



# Red Hat OpenShift Data Foundation 4.11

## Deploying and managing OpenShift Data Foundation using Google Cloud

Instructions on deploying and managing OpenShift Data Foundation on existing Red Hat OpenShift Container Platform Google Cloud clusters



## Red Hat OpenShift Data Foundation 4.11 Deploying and managing OpenShift Data Foundation using Google Cloud

---

Instructions on deploying and managing OpenShift Data Foundation on existing Red Hat OpenShift Container Platform Google Cloud clusters

## Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## Abstract

Read this document for instructions about how to install and manage Red Hat OpenShift Data Foundation using Red Hat OpenShift Container Platform on Google Cloud. Deploying and managing OpenShift Data Foundation 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

<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>5</b>
<b>PROVIDING FEEDBACK ON RED HAT DOCUMENTATION</b> .....	<b>6</b>
<b>PREFACE</b> .....	<b>7</b>
<b>CHAPTER 1. PREPARING TO DEPLOY OPENSIFT DATA FOUNDATION</b> .....	<b>8</b>
<b>CHAPTER 2. DEPLOYING OPENSIFT DATA FOUNDATION ON GOOGLE CLOUD</b> .....	<b>9</b>
2.1. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR	9
2.2. ENABLING CLUSTER-WIDE ENCRYPTION WITH KMS USING THE TOKEN AUTHENTICATION METHOD	11
2.3. ENABLING CLUSTER-WIDE ENCRYPTION WITH KMS USING THE KUBERNETES AUTHENTICATION METHOD	11
2.4. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER	14
2.5. VERIFYING OPENSIFT DATA FOUNDATION DEPLOYMENT	17
2.5.1. Verifying the state of the pods	17
2.5.2. Verifying the OpenShift Data Foundation cluster is healthy	19
2.5.3. Verifying the Multicloud Object Gateway is healthy	19
2.5.4. Verifying that the OpenShift Data Foundation specific storage classes exist	19
<b>CHAPTER 3. DEPLOY STANDALONE MULTICLOUD OBJECT GATEWAY</b> .....	<b>20</b>
3.1. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR	20
3.2. CREATING A STANDALONE MULTICLOUD OBJECT GATEWAY	21
<b>CHAPTER 4. UNINSTALLING OPENSIFT DATA FOUNDATION</b> .....	<b>24</b>
4.1. UNINSTALLING OPENSIFT DATA FOUNDATION IN INTERNAL MODE	24
<b>CHAPTER 5. STORAGE CLASSES AND STORAGE POOLS</b> .....	<b>25</b>
5.1. CREATING STORAGE CLASSES AND POOLS	25
5.2. CREATING A STORAGE CLASS FOR PERSISTENT VOLUME ENCRYPTION	26
<b>CHAPTER 6. CONFIGURE STORAGE FOR OPENSIFT CONTAINER PLATFORM SERVICES</b> .....	<b>29</b>
6.1. CONFIGURING IMAGE REGISTRY TO USE OPENSIFT DATA FOUNDATION	29
6.1.1. Configuring Multicloud Object Gateway as backend storage for OpenShift image registry	30
6.1.2. Configuring OpenShift Data Foundation CephFS as backend storage for OpenShift image registry	32
6.2. CONFIGURING MONITORING TO USE OPENSIFT DATA FOUNDATION	34
6.3. CLUSTER LOGGING FOR OPENSIFT DATA FOUNDATION	36
6.3.1. Configuring persistent storage	37
6.3.2. Configuring cluster logging to use OpenShift data Foundation	38
<b>CHAPTER 7. BACKING OPENSIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSIFT DATA FOUNDATION</b> .....	<b>41</b>
<b>CHAPTER 8. HOW TO USE DEDICATED WORKER NODES FOR RED HAT OPENSIFT DATA FOUNDATION</b> .	<b>43</b>
8.1. ANATOMY OF AN INFRASTRUCTURE NODE	43
8.2. MACHINE SETS FOR CREATING INFRASTRUCTURE NODES	43
8.3. MANUAL CREATION OF INFRASTRUCTURE NODES	44
8.4. TAINT A NODE FROM THE USER INTERFACE	45
<b>CHAPTER 9. SCALING STORAGE NODES</b> .....	<b>46</b>
9.1. REQUIREMENTS FOR SCALING STORAGE NODES	46
9.2. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSIFT DATA FOUNDATION NODES ON GOOGLE CLOUD INFRASTRUCTURE	46

9.3. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES	49
9.3.1. Adding a node to an installer-provisioned infrastructure	49
9.3.2. Scaling up storage capacity	50
<b>CHAPTER 10. MULTICLOUD OBJECT GATEWAY</b>	<b>51</b>
10.1. ABOUT THE MULTICLOUD OBJECT GATEWAY	51
10.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS	51
10.2.1. Accessing the Multicloud Object Gateway from the terminal	52
10.2.2. Accessing the Multicloud Object Gateway from the MCG command-line interface	54
10.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD	57
10.3.1. Creating a new backing store	57
10.3.2. Adding storage resources for hybrid or Multicloud using the MCG command line interface	58
10.3.2.1. Creating an AWS-backed backingstore	58
10.3.2.2. Creating an IBM COS-backed backingstore	60
10.3.2.3. Creating an Azure-backed backingstore	62
10.3.2.4. Creating a GCP-backed backingstore	64
10.3.2.5. Creating a local Persistent Volume-backed backingstore	66
10.3.3. Creating an s3 compatible Multicloud Object Gateway backingstore	68
10.3.4. Adding storage resources for hybrid and Multicloud using the user interface	69
10.3.5. Creating a new bucket class	71
10.3.6. Editing a bucket class	72
10.3.7. Editing backing stores for bucket class	72
10.4. MANAGING NAMESPACE BUCKETS	73
10.4.1. Amazon S3 API endpoints for objects in namespace buckets	74
10.4.2. Adding a namespace bucket using the Multicloud Object Gateway CLI and YAML	74
10.4.2.1. Adding an AWS S3 namespace bucket using YAML	75
10.4.2.2. Adding an IBM COS namespace bucket using YAML	77
10.4.2.3. Adding an AWS S3 namespace bucket using the Multicloud Object Gateway CLI	80
10.4.2.4. Adding an IBM COS namespace bucket using the Multicloud Object Gateway CLI	82
10.4.3. Adding a namespace bucket using the OpenShift Container Platform user interface	84
10.5. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS	85
10.5.1. Creating bucket classes to mirror data using the MCG command-line-interface	86
10.5.2. Creating bucket classes to mirror data using a YAML	86
10.6. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY	87
10.6.1. Introduction to bucket policies	87
10.6.2. Using bucket policies in Multicloud Object Gateway	87
10.6.3. Creating a user in the Multicloud Object Gateway	88
10.7. OBJECT BUCKET CLAIM	89
10.7.1. Dynamic Object Bucket Claim	90
10.7.2. Creating an Object Bucket Claim using the command line interface	92
10.7.3. Creating an Object Bucket Claim using the OpenShift Web Console	95
10.7.4. Attaching an Object Bucket Claim to a deployment	96
10.7.5. Viewing object buckets using the OpenShift Web Console	96
10.7.6. Deleting Object Bucket Claims	97
10.8. CACHING POLICY FOR OBJECT BUCKETS	97
10.8.1. Creating an AWS cache bucket	97
10.8.2. Creating an IBM COS cache bucket	99
10.9. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS	102
10.9.1. Scaling the Multicloud Object Gateway with storage nodes	102
10.10. AUTOMATIC SCALING OF MULTICLOUD OBJECT GATEWAY ENDPOINTS	102
<b>CHAPTER 11. MANAGING PERSISTENT VOLUME CLAIMS</b>	<b>104</b>
11.1. CONFIGURING APPLICATION PODS TO USE OPENSIFT DATA FOUNDATION	104

---

11.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS	106
11.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS	106
11.4. DYNAMIC PROVISIONING	106
11.4.1. About dynamic provisioning	106
11.4.2. Dynamic provisioning in OpenShift Data Foundation	107
11.4.3. Available dynamic provisioning plug-ins	107
<b>CHAPTER 12. VOLUME SNAPSHOTS</b>	<b>109</b>
12.1. CREATING VOLUME SNAPSHOTS	109
12.2. RESTORING VOLUME SNAPSHOTS	110
12.3. DELETING VOLUME SNAPSHOTS	112
<b>CHAPTER 13. VOLUME CLONING</b>	<b>114</b>
13.1. CREATING A CLONE	114
<b>CHAPTER 14. REPLACING STORAGE NODES</b>	<b>115</b>
14.1. REPLACING OPERATIONAL NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE	115
14.2. REPLACING FAILED NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE	116
<b>CHAPTER 15. REPLACING STORAGE DEVICES</b>	<b>119</b>
15.1. REPLACING OPERATIONAL OR FAILED STORAGE DEVICES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE	119
<b>CHAPTER 16. UPGRADING TO OPENSIFT DATA FOUNDATION</b>	<b>120</b>
16.1. OVERVIEW OF THE OPENSIFT DATA FOUNDATION UPDATE PROCESS	120
16.2. UPDATING RED HAT OPENSIFT DATA FOUNDATION 4.10 TO 4.11	121
16.3. UPDATING RED HAT OPENSIFT DATA FOUNDATION 4.11.X TO 4.11.Y	122
16.4. CHANGING THE UPDATE APPROVAL STRATEGY	124





## MAKING OPEN SOURCE MORE INCLUSIVE

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, create a Bugzilla ticket:

1. Go to the [Bugzilla](#) website.
2. In the **Component** section, choose **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**.

---

## PREFACE

Red Hat OpenShift Data Foundation supports deployment on existing Red Hat OpenShift Container Platform (RHOCP) Google Cloud clusters.



### NOTE

Only internal OpenShift Data Foundation clusters are supported on Google Cloud. See [Planning your deployment](#) for more information about deployment requirements.

To deploy OpenShift Data Foundation in internal mode, start with the requirements in [Preparing to deploy OpenShift Data Foundation](#) chapter and follow the appropriate deployment process based on your requirement:

- [Deploy OpenShift Data Foundation on Google Cloud](#)
- [Deploy standalone Multicloud Object Gateway component](#)

# CHAPTER 1. PREPARING TO DEPLOY OPENSIFT DATA FOUNDATION

Deploying OpenShift Data Foundation 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 Data Foundation, follow these steps:

1. Optional: If you want to enable cluster-wide encryption using an external Key Management System (KMS) then follow the steps:
  - Ensure that you have a valid Red Hat OpenShift Data Foundation Advanced subscription. To know how subscriptions for OpenShift Data Foundation work, see [knowledgebase article on OpenShift Data Foundation subscriptions](#).
  - When the Token authentication method is selected for encryption then refer to [Enabling cluster-wide encryption with the Token authentication using KMS](#).
  - When the Kubernetes authentication method is selected for encryption then refer to [Enabling cluster-wide encryption with the Kubernetes authentication using KMS](#).
  - Ensure that you are using signed certificates on your Vault servers.
2. Minimum starting node requirements  
An OpenShift Data Foundation cluster will be deployed with minimum configuration when the standard deployment resource requirement is not met. See [Resource requirements](#) section in Planning guide.
3. Disaster recovery requirements [Technology Preview]  
Disaster Recovery features supported by Red Hat OpenShift Data Foundation require all of the following prerequisites to successfully implement a disaster recovery solution:
  - A valid Red Hat OpenShift Data Foundation Advanced subscription
  - A valid Red Hat Advanced Cluster Management for Kubernetes subscription  
To know how subscriptions for OpenShift Data Foundation work, see [knowledgebase article on OpenShift Data Foundation subscriptions](#).

For detailed requirements, see [Configuring OpenShift Data Foundation Disaster Recovery for OpenShift Workloads](#) guide, and *Requirements and recommendations* section of the [Install guide](#) in Red Hat Advanced Cluster Management for Kubernetes documentation.

## CHAPTER 2. DEPLOYING OPENSIFT DATA FOUNDATION ON GOOGLE CLOUD

You can deploy OpenShift Data Foundation on OpenShift Container Platform using dynamic storage devices provided by Google Cloud installer-provisioned infrastructure. This enables you to create internal cluster resources and it results in internal provisioning of the base services, which helps to make additional storage classes available to applications.

Also, it is possible to deploy only the Multicloud Object Gateway (MCG) component with OpenShift Data Foundation. For more information, see [Deploy standalone Multicloud Object Gateway](#).



### NOTE

Only internal OpenShift Data Foundation 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 Data Foundation](#) chapter before proceeding with the below steps for deploying using dynamic storage devices:

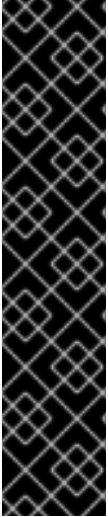
1. [Install the Red Hat OpenShift Data Foundation Operator](#).
2. [Create the OpenShift Data Foundation Cluster](#).

### 2.1. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR

You can install Red Hat OpenShift Data Foundation Operator using the Red Hat OpenShift Container Platform Operator Hub.

#### Prerequisites

- Access to an OpenShift Container Platform cluster using an account with **cluster-admin** and operator installation permissions.
- You must have at least three worker nodes in the Red Hat OpenShift Container Platform cluster. Each node should include one disk and requires 3 disks (PVs). However, one PV remains eventually unused by default. This is an expected behavior.
- For additional resource requirements, see the [Planning your deployment](#) guide.



## IMPORTANT

- When you need to override the cluster-wide default node selector for OpenShift Data Foundation, 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 Data Foundation resources are scheduled on that node. This helps you save on subscription costs. For more information, see the *How to use dedicated worker nodes for Red Hat OpenShift Data Foundation* section in the [Managing and Allocating Storage Resources](#) guide.

## Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators** → **OperatorHub**.
3. Scroll or type **OpenShift Data Foundation** into the **Filter by keyword** box to find the **OpenShift Data Foundation** Operator.
4. Click **Install**.
5. Set the following options on the **Install Operator** page:
  - a. Update Channel as **stable-4.11**.
  - 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 is created during the operator installation.
  - d. Select Approval Strategy as **Automatic** or **Manual**.  
If you select **Automatic** updates, then the Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your Operator without any intervention.  
  
If you select **Manual** updates, then the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to update the Operator to a newer version.
  - e. Ensure that the **Enable** option is selected for the **Console plugin**.
  - f. Click **Install**.

## Verification steps

- After the operator is successfully installed, a pop-up with a message, **Web console update is available** appears on the user interface. Click **Refresh web console** from this pop-up for the console changes to reflect.
- In the Web Console:
  - Navigate to Installed Operators and verify that the **OpenShift Data Foundation** Operator shows a green tick indicating successful installation.

- Navigate to **Storage** and verify if **Data Foundation** dashboard is available.

## 2.2. ENABLING CLUSTER-WIDE ENCRYPTION WITH KMS USING THE TOKEN AUTHENTICATION METHOD

You can enable the key value backend path and policy in the vault for token authentication.

### Prerequisites

- Administrator access to the vault.
- A valid Red Hat OpenShift Data Foundation Advanced subscription. For more information, see the [knowledgebase article on OpenShift Data Foundation subscriptions](#).
- Carefully, select a unique path name as the backend **path** that follows the naming convention since you cannot change it later.

### Procedure

1. Enable the Key/Value (KV) backend path in the vault.

For vault KV secret engine API, version 1:

```
$ vault secrets enable -path=odf kv
```

For vault KV secret engine API, version 2:

```
$ vault secrets enable -path=odf kv-v2
```

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

```
echo '
path "odf/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
path "sys/mounts" {
  capabilities = ["read"]
}' | vault policy write odf -
```

3. Create a token that matches the above policy:

```
$ vault token create -policy=odf -format json
```

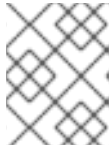
## 2.3. ENABLING CLUSTER-WIDE ENCRYPTION WITH KMS USING THE KUBERNETES AUTHENTICATION METHOD

You can enable the Kubernetes authentication method for cluster-wide encryption using the Key Management System (KMS).

### Prerequisites

- Administrator access to Vault.

- A valid Red Hat OpenShift Data Foundation Advanced subscription. For more information, see the [knowledgebase article on OpenShift Data Foundation subscriptions](#).
- The OpenShift Data Foundation operator must be installed from the Operator Hub.
- Select a unique path name as the backend **path** that follows the naming convention carefully. You cannot change this path name later.



## NOTE

Use of Vault namespaces are not supported with the Kubernetes authentication method in OpenShift Data Foundation 4.11.

## Procedure

1. Create a service account:

```
$ oc -n openshift-storage create serviceaccount <serviceaccount_name>
```

where, **<serviceaccount\_name>** specifies the name of the service account.

For example:

```
$ oc -n openshift-storage create serviceaccount odf-vault-auth
```

2. Create **clusterrolebindings** and **clusterroles**:

```
$ oc -n openshift-storage create clusterrolebinding vault-tokenreview-binding --
clusterrole=system:auth-delegator --serviceaccount=openshift-
storage:_<serviceaccount_name>_
```

For example:

```
$ oc -n openshift-storage create clusterrolebinding vault-tokenreview-binding --
clusterrole=system:auth-delegator --serviceaccount=openshift-storage:odf-vault-auth
```

3. Create a secret for the **serviceaccount** token and CA certificate.

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: odf-vault-auth-token
  namespace: openshift-storage
  annotations:
    kubernetes.io/service-account.name: <serviceaccount_name>
type: kubernetes.io/service-account-token
data: {}
EOF
```

where, **<serviceaccount\_name>** is the service account created in the earlier step.

4. Get the token and the CA certificate from the secret.



```
$ SA_JWT_TOKEN=$(oc -n openshift-storage get secret odf-vault-auth-token -o jsonpath="{.data['token']}" | base64 --decode; echo)
$ SA_CA_CERT=$(oc -n openshift-storage get secret odf-vault-auth-token -o jsonpath="{.data['ca.crt']}" | base64 --decode; echo)
```

- Retrieve the OCP cluster endpoint.

```
$ OCP_HOST=$(oc config view --minify --flatten -o jsonpath="{.clusters[0].cluster.server}")
```

- Fetch the service account issuer:

```
$ oc proxy &
$ proxy_pid=$!
$ issuer=$( curl --silent http://127.0.0.1:8001/.well-known/openid-configuration | jq -r
.issuer)
$ kill $proxy_pid
```

- Use the information collected in the previous step to setup the Kubernetes authentication method in Vault:

```
$ vault auth enable kubernetes
```

```
$ vault write auth/kubernetes/config \
  token_reviewer_jwt="$SA_JWT_TOKEN" \
  kubernetes_host="$OCP_HOST" \
  kubernetes_ca_cert="$SA_CA_CERT" \
  issuer="$issuer"
```



## IMPORTANT

To configure the Kubernetes authentication method in Vault when the issuer is empty:

```
$ vault write auth/kubernetes/config \
  token_reviewer_jwt="$SA_JWT_TOKEN" \
  kubernetes_host="$OCP_HOST" \
  kubernetes_ca_cert="$SA_CA_CERT"
```

- Enable the Key/Value (KV) backend path in Vault.  
For Vault KV secret engine API, version 1:

```
$ vault secrets enable -path=odf kv
```

For Vault KV secret engine API, version 2:

```
$ vault secrets enable -path=odf kv-v2
```

- Create a policy to restrict the users to perform a **write** or **delete** operation on the secret:

```
echo '
path "odf/*" {
```

```

capabilities = ["create", "read", "update", "delete", "list"]
}
path "sys/mounts" {
capabilities = ["read"]
}'| vault policy write odf -

```

10. Generate the roles:

```

$ vault write auth/kubernetes/role/odf-rook-ceph-op \
  bound_service_account_names=rook-ceph-system,rook-ceph-osd,noobaa \
  bound_service_account_namespaces=openshift-storage \
  policies=odf \
  ttl=1440h

```

The role **odf-rook-ceph-op** is later used while you configure the KMS connection details during the creation of the storage system.

```

$ vault write auth/kubernetes/role/odf-rook-ceph-osd \
  bound_service_account_names=rook-ceph-osd \
  bound_service_account_namespaces=openshift-storage \
  policies=odf \
  ttl=1440h

```

## 2.4. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER

Create an OpenShift Data Foundation cluster after you install the OpenShift Data Foundation operator.

### Prerequisites

- The OpenShift Data Foundation operator must be installed from the Operator Hub. For more information, see [Installing OpenShift Data Foundation Operator](#).
- Be aware that the default storage class of the Google Cloud platform 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:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: faster
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete

```

### Procedure

1. In the OpenShift Web Console, click **Operators → Installed Operators** to view all the installed operators.  
Ensure that the **Project** selected is **openshift-storage**.
2. Click on the **OpenShift Data Foundation** operator, and then click **Create StorageSystem**.

3. In the **Backing storage** page, select the following:
  - a. Select **Full Deployment** for the **Deployment type** option.
  - b. Select the **Use an existing StorageClass** option.
  - c. Select the **Storage Class**.  
By default, it is set as **standard**. However, if you created a storage class to use SSD based disks for better performance, you need to select that storage class.
  - d. Click **Next**.
4. In the **Capacity and nodes** page, provide the necessary information:
  - a. Select a value for **Requested Capacity** from the dropdown list. It is set to **2 TiB** by default.

**NOTE**

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

- b. In the **Select Nodes** section, select at least three available nodes.
  - c. Optional: Select the **Taint nodes** checkbox to dedicate the selected nodes for OpenShift Data Foundation.  
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 Data Foundation cluster requirements of an aggregated 30 CPUs and 72 GiB of RAM, a minimal cluster is deployed. For minimum starting node requirements, see the [Resource requirements](#) section in the *Planning* guide.
  - d. Click **Next**.
5. Optional: In the **Security and network** page, configure the following based on your requirements:
  - a. To enable encryption, select **Enable data encryption for block and file storage**
  - b. Select either one or both the encryption levels:
    - **Cluster-wide encryption**  
Encrypts the entire cluster (block and file).
    - **StorageClass encryption**  
Creates 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.
    - i. **Key Management Service Provider** is set to **Vault** by default.
    - ii. Select an **Authentication Method**.  
**Using Token authentication method**

- Enter a unique **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), **Port** number and **Token**.
- Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:
  - Enter the Key Value secret path in **Backend Path** that is dedicated and unique to OpenShift Data Foundation.
  - Optional: Enter **TLS Server Name** and **Vault Enterprise Namespace**
  - Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.
  - Click **Save**.

### Using Kubernetes authentication method

- Enter a unique Vault **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), **Port** number and **Role** name.
- Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:
  - Enter the Key Value secret path in **Backend Path** that is dedicated and unique to OpenShift Data Foundation.
  - Optional: Enter **TLS Server Name** and **Authentication Path** if applicable.
  - Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.
  - Click **Save**.

d. Click **Next**.

6. In the **Review and create** page, review the configuration details. To modify any configuration settings, click **Back**.

7. Click **Create StorageSystem**.

### Verification steps

- To verify the final Status of the installed storage cluster:
  - a. In the OpenShift Web Console, navigate to **Installed Operators** → **OpenShift Data Foundation** → **Storage System** → **ocs-storagecluster-storagesystem** → **Resources**.
  - b. Verify that **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.
- To verify that all components for OpenShift Data Foundation are successfully installed, see [Verifying your OpenShift Data Foundation deployment](#).

### Additional resources

To enable Overprovision Control alerts, refer to [Alerts](#) in Monitoring guide.

## 2.5. VERIFYING OPENSIFT DATA FOUNDATION DEPLOYMENT

Use this section to verify that OpenShift Data Foundation is deployed correctly.

### 2.5.1. Verifying the state of the pods

#### Procedure

1. Click **Workloads** → **Pods** from the OpenShift Web Console.
2. Select **openshift-storage** from the **Project** drop-down list.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

For more information on the expected number of pods for each component and how it varies depending on the number of nodes, see [Table 2.1, “Pods corresponding to OpenShift Data Foundation cluster”](#).

3. Set filter for Running and Completed pods to verify that the following pods are in **Running** and **Completed** state:

**Table 2.1. Pods corresponding to OpenShift Data Foundation cluster**

Component	Corresponding pods
OpenShift Data Foundation Operator	<ul style="list-style-type: none"> <li>● <b>ocs-operator-*</b> (1 pod on any storage node)</li> <li>● <b>ocs-metrics-exporter-*</b> (1 pod on any storage node)</li> <li>● <b>odf-operator-controller-manager-*</b> (1 pod on any storage node)</li> <li>● <b>odf-console-*</b> (1 pod on any storage node)</li> <li>● <b>csi-addons-controller-manager-*</b> (1 pod on any storage node)</li> </ul>
Rook-ceph Operator	<p><b>rook-ceph-operator-*</b></p> <p>(1 pod on any storage node)</p>

Component	Corresponding pods
Multicloud Object Gateway	<ul style="list-style-type: none"> <li>● <b>noobaa-operator-*</b> (1 pod on any storage node)</li> <li>● <b>noobaa-core-*</b> (1 pod on any storage node)</li> <li>● <b>noobaa-db-pg-*</b> (1 pod on any storage node)</li> <li>● <b>noobaa-endpoint-*</b> (1 pod on any storage node)</li> </ul>
MON	<p><b>rook-ceph-mon-*</b></p> <p>(3 pods distributed across storage nodes)</p>
MGR	<p><b>rook-ceph-mgr-*</b></p> <p>(1 pod on any storage node)</p>
MDS	<p><b>rook-ceph-mds-ocs-storagecluster-cephfilesystem-*</b></p> <p>(2 pods distributed across storage nodes)</p>
CSI	<ul style="list-style-type: none"> <li>● <b>cephfs</b> <ul style="list-style-type: none"> <li>○ <b>csi-cephfsplugin-*</b> (1 pod on each storage node)</li> <li>○ <b>csi-cephfsplugin-provisioner-*</b> (2 pods distributed across storage nodes)</li> </ul> </li> <li>● <b>rbd</b> <ul style="list-style-type: none"> <li>○ <b>csi-rbdplugin-*</b> (1 pod on each storage node)</li> <li>○ <b>csi-rbdplugin-provisioner-*</b> (2 pods distributed across storage nodes)</li> </ul> </li> </ul>
rook-ceph-crashcollector	<p><b>rook-ceph-crashcollector-*</b></p> <p>(1 pod on each storage node)</p>
OSD	<ul style="list-style-type: none"> <li>● <b>rook-ceph-osd-*</b> (1 pod for each device)</li> <li>● <b>rook-ceph-osd-prepare-ocs-deviceset-*</b> (1 pod for each device)</li> </ul>

## 2.5.2. Verifying the OpenShift Data Foundation cluster is healthy

### Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
3. In the **Status** card of the **Block and File** tab, verify that *Storage Cluster* has a green tick.
4. In the **Details** card, verify that the cluster information is displayed.

For more information on the health of the OpenShift Data Foundation cluster using the **Block and File** dashboard, see [Monitoring OpenShift Data Foundation](#).

## 2.5.3. Verifying the Multicloud Object Gateway is healthy

### Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
  - a. In the **Status card** of the **Object** tab, verify that both *Object Service* and *Data Resiliency* have a green tick.
  - b. In the **Details** card, verify that the MCG information is displayed.

