



# Red Hat OpenShift Data Foundation 4.15

## Managing and allocating storage resources

Instructions on how to allocate storage to core services and hosted applications in OpenShift Data Foundation, including snapshot and clone.



## Red Hat OpenShift Data Foundation 4.15 Managing and allocating storage resources

---

Instructions on how to allocate storage to core services and hosted applications in OpenShift Data Foundation, including snapshot and clone.

## 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

This document explains how to allocate storage to core services and hosted applications in Red Hat OpenShift Data Foundation.

## Table of Contents

<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>4</b>
<b>PROVIDING FEEDBACK ON RED HAT DOCUMENTATION</b> .....	<b>5</b>
<b>CHAPTER 1. OVERVIEW</b> .....	<b>6</b>
<b>CHAPTER 2. STORAGE CLASSES</b> .....	<b>7</b>
2.1. CREATING STORAGE CLASSES AND POOLS	7
2.2. STORAGE CLASS FOR PERSISTENT VOLUME ENCRYPTION	8
2.2.1. Access configuration for Key Management System (KMS)	8
2.2.1.1. Configuring access to KMS using vaulttokens	9
2.2.1.2. Configuring access to KMS using Thales CipherTrust Manager	9
2.2.1.3. Configuring access to KMS using vaulttenantsa	10
2.2.2. Creating a storage class for persistent volume encryption	14
2.2.2.1. Overriding Vault connection details using tenant ConfigMap	16
2.3. STORAGE CLASS WITH SINGLE REPLICA	17
2.3.1. Recovering after OSD lost from single replica	19
<b>CHAPTER 3. BLOCK POOLS</b> .....	<b>21</b>
3.1. CREATING A BLOCK POOL	21
3.2. UPDATING AN EXISTING POOL	21
3.3. DELETING A POOL	22
<b>CHAPTER 4. CONFIGURE STORAGE FOR OPENSIFT CONTAINER PLATFORM SERVICES</b> .....	<b>23</b>
4.1. CONFIGURING IMAGE REGISTRY TO USE OPENSIFT DATA FOUNDATION	23
4.2. USING MULTICLOUD OBJECT GATEWAY AS OPENSIFT IMAGE REGISTRY BACKEND STORAGE	25
4.3. CONFIGURING MONITORING TO USE OPENSIFT DATA FOUNDATION	28
4.4. OVERPROVISION LEVEL POLICY CONTROL [TECHNOLOGY PREVIEW]	31
4.5. CLUSTER LOGGING FOR OPENSIFT DATA FOUNDATION	32
4.5.1. Configuring persistent storage	33
4.5.2. Configuring cluster logging to use OpenShift data Foundation	34
<b>CHAPTER 5. CREATING MULTUS NETWORKS</b> .....	<b>37</b>
5.1. CREATING NETWORK ATTACHMENT DEFINITIONS	37
<b>CHAPTER 6. BACKING OPENSIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSIFT DATA FOUNDATION</b> .....	<b>40</b>
<b>CHAPTER 7. ADDING FILE AND OBJECT STORAGE TO AN EXISTING EXTERNAL OPENSIFT DATA FOUNDATION CLUSTER</b> .....	<b>42</b>
<b>CHAPTER 8. HOW TO USE DEDICATED WORKER NODES FOR RED HAT OPENSIFT DATA FOUNDATION</b> .	<b>46</b>
8.1. ANATOMY OF AN INFRASTRUCTURE NODE	46
8.2. MACHINE SETS FOR CREATING INFRASTRUCTURE NODES	46
8.3. MANUAL CREATION OF INFRASTRUCTURE NODES	47
8.4. TAINT A NODE FROM THE USER INTERFACE	48
<b>CHAPTER 9. MANAGING PERSISTENT VOLUME CLAIMS</b> .....	<b>50</b>
9.1. CONFIGURING APPLICATION PODS TO USE OPENSIFT DATA FOUNDATION	50
9.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS	51
9.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS	52
9.4. EXPANDING PERSISTENT VOLUME CLAIMS	52
9.5. DYNAMIC PROVISIONING	54
9.5.1. About dynamic provisioning	54

9.5.2. Dynamic provisioning in OpenShift Data Foundation	54
9.5.3. Available dynamic provisioning plug-ins	55
<b>CHAPTER 10. RECLAIMING SPACE ON TARGET VOLUMES</b> .....	<b>57</b>
10.1. ENABLING RECLAIM SPACE OPERATION USING ANNOTATING PERSISTENTVOLUMECLAIMS	57
10.2. ENABLING RECLAIM SPACE OPERATION USING RECLAIMSPACEJOB	58
10.3. ENABLING RECLAIM SPACE OPERATION USING RECLAIMSPACECRONJOB	59
10.4. CUSTOMISING TIMEOUTS REQUIRED FOR RECLAIM SPACE OPERATION	60
<b>CHAPTER 11. VOLUME SNAPSHOTS</b> .....	<b>62</b>
11.1. CREATING VOLUME SNAPSHOTS	62
11.2. RESTORING VOLUME SNAPSHOTS	63
11.3. DELETING VOLUME SNAPSHOTS	65
<b>CHAPTER 12. VOLUME CLONING</b> .....	<b>67</b>
12.1. CREATING A CLONE	67
<b>CHAPTER 13. MANAGING CONTAINER STORAGE INTERFACE (CSI) COMPONENT PLACEMENTS</b> .....	<b>68</b>
<b>CHAPTER 14. CREATING EXPORTS USING NFS</b> .....	<b>70</b>
14.1. ENABLING THE NFS FEATURE	70
14.2. CREATING NFS EXPORTS	71
14.3. CONSUMING NFS EXPORTS IN-CLUSTER	72
14.4. CONSUMING NFS EXPORTS EXTERNALLY FROM THE OPENSIFT CLUSTER	73
<b>CHAPTER 15. ANNOTATING ENCRYPTED RBD STORAGE CLASSES</b> .....	<b>76</b>



## 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**.

## CHAPTER 1. OVERVIEW

Read this document to understand how to create, configure, and allocate storage to core services or hosted applications in Red Hat OpenShift Data Foundation.

