name> --access-key=<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
   ```

   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:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: <backingstore-secret-name>
     namespace: openshift-storage
     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:

   ```yaml
   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: openshift-storage
  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.

### 10.4.2.2. 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
  ```

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:
  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
  ```
- For IBM Z infrastructure, use the following command:
  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

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

**NOTE**

Choose the correct Product Variant according to your architecture.

#### Procedure

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

```bash
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> -n openshift-storage
```

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.

10.4.2.3. Creating an Azure-backed backingstore

**Prerequisites**

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

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

  **NOTE**

  Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

- 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

  **NOTE**

  Choose the correct Product Variant according to your architecture.

**Procedure**

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

   ```bash
   noobaa backingstore create azure-blob <backingstore_name> --account-key=<AZURE ACCOUNT KEY> --account-name=<AZURE ACCOUNT NAME> --target-blob-container <blob container name>
   ```

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

   b. Replace `<AZURE ACCOUNT KEY>` and `<AZURE ACCOUNT NAME>` with an AZURE account key and account name you created for this purpose.

   c. Replace `<blob container name>` with an existing Azure blob container 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:
You can also add storage resources using a YAML:

1. Create a secret with the credentials:

apiVersion: v1
class: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  AccountName: <AZURE ACCOUNT NAME ENCODED IN BASE64>
  AccountKey: <AZURE ACCOUNT KEY ENCODED IN BASE64>

a. You must supply and encode your own Azure Account Name and Account Key using Base64, and use the results in place of `<AZURE ACCOUNT NAME ENCODED IN BASE64>` and `<AZURE ACCOUNT 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
class: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
labels:
  app: noobaa
name: bs
namespace: openshift-storage
spec:
  azureBlob:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
targetBlobContainer: <blob-container-name>
type: azure-blob

a. Replace `<blob-container-name>` with an existing Azure blob container 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.

10.4.2.4. Creating a GCP-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

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

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

**NOTE**

Choose the correct Product Variant according to your architecture.

**Procedure**

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

   ```
   noobaa backingstore create google-cloud-storage <backingstore_name> --private-key-json-file=<PATH TO GCP PRIVATE KEY JSON FILE> --target-bucket <GCP bucket name>
   ```

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

   b. Replace `<PATH TO GCP PRIVATE KEY JSON FILE>` with a path to your GCP private key created for this purpose.

   c. Replace `<GCP bucket name>` with an existing GCP object storage 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 "google-gcp"
   INFO[0002]  Created: Secret "backing-store-google-cloud-storage-gcp"
   ```

   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:  
      GoogleServiceAccountPrivateKeyJson: <GCP PRIVATE KEY ENCODED IN BASE64>
   ```
a. You must supply and encode your own GCP service account private key using Base64, and use the results in place of `<GCP PRIVATE KEY ENCODED IN BASE64>`.

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

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

```yaml
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
labels:
  app: noobaa
  name: bs
  namespace: openshift-storage
spec:
googleCloudStorage:
  secret:
    name: `<backingstore-secret-name>`
    namespace: openshift-storage
  targetBucket: `<target bucket>`
type: google-cloud-storage
```

a. Replace `<target bucket>` with an existing Google storage bucket. 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.