For more information on the health of the OpenShift Data Foundation cluster using the object service dashboard, see [Monitoring OpenShift Data Foundation](#).

## 2.5.4. Verifying that the OpenShift Data Foundation specific storage classes exist

### Procedure

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

## CHAPTER 3. DEPLOY STANDALONE MULTICLOUD OBJECT GATEWAY

Deploying only the Multicloud Object Gateway component with the OpenShift Data Foundation provides the flexibility in deployment and helps to reduce the resource consumption. Use this section to deploy only the standalone Multicloud Object Gateway component, which involves the following steps:

- Installing Red Hat OpenShift Data Foundation Operator
- Creating standalone Multicloud Object Gateway

### 3.1. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR

You can install Red Hat OpenShift Data Foundation Operator using the Red Hat OpenShift Container Platform Operator Hub.

#### Prerequisites

- Access to an OpenShift Container Platform cluster using an account with **cluster-admin** and operator installation permissions.
- You must have at least three worker nodes in the Red Hat OpenShift Container Platform cluster. Each node should include one disk and requires 3 disks (PVs). However, one PV remains eventually unused by default. This is an expected behavior.
- For additional resource requirements, see the [Planning your deployment](#) guide.



#### IMPORTANT

- When you need to override the cluster-wide default node selector for OpenShift Data Foundation, 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 Data Foundation resources are scheduled on that node. This helps you save on subscription costs. For more information, see the *How to use dedicated worker nodes for Red Hat OpenShift Data Foundation* section in the [Managing and Allocating Storage Resources](#) guide.

#### Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators** → **OperatorHub**.
3. Scroll or type **OpenShift Data Foundation** into the **Filter by keyword** box to find the **OpenShift Data Foundation Operator**.
4. Click **Install**.



5. Set the following options on the **Install Operator** page:
  - a. Update Channel as **stable-4.11**.
  - 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 is created during the operator installation.
  - d. Select Approval Strategy as **Automatic** or **Manual**.  
If you select **Automatic** updates, then the Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your Operator without any intervention.  
  
If you select **Manual** updates, then the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to update the Operator to a newer version.
  - e. Ensure that the **Enable** option is selected for the **Console plugin**.
  - f. Click **Install**.

### Verification steps

- After the operator is successfully installed, a pop-up with a message, **Web console update is available** appears on the user interface. Click **Refresh web console** from this pop-up for the console changes to reflect.
- In the Web Console:
  - Navigate to Installed Operators and verify that the **OpenShift Data Foundation** Operator shows a green tick indicating successful installation.
  - Navigate to **Storage** and verify if **Data Foundation** dashboard is available.

## 3.2. CREATING A STANDALONE MULTICLOUD OBJECT GATEWAY

You can create only the standalone Multicloud Object Gateway component while deploying OpenShift Data Foundation.

### Prerequisites

- Ensure that the OpenShift Data Foundation Operator is installed.

### Procedure

1. In the OpenShift Web Console, click **Operators** → **Installed Operators** to view all the installed operators.  
Ensure that the **Project** selected is **openshift-storage**.
2. Click **OpenShift Data Foundation** operator and then click **Create StorageSystem**.
3. In the **Backing storage** page, select the following:
  - a. Select **Multicloud Object Gateway** for **Deployment type**.
  - b. Select the **Use an existing StorageClass** option.

- c. Click **Next**.
4. Optional: In the **Security** page, select **Connect to an external key management service**
  - 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>'), **Port number**, and **Token**.
  - c. Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:
    - i. Enter the Key Value secret path in the **Backend Path** that is dedicated and unique to OpenShift Data Foundation.
    - ii. Optional: Enter **TLS Server Name** and **Vault Enterprise Namespace**
    - iii. Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate**, and **Client Private Key**.
    - iv. Click **Save**.
  - d. Click **Next**.
5. In the **Review and create** page, review the configuration details:  
To modify any configuration settings, click **Back**.
6. Click **Create StorageSystem**.

## Verification steps

### Verifying that the OpenShift Data Foundation cluster is healthy

1. In the OpenShift Web Console, click **Storage → Data Foundation**.
2. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
  - a. In the **Status card** of the **Object** tab, verify that both *Object Service* and *Data Resiliency* have a green tick.
  - b. In the **Details** card, verify that the MCG information is displayed.

### Verifying the state of the pods

1. Click **Workloads → Pods** from the OpenShift Web Console.
2. Select **openshift-storage** from the **Project** drop-down list and verify that the following pods are in **Running** state.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

Component	Corresponding pods
OpenShift Data Foundation Operator	<ul style="list-style-type: none"><li>● <b>ocs-operator-*</b> (1 pod on any storage node)</li><li>● <b>ocs-metrics-exporter-*</b> (1 pod on any storage node)</li><li>● <b>odf-operator-controller-manager-*</b> (1 pod on any storage node)</li><li>● <b>odf-console-*</b> (1 pod on any storage node)</li><li>● <b>csi-addons-controller-manager-*</b> (1 pod on any storage node)</li></ul>
Rook-ceph Operator	<b>rook-ceph-operator-*</b> (1 pod on any storage node)
Multicloud Object Gateway	<ul style="list-style-type: none"><li>● <b>noobaa-operator-*</b> (1 pod on any storage node)</li><li>● <b>noobaa-core-*</b> (1 pod on any storage node)</li><li>● <b>noobaa-db-pg-*</b> (1 pod on any storage node)</li><li>● <b>noobaa-endpoint-*</b> (1 pod on any storage node)</li></ul>

## CHAPTER 4. UNINSTALLING OPENSIFT DATA FOUNDATION

### 4.1. UNINSTALLING OPENSIFT DATA FOUNDATION IN INTERNAL MODE

To uninstall OpenShift Data Foundation in Internal mode, refer to the [knowledge base article on Uninstalling OpenShift Data Foundation](#).

## CHAPTER 5. STORAGE CLASSES AND STORAGE POOLS

The OpenShift Data Foundation 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 behaviour.

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 Data Foundation 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 Data Foundation cluster is in **Ready** state.

### Procedure

1. Click **Storage** → **StorageClasses**.
2. Click **Create Storage Class**
3. Enter the storage class **Name** and **Description**.
4. **Reclaim Policy** is set to **Delete** as the default option. Use this setting.  
If you change the reclaim policy to **Retain** in the storage class, the persistent volume (PV) remains in **Released** state even after deleting the persistent volume claim (PVC).
5. **Volume binding mode** is set to **WaitForConsumer** as the default option.  
If you choose the **Immediate** option, then the PV gets created immediately when creating the PVC.
6. Select **RBD** or **CephFS Provisioner** as the plugin for provisioning the persistent volumes.
7. Select an existing **Storage Pool** from the list or create a new pool.



## NOTE

The 2-way replication data protection policy is only supported for the non-default RBD pool. 2-way replication can be used by creating an additional pool. To know about Data Availability and Integrity considerations for replica 2 pools, see [Knowledgebase Customer Solution Article](#).

### 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.
8. Optional: Select **Enable Encryption** checkbox.
  9. Click **Create** to create the storage class.

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

### Prerequisites

Based on your use case, you must ensure to configure access to KMS for one of the following:

- Using **vaulttokens**: Ensure to configure access as described in [Configuring access to KMS using vaulttokens](#)
- Using **vaulttenantsa** (Technology Preview): Ensure to configure access as described in [Configuring access to KMS using vaulttenantsa](#)

### Procedure

1. In the OpenShift Web Console, navigate to **Storage** → **StorageClasses**.
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 either **Immediate** or **WaitForFirstConsumer** as the **Volume binding mode**. **WaitForConsumer** is set as the default option.

6. Select **RBD Provisioner `openshift-storage.rbd.csi.ceph.com`** which is the plugin used for provisioning the persistent volumes.
7. Select **Storage Pool** where the volume data is stored from the list or create a new pool.
8. Select the **Enable encryption** checkbox. There are two options available to set the KMS connection details:
  - **Select existing KMS connection:** Select an existing KMS connection from the drop-down list. The list is populated from the the connection details available in the **csi-kms-connection-details** ConfigMap.
  - **Create new KMS connection** This is applicable for **vaulttokens** only.
    - a. **Key Management Service Provider** is set to Vault by default.
    - b. Enter a unique **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), Port number and **Token**.
    - 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 Data Foundation.
      - ii. Optional: Enter **TLS Server Name** and **Vault Enterprise Namespace**
      - iii. Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.
      - iv. Click **Save**.
    - d. Click **Save**.
9. Click **Create**.
10. Edit the ConfigMap to add the **vaultBackend** 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

**vaultBackend** is an optional parameters 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 **vaultBackend** 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.

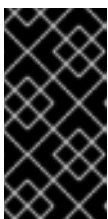
Example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: csi-kms-connection-details
[...]
data:
  1-vault: |-
    {
      "encryptionKMSType": "vaulttokens",
      "kmsServiceName": "1-vault",
      [...]
      "vaultBackend": "kv-v2"
    }
  2-vault: |-
    {
      "encryptionKMSType": "vaulttenantsa",
      [...]
      "vaultBackend": "kv"
    }
```

- iii. Click Save

## Next steps

- The storage class can be used to create encrypted persistent volumes. For more information, see [managing persistent volume claims](#).



## 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](#).



## CHAPTER 6. CONFIGURE STORAGE FOR OPENSIFT CONTAINER PLATFORM SERVICES

You can use OpenShift Data Foundation 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 Data Foundation 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 OPENSIFT DATA FOUNDATION

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.



### NOTE

Starting with OpenShift Container Platform 4.11, object storage is the recommended backend storage for configuring image registry.

Follow the instructions in this section to configure OpenShift Data Foundation 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.

## 6.1.1. Configuring Multicloud Object Gateway as backend storage for OpenShift image registry

You can use Multicloud Object Gateway (MCG) as OpenShift image registry backend storage in an on-prem OpenShift deployment starting from OpenShift Container Platform version 4.11.

### Prerequisites

- Administrative access to OpenShift Web Console.
- A running OpenShift Data Foundation cluster with MCG.

### Procedure

1. Create **ObjectBucketClaim** by following the steps in [Creating Object Bucket Claim](#).
2. Create an **image-registry-private-configuration-user** secret.
  - a. Go to the OpenShift web-console.
  - b. Click **ObjectBucketClaim** → **ObjectBucketClaim Data**.
  - c. In the **ObjectBucketClaim** data, look for **MCG access key** and **MCG secret key** in the **openshift-image-registry namespace**.
  - d. Create the secret using the following command:

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=<MCG Accesskey> --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=<MCG Secretkey> --namespace openshift-image-registry
```

3. Change the status of **managementState** of Image Registry Operator to **Managed**.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec": {"managementState": "Managed"}}'
```

4. Edit the **spec.storage** section of Image Registry Operator configuration file:
  - a. Get the **unique-bucket-name** and **regionEndpoint** under the **Object Bucket Claim Data** section from the Web Console **OR** you can also get the information on regionEndpoint and unique-bucket-name from the command:

```
$ oc describe noobaa
```

- b. Add **regionEndpoint** as <http://<Endpoint-name>:<port>> if the
  - storageclass is **ceph-rgw** storageclass and the
  - endpoint points to the internal SVC from the openshift-storage namespace.
- c. An **image-registry** pod spawns after you make the changes to the Operator registry configuration file.

```
$ oc edit configs.imageregistry.operator.openshift.io -n openshift-image-registry
```

```

apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  [..]
name: cluster
spec:
  [..]
storage:
  s3:

    bucket: <Unique-bucket-name>

    region: us-east-1 (Use this region as default)

    regionEndpoint: https://<Endpoint-name>:<port>

    virtualHostedStyle: false

```

5. Reset the image registry settings to default.

```
$ oc get pods -n openshift-image-registry
```

### Verification steps

- Run the following command to check if you have configured MCG as OpenShift Image Registry backend storage successfully.

```
$ oc get pods -n openshift-image-registry
```

### Example output

```

$ oc get pods -n openshift-image-registry

NAME                                READY  STATUS   RESTARTS  AGE
cluster-image-registry-operator-56d78bc5fb-bxcgv  2/2    Running  0         44d
image-pruner-1605830400-29r7k          0/1    Completed  0         10h
image-registry-b6c8f4596-ln88h        1/1    Running   0         17d
node-ca-2nxvz                          1/1    Running   0         44d
node-ca-dtwjd                          1/1    Running   0         44d
node-ca-h92rj                          1/1    Running   0         44d
node-ca-k9bkd                          1/1    Running   0         44d
node-ca-stkzc                          1/1    Running   0         44d
node-ca-xn8h4                          1/1    Running   0         44d

```

- (Optional) You can also the run the following command to verify if you have configured MCG as OpenShift Image Registry backend storage successfully.