- [Chapter 2, \*Storage classes\*](#) shows you how to create custom storage classes.
- [Chapter 3, \*Block pools\*](#) provides you with information on how to create, update and delete block pools.
- [Chapter 4, \*Configure storage for OpenShift Container Platform services\*](#) shows you how to use OpenShift Data Foundation for core OpenShift Container Platform services.
- [Chapter 6, \*Backing OpenShift Container Platform applications with OpenShift Data Foundation\*](#) provides information about how to configure OpenShift Container Platform applications to use OpenShift Data Foundation.
- [Adding file and object storage to an existing external OpenShift Data Foundation cluster](#)
- [Chapter 8, \*How to use dedicated worker nodes for Red Hat OpenShift Data Foundation\*](#) provides information about how to use dedicated worker nodes for Red Hat OpenShift Data Foundation.
- [Chapter 9, \*Managing Persistent Volume Claims\*](#) provides information about managing Persistent Volume Claim requests, and automating the fulfillment of those requests.
- [Chapter 10, \*Reclaiming space on target volumes\*](#) shows you how to reclaim the actual available storage space.
- [Chapter 11, \*Volume Snapshots\*](#) shows you how to create, restore, and delete volume snapshots.
- [Chapter 12, \*Volume cloning\*](#) shows you how to create volume clones.
- [Chapter 13, \*Managing container storage interface \(CSI\) component placements\*](#) provides information about setting tolerations to bring up container storage interface component on the nodes.

## CHAPTER 2. STORAGE CLASSES

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 custom storage classes to use other storage resources or to offer a different behavior to applications.



### NOTE

Custom storage classes are not supported for *external mode* OpenShift Data Foundation clusters.

## 2.1. 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. Choose a **Storage system** for your workloads.
8. 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.
9. Optional: Select **Enable Encryption** checkbox.
  10. Click **Create** to create the storage class.

## 2.2. STORAGE CLASS FOR PERSISTENT VOLUME ENCRYPTION

Persistent volume (PV) encryption guarantees isolation and confidentiality between tenants (applications). Before you can use PV encryption, you must create a storage class for PV encryption. Persistent volume encryption is only available for RBD PVs.

OpenShift Data Foundation supports storing encryption passphrases in HashiCorp Vault and Thales CipherTrust Manager. You can create an encryption enabled storage class using an external key management system (KMS) for persistent volume encryption. You need to configure access to the KMS before creating the storage class.



### NOTE

For PV encryption, you must have a valid Red Hat OpenShift Data Foundation Advanced subscription. For more information, see the [knowledgebase article on OpenShift Data Foundation subscriptions](#).

### 2.2.1. Access configuration for Key Management System (KMS)

Based on your use case, you need to configure access to KMS using one of the following ways:

- Using **vaulttokens**: allows users to authenticate using a token
- Using **Thales CipherTrust Manager**: uses Key Management Interoperability Protocol (KMIP)
- Using **vaulttenantsa** (Technology Preview): allows users to use **serviceaccounts** to authenticate with **Vault**



## IMPORTANT

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

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

### 2.2.1.1. Configuring access to KMS using vaulttokens

#### Prerequisites

- The OpenShift Data Foundation cluster is in **Ready** state.
- On the external key management system (KMS),
  - Ensure that a policy with a token exists and the key value backend path in **Vault** is enabled.
  - Ensure that you are using signed certificates on your **Vault** servers.

#### Procedure

Create a secret in the tenant's namespace.

1. In the OpenShift Container Platform web console, navigate to **Workloads → Secrets**
2. Click **Create → Key/value secret**
3. Enter **Secret Name** as **ceph-csi-kms-token**.
4. Enter **Key** as **token**.
5. Enter **Value**.  
It is the token from Vault. You can either click **Browse** to select and upload the file containing the token or enter the token directly in the text box.
6. Click **Create**.




## NOTE

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

### 2.2.1.2. Configuring access to KMS using Thales CipherTrust Manager

#### Prerequisites

1. Create a KMIP client if one does not exist. From the user interface, select **KMIP → Client Profile → Add Profile**.
  - a. Add the **CipherTrust** username to the **Common Name** field during profile creation.

2. Create a token by navigating to **KMIP → Registration Token → New Registration Token**. Copy the token for the next step.
3. To register the client, navigate to **KMIP → Registered Clients → Add Client**. Specify the **Name**. Paste the **Registration Token** from the previous step, then click **Save**.
4. Download the Private Key and Client Certificate by clicking **Save Private Key** and **Save Certificate** respectively.
5. To create a new KMIP interface, navigate to **Admin Settings → Interfaces → Add Interface**.
  - a. Select **KMIP Key Management Interoperability Protocol** and click **Next**.
  - b. Select a free **Port**.
  - c. Select **Network Interface** as **all**.
  - d. Select **Interface Mode** as **TLS, verify client cert, user name taken from client cert, auth request is optional**.
  - e. (Optional) You can enable hard delete to delete both meta-data and material when the key is deleted. It is disabled by default.
  - f. Select the CA to be used, and click **Save**.
6. To get the server CA certificate, click on the Action menu (  ) on the right of the newly created interface, and click **Download Certificate**.

### Procedure

1. To create a key to act as the Key Encryption Key (KEK) for storageclass encryption, follow the steps below:
  - a. Navigate to **Keys → Add Key**.
  - b. Enter **Key Name**.
  - c. Set the **Algorithm** and **Size** to **AES** and **256** respectively.
  - d. Enable **Create a key in Pre-Active state** and set the date and time for activation.
  - e. Ensure that **Encrypt** and **Decrypt** are enabled under **Key Usage**.
  - f. Copy the ID of the newly created Key to be used as the Unique Identifier during deployment.

### 2.2.1.3. Configuring access to KMS using vaulttenantsa

#### Prerequisites

- The OpenShift Data Foundation cluster is in **Ready** state.
- On the external key management system (KMS),
  - Ensure that a policy exists and the key value backend path in Vault is enabled.
  - Ensure that you are using signed certificates on your Vault servers.

- Create the following serviceaccount in the tenant namespace as shown below:

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ceph-csi-vault-sa
EOF
```

## Procedure

You need to configure the Kubernetes authentication method before OpenShift Data Foundation can authenticate with and start using **Vault**. The following instructions create and configure **serviceAccount**, **ClusterRole**, and **ClusterRoleBinding** required to allow OpenShift Data Foundation to authenticate with **Vault**.

1. Apply the following YAML to your Openshift cluster:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: rbd-csi-vault-token-review
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rbd-csi-vault-token-review
rules:
- apiGroups: ["authentication.k8s.io"]
  resources: ["tokenreviews"]
  verbs: ["create", "get", "list"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rbd-csi-vault-token-review
subjects:
- kind: ServiceAccount
  name: rbd-csi-vault-token-review
  namespace: openshift-storage
roleRef:
  kind: ClusterRole
  name: rbd-csi-vault-token-review
  apiGroup: rbac.authorization.k8s.io
```