10.4.2.5. Creating a local Persistent Volume-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
  ```

  **NOTE**

  Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

- 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](https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages)
Choose the correct Product Variant according to your architecture.

Procedure

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

   ```
   noobaa backingstore create pv-pool <backingstore_name> --num-volumes=<NUMBER OF VOLUMES> --pv-size-gb=<VOLUME SIZE> --storage-class=<LOCAL STORAGE CLASS>
   ```

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

   b. Replace `<NUMBER OF VOLUMES>` with the number of volumes you would like to create. Note that increasing the number of volumes scales up the storage.

   c. Replace `<VOLUME SIZE>` with the required size, in GB, of each volume

   d. Replace `<LOCAL STORAGE CLASS>` with the local storage class, recommended to use `ocs-storagecluster-ceph-rbd`

      The output will be similar to the following:

      ```
      INFO[0001]  Exists: NooBaa "noobaa"
      INFO[0002]  Exists: BackingStore "local-mcg-storage"
      ```

You can also add storage resources using a YAML:

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

   ```
   apiVersion: noobaa.io/v1alpha1
   kind: BackingStore
   metadata:
     finalizers:
     - noobaa.io/finalizer
   labels:
     app: noobaa
     name: <backingstore_name>
     namespace: openshift-storage
   spec:
     pvPool:
       numVolumes: <NUMBER OF VOLUMES>
       resources:
         requests:
           storage: <VOLUME SIZE>
       storageClass: <LOCAL STORAGE CLASS>
   type: pv-pool
   ```

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

   b. Replace `<NUMBER OF VOLUMES>` with the number of volumes you would like to create. Note that increasing the number of volumes scales up the storage.

   c. Replace `<VOLUME SIZE>` with the required size, in GB, of each volume. Note that the letter G should remain
10.4.3. Creating an s3 compatible Multicloud Object Gateway backingstore

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

Procedure

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

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

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

      oc get secret <RGW USER SECRET NAME> -o yaml -n openshift-storage

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

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

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

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

      The output will be similar to the following:

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

You can also create the backingstore using a YAML:

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

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

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

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

   a. Replace `<backingstore-secret-name>` with the name of the secret that was created with `CephObjectStore` in the previous step.
   
   b. Replace `<bucket-name>` with an existing RGW bucket name. This argument tells Multicloud Object Gateway which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.
   
   c. To get the `<RGW endpoint>`, see Accessing the RADOS Object Gateway S3 endpoint.

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

**Procedure**

1. In your OpenShift Storage console, click **Storage → Overview → Object tab → Multicloud Object Gateway** link.

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

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

5. Select the newly created connection and map it to the existing bucket.
6. Repeat these steps to create as many backing stores as needed.

**NOTE**

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

10.4.5. 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.

4. Click **Create Bucket Class**.

5. On the **Create new Bucket Class** page, perform the following:
   a. Select the bucket class type and enter a bucket class name.
      i. Select the **BucketClass type**. Choose one of the following options:
         - **Namespace**
           Data is stored on the NamespaceStores without performing de-duplication, compression or encryption.
         - **Standard**
           Data will be consumed by a Multicloud Object Gateway (MCG), deduped, compressed and encrypted.
By default, Standard is selected.

ii. Enter a **Bucket Class Name**.

iii. Click **Next**.

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.

c. Select at least 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.

   **NOTE**
   You need to select at least 2 backing stores when you select Policy Type as Mirror in previous step.

d. Review and confirm Bucket Class settings.

e. 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.

**10.4.6. Editing a bucket class**

Use the following procedure to edit the bucket class components through the YAML file by clicking the **edit** button on the OpenShift web console.

**Prerequisites**

- Administrator access to OpenShift.

**Procedure**

1. Log into the **OpenShift Web Console**.

2. Click **Operators → Installed Operators**.

3. Click **OpenShift Container Storage Operator**.

4. On the OpenShift Container Storage Operator page, scroll right and click the **Bucket Class** tab.
5. Click on the action menu (⋮) next to the Bucket class you want to edit.

6. Click Edit Bucket Class.

7. You are redirected to the YAML file, make the required changes in this file and click Save.

10.4.7. Editing backing stores for bucket class

Use the following procedure to edit an existing Multicloud Object Gateway bucket class to change the underlying backing stores used in a bucket class.

Prerequisites

- Administrator access to OpenShift Web Console.
- A bucket class.
- Backing stores.

Procedure

1. Click Operators → Installed Operators to view the installed operators.

2. Click OpenShift Container Storage Operator.

3. Click the Bucket Class tab.

4. Click on the action menu (⋮) next to the Bucket class you want to edit.

5. Click Edit Bucket Class Resources.

6. On the Edit Bucket Class Resources page, edit the bucket class resources either by adding a backing store to the bucket class or by removing a backing store from the bucket class. You can also edit bucket class resources created with one or two tiers and different placement policies.

   - To add a backing store to the bucket class, select the name of the backing store.
   - To remove a backing store from the bucket class, clear the name of the backing store.
7. Click Save.

10.5. MANAGING NAMESPACE BUCKETS

Namespace buckets let you connect data repositories on different providers together, so you can interact with all of your data through a single unified view. Add the object bucket associated with each provider to the namespace bucket, and access your data through the namespace bucket to see all of your object buckets at once. This lets you write to your preferred storage provider while reading from multiple other storage providers, greatly reducing the cost of migrating to a new storage provider.

1. **Connect your providers to the Multicloud Object Gateway**.

2. **Create a namespace resource for each of your providers** so they can be added to a namespace bucket.

3. **Add your namespace resources to a namespace bucket** and configure the bucket to read from and write to the appropriate namespace resources.

You can interact with objects in a namespace bucket using the S3 API. See S3 API endpoints for objects in namespace buckets for more information.

**NOTE**

A namespace bucket can only be used if its write target is available and functional.

10.5.1. Adding provider connections to the Multicloud Object Gateway

You need to add connections for each of your providers so that the Multicloud Object Gateway has access to the provider.

**Prerequisites**

- Administrative access to the OpenShift Console.

**Procedure**

1. In the OpenShift Console, click **Storage → Overview** and click the **Object** tab.
2. Click Multicloud Object Gateway and log in if prompted.

3. Click Accounts and select an account to add the connection to.

4. Click My Connections.

5. Click Add Connection.
   a. Enter a Connection Name.
   b. Your cloud provider is shown in the Service dropdown by default. Change the selection to use a different provider.
   c. Your cloud provider’s default endpoint is shown in the Endpoint field by default. Enter an alternative endpoint if required.
   d. Enter your Access Key for this cloud provider.
   e. Enter your Secret Key for this cloud provider.
   f. Click Save.

10.5.2. Adding namespace resources using the Multicloud Object Gateway

Add existing storage to Multicloud Storage Gateway as namespace resources so that they can be included in namespace buckets for a unified view of existing storage targets, such as Amazon Web Services S3 buckets, Microsoft Azure blobs, and IBM Cloud Object Storage buckets.

Prerequisites

- Administrative access to the OpenShift Console.
- Target connections (providers) are already added to the Multicloud Object Gateway. See Section 10.5.1, “Adding provider connections to the Multicloud Object Gateway” for details.

Procedure

1. In the OpenShift Console, click Storage → Overview and click on the Object tab.

2. Click Multicloud Storage Gateway and log in if prompted.

3. Click Resources, and click the Namespace Resources tab.

4. Click Create Namespace Resource
   a. In Target Connection, select the connection to be used for this namespace’s storage provider.
      If you need to add a new connection, click Add New Connection and enter your provider details; see Section 10.5.1, “Adding provider connections to the Multicloud Object Gateway” for more information.
   b. In Target Bucket, select the name of the bucket to use as a target.
   c. Enter a Resource Name for your namespace resource.
   d. Click Create.
Verification

- Verify that the new resource is listed with a green check mark in the **State** column, and 0 buckets in the **Connected Namespace Buckets** column.

### 10.5.3. Adding resources to namespace buckets using the Multicloud Object Gateway

Add namespace resources to namespace buckets for a unified view of your storage across various providers. You can also configure read and write behaviour so that only one provider accepts new data, while all providers allow existing data to be read.

**Prerequisites**

- Ensure that all namespace resources you want to handle in a bucket have been added to the Multicloud Object Gateway: [Adding namespace resources using the Multicloud Object Gateway](#).

**Procedure**

1. In the OpenShift Console, click **Storage → Overview** and click the **Object** tab.
2. Click **Multicloud Object Gateway** and log in if prompted.
3. Click **Buckets**, and click on the **Namespace Buckets** tab.
4. Click **Create Namespace Bucket**
   a. On the **Choose Name** tab, specify a **Name** for the namespace bucket and click **Next**.
   b. On the **Set Placement** tab:
      i. Under **Read Policy**, select the checkbox for each namespace resource that the namespace bucket should read data from.
      ii. Under **Write Policy**, specify which namespace resource the namespace bucket should write data to.
      iii. Click **Next**.
   c. Do not make changes on the **Set Caching Policy** tab in a production environment. This tab is provided as a Development Preview and is subject to support limitations.
   d. Click **Create**.

**Verification**

- Verify that the namespace bucket is listed with a green check mark in the **State** column, the expected number of read resources, and the expected write resource name.

### 10.5.4. Amazon S3 API endpoints for objects in namespace buckets

You can interact with objects in namespace buckets using the Amazon Simple Storage Service (S3) API.

Red Hat OpenShift Container Storage 4.6 onwards supports the following namespace bucket operations:
ListObjectVersions
• ListObjects
• PutObject
• CopyObject
• ListParts
• CreateMultipartUpload
• CompleteMultipartUpload
• UploadPart
• UploadPartCopy
• AbortMultipartUpload
• GetObjectAcl
• GetObject
• HeadObject
• DeleteObject
• DeleteObjects

See the Amazon S3 API reference documentation for the most up-to-date information about these operations and how to use them.

Additional resources
• Amazon S3 REST API Reference
• Amazon S3 CLI Reference

10.5.5. Adding a namespace bucket using the Multicloud Object Gateway CLI and YAML

For more information about namespace buckets, see Managing namespace buckets.

Depending on the type of your deployment and whether you want to use YAML or the Multicloud Object Gateway CLI, choose one of the following procedures to add a namespace bucket:

• Adding an AWS S3 namespace bucket using YAML
• Adding an IBM COS namespace bucket using YAML
• Adding an AWS S3 namespace bucket using the Multicloud Object Gateway CLI
• Adding an IBM COS namespace bucket using the Multicloud Object Gateway CLI

10.5.5.1. Adding an AWS S3 namespace bucket using YAML
Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Chapter 2, Accessing the Multicloud Object Gateway with your applications

Procedure

1. Create a secret with the credentials:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: <namespacestore-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>`. ii. Replace `<namespacestore-secret-name>` with a unique name.

2. Create a NamespaceStore resource using OpenShift Custom Resource Definitions (CRDs). A NamespaceStore represents underlying storage to be used as a read or write target for the data in the Multicloud Object Gateway namespace buckets. To create a NamespaceStore resource, apply the following YAML:

   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: NamespaceStore
   metadata:
     finalizers:
     - noobaa.io/finalizer
     labels:
       app: noobaa
       name: <resource-name>
     namespace: openshift-storage
   spec:
     awsS3:
       secret:
         name: <namespacestore-secret-name>
         namespace: <namespace-secret>
         targetBucket: <target-bucket>
       type: aws-s3
   ```

   a. Replace `<resource-name>` with the name you want to give to the resource.
   b. Replace `<namespacestore-secret-name>` with the secret created in step 1.
   c. Replace `<namespace-secret>` with the namespace where the secret can be found.
   d. Replace `<target-bucket>` with the target bucket you created for the NamespaceStore.
3. Create a namespace bucket class that defines a namespace policy for the namespace buckets. The namespace policy requires a type of either single or multi.