```
$ oc describe pod <image-registry-name>
```

### Example output

```
$ oc describe pod image-registry-b6c8f4596-ln88h
```

## Environment:

```

REGISTRY_STORAGE_S3_REGIONENDPOINT:  http://s3.openshift-storage.svc

REGISTRY_STORAGE:                      s3

REGISTRY_STORAGE_S3_BUCKET:            bucket-registry-mcg

REGISTRY_STORAGE_S3_REGION:           us-east-1

REGISTRY_STORAGE_S3_ENCRYPT:           true

REGISTRY_STORAGE_S3_VIRTUALHOSTEDSTYLE: false

REGISTRY_STORAGE_S3_USEDUALSTACK:     true

REGISTRY_STORAGE_S3_ACCESSKEY:        <set to the key
'REGISTRY_STORAGE_S3_ACCESSKEY' in secret 'image-registry-private-configuration'>
Optional: false

REGISTRY_STORAGE_S3_SECRETKEY:        <set to the key
'REGISTRY_STORAGE_S3_SECRETKEY' in secret 'image-registry-private-configuration'>
Optional: false

REGISTRY_HTTP_ADDR:                    :5000

REGISTRY_HTTP_NET:                     tcp

REGISTRY_HTTP_SECRET:
57b943f691c878e342bac34e657b702bd6ca5488d51f839fecafa918a79a5fc6ed70184cab04760
1403c1f383e54d458744062dcaaa483816d82408bb56e686f

REGISTRY_LOG_LEVEL:                    info

REGISTRY_OPENSHIFT_QUOTA_ENABLED:      true

REGISTRY_STORAGE_CACHE_BLOBDESCRIPTOR: inmemory

REGISTRY_STORAGE_DELETE_ENABLED:      true

REGISTRY_OPENSHIFT_METRICS_ENABLED:   true

REGISTRY_OPENSHIFT_SERVER_ADDR:       image-registry.openshift-image-
registry.svc:5000

REGISTRY_HTTP_TLS_CERTIFICATE:         /etc/secrets/tls.crt

REGISTRY_HTTP_TLS_KEY:                 /etc/secrets/tls.key

```

## 6.1.2. Configuring OpenShift Data Foundation CephFS as backend storage for OpenShift image registry

### Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Data Foundation 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** → **StorageClasses** 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 OPENSIFT DATA FOUNDATION

OpenShift Data Foundation provides a monitoring stack that comprises of Prometheus and Alert Manager.

Follow the instructions in this section to configure OpenShift Data Foundation 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 Data Foundation 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** → **StorageClasses** to view available storage classes.

### Procedure

1. In the OpenShift Web Console, go to **Workloads** → **Config Maps**.
2. Set the **Project** dropdown to **openshift-monitoring**.
3. Click **Create Config Map**.
4. Define a new **cluster-monitoring-config** Config Map using the following example. Replace the content in angle brackets (<, >) with your own values, for example, **retention: 24h** or **storage: 40Gi**.

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

#### Example cluster-monitoring-config Config Map

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

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

#### Verification steps

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

Figure 6.1. Monitoring storage created and bound

Project: openshift-monitoring ▾

Persistent Volume Claims

Create Persistent Volume Claim

Filter by name...

0 Pending 5 Bound 0 Lost Select All Filters 5 Items

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

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**.

Figure 6.2. Persistent Volume Claims attached to alertmanager-main-\* pod

Volumes

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

3. Verify that the new **prometheus-k8s-\*** pods appear with a state of **Running**.
  - a. Click the new **prometheus-k8s-\*** pods to view the pod details.
  - b. Scroll down to **Volumes** and verify that the volume has a **Type**, **ocs-prometheus-claim** that matches one of your new Persistent Volume Claims, for example, **ocs-prometheus-claim-prometheus-k8s-0**.

Figure 6.3. Persistent Volume Claims attached to prometheus-k8s-\* pod

Volumes

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

## 6.3. CLUSTER LOGGING FOR OPENSIFT DATA FOUNDATION



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 Data Foundation 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 Data Foundation to have OpenShift Data Foundation 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:

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

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



### NOTE

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

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

For more information, see [Configuring cluster logging](#).

### 6.3.2. Configuring cluster logging to use OpenShift data Foundation

Follow the instructions in this section to configure OpenShift Data Foundation as storage for the OpenShift cluster logging.



#### NOTE

You can obtain all the logs when you configure logging for the first time in OpenShift Data Foundation. 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 Data Foundation Operator is installed and running in the **openshift-storage** namespace.
- Cluster logging Operator is installed and running in the **openshift-logging** namespace.

#### Procedure

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

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

```

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

```

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

```

spec:
[...]
```

```

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

```

6. Click **Save**.

## Verification steps

1. Verify that the Persistent Volume Claims are bound to the **elasticsearch** pods.
  - a. Go to **Storage → Persistent Volume Claims**
  - b. Set the **Project** dropdown to **openshift-logging**.
  - c. Verify that Persistent Volume Claims are visible with a state of **Bound**, attached to **elasticsearch-\*** pods.

Figure 6.4. Cluster logging created and bound

Name	Namespace	Status	Persistent Volume	Requested
elasticsearch-elasticsearch-cdm-9r6z4biv-1	openshift-logging	Bound	pvc-8993013d-1a6e-11ea-8d2f-027bataef61a	200G
elasticsearch-elasticsearch-cdm-9r6z4biv-2	openshift-logging	Bound	pvc-89947c90-1a6e-11ea-8d2f-027bataef61a	200G
elasticsearch-elasticsearch-cdm-9r6z4biv-3	openshift-logging	Bound	pvc-8995f557-1a6e-11ea-8d2f-027bataef61a	200G

2. Verify that the new cluster logging is being used.
  - a. Click **Workload → Pods**

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

```
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 Data Foundation in the uninstall chapter of the respective deployment guide.

## CHAPTER 7. BACKING OPENSIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSIFT DATA FOUNDATION

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

### Prerequisites

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

### Procedure

1. In the OpenShift Web Console, perform one of the following:

- Click **Workloads → Deployments**.

In the Deployments page, you can do one of the following:

- Select any existing deployment and click **Add Storage** option from the **Action** menu (⋮).
- Create a new deployment and then add storage.
  - i. Click **Create Deployment** to create a new deployment.
  - ii. Edit the **YAML** based on your requirement to create a deployment.
  - iii. Click **Create**.
- iv. Select **Add Storage** from the **Actions** drop-down menu on the top right of the page.

- Click **Workloads → Deployment Configs**

In the Deployment Configs page, you can do one of the following:

- Select any existing deployment and click **Add Storage** option from the **Action** menu (⋮).
- Create a new deployment and then add storage.
  - i. Click **Create Deployment Config** to create a new deployment.
  - ii. Edit the **YAML** based on your requirement to create a deployment.
  - iii. Click **Create**.
- iv. Select **Add Storage** from the **Actions** drop-down menu on the top right of the page.

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

- Click the **Use existing claim** option and select a suitable PVC from the drop-down list.

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

**NOTE**

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

- d. Select the size of the desired storage capacity.

**NOTE**

You can expand 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 OPENSIFT DATA FOUNDATION

Any Red Hat OpenShift Container Platform subscription requires an OpenShift Data Foundation subscription. However, you can save on the OpenShift Container Platform subscription costs if you are using infrastructure nodes to schedule OpenShift Data Foundation resources.

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 Data Foundation 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 Data Foundation entitlements are necessary for the nodes running OpenShift Data Foundation.

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

Adding an OpenShift Data Foundation taint with a **NoSchedule** effect is also required so that the **infra** node will only schedule OpenShift Data Foundation 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 Data Foundation resources to be scheduled on the tainted nodes.



#### NOTE

Adding storage taint on nodes might require toleration handling for the other **daemonset** pods such as **openshift-dns daemonset**. For information about how to manage the tolerations, see Knowledgebase article: <https://access.redhat.com/solutions/6592171>.

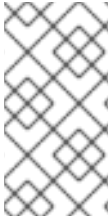
Example of the taint and labels required on infrastructure node that will be used to run OpenShift Data Foundation services:

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

### 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.

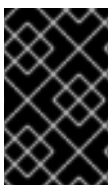


## NOTE

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 Data Foundation 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 Data Foundation services.

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



## IMPORTANT

If you add a taint to the infrastructure nodes, you also need to add tolerations to the taint for other workloads, for example, the fluentd pods. For more information, see the Red Hat Knowledgebase solution [Infrastructure Nodes in OpenShift 4](#).

## 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 Data Foundation services, and that these nodes have sufficient CPU and memory resources. To avoid the RHOCP subscription cost, the following is required:

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

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



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



### WARNING

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

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

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

## 8.4. TAINT A NODE FROM THE USER INTERFACE

This section explains the procedure to taint nodes after the OpenShift Data Foundation deployment.

### Procedure

1. In the OpenShift Web Console, click **Compute** → **Nodes**, and then select the node which has to be tainted.
2. In the **Details** page click on **Edit taints**.
3. Enter the values in the **Key** <nodes.openshift.ocs.io/storage>, **Value** <true> and in the **Effect**<Noschedule> field.
4. Click Save.

### Verification steps

- Follow the steps to verify that the node has tainted successfully:
  - Navigate to **Compute** → **Nodes**.
  - Select the node to verify its status, and then click on the **YAML** tab.
  - In the **specs** section check the values of the following parameters:

```
Taints:
  Key: nodes.openshift.ocs.io/storage
  Value: true
  Effect: Noschedule
```

### Additional resources

For more information, refer to [Creating the OpenShift Data Foundation cluster on VMware vSphere](#) .

## CHAPTER 9. SCALING STORAGE NODES

To scale the storage capacity of OpenShift Data Foundation, you can do either of the following:

- **Scale up storage nodes** - Add storage capacity to the existing OpenShift Data Foundation 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 Data Foundation 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 DATA FOUNDATION NODES ON GOOGLE CLOUD INFRASTRUCTURE

To increase the storage capacity in a dynamically created storage cluster on an user-provisioned infrastructure, you can add storage capacity and performance to your configured Red Hat OpenShift Data Foundation worker nodes.

#### Prerequisites

- You have administrative privilege to the OpenShift Container Platform Console.
- You have a running OpenShift Data Foundation Storage Cluster.

- The disk should be of the same size and type as used during initial deployment.

### Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators** → **Installed Operators**
3. Click **OpenShift Data Foundation** Operator.
4. Click the **Storage Systems** tab.
  - a. Click the **Action Menu ( ⋮ )** on the far right of the storage system name to extend the options menu.
  - b. Select **Add Capacity** from the options menu.
  - c. Select the **Storage Class**. Choose the storage class which you wish to use to provision new storage devices.  
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 Data Foundation uses a replica count of 3.
  - d. Click **Add**.
5. To check the status, navigate to **Storage** → **Data Foundation** and verify that **Storage System** in the Status card has a green tick.

### Verification steps

- Verify the **Raw Capacity** card.
  - a. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
  - b. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
  - c. In the **Block and File** tab, check the **Raw Capacity** 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 Persistent Volume Claims (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.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

- 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.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

- Optional: If cluster-wide encryption is enabled on the cluster, verify that the new OSD devices are encrypted.
  - a. Identify the nodes where the new OSD pods are running.

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

#### <OSD-pod-name>

Is the name of the OSD pod.  
For example:

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

Example output:

```
NODE
compute-1
```

- b. For each of the nodes identified in the previous step, do the following:
  - i. Create a debug pod and open a chroot environment for the selected hosts.

```
$ oc debug node/<node-name>
```

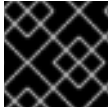
#### <node-name>

Is the name of the node.

```
$ chroot /host
```

- ii. Check for the **crypt** keyword beside the **ocs-deviceSet** names.

```
$ lsblk
```



### IMPORTANT

Cluster reduction is supported only with the [Red Hat Support Team's](#) assistance.

## 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 to an installer-provisioned infrastructure

#### Prerequisites

- You have administrative privilege to the OpenShift Container Platform Console.
- You have a running OpenShift Data Foundation Storage Cluster.

#### Procedure

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

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

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

2. On the OpenShift web console, click **Workloads** → **Pods**, confirm that at least the following pods on the new node are in **Running** state:
  - **csi-cephfsplugin-\***
  - **csi-rbdplugin-\***

### 9.3.2. Scaling up storage capacity

After you add a new node to OpenShift Data Foundation, 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 Multicloud Object Gateway (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 Data Foundation Platform.
- Download the MCG command-line interface for easier management.

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



#### NOTE

Specify the appropriate architecture for enabling the repositories using the subscription manager.

- For IBM Power, use the following command:

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

- For IBM Z infrastructure, use the following command:

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

- Alternatively, you can install the MCG package from the OpenShift Data Foundation RPMs found at [Download RedHat OpenShift Data Foundation page](#).



#### NOTE

Choose the correct Product Variant according to your architecture.

You can access the relevant endpoint, access key, and secret access key in 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"](#)

For example:

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

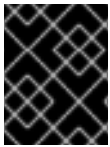
If the client application tries to access `https://<bucket-name>.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com`

#### <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 Multicloud Object Gateway (MCG) endpoint, including its access key (**AWS\_ACCESS\_KEY\_ID** value) and secret access key (**AWS\_SECRET\_ACCESS\_KEY** value).

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

The output will look similar to the following:

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



Welcome to NooBaa!  
-----

Welcome to NooBaa!  
-----

NooBaa Core Version:  
NooBaa Operator Version:

Lets get started:

#### 1. Connect to Management console:

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

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

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

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

#### 2. Test S3 client:

```
kubectl port-forward -n openshift-storage service/s3 10443:443 &
```

**1**

```
NOOBAA_ACCESS_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r  
'data.AWS_ACCESS_KEY_ID|@base64d')
```

**2**

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

Services:

Service Mgmt:

External DNS:

```
https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
```

```
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-
```

2.elb.amazonaws.com:443

Internal DNS:

```
https://noobaa-mgmt.openshift-storage.svc:443
```

Internal IP:

```
https://172.30.235.12:443
```

Node Ports:

```
https://10.0.142.103:31385
```

Pod Ports:

```
https://10.131.0.19:8443
```

serviceS3:

External DNS: **3**

```
https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
```

```
https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443
```

Internal DNS:

```

https://s3.openshift-storage.svc:443
Internal IP:
https://172.30.86.41:443
Node Ports:
https://10.0.142.103:31011
Pod Ports:
https://10.131.0.19:6443

```

- 1 access key (**AWS\_ACCESS\_KEY\_ID** value)
- 2 secret access key (**AWS\_SECRET\_ACCESS\_KEY** value)
- 3 MCG endpoint



#### NOTE

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

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

### Prerequisites

- Download the MCG command-line interface.

```

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

```



#### NOTE

Specify the appropriate architecture for enabling the repositories using the subscription manager.