2. Create a secret for serviceaccount token and CA certificate.

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: rbd-csi-vault-token-review-token
  namespace: openshift-storage
  annotations:
    kubernetes.io/service-account.name: "rbd-csi-vault-token-review"
```

```
type: kubernetes.io/service-account-token
data: {}
EOF
```

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

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

4. Retrieve the OpenShift cluster endpoint.

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

5. Use the information collected in the previous steps to set up the kubernetes authentication method in Vault as shown:

```
$ 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"
```

6. Create a role in Vault for the tenant namespace:

```
$ vault write "auth/kubernetes/role/csi-kubernetes" bound_service_account_names="ceph-
csi-vault-sa" bound_service_account_namespaces=<tenant_namespace> policies=
<policy_name_in_vault>
```

**csi-kubernetes** is the default role name that OpenShift Data Foundation looks for in Vault. The default service account name in the tenant namespace in the OpenShift Data Foundation cluster is **ceph-csi-vault-sa**. These default values can be overridden by creating a ConfigMap in the tenant namespace.

For more information about overriding the default names, see [Overriding Vault connection details using tenant ConfigMap](#).

## Sample YAML

- To create a storageclass that uses the **vaulttenantsa** method for PV encryption, you must either edit the existing ConfigMap or create a ConfigMap named **csi-kms-connection-details** that will hold all the information needed to establish the connection with Vault.

The sample yamll given below can be used to update or create the **csi-kms-connection-detail** ConfigMap:

```
apiVersion: v1
data:
  vault-tenant-sa: |-
    {
      "encryptionKMSType": "vaulttenantsa",
      "vaultAddress": "<https://hostname_or_ip_of_vault_server:port>",
      "vaultTLSServerName": "<vault TLS server name>",
      "vaultAuthPath": "/v1/auth/kubernetes/login",
```



```

"vaultAuthNamespace": "<vault auth namespace name>"
"vaultNamespace": "<vault namespace name>",
"vaultBackendPath": "<vault backend path name>",
"vaultCAFromSecret": "<secret containing CA cert>",
"vaultClientCertFromSecret": "<secret containing client cert>",
"vaultClientCertKeyFromSecret": "<secret containing client private key>",
"tenantSAName": "<service account name in the tenant namespace>"
}
metadata:
  name: csi-kms-connection-details

```

<b>encryptionKMSType</b>	Set to <b>vaulttenantsa</b> to use service accounts for authentication with vault.
<b>vaultAddress</b>	The hostname or IP address of the vault server with the port number.
<b>vaultTLSServerName</b>	(Optional) The vault TLS server name
<b>vaultAuthPath</b>	(Optional) The path where kubernetes auth method is enabled in Vault. The default path is <b>kubernetes</b> . If the auth method is enabled in a different path other than <b>kubernetes</b> , this variable needs to be set as <b>"/v1/auth/&lt;path&gt;/login"</b> .
<b>vaultAuthNamespace</b>	(Optional) The Vault namespace where kubernetes auth method is enabled.
<b>vaultNamespace</b>	(Optional) The Vault namespace where the backend path being used to store the keys exists
<b>vaultBackendPath</b>	The backend path in Vault where the encryption keys will be stored
<b>vaultCAFromSecret</b>	The secret in the OpenShift Data Foundation cluster containing the CA certificate from Vault
<b>vaultClientCertFromSecret</b>	The secret in the OpenShift Data Foundation cluster containing the client certificate from Vault
<b>vaultClientCertKeyFromSecret</b>	The secret in the OpenShift Data Foundation cluster containing the client private key from Vault
<b>tenantSAName</b>	(Optional) The service account name in the tenant namespace. The default value is <b>ceph-csi-vault-sa</b> . If a different name is to be used, this variable has to be set accordingly.

## 2.2.2. 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](#)
- Using Thales CipherTrust Manager (using KMIP): Ensure to configure access as described in [Configuring access to KMS using Thales CipherTrust Manager](#)

### 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.
    - a. Select the **Provider** from the drop down.
    - b. Select the **Key service** for the given provider from the list.
  - **Create new KMS connection** This is applicable for **vaulttokens** and **Thales CipherTrust Manager (using KMIP)** only.
    - a. Select the **Key Management Service Provider**.
    - b. If **Vault** is selected as the **Key Management Service Provider**, follow these steps:
      - i. Enter a unique **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), Port number and **Token**.
      - ii. Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:

- A. Enter the Key Value secret path in **Backend Path** that is dedicated and unique to OpenShift Data Foundation.
  - B. Optional: Enter **TLS Server Name** and **Vault Enterprise Namespace**.
  - C. Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.
  - D. Click **Save**.
- c. If **Thales CipherTrust Manager (using KMIP)** is selected as the **Key Management Service Provider**, follow these steps:
- i. Enter a unique **Connection Name**.
  - ii. In the **Address** and **Port** sections, enter the IP of Thales CipherTrust Manager and the port where the KMIP interface is enabled. For example, **Address: 123.34.3.2**, **Port: 5696**.
  - iii. Upload the **Client Certificate**, **CA certificate**, and **Client Private Key**.
  - iv. Enter the **Unique Identifier** for the key to be used for encryption and decryption, generated above.
  - v. The **TLS Server** field is optional and used when there is no DNS entry for the KMIP endpoint. For example, **kmip\_all\_<port>.ciphertrustmanager.local**.
- d. Click **Save**.
- e. Click **Create**.
9. 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.

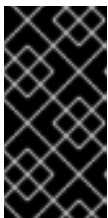
Example:

```
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](#).

#### 2.2.2.1. Overriding Vault connection details using tenant ConfigMap

The Vault connections details can be reconfigured per tenant by creating a ConfigMap in the Openshift namespace with configuration options that differ from the values set in the **csi-kms-connection-details** ConfigMap in the **openshift-storage** namespace. The ConfigMap needs to be located in the tenant namespace. The values in the ConfigMap in the tenant namespace will override the values set in the **csi-kms-connection-details** ConfigMap for the encrypted Persistent Volumes created in that namespace.