- A namespace policy of type single requires the following configuration:

```yaml
apiversion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
    name: <my-bucket-class>
    namespace: openshift-storage
spec:
  namespacePolicy:
    type:
      single:
        resource: <resource>
```

Replace `<my-bucket-class>` with a unique namespace bucket class name.

Replace `<resource>` with the name of a single namespace-store that will define the read and write target of the namespace bucket.

- A namespace policy of type multi requires the following configuration:

```yaml
apiversion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
    name: <my-bucket-class>
    namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
        - <read-resources>
        - <read-resources>
```

Replace `<my-bucket-class>` with a unique bucket class name.

Replace `<write-resource>` with the name of a single namespace-store that will define the write target of the namespace bucket.

Replace `<read-resources>` with a list of the names of the namespace-stores that will define the read targets of the namespace bucket.

4. Apply the following YAML to create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in step 2.

```yaml
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
```
namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>

NOTE
For IBM Power Systems and IBM Z infrastructure use storageClassName as
openshift-storage.noobaa.io

a. Replace <my-bucket-class> with the bucket class created in the previous step.

Once the OBC is provisioned by the operator, a bucket is created in the Multicloud Object Gateway, and
the operator creates a Secret and ConfigMap with the same name of the OBC on the same namespace
of the OBC.

10.5.5.2. Adding an IBM COS namespace bucket using YAML

Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Chapter 2, Accessing the Multicloud Object
  Gateway with your applications

Procedure

1. Create a secret with the credentials:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-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 `<namespacestore-secret-name>` with a unique name.

2. Create a NamespaceStore resource using OpenShift Custom Resource Definitions (CRDs). A
   NamespaceStore represents underlying storage to be used as a read or write target for the data
   in the Multicloud Object Gateway namespace buckets. To create a NamespaceStore resource,
   apply the following YAML:

```yaml
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
```
metadata:
finalizers:
- noobaa.io/finalizer
labels:
  app: noobaa
  name: bs
  namespace: openshift-storage
spec:
s3Compatible:
  endpoint: <IBM COS ENDPOINT>
  secret:
    name: <namespacestore-secret-name>
    namespace: <namespace-secret>
    signatureVersion: v2
  targetBucket: <target-bucket>
type: ibm-cos

a. Replace `<IBM COS ENDPOINT>` with the appropriate IBM COS endpoint.

b. Replace `<namespacestore-secret-name>` with the secret created in step 1.

c. Replace `<namespace-secret>` with the namespace where the secret can be found.

d. Replace `<target-bucket>` with the target bucket you created for the NamespaceStore.

3. Create a namespace bucket class that defines a namespace policy for the namespace buckets. The namespace policy requires a type of either `single` or `multi`.

- A namespace policy of type `single` requires the following configuration:

  ```yaml
  apiVersion: noobaa.io/v1alpha1
  kind: BucketClass
  metadata:
    labels:
      app: noobaa
      name: <my-bucket-class>
      namespace: openshift-storage
  spec:
    namespacePolicy:
      type:
      single:
        resource: <resource>
  ```

  Replace `<my-bucket-class>` with a unique namespace bucket class name.

  Replace `<resource>` with the name of a single namespace-store that will define the read and write target of the namespace bucket.

- A namespace policy of type `multi` requires the following configuration:

  ```yaml
  apiVersion: noobaa.io/v1alpha1
  kind: BucketClass
  metadata:
    labels:
      app: noobaa
      name: <my-bucket-class>
  ```
namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>

Replace `<my-bucket-class>` with a unique bucket class name.

Replace `<write-resource>` with the name of a single namespace-store that will define the write target of the namespace bucket.

Replace `<read-resources>` with a list of the names of namespace-stores that will define the read targets of the namespace bucket.

4. Apply the following YAML to create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in step 2.

```yaml
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

**NOTE**

For IBM Power Systems and IBM Z infrastructure use `storageClassName` as `openshift-storage.noobaa.io`

a. Replace `<my-bucket-class>` with the bucket class created in the previous step.

Once the OBC is provisioned by the operator, a bucket is created in the Multicloud Object Gateway, and the operator creates a Secret and ConfigMap with the same name of the OBC on the same namespace of the OBC.

10.5.5.3. Adding an AWS S3 namespace bucket using the Multicloud Object Gateway CLI

**Prerequisites**

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Chapter 2, Accessing the Multicloud Object Gateway with your applications
- Download the Multicloud Object Gateway command-line interface:
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```shell
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

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/package](https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package).

**NOTE**

Choose the correct Product Variant according to your architecture.

**Procedure**

1. Create a NamespaceStore resource. A NamespaceStore represents an underlying storage to be used as a read or write target for the data in Multicloud Object Gateway namespace buckets. From the MCG command-line interface, run the following command:

   ```shell
   noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
   ```

   a. Replace `<namespacestore>` with the name of the NamespaceStore.

   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.

2. Create a namespace bucket class that defines a namespace policy for the namespace buckets. The namespace policy requires a type of either `single` or `multi`.

   - Run the following command to create a namespace bucket class with a namespace policy of type `single`:

     ```shell
     noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --resource <resource> -n openshift-storage
     ```

     Replace `<resource-name>` with the name you want to give the resource.

     Replace `<my-bucket-class>` with a unique bucket class name.

     Replace `<resource>` with a single namespace-store that will define the read and write target of the namespace bucket.
Run the following command to create a namespace bucket class with a namespace policy of type `multi`:

```
noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-resource <write-resource> --read-resources <read-resources> -n openshift-storage
```

Replace `<resource-name>` with the name you want to give the resource.

Replace `<my-bucket-class>` with a unique bucket class name.

Replace `<write-resource>` with a single namespace-store that will define the write target of the namespace bucket.

Replace `<read-resources>` with a list of namespace-stores separated by commas that will define the read targets of the namespace bucket.

3. Run the following command to create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in step 2.