- For IBM Power, use the following command:

```

# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms

```

- For IBM Z infrastructure, use the following command:

```

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

```

```
#-----#
```

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

```
#-----#
```

```
ExternalDNS : [https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443]
```

```
ExternalIP : []
```

```
NodePorts : [https://10.0.142.103:31385]
```

```
InternalDNS : [https://noobaa-mgmt.openshift-storage.svc:443]
```

```
InternalIP : [https://172.30.235.12:443]
```

```
PodPorts : [https://10.131.0.19:8443]
```

```
#-----#
```

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

```
#-----#
```

```
email : admin@noobaa.io
```

```
password : HKLbH1rSuVU0l/souIkSiA==
```

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

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

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

```
2
AWS_ACCESS_KEY_ID : jVmAsu9FsvRHYmfjTiHV
```

```
3
AWS_SECRET_ACCESS_KEY : E//420VNedJfATvVSmDz6FMtsSAzuBv6z180PT5c
```

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

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

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

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

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

No OBC's found.

- 1 endpoint
- 2 access key
- 3 secret access key

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

For example:

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

```
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. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD

### 10.3.1. Creating a new backing store

Use this procedure to create a new backing store in OpenShift Data Foundation.

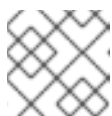
#### Prerequisites

- Administrator access to OpenShift Data Foundation.

#### Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. Click the **Backing Store** tab.
3. Click **Create Backing Store**.
4. 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 the 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 10.3.2, "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 the **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 the MCG that it can use this bucket for the system.
5. Click **Create Backing Store**.

## Verification steps

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. Click the **Backing Store** tab to view all the backing stores.

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

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

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

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

- For creating an AWS-backed backingstore, see [Section 10.3.2.1, "Creating an AWS-backed backingstore"](#)
- For creating an IBM COS-backed backingstore, see [Section 10.3.2.2, "Creating an IBM COS-backed backingstore"](#)
- For creating an Azure-backed backingstore, see [Section 10.3.2.3, "Creating an Azure-backed backingstore"](#)
- For creating a GCP-backed backingstore, see [Section 10.3.2.4, "Creating a GCP-backed backingstore"](#)
- For creating a local Persistent Volume-backed backingstore, see [Section 10.3.2.5, "Creating a local Persistent Volume-backed backingstore"](#)

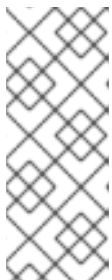
For VMware deployments, skip to [Section 10.3.3, "Creating an s3 compatible Multicloud Object Gateway backingstore"](#) for further instructions.

#### 10.3.2.1. Creating an AWS-backed backingstore

##### Prerequisites

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

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



##### NOTE

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

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

- Alternatively, you can install the MCG package from the OpenShift Data Foundation 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**

*Using MCG command-line interface*

- 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
```

**<backingstore\_name>**

The name of the backingstore.

**<AWS ACCESS KEY> and <AWS SECRET ACCESS KEY>**

The AWS access key ID and secret access key you created for this purpose.

**<bucket-name>**

The existing AWS bucket name. This argument indicates to the MCG 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"
```

*Adding storage resources using a YAML*

1. Create a secret with the credentials:

```
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>
```

**<AWS ACCESS KEY> and <AWS SECRET ACCESS KEY>**

Supply and encode your own AWS access key ID and secret access key using Base64, and use the results for **<AWS ACCESS KEY ID ENCODED IN BASE64>** and **<AWS SECRET ACCESS KEY ENCODED IN BASE64>**.

**<backingstore-secret-name>**

The name of the backingstore secret created in the previous step.

- Apply the following YAML for a specific backing store:

```

apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
    type: aws-s3

```

**<bucket-name>**

The existing AWS bucket name.

**<backingstore-secret-name>**

The name of the backingstore secret created in the previous step.

### 10.3.2.2. Creating an IBM COS-backed backingstore

#### Prerequisites

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

```

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

```



#### NOTE

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

- For IBM Power, use the following command:

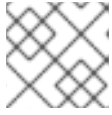
```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- For IBM Z infrastructure, use the following command:

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

- Alternatively, you can install the MCG package from the OpenShift Data Foundation 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**

*Using command-line interface*

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

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

**<backingstore\_name>**

The name of the backingstore.

**<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, and <IBM COS ENDPOINT>**

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.

**<bucket-name>**

An existing IBM bucket name. This argument indicates MCG about the 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"
```

*Adding storage resources using an YAML*

1. Create a secret with the credentials:

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

**<IBM COS ACCESS KEY ID ENCODED IN BASE64> and <IBM COS SECRET ACCESS KEY ENCODED IN BASE64>**

Provide and encode your own IBM COS access key ID and secret access key using Base64, and use the results in place of these attributes respectively.

**<backingstore-secret-name>**

The name of the backingstore secret.

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
```

#### <bucket-name>

an existing IBM COS bucket name. This argument indicates to MCG about the bucket to use as a target bucket for its backingstore, and subsequently, data storage and administration.

#### <endpoint>

A regional endpoint that corresponds to the location of the existing IBM bucket name. This argument indicates to MCG about the endpoint to use for its backingstore, and subsequently, data storage and administration.

#### <backingstore-secret-name>

The name of the secret created in the previous step.

### 10.3.2.3. Creating an Azure-backed backingstore

#### Prerequisites

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

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

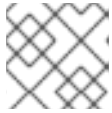


#### NOTE

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

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

- Alternatively, you can install the MCG package from the OpenShift Data Foundation 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**

*Using the MCG command-line interface*

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

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

**<backingstore\_name>**

The name of the backingstore.

**<AZURE ACCOUNT KEY> and <AZURE ACCOUNT NAME>**

An AZURE account key and account name you created for this purpose.

**<blob container name>**

An existing Azure blob container name. This argument indicates to MCG about the bucket to use as a target bucket for its backingstore, and subsequently, data storage and administration.

The output will be similar to the following:

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

*Adding storage resources using a YAML*

1. Create a secret with the credentials:

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

**<AZURE ACCOUNT NAME ENCODED IN BASE64> and <AZURE ACCOUNT KEY ENCODED IN BASE64>**

Supply and encode your own Azure Account Name and Account Key using Base64, and use the results in place of these attributes respectively.

**<backingstore-secret-name>**

A unique name of backingstore secret.

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:
  azureBlob:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBlobContainer: <blob-container-name>
    type: azure-blob

```

**<blob-container-name>**

An existing Azure blob container name. This argument indicates to the MCG about the bucket to use as a target bucket for its backingstore, and subsequently, data storage and administration.

**<backingstore-secret-name>**

with the name of the secret created in the previous step.

### 10.3.2.4. Creating a GCP-backed backingstore

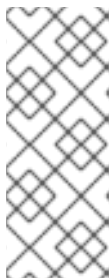
#### Prerequisites

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

```

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

```



**NOTE**

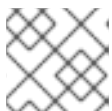
Specify the appropriate architecture for enabling the repositories using the subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```

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

```

- Alternatively, you can install the MCG package from the OpenShift Data Foundation 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

Using the MCG command-line interface

- 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> -n
openshift-storage
```

### <backingstore\_name>

Name of the backingstore.

### <PATH TO GCP PRIVATE KEY JSON FILE>

A path to your GCP private key created for this purpose.

### <GCP bucket name>

An existing GCP object storage bucket name. This argument tells the MCG 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"
```

Adding storage resources using a YAML

- 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>
```

### <GCP PRIVATE KEY ENCODED IN BASE64>

Provide and encode your own GCP service account private key using Base64, and use the results for this attribute.

### <backingstore-secret-name>

A unique name of the backingstore secret.

- 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:
  googleCloudStorage:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <target bucket>
    type: google-cloud-storage

```

**<target bucket>**

An existing Google storage bucket. This argument indicates to the MCG about the bucket to use as a target bucket for its backing store, and subsequently, data storage dfdand administration.

**<backingstore-secret-name>**

The name of the secret created in the previous step.

**10.3.2.5. Creating a local Persistent Volume-backed backingstore****Prerequisites**

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

```

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

```

**NOTE**

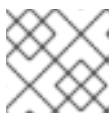
Specify the appropriate architecture for enabling the repositories using the subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```

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

```

- Alternatively, you can install the MCG package from the OpenShift Data Foundation 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)

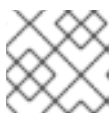
**NOTE**

Choose the correct Product Variant according to your architecture.

**Procedure**

*Using the MCG command-line interface*

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

**NOTE**

This command must be run from within the **openshift-storage** namespace.

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

**<backingstore\_name>**

The name of the backingstore.

**<NUMBER OF VOLUMES>**

The number of volumes you would like to create. Note that increasing the number of volumes scales up the storage.

**<VOLUME SIZE>**

Required size in GB of each volume.

**<LOCAL STORAGE CLASS>**

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

### 10.3.3. Creating an s3 compatible Multicloud Object Gateway backingstore

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

#### Procedure

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



#### NOTE

This command must be run from within the **openshift-storage** namespace.

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

- 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 the MCG which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.
- e. To get the **<RGW endpoint>**, see [Accessing the RADOS Object Gateway S3 endpoint](#). The output will be similar to the following:

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

You can also create the backingstore using a YAML:

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

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



- a. Replace **<RGW-Username>** and **<Display-name>** with a unique username and display name.
2. Apply the following YAML for an S3-Compatible backing store:

```

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

```

- a. Replace **<backingstore-secret-name>** with the name of the secret that was created with **CephObjectStore** in the previous step.
- b. Replace **<bucket-name>** with an existing RGW bucket name. This argument tells the MCG 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](#).

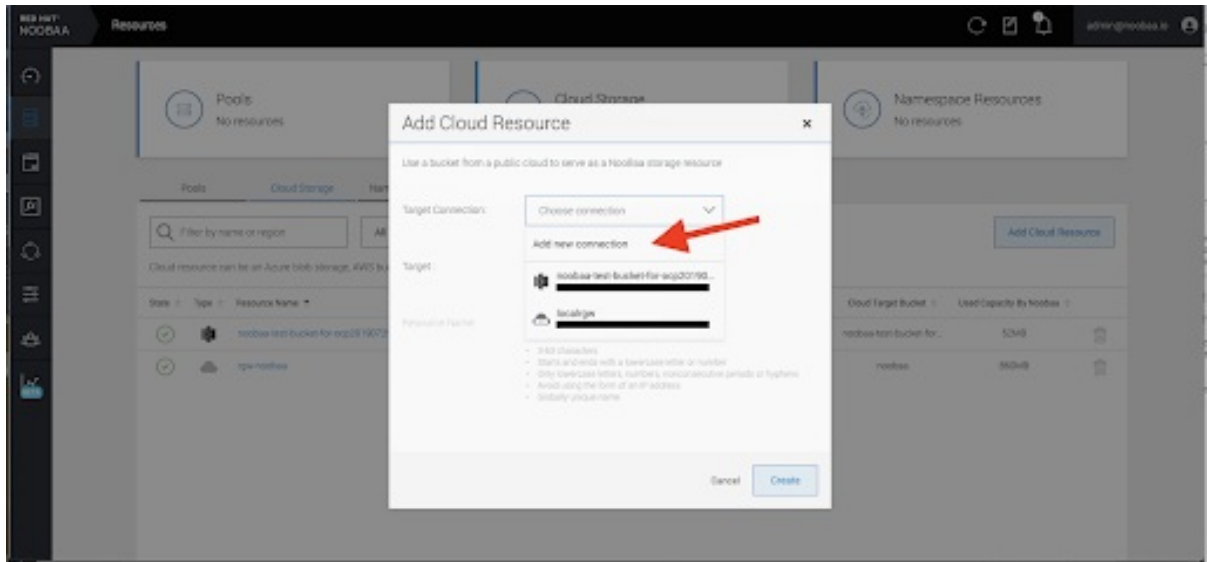
### 10.3.4. Adding storage resources for hybrid and Multicloud using the user interface

#### Procedure

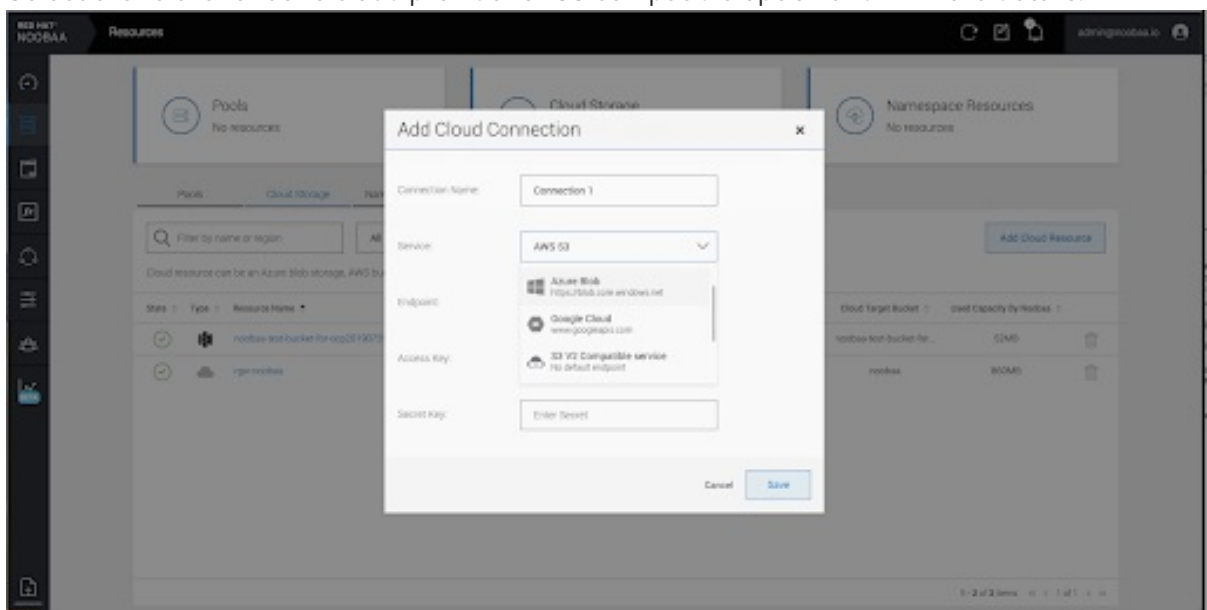
1. In the OpenShift Web Console, click **Storage → Data Foundation**.
2. In the **Storage Systems** tab, select the storage system and then click **Overview → Object** tab.
3. Select the **Multicloud Object Gateway** link.
1. Select the **Resources** tab in the left, highlighted below. From the list that populates, select **Add Cloud Resource**.



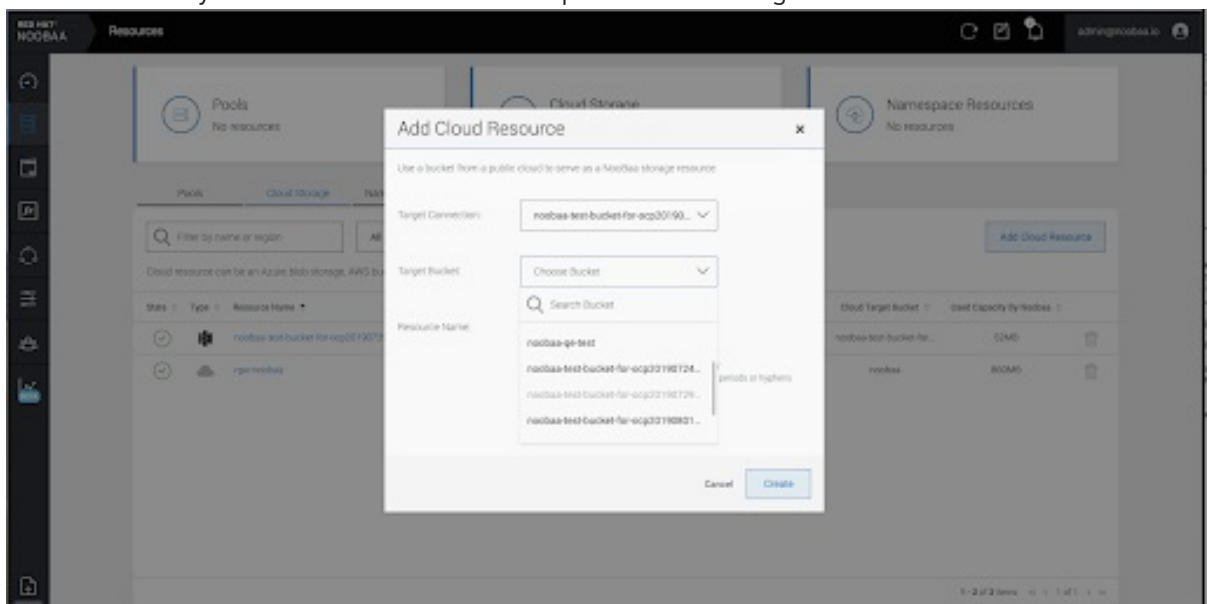
2. Select **Add new connection**.



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



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



5. 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.

### 10.3.5. Creating a new bucket class

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

Use this procedure to create a bucket class in OpenShift Data Foundation.

#### Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. Click the **Bucket Class** tab.
3. Click **Create Bucket Class**.
4. 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:
      - **Standard**: data will be consumed by a Multicloud Object Gateway (MCG), deduped, compressed and encrypted.
      - **Namespace**: data is stored on the NamespaceStores without performing de-duplication, compression or encryption. 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. In the OpenShift Web Console, click **Storage → Data Foundation**.
2. Click the **Bucket Class** tab and search the new Bucket Class.

### 10.3.6. 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 Web Console.

#### Procedure

1. In the OpenShift Web Console, click **Storage → Data Foundation**.
2. Click the **Bucket Class** tab.
3. Click the Action Menu ( **⋮** ) next to the Bucket class you want to edit.
4. Click **Edit Bucket Class**.
5. You are redirected to the **YAML** file, make the required changes in this file and click **Save**.

### 10.3.7. Editing backing stores for bucket class

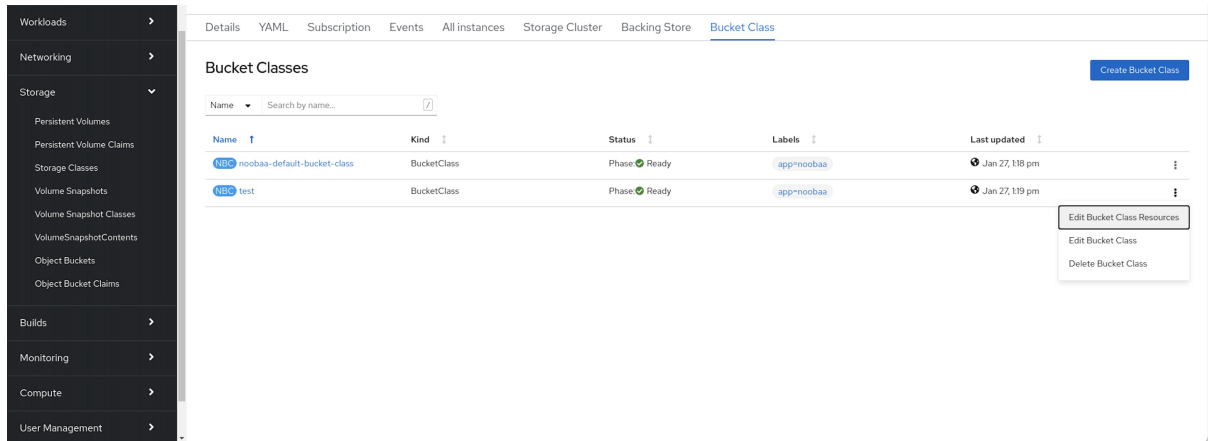
Use the following procedure to edit an existing Multicloud Object Gateway (MCG) bucket class to change the underlying backing stores used in a bucket class.

#### Prerequisites

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

#### Procedure

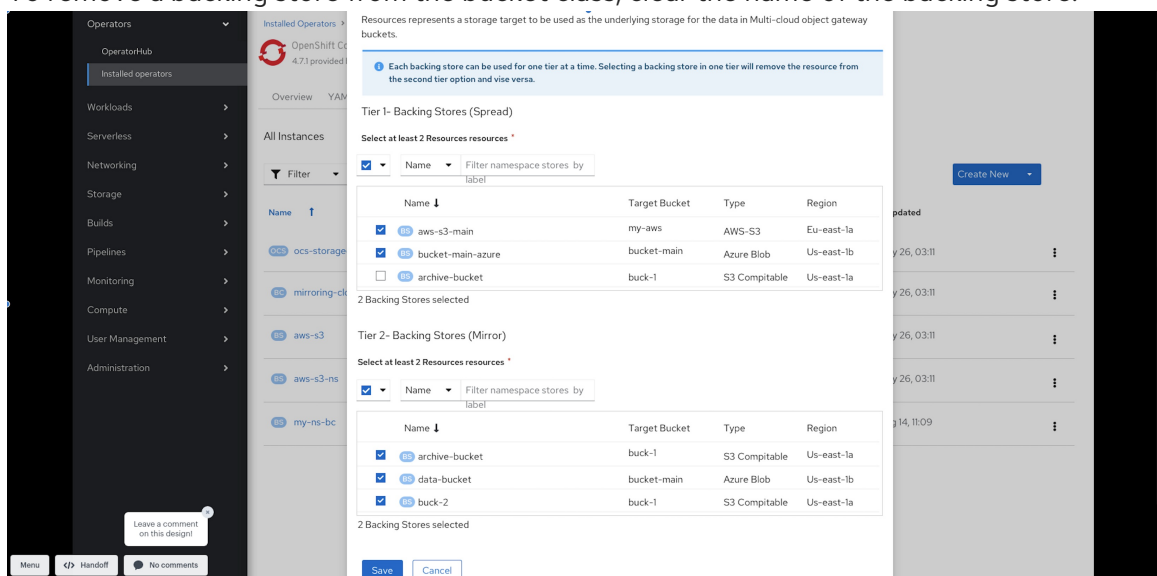
1. In the OpenShift Web Console, click **Storage → Data Foundation**.
2. Click the **Bucket Class** tab.
3. Click the Action Menu ( **⋮** ) next to the Bucket class you want to edit.



4. Click **Edit Bucket Class Resources**.

5. 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.



6. Click **Save**.

## 10.4. 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.

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.

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

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

Red Hat OpenShift Data Foundation 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](#)

### 10.4.2. 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](#)

### 10.4.2.1. Adding an AWS S3 namespace bucket using YAML

#### Prerequisites

- Openshift Container Platform with OpenShift Data Foundation operator installed.
- Access to the Multicloud Object Gateway (MCG).  
For information, see Chapter 2, [Accessing the Multicloud Object Gateway with your applications](#).

#### Procedure

1. Create a secret with the credentials:

```

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>

```

where **<namespacestore-secret-name>** is a unique NamespaceStore name.

You must provide 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>**.

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 MCG namespace buckets.

To create a NamespaceStore resource, apply the following 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

```

**<resource-name>**

The name you want to give to the resource.

**<namespacestore-secret-name>**

The secret created in the previous step.

**<namespace-secret>**

The namespace where the secret can be found.

**<target-bucket>**

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:

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

**<my-bucket-class>**

The unique namespace bucket class name.

**<resource>**

The name of a single NamespaceStore that defines the read and write target of the namespace bucket.

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

```
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>
```



**<my-bucket-class>**

A unique bucket class name.

**<write-resource>**

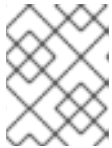
The name of a single NamespaceStore that defines the **write** target of the namespace bucket.

**<read-resources>**

A list of the names of the NamespaceStores that defines the **read** targets of the namespace bucket.

4. Create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in the earlier step using the following YAML:

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

**NOTE**

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

**<resource-name>**

The name you want to give to the resource.

**<my-bucket>**

The name you want to give to the bucket.

**<my-bucket-class>**

The bucket class created in the previous step.

After the OBC is provisioned by the operator, a bucket is created in the MCG, and the operator creates a **Secret** and **ConfigMap** with the same name and in the same namespace as that of the OBC.

#### 10.4.2.2. Adding an IBM COS namespace bucket using YAML

##### Prerequisites

- Openshift Container Platform with OpenShift Data Foundation operator installed.
- Access to the Multicloud Object Gateway (MCG), see Chapter 2, [Accessing the Multicloud Object Gateway with your applications](#).

##### Procedure

1. Create a secret with the credentials:

■

```

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>

```

#### <namespacestore-secret-name>

A unique NamespaceStore name.

You must provide 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>**.

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 MCG namespace buckets.

To create a NamespaceStore resource, apply the following 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

```

#### <IBM COS ENDPOINT>

The appropriate IBM COS endpoint.

#### <namespacestore-secret-name>

The secret created in the previous step.

#### <namespace-secret>

The namespace where the secret can be found.

#### <target-bucket>

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**.

- The namespace policy of type **single** requires the following configuration:

```

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

```

#### <my-bucket-class>

The unique namespace bucket class name.

#### <resource>

The name of a single NamespaceStore that defines the **read** and **write** target of the namespace bucket.

- The namespace policy of type **multi** requires the following configuration:

```

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>

```

#### <my-bucket-class>

The unique bucket class name.

#### <write-resource>

The name of a single NamespaceStore that defines the write target of the namespace bucket.

#### <read-resources>

A list of the NamespaceStores names that defines the **read** targets of the namespace bucket.

4. To create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in the previous step, apply the following YAML:

```

apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim

```

```

metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>

```



#### NOTE

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

#### <resource-name>

The name you want to give to the resource.

#### <my-bucket>

The name you want to give to the bucket.

#### <my-bucket-class>

The bucket class created in the previous step.

After the OBC is provisioned by the operator, a bucket is created in the MCG, and the operator creates a **Secret** and **ConfigMap** with the same name and in the same namespace as that of the OBC.

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

#### Prerequisites

- OpenShift Container Platform with OpenShift Data Foundation operator installed.
- Access to the Multicloud Object Gateway (MCG), see Chapter 2, [Accessing the Multicloud Object Gateway with your applications](#).
- Download the MCG command-line interface:

```

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

```



#### NOTE

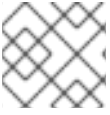
Specify the appropriate architecture for enabling the repositories using subscription manager. For instance, in case of IBM Z infrastructure use the following command:

```

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

```

Alternatively, you can install the MCG package from the OpenShift Data Foundation 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. In the MCG command-line interface, create a NamespaceStore resource.  
A NamespaceStore represents an underlying storage to be used as a **read** or **write** target for the data in MCG namespace buckets.

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

**<namespacestore>**

The name of the NamespaceStore.

**<AWS ACCESS KEY> and <AWS SECRET ACCESS KEY>**

The AWS access key ID and secret access key you created for this purpose.

**<bucket-name>**

The existing AWS bucket name. This argument tells the MCG 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 can be either **single** or **multi**.

- To create a namespace bucket class with a namespace policy of type **single**:

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

**<resource-name>**

The name you want to give the resource.

**<my-bucket-class>**

A unique bucket class name.

**<resource>**

A single namespace-store that defines the **read** and **write** target of the namespace bucket.

- 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
```

**<resource-name>**

The name you want to give the resource.