## Procedure

1. Ensure that you are in the tenant namespace.
2. Click on **Workloads → ConfigMaps**.
3. Click on **Create ConfigMap**.
4. The following is a sample yaml. The values to be overridden for the given tenant namespace can be specified under the **data** section as shown below:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ceph-csi-kms-config
data:
  vaultAddress: "<vault_address:port>"
  vaultBackendPath: "<backend_path>"
  vaultTLSServerName: "<vault_tls_server_name>"
  vaultNamespace: "<vault_namespace>"
```

5. After the yaml is edited, click on **Create**.

## 2.3. STORAGE CLASS WITH SINGLE REPLICA

You can create a storage class with a single replica to be used by your applications. This avoids redundant data copies and allows resiliency management on the application level.



### WARNING

Enabling this feature creates a single replica pool without data replication, increasing the risk of data loss, data corruption, and potential system instability if your application does not have its own replication. If any OSDs are lost, this feature requires very disruptive steps to recover. All applications can lose their data, and must be recreated in case of a failed OSD.

## Procedure

1. Enable the single replica feature using the following command:

```
$ oc patch storagecluster ocs-storagecluster -n openshift-storage --type json --patch '[{"op": "replace", "path": "/spec/managedResources/cephNonResilientPools/enable", "value": true }]
```

2. Verify **storagecluster** is in **Ready** state:

```
$ oc get storagecluster
```

Example output:

■

NAME	AGE	PHASE	EXTERNAL	CREATED AT	VERSION
ocs-storagecluster	10m	Ready		2024-02-05T13:56:15Z	4.15.0

3. New **cephblockpools** are created for each failure domain. Verify **cephblockpools** are in **Ready** state:

```
$ oc get cephblockpools
```

Example output:

NAME	PHASE
ocs-storagecluster-cephblockpool	Ready
ocs-storagecluster-cephblockpool-us-east-1a	Ready
ocs-storagecluster-cephblockpool-us-east-1b	Ready
ocs-storagecluster-cephblockpool-us-east-1c	Ready

4. Verify new storage classes have been created:

```
$ oc get storageclass
```

Example output:

NAME	PROVISIONER	RECLAIMPOLICY
gp2 (default)	kubernetes.io/aws-ebs	Delete
WaitForFirstConsumer true	104m	
gp2-csi	ebs.csi.aws.com	Delete
WaitForFirstConsumer true	104m	
gp3-csi	ebs.csi.aws.com	Delete
WaitForFirstConsumer true	104m	
ocs-storagecluster-ceph-non-resilient-rbd	openshift-storage.rbd.csi.ceph.com	Delete
WaitForFirstConsumer true	46m	
ocs-storagecluster-ceph-rbd	openshift-storage.rbd.csi.ceph.com	Delete
Immediate true	52m	
ocs-storagecluster-cephfs	openshift-storage.cephfs.csi.ceph.com	Delete
Immediate true	52m	
openshift-storage.noobaa.io	openshift-storage.noobaa.io/obc	Delete
Immediate false	50m	

5. New OSD pods are created; 3 osd-prepare pods and 3 additional pods. Verify new OSD pods are in **Running** state:

```
$ oc get pods | grep osd
```

Example output:

rook-ceph-osd-0-6dc76777bc-snhnm	2/2	Running	0	9m50s
rook-ceph-osd-1-768bdfdc4-h5n7k	2/2	Running	0	9m48s
rook-ceph-osd-2-69878645c4-bkdlq	2/2	Running	0	9m37s
rook-ceph-osd-3-64c44d7d76-zfxq9	2/2	Running	0	5m23s
rook-ceph-osd-4-654445b78f-nsgjb	2/2	Running	0	5m23s
rook-ceph-osd-5-5775949f57-vz6jp	2/2	Running	0	5m22s
rook-ceph-osd-prepare-ocs-device-set-gp2-0-data-0x6t87-59swf	0/1	Completed	0	10m

```
rook-ceph-osd-prepare-ocs-deviceset-gp2-1-data-0klwr7-bk45t 0/1 Completed 0
10m
rook-ceph-osd-prepare-ocs-deviceset-gp2-2-data-0mk2cz-jx7zv 0/1 Completed 0
10m
```

### 2.3.1. Recovering after OSD lost from single replica

When using replica 1, a storage class with a single replica, data loss is guaranteed when an OSD is lost. Lost OSDs go into a failing state. Use the following steps to recover after OSD loss.

#### Procedure

Follow these recovery steps to get your applications running again after data loss from replica 1. You first need to identify the domain where the failing OSD is.

1. If you know which failure domain the failing OSD is in, run the following command to get the exact **replica1-pool-name** required for the next steps. If you do not know where the failing OSD is, skip to step 2.

```
$ oc get cephblockpools
```

Example output:

```
NAME                                PHASE
ocs-storagecluster-cephblockpool    Ready
ocs-storagecluster-cephblockpool-us-south-1 Ready
ocs-storagecluster-cephblockpool-us-south-2 Ready
ocs-storagecluster-cephblockpool-us-south-3 Ready
```

Copy the corresponding failure domain name for use in next steps, then skip to step 4.

2. Find the OSD pod that is in **Error** state or **CrashLoopBackoff** state to find the failing OSD:

```
$ oc get pods -nopenshift-storage -l app=rook-ceph-osd | grep 'CrashLoopBackOff|Error'
```

3. Identify the replica-1 pool that had the failed OSD.

- a. Identify the node where the failed OSD was running:

```
failed_osd_id=0 #replace with the ID of the failed OSD
```

- b. Identify the failureDomainLabel for the node where the failed OSD was running:

```
failure_domain_label=$(oc get storageclass ocs-storagecluster-ceph-non-resilient-rbd -o
yaml | grep domainLabel |head -1 |awk -F:' ' '{print $2}')
```

```
failure_domain_value=$(oc get pods $failed_osd_id -oyaml |grep topology-location-zone
|awk '{print $2}')
```

The output shows the replica-1 pool name whose OSD is failing, for example:

```
replica1-pool-name= "ocs-storagecluster-cephblockpool-$failure_domain_value"
```

where **\$failure\_domain\_value** is the failureDomainName.

4. Delete the replica-1 pool.

a. Connect to the toolbox pod:

```
toolbox=$(oc get pod -l app=rook-ceph-tools -n openshift-storage -o  
jsonpath='{.items[*].metadata.name}')
```

```
oc rsh $toolbox -n openshift-storage
```

b. Delete the replica-1 pool. Note that you have to enter the replica-1 pool name twice in the command, for example:

```
ceph osd pool rm <replica1-pool-name> <replica1-pool-name> --yes-i-really-really-  
mean-it
```

Replace **replica1-pool-name** with the failure domain name identified earlier.

5. Purge the failing OSD by following the steps in section "Replacing operational or failed storage devices" based on your platform in the [Replacing devices](#) guide.

6. Restart the rook-ceph operator:

```
$ oc delete pod -l rook-ceph-operator -nopenshift-storage
```

7. Recreate any affected applications in that availability zone to start using the new pool with same name.



## CHAPTER 3. BLOCK POOLS

The OpenShift Data Foundation operator installs a default set of storage pools depending on the platform in use. These default storage pools are owned and controlled by the operator and it cannot be deleted or modified. With OpenShift Container Platform, you can create multiple custom 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 block pools are not supported for *external mode* OpenShift Data Foundation clusters.

### 3.1. CREATING A BLOCK POOL

#### Prerequisites

- You must be logged into the OpenShift Container Platform web console as an administrator.

#### Procedure

1. Click **Storage** → **Data Foundation**
2. In the **Storage systems** tab, select the storage system and then click the **BlockPools** tab.
3. Click **Create Block Pool**
4. Enter **Pool name**.



### NOTE

Using 2-way replication data protection policy is not supported for the default pool. However, you can use 2-way replication if you are creating an additional pool.

5. Select **Data protection policy** as either **2-way Replication** or **3-way Replication**.
6. Optional: Select **Enable compression** checkbox 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 is not compressed.
7. Click **Create**.

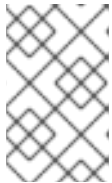
### 3.2. UPDATING AN EXISTING POOL

#### Prerequisites

- You must be logged into the OpenShift Container Platform web console as an administrator.

## Procedure

1. Click **Storage → Data Foundation**
2. In the **Storage systems** tab, select the storage system and then click **BlockPools**.
3. Click the Action Menu ( ⋮ ) at the end the pool you want to update.
4. Click **Edit Block Pool**.
5. Modify the form details as follows:



### NOTE

Using 2-way replication data protection policy is not supported for the default pool. However, you can use 2-way replication if you are creating an additional pool.

- a. Change the **Data protection policy** to either 2-way Replication or 3-way Replication.
  - b. Enable or disable the compression option.  
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 is not compressed.
6. Click **Save**.

## 3.3. DELETING A POOL

Use this procedure to delete a pool in OpenShift Data Foundation.

### Prerequisites

- You must be logged into the OpenShift Container Platform web console as an administrator.

### Procedure

1. . Click **Storage → Data Foundation**
2. In the **Storage systems** tab, select the storage system and then click the **BlockPools** tab.
3. Click the Action Menu ( ⋮ ) at the end the pool you want to delete.
4. Click **Delete Block Pool**
5. Click **Delete** to confirm the removal of the Pool.



### NOTE

A pool cannot be deleted when it is bound to a PVC. You must detach all the resources before performing this activity.

## CHAPTER 4. CONFIGURE STORAGE FOR OPENSIFT CONTAINER PLATFORM SERVICES

You can use OpenShift Data Foundation to provide storage for OpenShift Container Platform services such as the following:

- OpenShift image registry
- OpenShift monitoring
- OpenShift logging (Loki)

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 a plenty of storage capacity for the following OpenShift services that you configure:

- OpenShift image registry
- OpenShift monitoring
- OpenShift logging (Loki)
- OpenShift tracing platform (Tempo)

If the storage for these critical services runs out of space, the OpenShift 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](#) of *Monitoring* guide in the OpenShift Container Platform documentation for details.

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

### 4.1. CONFIGURING IMAGE REGISTRY TO USE 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.

Follow the instructions in this section to configure OpenShift Data Foundation as storage for the Container Image Registry. On AWS, it is not required to change the storage for the registry. However, it is recommended to change the storage to OpenShift Data Foundation Persistent Volume for vSphere and Bare metal platforms.



## WARNING


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

## Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift 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**.

## 4.2. USING MULTICLOUD OBJECT GATEWAY AS OPENSIFT IMAGE REGISTRY BACKEND STORAGE

You can use Multicloud Object Gateway (MCG) as OpenShift Container Platform (OCP) Image Registry backend storage in an on-prem OpenShift deployment.

To configure MCG as a backend storage for the OCP image registry, follow the steps mentioned in the procedure.

### Prerequisites

- Administrative access to OCP 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 the 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 the 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

```

## 4.3. 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 4.1. 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 ↓	
<b>PVC</b> my-alertmanager-claim-alertmanager-main-0	openshift-monitoring	Bound	pvc-d00428a5-0ce6-11ea-8fe8-023bdfa29edc	40Gi	⋮
<b>PVC</b> my-alertmanager-claim-alertmanager-main-1	openshift-monitoring	Bound	pvc-d00be111-0ce6-11ea-8fe8-023bdfa29edc	40Gi	⋮
<b>PVC</b> my-alertmanager-claim-alertmanager-main-2	openshift-monitoring	Bound	pvc-d01ac717-0ce6-11ea-8fe8-023bdfa29edc	40Gi	⋮
<b>PVC</b> my-prometheus-claim-prometheus-k8s-0	openshift-monitoring	Bound	pvc-ce290f1b-0ce6-11ea-8fe8-023bdfa29edc	40Gi	⋮
<b>PVC</b> 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 4.2. 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	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 4.3. Persistent Volume Claims attached to prometheus-k8s-\* pod

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

## 4.4. OVERPROVISION LEVEL POLICY CONTROL [TECHNOLOGY PREVIEW]

Overprovision control is a mechanism that enables you to define a quota on the amount of Persistent Volume Claims (PVCs) consumed from a storage cluster, based on the specific application namespace.

When you enable the overprovision control mechanism, it prevents you from overprovisioning the PVCs consumed from the storage cluster. OpenShift provides flexibility for defining constraints that limit the aggregated resource consumption at cluster scope with the help of **ClusterResourceQuota**. For more information see, [OpenShift ClusterResourceQuota](#).

With overprovision control, a **ClusterResourceQuota** is initiated, and you can set the storage capacity limit for each storage class. The alarm triggers when 80% of the capacity limit is consumed.



### NOTE

Overprovision level policy control is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information, refer to [Technology Preview Features Support Scope](#).

For more information about OpenShift Data Foundation deployment, refer to [Product Documentation](#) and select the deployment procedure according to the platform.

### Prerequisites

- Ensure that the OpenShift Data Foundation cluster is created.

### Procedure

1. Deploy **storagecluster** either from the command line interface or the user interface.
2. Label the application namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: <desired_name>
  labels:
    storagequota: <desired_label>
```

<desired\_name>

Specify a name for the application namespace, for example, **quota-rbd**.

<desired\_label>

Specify a label for the storage quota, for example, **storagequota1**.

3. Edit the **storagecluster** to set the quota limit on the storage class.

```
$ oc edit storagecluster -n openshift-storage <ocs_storagecluster_name>
```

<ocs\_storagecluster\_name>

Specify the name of the storage cluster.

4. Add an entry for Overprovision Control with the desired hard limit into the **StorageCluster.Spec**:

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
spec:
  [...]
  overprovisionControl:
    - capacity: <desired_quota_limit>
      storageClassName: <storage_class_name>
      quotaName: <desired_quota_name>
      selector:
        labels:
          matchLabels:
            storagequota: <desired_label>
  [...]