```
noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
```

   a. Replace `<bucket-name>` with a bucket name of your choice.

   b. Replace `<custom-bucket-class>` with the name of the bucket class created in step 2.

Once the OBC is provisioned by the operator, a bucket is created in the Multicloud Object Gateway, and the operator creates a Secret and ConfigMap with the same name of the OBC on the same namespace of the OBC.

### 10.5.5.4. Adding an IBM COS namespace bucket using the Multicloud Object Gateway CLI

#### Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Chapter 2, *Accessing the Multicloud Object Gateway with your applications*
- Download the Multicloud Object Gateway command-line interface:

  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
  # yum install mcg
  ```
NOTE
Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:
  
  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
  ```

- For IBM Z infrastructure, use the following command:
  
  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

Alternatively, you can install the mcg package from the OpenShift Container Storage RPMs found here [link]

NOTE
Choose the correct Product Variant according to your architecture.

Procedure

1. Create a NamespaceStore resource. A NamespaceStore represents an underlying storage to be used as a read or write target for the data in Multicloud Object Gateway namespace buckets. From the MCG command-line interface, run the following command:

   ```bash
   noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
   ```

   a. Replace `<namespacestore>` with the name of the NamespaceStore.

   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.

   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.

2. Create a namespace bucket class that defines a namespace policy for the namespace buckets. The namespace policy requires a type of either `single` or `multi`.

   - Run the following command to create a namespace bucket class with a namespace policy of type `single`:
     
     ```bash
     noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --resource <resource> -n openshift-storage
     ```

     Replace `<resource-name>` with the name you want to give the resource.

     Replace `<my-bucket-class>` with a unique bucket class name.
Replace `<resource>` with a single namespace-store that will define the read and write target of the namespace bucket.

- Run the following command to create a namespace bucket class with a namespace policy of type `multi`:
  
  ```
  noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-resource <write-resource> --read-resources <read-resources> -n openshift-storage
  ```

  Replace `<resource-name>` with the name you want to give the resource.

  Replace `<my-bucket-class>` with a unique bucket class name.

  Replace `<write-resource>` with a single namespace-store that will define the write target of the namespace bucket.

  Replace `<read-resources>` with a list of namespace-stores separated by commas that will define the read targets of the namespace bucket.

  3. Run the following command to create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in step 2.

  ```
  noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
  ```

  a. Replace `<bucket-name>` with a bucket name of your choice.

  b. Replace `<custom-bucket-class>` with the name of the bucket class created in step 2.

Once the OBC is provisioned by the operator, a bucket is created in the Multicloud Object Gateway, and the operator creates a Secret and ConfigMap with the same name of the OBC on the same namespace of the OBC.

**10.6. 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 10.4, “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 10.6.1, “Creating bucket classes to mirror data using the MCG command-line-interface”

- Section 10.6.2, “Creating bucket classes to mirror data using a YAML”

- Section 10.6.3, “Configuring buckets to mirror data using the user interface”
10.6.1. 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 placement-bucketclass 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
   ```

10.6.2. Creating bucket classes to mirror data using a YAML

1. Apply the following YAML.

   ```
   apiVersion: noobaa.io/v1alpha1
   kind: BucketClass
   metadata:
     app: noobaa
     name: <bucket-class-name>
     namespace: openshift-storage
   spec:
     placementPolicy:
       tiers:
         - backingStores:
           - <backing-store-1>
           - <backing-store-2>
       placement: Mirror
   ```

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

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

   For more information about OBCs, see Section 10.8, “Object Bucket Claim”.

10.6.3. Configuring buckets to mirror data using the user interface

1. In your OpenShift Storage console, Click Storage → Overview → Object tab → Multicloud Object Gateway link.

2. On the NooBaa page, click the buckets icon on the left side. You will see a list of your buckets:
3. Click the bucket you want to update.

4. Click **Edit Tier 1 Resources**

5. Select **Mirror** and check the relevant resources you want to use for this bucket. In the following example, the data between `noobaa-default-backing-store` which is on RGW and `AWS-backingstore` which is on AWS is mirrored:

6. Click **Save**.

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

**10.7. 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.

10.7.1. 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](#).

10.7.2. Using bucket policies

Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see [Section 10.2, “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:

   ```json
   {
   "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 with regard to access permissions.

   For details on these elements and examples of how they can be used to control the access permissions, 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 10.7.3, “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:
Replace `ENDPOINT` with the S3 endpoint
Replace `MyBucket` with the bucket to set the policy on
Replace `BucketPolicy` with the bucket policy JSON file
Add `--no-verify-ssl` if you are using the default self signed certificates

For example:

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

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

**NOTE**

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

**NOTE**

Bucket policy conditions are not supported.

10.7.3. 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 10.2, “Accessing the Multicloud Object Gateway with your applications”

**Procedure**

1. In your OpenShift Storage console, navigate to **Storage → Overview → Object** tab → select the **Multicloud Object Gateway** link:
2. Under the **Accounts** tab, click **Create Account**

3. Select **S3 Access Only**, provide the **Account Name**, for example, **john.doe@example.com**. Click **Next**: 
4. Select S3 default placement, for example, noobaa-default-backing-store. Select Buckets Permissions. A specific bucket or all buckets can be selected. Click Create:
10.8. 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 10.8.1, "Dynamic Object Bucket Claim"
- Section 10.8.2, "Creating an Object Bucket Claim using the command line interface"
- Section 10.8.3, "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.

10.8.1. 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:

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

These lines are the Object Bucket Claim itself.

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

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

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

```yaml
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
```

Red Hat OpenShift Container Storage 4.8 Deploying and managing OpenShift Container Storage using Google Cl
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

IMPORTANT

Retrieve the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY. The names are used so that it is compatible with the AWS S3 API. You need to specify the keys while performing S3 operations, especially when you read, write or list from the Multicloud Object Gateway (MCG) bucket. The keys are encoded in Base64. Decode the keys before using them.

   # oc get secret <obc_name> -o yaml

   Specify the name of the object bucket claim.

10.8.2. 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
```

**NOTE**

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:
  
  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
  ```

- For IBM Z infrastructure, use the following command:

  ```
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

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:

```yaml
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-ocl
  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:

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

Example output:

```yaml
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVV94cmY=
  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-ocl
  noobaa-domain: openshift-storage.noobaa.io
name: test21obc
namespace: openshift-storage
ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
```
controller: true
class: 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:

```yaml
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.

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

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

**Prerequisites**

- Administrative access to the OpenShift Web Console.
In order for your applications to communicate with the OBC, you need to use the configmap and secret. For more information about this, see Section 10.8.1, “Dynamic Object Bucket Claim”.

Procedure

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

   ![Create Object Bucket Claim](image)

   **Create Object Bucket Claim**

   **Object Bucket Claim Name**
   
   my-object-bucket
   
   If not provided, a generic name will be generated.

   **Storage Class**

   ![Storage Class](image)

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

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

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

   **Internal mode**

   ![Internal mode](image)

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

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

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

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

**NOTE**

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

5. Click **Create**.
   Once you create the OBC, you are redirected to its detail page:
10.8.4. Attaching an Object Bucket Claim to a deployment

Once created, Object Bucket Claims (OBCs) can be attached to specific deployments.

Prerequisites

- Administrative access to the OpenShift Web Console.

Procedure

1. On the left navigation bar, click **Storage → Object Bucket Claims**.
2. Click the action menu (⋮) next to the OBC you created.
3. From the drop down menu, select **Attach to Deployment**
4. Select the desired deployment from the Deployment Name list, then click **Attach**.
10.8.5. Viewing object buckets using the OpenShift Web Console

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

Prerequisites

- Administrative access to the OpenShift Web Console.

Procedure

To view the object bucket details:

1. Log into the OpenShift Web Console.

2. On the left navigation bar, click **Storage → Object Buckets**:

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

3. Select the object bucket you want to see details for. You are navigated to the object bucket’s details page:
10.8.6. Deleting Object Bucket Claims

Prerequisites

- Administrative access to the OpenShift Web Console.

Procedure

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

2. Click on the action menu (⋮) next to the Object Bucket Claim you want to delete.

3. Select **Delete Object Bucket Claim** from menu.
4. Click **Delete**.

### Additional Resources

- Section 10.8, “Object Bucket Claim”

### 10.9. CACHING POLICY FOR OBJECT BUCKETS

A cache bucket is a namespace bucket with a hub target and a cache target. The hub target is an S3 compatible large object storage bucket. The cache bucket is the local Multicloud Object Gateway bucket. You can create a cache bucket that caches an AWS bucket or an IBM COS bucket.