**<my-bucket-class>**

A unique bucket class name.

**<write-resource>**

A single namespace-store that defines the **write** target of the namespace bucket.

**<read-resources>**

A list of namespace-stores separated by commas that defines the **read** targets of the namespace bucket.

3. Create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in the previous step.

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

**<bucket-name>**

A bucket name of your choice.

**<custom-bucket-class>**

The name of the bucket class created in the previous step.

After the OBC is provisioned by the operator, a bucket is created in the MCG, and the operator creates a **Secret** and a **ConfigMap** with the same name and in the same namespace as that of the OBC.

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

##### Prerequisites

- Openshift Container Platform with OpenShift Data Foundation operator installed.
- Access to the Multicloud Object Gateway (MCG), see Chapter 2, [Accessing the Multicloud Object Gateway with your applications](#).
- Download the MCG command-line interface:

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



##### NOTE

Specify the appropriate architecture for enabling the repositories using subscription manager.

- For IBM Power, use the following command:

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

- For IBM Z infrastructure, use the following command:

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

Alternatively, you can install the MCG package from the OpenShift Data Foundation 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. In the MCG command-line interface, create a NamespaceStore resource.  
A NamespaceStore represents an underlying storage to be used as a **read** or **write** target for the data in the MCG namespace buckets.

```
$ 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
```

### <namespacestore>

The name of the NamespaceStore.

### <IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, <IBM COS ENDPOINT>

An IBM access key ID, secret access key, and the appropriate regional endpoint that corresponds to the location of the existing IBM bucket.

### <bucket-name>

An existing IBM bucket name. This argument tells the MCG 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**.

- To create a namespace bucket class with a namespace policy of type **single**:

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

### <resource-name>

The name you want to give the resource.

### <my-bucket-class>

A unique bucket class name.

### <resource>

A single NamespaceStore that defines the **read** and **write** target of the namespace bucket.

- 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
```

### <resource-name>

The name you want to give the resource.

### <my-bucket-class>

A unique bucket class name.

### <write-resource>

A single NamespaceStore that defines the **write** target of the namespace bucket.

### <read-resources>

A comma-separated list of NamespaceStores that defines the **read** targets of the namespace bucket.

3. Create a bucket using an Object Bucket Class (OBC) resource that uses the bucket class defined in the earlier step.

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

**<bucket-name>**

A bucket name of your choice.

**<custom-bucket-class>**

The name of the bucket class created in the previous step.

After the OBC is provisioned by the operator, a bucket is created in the MCG, and the operator creates a **Secret** and **ConfigMap** with the same name and in the same namespace as that of the OBC.

### 10.4.3. Adding a namespace bucket using the OpenShift Container Platform user interface

You can add namespace buckets using the OpenShift Container Platform user interface. For information about namespace buckets, see [Managing namespace buckets](#).

#### Prerequisites

- OpenShift Container Platform with OpenShift Data Foundation operator installed.
- Access to the Multicloud Object Gateway (MCG).

#### Procedure

1. Log into the OpenShift Web Console.
2. Click **Storage** → **Data Foundation**.
3. Click the **Namespace Store** tab to create a **namespacestore** resources to be used in the namespace bucket.
  - a. Click **Create namespace store**
  - b. Enter a namespacestore name.
  - c. Choose a provider.
  - d. Choose a region.
  - e. Either select an existing secret, or click **Switch to credentials** to create a secret by entering a secret key and secret access key.
  - f. Choose a target bucket.
  - g. Click **Create**.
  - h. Verify that the namespacestore is in the **Ready** state.
  - i. Repeat these steps until you have the desired amount of resources.
4. Click the **Bucket Class** tab → **Create a new Bucket Class**



- a. Select the **Namespace** radio button.
  - b. Enter a Bucket Class name.
  - c. (Optional) Add description.
  - d. Click **Next**.
5. Choose a namespace policy type for your namespace bucket, and then click **Next**.
  6. Select the target resources.
    - If your namespace policy type is **Single**, you need to choose a read resource.
    - If your namespace policy type is **Multi**, you need to choose read resources and a write resource.
    - If your namespace policy type is **Cache**, you need to choose a Hub namespace store that defines the read and write target of the namespace bucket.
  7. Click **Next**.
  8. Review your new bucket class, and then click **Create Bucketclass**.
  9. On the **BucketClass** page, verify that your newly created resource is in the **Created** phase.
  10. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
  11. In the **Status** card, click **Storage System** and click the storage system link from the pop up that appears.
  12. In the **Object** tab, click **Multicloud Object Gateway** → **Buckets** → **Namespace Buckets** tab .
  13. 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 created in the earlier step that the namespace bucket should read data from.
      - ii. If the namespace policy type you are using is **Multi**, then Under **Write Policy**, specify which namespace resource the namespace bucket should write data to.
      - iii. Click **Next**.
    - c. Click **Create**.

#### Verification steps

- 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. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS

You can use the simplified process of the Multicloud Object Gateway (MCG) to span data across cloud

providers and clusters. Before you create a bucket class that reflects the data management policy and mirroring, you must add a backing storage that can be used by the MCG. For information, see Chapter 4, [Section 10.3, “Adding storage resources for hybrid or Multicloud”](#).

You can set up mirroring data by using the OpenShift UI, YAML or MCG command-line interface.

See the following sections:

- [Section 6.2, Section 10.5.1, “Creating bucket classes to mirror data using the MCG command-line-interface”](#)
- [Section 6.3, Section 10.5.2, “Creating bucket classes to mirror data using a YAML”](#)

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

#### Prerequisites

- Ensure to download Multicloud Object Gateway (MCG) command-line interface.

#### Procedure

1. From the Multicloud Object Gateway (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 to generate a new bucket that will be mirrored between two locations:

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

### 10.5.2. Creating bucket classes to mirror data using a YAML

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

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    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 10.7, "Object Bucket Claim"](#).

## 10.6. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY

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

### 10.6.1. Introduction to 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](#).

### 10.6.2. Using bucket policies in Multicloud Object Gateway

#### Prerequisites

- A running OpenShift Data Foundation Platform.
- Access to the Multicloud Object Gateway (MCG), see [Section 10.2, "Accessing the Multicloud Object Gateway with your applications"](#)

#### Procedure

To use bucket policies in the MCG:

1. Create the bucket policy in JSON format.  
For example:

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

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

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

- Replace ***ENDPOINT*** with the S3 endpoint.
- Replace ***MyBucket*** with the bucket to set the policy on.
- Replace ***BucketPolicy*** with the bucket policy JSON file.
- Add **--no-verify-ssl** if you are using the default self signed certificates.  
For example:

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

For more information on the **put-bucket-policy** command, see the [AWS CLI Command Reference for put-bucket-policy](#).



#### NOTE

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



#### NOTE

Bucket policy conditions are not supported.

#### Additional resources

- 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](#) .

### 10.6.3. Creating a user in the Multicloud Object Gateway

#### Prerequisites

- A running OpenShift Data Foundation Platform.
- Download the MCG command-line interface for easier management.

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

**NOTE**

Specify the appropriate architecture for enabling the repositories using the subscription manager.

- For IBM Power, use the following command:

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

- For IBM Z infrastructure, use the following command:

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

- Alternatively, you can install the MCG package from the OpenShift Data Foundation RPMs found at [Download RedHat OpenShift Data Foundation page](#).

**NOTE**

Choose the correct Product Variant according to your architecture.

**Procedure**

Execute the following command to create an MCG user account:

```
noobaa account create <noobaa-account-name> [--allow_bucket_create=true] [--allowed_buckets=[]]
[--default_resource=""] [--full_permission=false]
```

**<noobaa-account-name>**

Specify the name of the new MCG user account.

**--allow\_bucket\_create**

Allows the user to create new buckets.

**--allowed\_buckets**

Sets the user's allowed bucket list (use commas or multiple flags).

**--default\_resource**

Sets the default resource. The new buckets are created on this default resource (including the future ones).

**--full\_permission**

Allows this account to access all existing and future buckets.

**IMPORTANT**

You need to provide permission to access at least one bucket or full permission to access all the buckets.

**10.7. 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 in three ways:

- [Section 10.7.1, “Dynamic Object Bucket Claim”](#)
- [Section 10.7.2, “Creating an Object Bucket Claim using the command line interface”](#)
- [Section 10.7.3, “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.

### 10.7.1. Dynamic Object Bucket Claim

Similar to Persistent Volumes, you can add the details of the Object Bucket claim (OBC) 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.



#### NOTE

The Multicloud Object Gateway endpoints uses self-signed certificates only if OpenShift uses self-signed certificates. Using signed certificates in OpenShift automatically replaces the Multicloud Object Gateway endpoints certificates with signed certificates. Get the certificate currently used by Multicloud Object Gateway by accessing the endpoint via the browser. See [Accessing the Multicloud Object Gateway with your applications](#) for more information.

#### Procedure

1. Add the following lines to your application YAML:

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

These lines are the OBC itself.

- a. Replace **<obc-name>** with the a unique OBC name.
  - b. Replace **<obc-bucket-name>** with a unique bucket name for your OBC.
2. To automate the use of the OBC add more lines to the YAML file.  
For example:

```
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

```

The example is the mapping between the bucket claim result, which is a configuration map with data and a secret with the credentials. This specific job claims the Object Bucket from NooBaa, which creates a bucket and an account.

- a. Replace all instances of **<obc-name>** with your OBC name.
  - b. Replace **<your application image>** with your application image.
3. Apply the updated YAML file:

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

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> -o yaml
```

Replace **obc-name** with the name of your OBC.

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
```

**<obc\_name>**

Specify the name of the object bucket claim.

## 10.7.2. Creating an Object Bucket Claim using the command line interface

When creating an Object Bucket Claim (OBC) 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 Multicloud Object Gateway (MCG) command-line interface.

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

### NOTE

Specify the appropriate architecture for enabling the repositories using the subscription manager.

- For IBM Power, use the following command:

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

- For IBM Z infrastructure, use the following command:

```
# subscription-manager repos --enable=rh-odf-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 OBC name, for example, **myappobc**.

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

For example:

```
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 OBC:

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

For example:

```
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 OBC:

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

For example:

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

```
storageClassName: openshift-storage.noobaa.io
status:
  phase: Bound
```

4. Inside of your **openshift-storage** namespace, you can find the configuration map and the secret to use this OBC. The CM and the secret have the same name as the OBC. Run the following command to view the secret:

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

For example:

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
  Wi9kcFluSWxHRzIWaFlzNk1hc0xma2JXcjM1MVhqa051SIBleXpmOQ==
kind: Secret
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
  resourceVersion: "40751"
  selfLink: /api/v1/namespaces/openshift-storage/secrets/test21obc
  uid: 65117c1c-f662-11e9-9094-0a5305de57bb
type: Opaque
```

The secret gives you the S3 access credentials.

5. Run the following command to view the configuration map:

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

For example:

```
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.

### 10.7.3. 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 10.7.1, "Dynamic Object Bucket Claim"](#).

#### Procedure

1. Log into the OpenShift Web Console.
2. On the left navigation bar, click **Storage** → **Object Bucket Claims** → **Create Object Bucket Claim**.
  - a. Enter a name for your object bucket claim and select the appropriate storage class based on your deployment, internal or external, from the dropdown menu:

##### Internal mode

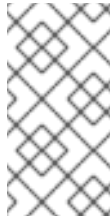
The following storage classes, which were created after deployment, are available for use:

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

##### External mode

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

- **ocs-external-storagecluster-ceph-rgw** uses the RGW
- **openshift-storage.noobaa.io** uses the MCG



#### NOTE

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

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


### 10.7.4. 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

- On the left navigation bar, click **Storage → Object Bucket Claims**.
- Click the Action menu (  ) next to the OBC you created.
  - From the drop-down menu, select **Attach to Deployment**
  - Select the desired deployment from the Deployment Name list, then click **Attach**.

### 10.7.5. 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


- Log into the OpenShift Web Console.
- On the left navigation bar, click **Storage → Object Buckets**.  
Optional: 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.
- Select the object bucket of which you want to see the details. Once selected you are navigated to the **Object Bucket Details** page.

## 10.7.6. Deleting Object Bucket Claims

### Prerequisites

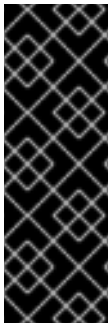
- Administrative access to the OpenShift Web Console.

### Procedure

1. On the left navigation bar, click **Storage** → **Object Bucket Claims**.
2. Click the Action menu (  ) next to the Object Bucket Claim (OBC) you want to delete.
  - a. Select **Delete Object Bucket Claim**.
  - b. Click **Delete**.