```

<desired\_quota\_limit>

Specify a desired quota limit for the storage class, for example, **27Ti**.

<storage\_class\_name>

Specify the name of the storage class for which you want to set the quota limit, for example, **ocs-storagecluster-ceph-rbd**.

<desired\_quota\_name>

Specify a name for the storage quota, for example, **quota1**.

<desired\_label>

Specify a label for the storage quota, for example, **storagequota1**.

5. Save the modified **storagecluster**.
6. Verify that the **clusterresourcequota** is defined.



#### NOTE

Expect the **clusterresourcequota** with the **quotaName** that you defined in the previous step, for example, **quota1**.

```
$ oc get clusterresourcequota -A
```

```
$ oc describe clusterresourcequota -A
```

## 4.5. 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.

#### 4.5.1. Configuring persistent storage

You can configure a persistent storage class and size for the Elasticsearch cluster using the storage class name and size parameters. The Cluster Logging Operator creates a Persistent Volume Claim for each data node in the Elasticsearch cluster based on these parameters. For example:

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

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



### NOTE

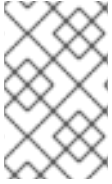
Omission of the storage block will result in a deployment backed by default storage. For example:

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

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

## 4.5.2. 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 4.4. 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 5. CREATING MULTUS NETWORKS

OpenShift Container Platform uses the Multus CNI plug-in to allow chaining of CNI plug-ins. You can configure your default pod network during cluster installation. The default network handles all ordinary network traffic for the cluster.

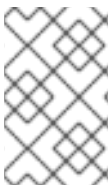
You can define an additional network based on the available CNI plug-ins and attach one or more of these networks to your pods. To attach additional network interfaces to a pod, you must create configurations that define how the interfaces are attached.

You specify each interface by using a NetworkAttachmentDefinition (NAD) custom resource (CR). A CNI configuration inside each of the NetworkAttachmentDefinition defines how that interface is created.

OpenShift Data Foundation uses the CNI plug-in called macvlan. Creating a macvlan-based additional network allows pods on a host to communicate with other hosts and pods on those hosts using a physical network interface. Each pod that is attached to a macvlan-based additional network is provided a unique MAC address.

### 5.1. CREATING NETWORK ATTACHMENT DEFINITIONS

To utilize Multus, an already working cluster with the correct networking configuration is required, see [Requirements for Multus configuration](#). The newly created NetworkAttachmentDefinition (NAD) can be selected during the Storage Cluster installation. This is the reason they must be created before the Storage Cluster.



#### NOTE

Network attachment definitions can only use the **whereabouts** IP address management (IPAM), and it must specify the **range** field. **ipRanges** and plugin chaining are not supported.

You can select the newly created **NetworkAttachmentDefinition** (NAD) during the Storage Cluster installation. This is the reason you must create the NAD before you create the Storage Cluster.

As detailed in the Planning Guide, the Multus networks you create depend on the number of available network interfaces you have for OpenShift Data Foundation traffic. It is possible to separate all of the storage traffic onto one of the two interfaces (one interface used for default OpenShift SDN) or to further segregate storage traffic into client storage traffic (public) and storage replication traffic (private or cluster).

The following is an example **NetworkAttachmentDefinition** for all the storage traffic, public and cluster, on the same interface. It requires one additional interface on all schedulable nodes (OpenShift default SDN on separate network interface):

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-public-cluster
  namespace: openshift-storage
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "macvlan",
```

```
"master": "ens2",
"mode": "bridge",
"ipam": {
  "type": "whereabouts",
  "range": "192.168.1.0/24"
}
}'
```



## NOTE

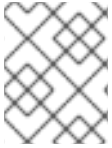
All network interface names must be the same on all the nodes attached to the Multus network (that is, **ens2** for **ocs-public-cluster**).

The following is an example **NetworkAttachmentDefinition** for storage traffic on separate Multus networks, public, for client storage traffic, and cluster, for replication traffic. It requires two additional interfaces on OpenShift nodes hosting object storage device (OSD) pods and one additional interface on all other schedulable nodes (OpenShift default SDN on separate network interface):

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-public
  namespace: openshift-storage
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "macvlan",
    "master": "ens2",
    "mode": "bridge",
    "ipam": {
      "type": "whereabouts",
      "range": "192.168.1.0/24"
    }
  }'
```

Example **NetworkAttachmentDefinition**:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-cluster
  namespace: openshift-storage
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "macvlan",
    "master": "ens3",
    "mode": "bridge",
    "ipam": {
      "type": "whereabouts",
      "range": "192.168.2.0/24"
    }
  }'
```

**NOTE**

All network interface names must be the same on all the nodes attached to the Multus networks (that is, **ens2** for **ocs-public**, and **ens3** for **ocs-cluster**).

## CHAPTER 6. 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 7. ADDING FILE AND OBJECT STORAGE TO AN EXISTING EXTERNAL OPENSIFT DATA FOUNDATION CLUSTER

When OpenShift Data Foundation is configured in external mode, there are several ways to provide storage for persistent volume claims and object bucket claims.

- Persistent volume claims for block storage are provided directly from the external Red Hat Ceph Storage cluster.
- Persistent volume claims for file storage can be provided by adding a Metadata Server (MDS) to the external Red Hat Ceph Storage cluster.
- Object bucket claims for object storage can be provided either by using the Multicloud Object Gateway or by adding the Ceph Object Gateway to the external Red Hat Ceph Storage cluster.

Use the following process to add file storage (using Metadata Servers) or object storage (using Ceph Object Gateway) or both to an external OpenShift Data Foundation cluster that was initially deployed to provide only block storage.

### Prerequisites

- OpenShift Data Foundation 4.15 is installed and running on the OpenShift Container Platform version 4.15 or above. Also, the OpenShift Data Foundation Cluster in external mode is in the **Ready** state.
- Your external Red Hat Ceph Storage cluster is configured with one or both of the following:
  - a Ceph Object Gateway (RGW) endpoint that can be accessed by the OpenShift Container Platform cluster for object storage
  - a Metadata Server (MDS) pool for file storage
- Ensure that you know the parameters used with the **ceph-external-cluster-details-exporter.py** script during external OpenShift Data Foundation cluster deployment.

### Procedure

1. Download the OpenShift Data Foundation version of the **ceph-external-cluster-details-exporter.py** python script using the following command:

```
oc get csv $(oc get csv -n openshift-storage | grep ocs-operator | awk '{print $1}') -n
openshift-storage -o
jsonpath='{.metadata.annotations.external\.features\.ocs\.openshift\.io/export-script}' | base64
--decode > ceph-external-cluster-details-exporter.py
```

2. Update permission caps on the external Red Hat Ceph Storage cluster by running **ceph-external-cluster-details-exporter.py** on any client node in the external Red Hat Ceph Storage cluster. You may need to ask your Red Hat Ceph Storage administrator to do this.