**IMPORTANT**

Cache buckets are a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information, see [Technology Preview Features Support Scope](#).

- **AWS S3**
- **IBM COS**

### 10.9.1. Creating an AWS cache bucket

**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
  ```
NOTE

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

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/package.

NOTE

Choose the correct Product Variant according to your architecture.

Procedure

1. Create a NamespaceStore resource. A NamespaceStore represents an underlying storage to be used as a read or write target for the data in Multicloud Object Gateway namespace buckets. From the MCG command-line interface, run the following command:

```
nobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name>
```

   a. Replace `<namespacestore>` with the name of the namespacestore.

   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.

You can also add storage resources by applying a YAML. First create a secret with credentials:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-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>
```

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>`.

Replace `<namespacestore-secret-name>` with a unique name.

Then apply the following YAML:
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
labels:
  app: noobaa
name: <namespacestore>
namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
type: aws-s3

1. Replace `<namespacestore>` with a unique name.

2. Replace `<namespacestore-secret-name>` with the secret created in the previous step.

3. Replace `<namespace-secret>` with the namespace used to create the secret in the previous step.

4. Replace `<target-bucket>` with the AWS S3 bucket you created for the namespacestore.

2. Run the following command to create a bucket class:

   ```bash
   noobaa bucketclass create namespace-bucketclass cache <my-cache-bucket-class> --backingstores <backing-store> --hub-resource <namespacestore>
   ```

   a. Replace `<my-cache-bucket-class>` with a unique bucket class name.

   b. Replace `<backing-store>` with the relevant backing store. You can list one or more backing stores separated by commas in this field.

   c. Replace `<namespacestore>` with the namespacestore created in the previous step.

3. Run the following command to create a bucket using an Object Bucket Claim resource that uses the bucket class defined in step 2.

   ```bash
   noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
   ```

   a. Replace `<my-bucket-claim>` with a unique name.

   b. Replace `<custom-bucket-class>` with the name of the bucket class created in step 2.

10.9.2. Creating an IBM COS cache bucket

**Prerequisites**

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

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

Specify the appropriate architecture for enabling the repositories using subscription manager. For instance,

- For IBM Power Systems, use the following command:

  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
  ```

- For IBM Z infrastructure, use the following command:

  ```bash
  # subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
  ```

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/package.

NOTE

Choose the correct Product Variant according to your architecture.

Procedure

1. Create a NamespaceStore resource. A NamespaceStore represents an underlying storage to be used as a read or write target for the data in Multicloud Object Gateway namespace buckets. From the MCG command-line interface, run the following command:

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

   a. Replace `<namespacestore>` with the name of the NamespaceStore.

   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.

   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. You can also add storage resources by applying a YAML. First, Create a secret with the credentials:

   ```yaml
   apiVersion: v1
   kind: Secret
   metadata:
     name: <namespacestore-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>
   ```
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>`.

Replace `<namespacestore-secret-name>` with a unique name.

Then apply the following YAML:

```yaml
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
    name: <namespacestore>
    namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
  secret:
    name: <backingstore-secret-name>
    namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

d. Replace `<namespacestore>` with a unique name.

e. Replace `<IBM COS ENDPOINT>` with the appropriate IBM COS endpoint.

f. Replace `<backingstore-secret-name>` with the secret created in the previous step.

g. Replace `<namespace-secret>` with the namespace used to create the secret in the previous step.

h. Replace `<target-bucket>` with the AWS S3 bucket you created for the namespacestore.

2. Run the following command to create a bucket class:

```
noobaa bucketclass create namespace-bucketclass cache <my-bucket-class> --backingstores <backing-store> --hubResource <namespacestore>
```

a. Replace `<my-bucket-class>` with a unique bucket class name.

b. Replace `<backing-store>` with the relevant backing store. You can list one or more backingstores separated by commas in this field.

c. Replace `<namespacestore>` with the namespacestore created in the previous step.

3. Run the following command to create a bucket using an Object Bucket Claim resource that uses the bucket class defined in step 2.

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. Replace `<my-bucket-claim>` with a unique name.
b. Replace `<custom-bucket-class>` with the name of the bucket class created in step 2.

10.10. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS

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

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

- Storage service
- S3 endpoint service

10.10.1. S3 endpoints in the Multicloud Object Gateway

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

10.10.2. Scaling with storage nodes

Prerequisites

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

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

Procedure

1. In the Multicloud Object Gateway user interface, from the Overview page, click Add Storage Resources:
2. In the window, click **Deploy Kubernetes Pool**

3. In the **Create Pool** step create the target pool for the future installed nodes.
4. In the **Configure** step, configure the number of requested pods and the size of each PV. For each new pod, one PV is be created.

5. In the **Review** step, you can find the details of the new pool and select the deployment method you wish to use: local or external deployment. If local deployment is selected, the Kubernetes nodes will deploy within the cluster. If external deployment is selected, you will be provided with a YAML file to run externally.
6. All nodes will be assigned to the pool you chose in the first step, and can be found under Resources → Storage resources → Resource name:

![Resource Management Screen]

10.11. AUTOMATIC SCALING OF MULTICLOUD OBJECT GATEWAY ENDPOINTS

The number of MultiCloud Object Gateway (MCG) endpoints scale automatically when the load on the MCG S3 service increases or decreases. {product-name-short} clusters are deployed with one active MCG endpoint. Each MCG endpoint pod is configured by default with 1 CPU and 2Gi memory request, with limits matching the request. When the CPU load on the endpoint crosses over an 80% usage threshold for a consistent period of time, a second endpoint is deployed lowering the load on the first endpoint. When the average CPU load on both endpoints falls below the 80% threshold for a consistent period of time, one of the endpoints is deleted. This feature improves performance and serviceability of the MCG.
11.1. 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. For Rados Block Device (RBD), if the Access mode is ReadWriteOnce (RWO), select the required **Volume mode**. The default volume mode is **Filesystem**.
      
      v. Specify a **Size** as per application requirement.
      
      vi. **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.

11.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS

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

Prerequisites
Administrator access to OpenShift Container Storage.

Procedure

1. Log in to OpenShift Web Console.
2. Click Storage → Persistent Volume Claims
3. Search for the required PVC name by using the Filter textbox. You can also filter the list of PVCs by Name or Label to narrow down the list
4. Check the Status column corresponding to the required PVC.
5. Click the required Name to view the PVC details.

11.3. 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 Storage → Overview → Block and File
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.

11.4. DYNAMIC PROVISIONING

11.4.1. 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.