## 10.8. 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](#)

### 10.8.1. Creating an AWS cache bucket

#### Prerequisites

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

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



### NOTE

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

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

Alternatively, you can install the MCG package from the OpenShift Data Foundation 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 the MCG namespace buckets. From the MCG command-line interface, run the following command:

```
noobaa 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 the MCG 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:

```
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
```

- d. Replace **<namespacestore>** with a unique name.
  - e. Replace **<namespacestore-secret-name>** with the secret created in the previous step.
  - f. Replace **<namespace-secret>** with the namespace used to create the secret in the previous step.
  - g. 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-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 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 (OBC) 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.

## 10.8.2. Creating an IBM COS cache bucket

### Prerequisites

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

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

**NOTE**

Specify the appropriate architecture for enabling the repositories using the subscription manager.

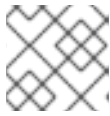
- For IBM Power, use the following command:

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

- For IBM Z infrastructure, use the following command:

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

Alternatively, you can install the MCG package from the OpenShift Data Foundation 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**

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

```
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>
```

- Replace **<namespacestore>** with the name of the NamespaceStore.
- 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.
- Replace **<bucket-name>** with an existing IBM bucket name. This argument tells the MCG 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:

```
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:

```
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.

## 10.9. 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

### 10.9.1. Scaling the Multicloud Object Gateway with storage nodes

#### Prerequisites

- A running OpenShift Data Foundation cluster on OpenShift Container Platform with access to the Multicloud Object Gateway (MCG).

A storage node in the MCG is a NooBaa daemon container attached to one or more Persistent Volumes (PVs) 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

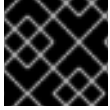
1. Log in to **OpenShift Web Console**.
2. From the MCG user interface, click **Overview** → **Add Storage Resources**.
3. In the window, click **Deploy Kubernetes Pool**
4. In the **Create Pool** step create the target pool for the future installed nodes.
5. In the **Configure** step, configure the number of requested pods and the size of each PV. For each new pod, one PV is to be created.
6. 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.
7. All nodes will be assigned to the pool you chose in the first step, and can be found under **Resources** → **Storage resources** → **Resource name**.

## 10.10. 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. OpenShift Data Foundation 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.

## CHAPTER 11. MANAGING PERSISTENT VOLUME CLAIMS



### IMPORTANT

Expanding PVCs is not supported for PVCs backed by OpenShift Data Foundation.

### 11.1. CONFIGURING APPLICATION PODS TO USE OPENSIFT DATA FOUNDATION

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

#### Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Data Foundation 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 Data Foundation are available. In OpenShift Web Console, click **Storage** → **StorageClasses** 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 Data Foundation.
    - ii. Specify the PVC **Name**, for example, **myclaim**.
    - iii. Select the required **Access Mode**.



### NOTE

The **Access Mode, Shared access (RWX)** is not supported in IBM FlashSystem.

- 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 **volumes:** 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 **volumes:** 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 Data Foundation.

### 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. In the OpenShift Web Console, click **Storage** → **Data Foundation**
2. In the **Storage systems** tab, select the storage system and then click **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 Data Foundation

Red Hat OpenShift Data Foundation 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 Data Foundation 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 Data Foundation 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:

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



### IMPORTANT

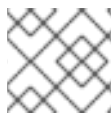
Any chosen provisioner plug-in also requires configuration for the relevant cloud, host, or third-party provider as per the relevant documentation.



## 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 Data Foundation 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.

**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.

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](mailto: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](mailto: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.

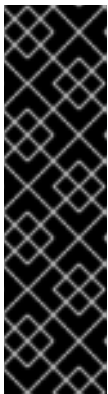


#### NOTE

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](mailto: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:

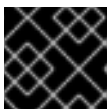
```
$ oc adm drain <node_name> --force --delete-emptydir-data=true --ignore-daemonsets
```



#### 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 Data Foundation 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 Data Foundation label to the new node:

```
$ 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:

```
$ 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 Data Foundation pods are in **Running** state.
4. Verify that new OSD pods are running on the replacement node.

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

5. Optional: If 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-device** name(s)

```
$ lsblk
```

6. If verification steps fail, [contact Red Hat Support](#).

## 14.2. REPLACING FAILED NODES ON GOOGLE CLOUD INSTALLER-PROVISIONED INFRASTRUCTURE

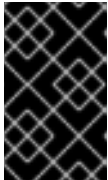
Perform this procedure to replace a failed node which is not operational on Google Cloud installer-provisioned infrastructure (IPI) for OpenShift Data Foundation.

### 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 Data Foundation 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 Data Foundation label to the new node:

```
$ 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:

```
$ 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 Data Foundation pods are in **Running** state.
4. Verify that new OSD pods are running on the replacement node.

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

5. Optional: If 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-device** 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. UPGRADING TO OPENSIFT DATA FOUNDATION

## 16.1. OVERVIEW OF THE OPENSIFT DATA FOUNDATION UPDATE PROCESS

This chapter helps you to upgrade between the minor releases and z-streams for all Red Hat OpenShift Data Foundation deployments (Internal, Internal-Attached and External). The upgrade process remains the same for all deployments.

You can upgrade OpenShift Data Foundation and its components, either between minor releases like 4.10 and 4.11, or between z-stream updates like 4.11.0 and 4.11.1 by enabling automatic updates (if not done so during operator installation) or performing manual updates. When a new z-stream release becomes available, the upgrade process triggers automatically if the update strategy was set to Automatic.

You also need to upgrade the different parts of Red Hat OpenShift Data Foundation in the following order for both internal and external mode deployments:

1. **Update OpenShift Container Platform** according to the [Updating clusters](#) documentation for OpenShift Container Platform.
2. **Update Red Hat OpenShift Data Foundation.**
  - 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 Data Foundation as well as Local Storage Operator when in use.
  - b. **For updating between minor releases**, see [Updating Red Hat OpenShift Data Foundation 4.10 to 4.11](#).
  - c. **For updating between z-stream releases**, see [Updating Red Hat OpenShift Data Foundation 4.11.x to 4.11.y](#).
  - d. **For updating external mode deployments** you must also perform the steps from section [Updating the Red Hat OpenShift Data Foundation external secret](#) .
  - e. **If you use local storage, then update the Local Storage operator** See [Checking for Local Storage Operator deployments](#) if you are unsure.

### Update considerations

Review the following important considerations before you begin.

- The Red Hat OpenShift Container Platform version is the same as Red Hat OpenShift Data Foundation.  
See the [Interoperability Matrix](#) for more information about supported combinations of OpenShift Container Platform and Red Hat OpenShift Data Foundation.
- To know whether your cluster was deployed in internal or external mode, refer to the [knowledgebase article](#) on *How to determine if ODF cluster has storage in internal or external mode*.
- The Local Storage Operator is fully supported only when the Local Storage Operator version matches the Red Hat OpenShift Container Platform version.

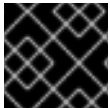
- The flexible scaling feature is available only in new deployments of OpenShift Data Foundation. For more information, see [Scaling storage guide](#).

## 16.2. UPDATING RED HAT OPENSIFT DATA FOUNDATION 4.10 TO 4.11

This chapter helps you to upgrade between the minor releases for all Red Hat OpenShift Data Foundation deployments (Internal, Internal-Attached and External). The upgrade process remains the same for all deployments. The Only difference is what gets upgraded and what's not.

- For Internal and Internal-attached deployments, upgrading OpenShift Data Foundation upgrades all OpenShift Data Foundation services including the backend Ceph Storage cluster.
- For External mode deployments, upgrading OpenShift Data Foundation only upgrades the OpenShift Data Foundation service while the backend Ceph storage cluster remains untouched and needs to be upgraded separately.

We recommend upgrading RHCS along with OpenShift Data Foundation in order to get new feature support, security fixes, and other bug fixes. Since we do not have a strong dependency on RHCS upgrade, you can upgrade the OpenShift Data Foundation operator first followed by RHCS upgrade or vice-versa. See [solution](#) to know more about Red Hat Ceph Storage releases.

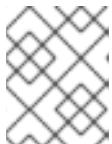


### IMPORTANT

Upgrading to 4.11 directly from any version older than 4.10 is unsupported.

### Prerequisites

- Ensure that the **OpenShift Container Platform** cluster has been updated to the latest stable release of version 4.11.X, see [Updating Clusters](#).
- Ensure that the **OpenShift Data Foundation** cluster is healthy and data is resilient.
  - Navigate to **Storage → Data Foundation → Storage Systemstab** and then click on the storage system name.
  - Check for the green tick on the status card of both **Overview - Block and File** and **Object** tabs. Green tick indicates that the *storage cluster*, *object service* and *data resiliency* are all healthy.
- Ensure that all **OpenShift Data Foundation** Pods, including the operator pods, are in **Running** state in the **openshift-storage** namespace.  
To view the state of the pods, on the OpenShift Web Console, click **Workloads → Pods** Select **openshift-storage** from the **Project** drop-down list.



### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

- Ensure that you have sufficient time to complete the OpenShift Data Foundation update process, as the update time varies depending on the number of OSDs that run in the cluster.

### Procedure

1. On the OpenShift Web Console, navigate to **Operators → Installed Operators**

2. Select **openshift-storage** project.
3. Click the OpenShift Data Foundation operator name.
4. Click the **Subscription** tab and click the link under **Update Channel**.
5. Select the **Stable-4.11** update channel and **Save** it.
6. If the **Upgrade status** shows **requires approval**, click on **requires approval**.
  - a. On the Install Plan Details page, click **Preview Install Plan**.
  - b. Review the install plan and click **Approve**.  
Wait for the **Status** to change from **Unknown** to **Created**.
7. Navigate to **Operators → Installed Operators**.
8. Select the **openshift-storage** project.  
Wait for the OpenShift Data Foundation Operator **Status** to change to **Up to date**.

### Verification steps

- Check the **Version** below the OpenShift Data Foundation name and check the operator status.
  - Navigate to **Operators → Installed Operators** and select the **openshift-storage** project.
  - When the upgrade completes, the version updates to a new version number for OpenShift Data Foundation and status changes to **Succeeded** with a green tick.
- Verify that the **OpenShift Data Foundation** cluster is healthy and data is resilient.
  - Navigate to **Storage → Data Foundation → Storage Systems** tab and then click on the storage system name.
  - Check for the green tick on the status card of **Overview- Block and File** and **Object** tabs. Green tick indicates that the storage cluster, object service and data resiliency is healthy.
- If verification steps fail, [contact Red Hat Support](#).



### IMPORTANT

After updating external mode deployments, you must also update the external secret. For instructions, see [Updating the OpenShift Data Foundation external secret](#).

### Additional Resources

If you face any issues while updating OpenShift Data Foundation, see the *Commonly required logs for troubleshooting* section in the [Troubleshooting guide](#).

## 16.3. UPDATING RED HAT OPENSIFT DATA FOUNDATION 4.11.X TO 4.11.Y

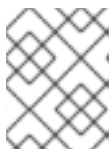
This chapter helps you to upgrade between the z-stream release for all Red Hat OpenShift Data Foundation deployments (Internal, Internal-Attached and External). The upgrade process remains the same for all deployments. The Only difference is what gets upgraded and what's not.

- For Internal and Internal-attached deployments, upgrading OpenShift Data Foundation upgrades all OpenShift Data Foundation services including the backend Ceph Storage cluster.
- For External mode deployments, upgrading OpenShift Data Foundation only upgrades the OpenShift Data Foundation service while the backend Ceph storage cluster remains untouched and needs to be upgraded separately.  
Hence, we recommend upgrading RHCS along with OpenShift Data Foundation in order to get new feature support, security fixes, and other bug fixes. Since we do not have a strong dependency on RHCS upgrade, you can upgrade the OpenShift Data Foundation operator first followed by RHCS upgrade or vice-versa. See [solution](#) to know more about Red Hat Ceph Storage releases.

When a new z-stream release becomes available, the upgrade process triggers automatically if the update strategy was set to **Automatic**. If the update strategy is set to **Manual** then use the following procedure.

### Prerequisites

- Ensure that the **OpenShift Container Platform** cluster has been updated to the latest stable release of version 4.11.X, see [Updating Clusters](#).
- Ensure that the **OpenShift Data Foundation** cluster is healthy and data is resilient.
  - Navigate to **Storage → Data Foundation → Storage Systemstab** and then click on the storage system name.
  - Check for the green tick on the status card of **Overview - Block and File** and **Object** tabs. Green tick indicates that the storage cluster, object service and data resiliency is healthy.
- Ensure that all **OpenShift Data Foundation** Pods, including the operator pods, are in **Running** state in the **openshift-storage** namespace.  
To view the state of the pods, on the OpenShift Web Console, click **Workloads → Pods** Select **openshift-storage** from the **Project** drop-down list.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

- Ensure that you have sufficient time to complete the OpenShift Data Foundation update process, as the update time varies depending on the number of OSDs that run in the cluster.

### Procedure

1. On the OpenShift Web Console, navigate to **Operators → Installed Operators**
2. Select **openshift-storage** project.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

3. Click the **OpenShift Data Foundation** operator name.

4. Click the **Subscription** tab.
5. If the **Upgrade Status** shows **require approval**, click on **requires approval** link.
6. On the **InstallPlan Details** page, click **Preview Install Plan**.
7. Review the install plan and click **Approve**.
8. Wait for the Status to change from **Unknown** to **Created**.

### Verification steps

- Check the **Version** below the OpenShift Data Foundation name and check the operator status.
  - Navigate to **Operators → Installed Operators** and select the **openshift-storage** project.
  - When the upgrade completes, the version updates to a new version number for OpenShift Data Foundation and status changes to **Succeeded** with a green tick.
- Verify that the OpenShift Data Foundation cluster is healthy and data is resilient.
  - Navigate to **Storage → Data Foundation → Storage System** tab and then click on the storage system name.
  - Check for the green tick on the status card of **Overview - Block and File** and **Object** tabs. Green tick indicates that the storage cluster, object service and data resiliency is healthy.
- If verification steps fail, [contact Red Hat Support](#).

## 16.4. CHANGING THE UPDATE APPROVAL STRATEGY

To ensure that the storage system gets updated automatically when a new update is available in the same channel, we recommend keeping the update approval strategy to **Automatic**. Changing the update approval strategy to **Manual** will need manual approval for each upgrade.

### Procedure

1. Navigate to **Operators → Installed Operators**
2. Select **openshift-storage** from the **Project** drop-down list.



#### NOTE

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

3. Click on **OpenShift Data Foundation** operator name
4. Go to the **Subscription** tab.
5. Click on the **pencil** icon for changing the **Update approval**.
6. Select the update approval strategy and click **Save**.

### Verification steps



- Verify that the Update approval shows the newly selected approval strategy below it.