```
# python3 ceph-external-cluster-details-exporter.py --upgrade \
--run-as-user=ocs-client-name \
--rgw-pool-prefix rgw-pool-prefix
```

**--run-as-user**

The client name used during OpenShift Data Foundation cluster deployment. Use the default client name **client.healthchecker** if a different client name was not set.

**--rgw-pool-prefix**

The prefix used for the Ceph Object Gateway pool. This can be omitted if the default prefix is used.

3. Generate and save configuration details from the external Red Hat Ceph Storage cluster.

- a. Generate configuration details by running **ceph-external-cluster-details-exporter.py** on any client node in the external Red Hat Ceph Storage cluster.

```
# python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbd-block-pool-name --monitoring-endpoint ceph-mgr-prometheus-exporter-endpoint --monitoring-endpoint-port ceph-mgr-prometheus-exporter-port --run-as-user ocs-client-name --rgw-endpoint rgw-endpoint --rgw-pool-prefix rgw-pool-prefix
```

**--monitoring-endpoint**

Is optional. It accepts comma separated list of IP addresses of active and standby mgrs reachable from the OpenShift Container Platform cluster. If not provided, the value is automatically populated.

**--monitoring-endpoint-port**

Is optional. It is the port associated with the ceph-mgr Prometheus exporter specified by **--monitoring-endpoint**. If not provided, the value is automatically populated.

**--run-as-user**

The client name used during OpenShift Data Foundation cluster deployment. Use the default client name **client.healthchecker** if a different client name was not set.

**--rgw-endpoint**

Provide this parameter to provision object storage through Ceph Object Gateway for OpenShift Data Foundation. (optional parameter)

**--rgw-pool-prefix**

The prefix used for the Ceph Object Gateway pool. This can be omitted if the default prefix is used.

User permissions are updated as shown:

```
caps: [mgr] allow command config
caps: [mon] allow r, allow command quorum_status, allow command version
caps: [osd] allow rwx pool=default.rgw.meta, allow r pool=.rgw.root, allow rw pool=default.rgw.control, allow rx pool=default.rgw.log, allow x pool=default.rgw.buckets.index
```



**NOTE**

Ensure that all the parameters (including the optional arguments) except the Ceph Object Gateway details (if provided), are the same as what was used during the deployment of OpenShift Data Foundation in external mode.

- b. Save the output of the script in an **external-cluster-config.json** file. The following example output shows the generated configuration changes in bold text.