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

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

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

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

11.4.3. 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:

<table>
<thead>
<tr>
<th>Storage type</th>
<th>Provisioner plug-in name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Cinder</td>
<td>kubernetes.io/cinder</td>
<td></td>
</tr>
</tbody>
</table>
### CHAPTER 11. MANAGING PERSISTENT VOLUME CLAIMS

<table>
<thead>
<tr>
<th>Storage type</th>
<th>Provisioner plug-in name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Elastic Block Store (EBS)</td>
<td>kubernetes.io/aws-ebs</td>
<td>For dynamic provisioning when using multiple clusters in different zones, tag each node with <code>Key=kubernetes.io/cluster/&lt;cluster_name&gt;</code>, <code>Value=&lt;cluster_id&gt;</code> where <code>&lt;cluster_name&gt;</code> and <code>&lt;cluster_id&gt;</code> are unique per cluster.</td>
</tr>
<tr>
<td>AWS Elastic File System (EFS)</td>
<td></td>
<td>Dynamic provisioning is accomplished through the EFS provisioner pod and not through a provisioner plug-in.</td>
</tr>
<tr>
<td>Azure Disk</td>
<td>kubernetes.io/azure-disk</td>
<td></td>
</tr>
<tr>
<td>Azure File</td>
<td>kubernetes.io/azure-file</td>
<td>The <code>persistent-volume-binder</code> ServiceAccount requires permissions to create and get Secrets to store the Azure storage account and keys.</td>
</tr>
<tr>
<td>GCE Persistent Disk (gcePD)</td>
<td>kubernetes.io/gce-pd</td>
<td>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.</td>
</tr>
<tr>
<td>VMware vSphere</td>
<td>kubernetes.io/vsphere-volume</td>
<td></td>
</tr>
<tr>
<td>Red Hat Virtualization</td>
<td>csi.ovirt.org</td>
<td></td>
</tr>
</tbody>
</table>

**IMPORTANT**

Any chosen provisioner plug-in also requires configuration for the relevant cloud, host, or third-party provider as per the relevant documentation.
CHAPTER 12. VOLUME SNAPSHOTS

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

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

**NOTE**

You cannot schedule periodic creation of snapshots.

12.1. CREATING VOLUME SNAPSHOTS

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

**Prerequisites**

- For a consistent snapshot, the PVC should be in **Bound** state and not be in use. Ensure to stop all IO before taking the snapshot.

**NOTE**

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

**Procedure**

**From the Persistent Volume Claims page**

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

2. To create a volume snapshot, do one of the following:
   - Beside the desired PVC, click Action menu (⋯) → **Create Snapshot**
   - Click on the PVC for which you want to create the snapshot and click **Actions → Create Snapshot**.

3. Enter a **Name** for the volume snapshot.

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

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

**From the Volume Snapshots page**

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

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

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

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

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

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

**Verification steps**

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

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

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

### 12.2. RESTORING VOLUME SNAPSHOTs

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

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

**Procedure**

**From the Persistent Volume Claims page**

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

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

2. Click on the PVC name with the volume snapshot to restore a volume snapshot as a new PVC.

3. In the **Volume Snapshots** tab, click the Action menu ( ⋮ ) next to the volume snapshot you want to restore.

4. Click **Restore as new PVC**

5. Enter a name for the new PVC.

6. Select the **Storage Class** name.
For Rados Block Device (RBD), you must select a storage class with the same pool as that of the parent PVC. Restoring the snapshot of an encrypted PVC using a storage class where encryption is not enabled and vice versa is not supported.

7. Select the **Access Mode** of your choice.

**IMPORTANT**

The ReadOnlyMany (ROX) access mode is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with ReadOnlyMany feature, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on availability and work schedules. See Creating a clone or restoring a snapshot with the new readonly access mode to use the ROX access mode.

8. (Optional) For RBD, select **Volume mode**.

9. Click **Restore**. You are redirected to the new PVC details page.

**From the Volume Snapshots page**

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

2. In the **Volume Snapshots** tab, click the Action menu (⋮) next to the volume snapshot you want to restore.

3. Click **Restore as new PVC**.

4. Enter a name for the new PVC.

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

**NOTE**

For Rados Block Device (RBD), you must select a storage class with the same pool as that of the parent PVC. Restoring the snapshot of an encrypted PVC using a storage class where encryption is not enabled and vice versa is not supported.

6. Select the **Access Mode** of your choice.
IMPORTANT

The ReadOnlyMany (ROX) access mode is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with ReadOnlyMany feature, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on availability and work schedules. See Creating a clone or restoring a snapshot with the new readonly access mode to use the ROX access mode.

7. (Optional) For RBD, select Volume mode.

8. Click Restore. You are redirected to the new PVC details page.

Verification steps

- Click Storage → Persistent Volume Claims from the OpenShift Web Console and confirm that the new PVC is listed in the Persistent Volume Claims page.

- Wait for the new PVC to reach Bound state.

12.3. DELETING VOLUME SNAPSHOTS

Prerequisites

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

Procedure

From Persistent Volume Claims page

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

2. Click on the PVC name which has the volume snapshot that needs to be deleted.

3. In the Volume Snapshots tab, beside the desired volume snapshot, click Action menu (⋯) → Delete Volume Snapshot

From Volume Snapshots page

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

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

Verification steps

- Ensure that the deleted volume snapshot is not present in the Volume Snapshots tab of the PVC details page.
- Click **Storage → Volume Snapshots** and ensure that the deleted volume snapshot is not listed.
CHAPTER 13. VOLUME CLONING

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

13.1. CREATING A CLONE

Prerequisites

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

**NOTE**

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

Procedure

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

2. To create a clone, do one of the following:
   - Beside the desired PVC, click Action menu (⋮) → **Clone PVC**.
   - Click on the PVC that you want to clone and click **Actions → Clone PVC**.

3. Enter a **Name** for the clone.

4. Select the access mode of your choice.

**IMPORTANT**

The ReadOnlyMany (ROX) access mode is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with ReadOnlyMany feature, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on availability and work schedules. See [Creating a clone or restoring a snapshot with the new readonly access mode](#) to use the ROX access mode.

5. Click **Clone**. You are redirected to the new PVC details page.

6. Wait for the cloned PVC status to become **Bound**. The cloned PVC is now available to be consumed by the pods. This cloned PVC is independent of its dataSource PVC.
CHAPTER 14. REPLACING STORAGE NODES

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

- Section 14.1, “Replacing operational nodes on Google Cloud installer-provisioned infrastructure”
- Section 14.2, “Replacing failed nodes on Google Cloud installer-provisioned infrastructure”

14.1. REPLACING OPERATIONAL NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE

Use this procedure to replace an operational node on Google Cloud 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:
  
  ```bash
  $ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
  ```

**Verification steps**

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

   ```bash
   $ 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. Verify that new OSD pods are running on the replacement node.

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

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

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

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

      ```bash
      $ lsblk
      ```

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

### 14.2. REPLACING FAILED NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE

Perform this procedure to replace a failed node which is not operational on Google Cloud 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 the web user interface**
    a. For the new node, click **Action Menu (⋮) → Edit Labels**
    b. Add `cluster.ocs.openshift.io/openshift-storage` and click **Save**.

    **From the command line interface**
    
    - Execute the following command to apply the OpenShift Container Storage label to the new node:
      
      ```bash
      $ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
      ```

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

**Verification steps**

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

   ```bash
   $ 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. Verify that new OSD pods are running on the replacement node.

   ```bash
   $ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
   ```
5. (Optional) If cluster-wide encryption is enabled on the cluster, verify that the new OSD devices are encrypted.
   For each of the new nodes identified in previous step, do the following:
   
   a. Create a debug pod and open a chroot environment for the selected host(s).
      
      $ oc debug node/<node name>
      $ chroot /host
   
   b. Run "lsblk" and check for the "crypt" keyword beside the `ocs-deviceset` name(s)
      
      $ lsblk
   
6. If verification steps fail, contact Red Hat Support.
CHAPTER 15. REPLACING STORAGE DEVICES

15.1. REPLACING OPERATIONAL OR FAILED STORAGE DEVICES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE

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

- Replacing operational nodes on Google Cloud installer-provisioned infrastructure
- Replacing failed nodes on Google Cloud installer-provisioned infrastructures.
CHAPTER 16. UPDATING OPENSHIFT CONTAINER STORAGE

16.1. OVERVIEW OF THE OPENSHIFT CONTAINER STORAGE UPDATE PROCESS

You can upgrade Red Hat OpenShift Container Storage and its components, either between minor releases like 4.7 and 4.8, or between batch updates like 4.8.0 and 4.8.1.

You need to upgrade the different parts of OpenShift Container Storage in a specific order.

1. Update OpenShift Container Platform according to the Updating clusters documentation for OpenShift Container Platform.

2. Update OpenShift Container Storage.
   a. To prepare a disconnected environment for updates see Operators guide to using Operator Lifecycle Manager on restricted networks to be able to update OpenShift Container Storage as well as Local Storage Operator when in use.
   b. Update the OpenShift Container Storage operator using the appropriate process for your setup:
      - Update OpenShift Container Storage in internal mode

Update considerations

Review the following important considerations before you begin.

- Red Hat recommends using the same version of Red Hat OpenShift Container Platform with Red Hat OpenShift Container Storage.
  See the Interoperability Matrix for more information about supported combinations of OpenShift Container Platform and OpenShift Container Storage.

- The Local Storage Operator is fully supported only when the Local Storage Operator version matches the Red Hat OpenShift Container Platform version.

16.2. PREPARING TO UPDATE IN A DISCONNECTED ENVIRONMENT

When your Red Hat OpenShift Container Storage environment is not directly connected to the internet, some additional configuration is required to provide the Operator Lifecycle Manager (OLM) with alternatives to the default Operator Hub and image registries.

See the OpenShift Container Platform documentation for more general information: Updating an Operator catalog image.

To configure your cluster for disconnected update:

1. Configure authentication for an alternative registry.
2. Build and mirror the Red Hat operator catalog.
3. Creating Operator imageContentSourcePolicy
4. Updating redhat-operator catalogssource

When these steps are complete, Continue with update as usual.
16.2.1. Adding mirror registry authentication details

Prerequisites

- Verify that your existing disconnected cluster uses OpenShift Container Platform 4.3 or higher.
- Verify that you have an oc client version of 4.4 or higher.
- Prepare a mirror host with a mirror registry. See Preparing your mirror host for details.

Procedure

1. Log in to the OpenShift Container Platform cluster using the cluster-admin role.

2. Locate your auth.json file.
   This file is generated when you use podman or docker to log in to a registry. It is located in one of the following locations:
   - ~/.docker/auth.json
   - /run/user/<UID>/containers/auth.json
   - /var/run/containers/<UID>/auth.json

3. Obtain your unique Red Hat registry pull secret and paste it into your auth.json file. It will look something like this.

   ```
   {
       "auths": {
           "cloud.openshift.com": {
               "auth": "**************",
               "email": "user@example.com"
           },
           "quay.io": {
               "auth": "**************",
               "email": "user@example.com"
           },
           "registry.connect.redhat.com": {
               "auth": "**************",
               "email": "user@example.com"
           },
           "registry.redhat.io": {
               "auth": "**************",
               "email": "user@example.com"
           }
       }
   }
   ```

4. Export environment variables with the appropriate details for your setup.

   ```
   $ export AUTH_FILE="<location_of_auth.json>"
   $ export MIRROR_REGISTRY_DNS="<your_registry_url>:<port>"
   ```

5. Use podman to log in to the mirror registry and store the credentials in the $AUTH_FILE.
$ podman login ${MIRROR_REGISTRY_DNS} --tls-verify=false --authfile ${AUTH_FILE}

This adds the mirror registry to the auth.json file.

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "***************",
      "email": "user@example.com"
    },
    "quay.io": {
      "auth": "***************",
      "email": "user@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "***************",
      "email": "user@example.com"
    },
    "registry.redhat.io": {
      "auth": "***************",
      "email": "user@example.com"
    },
    "<mirror_registry>": {
      "auth": "***************"
    }
  }
}
```

### 16.2.2. Building and mirroring the Red Hat operator catalog

Follow this process on a host that has access to Red Hat registries to create a mirror of those registries.

**Prerequisites**

- Run these commands as a cluster administrator.
- Be aware that mirroring the redhat-operator catalog can take hours to complete, and requires substantial available disk space on the mirror host.

**Procedure**

1. Build the catalog for redhat-operators.
   
   Set --from to the ose-operator-registry base image using the tag that matches the target OpenShift Container Platform cluster major and minor version.

   ```bash
   $ oc adm catalog build --appregistry-org redhat-operators \
   --from=registry.redhat.io/openshift4/ose-operator-registry:v4.7 \
   --to=${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
   --registry-config=${AUTH_FILE} \
   --filter-by-os="linux/amd64" --insecure
   ```
NOTE

For IBM Power Systems and IBM Z infrastructure specify value of `filter-by-os` as `linux/ppc64le`, and `linux/s390x` respectively.

2. Mirror the catalog for **redhat-operators**.
   This is a long operation and can take 1-5 hours. Make sure there is 100 GB available disk space on the mirror host.

   ```bash
   $ oc adm catalog mirror ${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2
   ${MIRROR_REGISTRY_DNS} --registry-config=${AUTH_FILE} --insecure
   ```

16.2.3. Creating Operator **imageContentSourcePolicy**

After the `oc adm catalog mirror` command is completed, the `imageContentSourcePolicy.yaml` file gets created. The output directory for this file is usually, `./[catalog image name]-manifests). Use this procedure to add any missing entries to the `.yaml` file and apply them to cluster.

**Procedure**

1. Check the content of this file for the mirrors mapping shown as follows:

   ```yaml
   spec:
   repositoryDigestMirrors:
   - mirrors:
     - <your_registry>/ocs4
       source: registry.redhat.io/ocs4
     - mirrors:
       - <your_registry>/rhceph
         source: registry.redhat.io/rhceph
     - mirrors:
       - <your_registry>/openshift4
         source: registry.redhat.io/openshift4
     - mirrors:
       - <your_registry>/rhscl
         source: registry.redhat.io/rhscl
   ```

2. Add any missing entries to the end of the `imageContentSourcePolicy.yaml` file.

3. Apply the `imageContentSourcePolicy.yaml` file to the cluster.

   ```bash
   $ oc apply -f ./[output dir]/imageContentSourcePolicy.yaml
   ```

Once the Image Content Source Policy is updated, all the nodes (master, infra, and workers) in the cluster need to be updated and rebooted. This process is automatically handled through the Machine Config Pool operator and take up to 30 minutes although the exact elapsed time might vary based on the number of nodes in your OpenShift cluster. You can monitor the update process by using the `oc get mcp` command or the `oc get node` command.

16.2.4. Updating **redhat-operator** CatalogSource

**Procedure**
1. Recreate a **CatalogSource** object that references the catalog image for Red Hat operators.

   **NOTE**

   Make sure you have mirrored the correct catalog source with the correct version (that is, v2).

   Save the following in a **redhat-operator-catalogsource.yaml** file, remembering to replace `<your_registry>` with your mirror registry URL:

   ```yaml
   apiVersion: operators.coreos.com/v1alpha1
   kind: CatalogSource
   metadata:
     name: redhat-operators
     namespace: openshift-marketplace
   spec:
     sourceType: grpc
     icon:
       base64data:
       PHN2ZyBpZD0iTGF5ZXJfMSIgZGF0YS1uYW1lPSJMYXllciAxIiB4bWxucz0iaHR0cDovL3d3dy55My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAxOTIgMTQ1Ij48ZGVmcz48c3R5bGU+LmNscy0xe2ZzbGw6I2UwMjI4IiBzdmcnIHZlcnNvbiBvZmZzZXQ9IjAgIj48ZTV5PSJhY2VudHJ5PSJhY2VudHJ5IiBvbmx5PSIiIj48cGF0YTI8L3N2Zz4=
       mediatype: image/svg+xml
     image: <your_registry>/olm/redhat-operators:v2
     displayName: Redhat Operators Catalog
     publisher: Red Hat
   ```

2. Create a catalogsource using the redhat-operator-catalogsource.yaml file:

   ```bash
   $ oc apply -f redhat-operator-catalogsource.yaml
   ```

3. Verify that the new **redhat-operator** pod is running.

   ```bash
   $ oc get pod -n openshift-marketplace | grep redhat-operators
   ```

16.2.5. Continue to update
After your alternative catalog source is configured, you can continue to the appropriate update process:

- Updating OpenShift Container Storage in internal mode

16.3. UPDATING OPENSSHIFT CONTAINER STORAGE IN INTERNAL MODE

Use the following procedures to update your OpenShift Container Storage cluster deployed in internal mode.

16.3.1. Enabling automatic updates for OpenShift Container Storage operator in internal mode

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

Prerequisites

- Under Block and File in the Status card, confirm that the Storage Cluster and Data Resiliency has a green tick mark.
- Under Object in the Status card, confirm that both Object Service and Data Resiliency are in Ready state (green tick).
- Update the OpenShift Container Platform cluster to the latest stable release of version 4.8.X, see Updating Clusters.
- Switch the Red Hat OpenShift Container Storage channel from stable-4.7 to stable-4.8. For details about channels, see OpenShift Container Storage upgrade channels and releases.

**NOTE**

You are required to switch channels only when you are updating minor versions (for example, updating from 4.7 to 4.8) and not when updating between batch updates of 4.8 (for example, updating from 4.8.0 to 4.8.1).

- Ensure that all OpenShift Container Storage Pods, including the operator pods, are in Running state in the openshift-storage namespace.
  To view the state of the pods, click Workloads → Pods from the left pane of the OpenShift Web Console. Select openshift-storage from the Project drop down list.
- Ensure that you have sufficient time to complete the Openshift Container Storage update process, as the update time varies depending on the number of OSDs that run in the cluster.

Procedure

1. Log in to OpenShift Web Console.
2. Click Operators → Installed Operators
3. Select the openshift-storage project.
4. Click the OpenShift Container Storage operator name.
5. Click the Subscription tab and click the link under Approval.

6. Select Automatic (default) and click Save.

7. Perform one of the following depending on the Upgrade Status:

   - **Upgrade Status shows requires approval**

     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. On the OpenShift Web Console, navigate to Storage → Overview → Object tab.
   - In the Status card, verify that both Object Service and Data Resiliency are in Ready state (green tick).

2. On the OpenShift Web Console, navigate to Storage → Overview → Block and File tab.
   - In the Status card, verify that the Storage Cluster and Data Resiliency has a green tick mark.

3. Click Operators → Installed Operators → OpenShift Container Storage Operator. Under Storage Cluster, verify that the cluster service status is Ready.

**NOTE**

Once updated from OpenShift Container Storage version 4.7 to 4.8, the Version field here will still display 4.7. This is because the ocs-operator does not update the string represented in this field.
4. Ensure that all OpenShift Container Storage Pods, including the operator pods, are in Running state in the **openshift-storage namespace**.
To view the state of the pods, click **Workloads → Pods**. Select **openshift-storage** from the **Project** drop down list.

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

**NOTE**
The flexible scaling feature is available only in the new deployments of Red Hat OpenShift Container Storage 4.7. Storage clusters upgraded to the 4.7 version do not support flexible scaling.

**Additional Resources**
If you face any issues while updating OpenShift Container Storage, see the **Commonly required logs for troubleshooting** section in the **Troubleshooting guide**.

**16.3.2. Manually updating OpenShift Container Storage operator in internal mode**

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

**Prerequisites**
- Under **Block and File** in the **Status** card, confirm that the **Storage Cluster** and **Data Resiliency** has a green tick mark.
- Under **Object** in the **Status** card, confirm that both **Object Service** and **Data Resiliency** are in **Ready** state (green tick).
- Update the OpenShift Container Platform cluster to the latest stable release of version 4.8.X, see **Updating Clusters**.
- Switch the Red Hat OpenShift Container Storage channel from **stable-4.7** to **stable-4.8**. For details about channels, see **OpenShift Container Storage upgrade channels and releases**.

**NOTE**
You are required to switch channels only when you are updating minor versions (for example, updating from 4.7 to 4.8) and not when updating between batch updates of 4.8 (for example, updating from 4.8.0 to 4.8.1).

- Ensure that all OpenShift Container Storage Pods, including the operator pods, are in Running state in the **openshift-storage namespace**.
  To view the state of the pods, click **Workloads → Pods** from the left pane of the OpenShift Web Console. Select **openshift-storage** from the **Project** drop down list.
- Ensure that you have sufficient time to complete the OpenShift Container Storage update process, as the update time varies depending on the number of OSDs that run in the cluster.

**Procedure**
1. Log in to OpenShift Web Console.
2. Click **Operators → Installed Operators**

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

4. Click the **OpenShift Container Storage** operator name.

5. Click the **Subscription** tab and click the link under **Approval**.

6. Select **Manual** and click **Save**.

7. Wait for the **Upgrade Status** to change to **Upgrading**.

8. If the **Upgrade Status** shows **requires approval**, click on **requires approval**.

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

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

11. Wait for the **Status** to change from **Unknown** to **Created**.

12. Click **Operators → Installed Operators**

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

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

**Verification steps**

1. On the OpenShift Web Console, navigate to **Storage → Overview → Object** tab.
   - In the **Status** card, verify that both **Object Service** and **Data Resiliency** are in **Ready** state (green tick).

2. On the OpenShift Web Console, navigate to **Storage → Overview → Block and File** tab.
   - In the **Status** card, verify that the **Storage Cluster** and **Data Resiliency** has a green tick mark.

3. Click **Operators → Installed Operators → OpenShift Container Storage Operator**. Under **Storage Cluster**, verify that the cluster service status is **Ready**.

   **NOTE**
   
   Once updated from OpenShift Container Storage version 4.7 to 4.8, the **Version** field here will still display 4.7. This is because the **ocs-operator** does not update the string represented in this field.

4. Ensure that all OpenShift Container Storage Pods, including the operator pods, are in **Running** state in the **openshift-storage namespace**.
   
   To view the state of the pods, click **Workloads → Pods** from the left pane of the OpenShift Web Console. Select **openshift-storage** from the **Project** drop down list.

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

**Additional Resources**

If you face any issues while updating OpenShift Container Storage, see the **Commonly required logs for troubleshooting** section in the **Troubleshooting guide**.