```

[{"name": "rook-ceph-mon-endpoints", "kind": "ConfigMap", "data": {"data":
"xxx.xxx.xxx.xxx:xxxx", "maxMonId": "0", "mapping": "{}"}}, {"name": "rook-ceph-mon",
"kind": "Secret", "data": {"admin-secret": "admin-secret", "fsid": "<fs-id>", "mon-secret":
"mon-secret"}}, {"name": "rook-ceph-operator-creds", "kind": "Secret", "data": {"userID": "
<user-id>", "userKey": "<user-key>"}}, {"name": "rook-csi-rbd-node", "kind": "Secret",
"data": {"userID": "csi-rbd-node", "userKey": "<user-key>"}}, {"name": "ceph-rbd", "kind":
"StorageClass", "data": {"pool": "<pool>"}}, {"name": "monitoring-endpoint", "kind":
"CephCluster", "data": {"MonitoringEndpoint": "xxx.xxx.xxx.xxx", "MonitoringPort":
"xxxx"}}, {"name": "rook-ceph-dashboard-link", "kind": "Secret", "data": {"userID": "ceph-
dashboard-link", "userKey": "<user-key>"}}, {"name": "rook-csi-rbd-provisioner", "kind":
"Secret", "data": {"userID": "csi-rbd-provisioner", "userKey": "<user-key>"}}, {"name":
"rook-csi-cephfs-provisioner", "kind": "Secret", "data": {"adminID": "csi-cephfs-
provisioner", "adminKey": "<admin-key>"}}, {"name": "rook-csi-cephfs-node", "kind":
"Secret", "data": {"adminID": "csi-cephfs-node", "adminKey": "<admin-key>"}}, {"name":
"cephfs", "kind": "StorageClass", "data": {"fsName": "cephfs", "pool": "cephfs_data"}},
{"name": "ceph-rgw", "kind": "StorageClass", "data": {"endpoint": "xxx.xxx.xxx.xxx:xxxx",
"poolPrefix": "default"}}, {"name": "rgw-admin-ops-user", "kind": "Secret", "data":
{"accessKey": "<access-key>", "secretKey": "<secret-key>"}]}

```

4. Upload the generated JSON file.
  - a. Log in to the OpenShift web console.
  - b. Click **Workloads** → **Secrets**.
  - c. Set **project** to **openshift-storage**.
  - d. Click on **rook-ceph-external-cluster-details**.
  - e. Click **Actions** ( ⋮ ) → **Edit Secret**
  - f. Click **Browse** and upload the **external-cluster-config.json** file.
  - g. Click **Save**.

### Verification steps

- To 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.
  - On the **Overview** → **Block and File** tab, check the Status card to confirm that the *Storage Cluster* has a green tick indicating it is healthy.
- If you added a Metadata Server for file storage:
  - a. Click **Workloads** → **Pods** and verify that **csi-cephfsplugin-\*** pods are created new and are in the **Running** state.
  - b. Click **Storage** → **Storage Classes** and verify that the **ocs-external-storagecluster-cephfs** storage class is created.
- If you added the Ceph Object Gateway for object storage:
  - a. Click **Storage** → **Storage Classes** and verify that the **ocs-external-storagecluster-ceph-rgw** storage class is created.



- b. To 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.
- c. Click the **Object** tab and confirm *Object Service* and *Data resiliency* has a green tick indicating it is healthy.

## 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: [Openshift-dns daemonsets doesn't include toleration to run on nodes with taints](#).

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: node.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. MANAGING PERSISTENT VOLUME CLAIMS

### 9.1. 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**.

## 9.2. 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.

## 9.3. 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.

## 9.4. EXPANDING PERSISTENT VOLUME CLAIMS

OpenShift Data Foundation 4.6 onwards has the ability to expand Persistent Volume Claims providing more flexibility in the management of persistent storage resources.

Expansion is supported for the following Persistent Volumes:

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



- PVC with ReadWriteOnce (RWO) access that is based on Ceph RADOS Block Devices (RBDs) with volume mode **Block**.
- PVC with ReadWriteOncePod (RWOP) that is based on Ceph File System (CephFS) or Network File System (NFS) for volume mode **Filesystem**.
- PVC with ReadWriteOncePod (RWOP) access that is based on Ceph RADOS Block Devices (RBDs) with volume mode **Filesystem**. With RWOP access mode, you mount the volume as read-write by a single pod on a single node.



## IMPORTANT

The ReadWriteOncePod (RWOP) access mode is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).



## NOTE

PVC expansion is not supported for OSD, MON and encrypted PVCs.

## Prerequisites

- Administrator access to OpenShift Web Console.

## Procedure

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

The screenshot shows the OpenShift Web Console interface for Persistent Volume Claims. The table lists three PVCs:

Name	Namespace	Status	Persistent Volume	Capacity	Used	Storage Class
pvc-db-moobaa-db-0	openshift-storage	Bound	pvc-81a35e5a-357f-45f4-93c4-8e673f58f661	50 GiB	3551 MiB	ocs-storagecluster-ceph-
pvc-ocs-devicset-0-data-0-r6sw5	openshift-storage	Bound	pvc-7185f4d-caaa-4a00-8a94-da495f7c5b06	512 GiB	-	ocs-storagecluster-ceph-
pvc-ocs-devicset-1-data-0-5wh9l	openshift-storage	Bound	pvc-547b053-cd21-404d-a77f-d861336937f8	512 GiB	-	ocs-storagecluster-ceph-

The action menu for the first PVC is open, showing options: Expand PVC, Edit Labels, Edit Annotations, Edit Persistent Volume Claim, and Delete Persistent Volume Claim.

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

## Expand Persistent Volume Claim

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

Size \*

50	GiB ▼
----	-------

Cancel

Expand

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



### NOTE

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

## 9.5. DYNAMIC PROVISIONING

### 9.5.1. About dynamic provisioning

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

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

Many storage types are available for use as persistent volumes in OpenShift Container Platform. Storage plug-ins might support static provisioning, dynamic provisioning or both provisioning types.

### 9.5.2. 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**.
- Create a PVC with ReadWriteOncePod (RWOP) access that is based on CephFS, NFS and RBD. With RWOP access mode, you mount the volume as read-write by a single pod on a single node.



### IMPORTANT

The ReadWriteOncePod (RWOP) access mode is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

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

### 9.5.3. Available dynamic provisioning plug-ins

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

Storage type	Provisioner plug-in name	Notes
OpenStack Cinder	<b>kubernetes.io/cinder</b>	

Storage type	Provisioner plug-in name	Notes
AWS Elastic Block Store (EBS)	<b>kubernetes.io/aws-ebs</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 10. RECLAIMING SPACE ON TARGET VOLUMES

The deleted files or chunks of zero data Sometimes take up storage space on the Ceph cluster resulting in inaccurate reporting of the available storage space. The reclaim space operation removes such discrepancies by executing the following operations on the target volume:

- **fstrim** - This operation is executed on volumes that are in **Filesystem** mode and only if the volume is mounted to a pod at the time of execution of reclaim space operation.
- **rbd sparsify** - This operation is executed when the volume is not attached to any pods and reclaims the space occupied by chunks of 4M-sized zeroed data.



### NOTE

- The reclaim space operation is supported only by the Ceph RBD volumes.
- The reclaim space operation involves a performance penalty when it is being executed.

You can use one of the following methods to reclaim the space:

- Enabling reclaim space operation using Annotating PersistentVolumeClaims (Recommended method to use for enabling reclaim space operation)
- Enabling reclaim space operation using ReclaimSpaceJob
- Enabling reclaim space operation using ReclaimSpaceCronJob

### 10.1. ENABLING RECLAIM SPACE OPERATION USING ANNOTATING PERSISTENTVOLUMECLAIMS

Use this procedure to annotate **PersistentVolumeClaims** so that it can invoke the reclaim space operation automatically based on a given schedule.



### NOTE

- The schedule value is in the same format as the [Kubernetes CronJobs](#) which sets the and/or interval of the recurring operation request.
- Recommended schedule interval is **@weekly**. If the schedule interval value is empty or in an invalid format, then the default schedule value is set to **@weekly**.
- Minimum supported interval between each scheduled operation is at least 24 hours. For example, **@daily** (At 00:00 every day) or **03\*\*\*** (At 3:00 every day).
- Schedule the **ReclaimSpace** operation during off-peak, maintenance window, or the interval when the workload input/output is expected to be low.
- **ReclaimSpaceCronJob** is recreated when the **schedule** is modified. It is automatically deleted when the annotation is removed.

### Procedure

1. Get the persistent volume claim (PVC) details.

```
$ oc get pvc data-pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
data-pvc	Bound	pvc-f37b8582-4b04-4676-88dd-e1b95c6abf74	1Gi	RWO
storageclass-ceph-rbd		20h		ocs-

2. Add annotation **reclaimspace.csiaddons.openshift.io/schedule=@monthly** to the PVC to create **reclaimspacecronjob**.

```
$ oc annotate pvc data-pvc "reclaimspace.csiaddons.openshift.io/schedule=@monthly"
```

```
persistentvolumeclaim/data-pvc annotated
```

3. Verify that **reclaimspacecronjob** is created in the format, "**<pvc-name>-xxxxxxx**".

```
$ oc get reclaimspacecronjobs.csiaddons.openshift.io
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LASTSCHEDULE	AGE
data-pvc-1642663516	@monthly			3s	

4. Modify the schedule to run this job automatically.

```
$ oc annotate pvc data-pvc "reclaimspace.csiaddons.openshift.io/schedule=@weekly" --
overwrite=true
```

```
persistentvolumeclaim/data-pvc annotated
```

5. Verify that the schedule for **reclaimspacecronjob** has been modified.

```
$ oc get reclaimspacecronjobs.csiaddons.openshift.io
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LASTSCHEDULE	AGE
data-pvc-1642664617	@weekly			3s	

## 10.2. ENABLING RECLAIM SPACE OPERATION USING RECLAIMSPACEJOB

**ReclaimSpaceJob** is a namespaced custom resource (CR) designed to invoke reclaim space operation on the target volume. This is a one time method that immediately starts the reclaim space operation. You have to repeat the creation of **ReclaimSpaceJob** CR to repeat the reclaim space operation when required.



## NOTE

- Recommended interval between the reclaim space operations is **weekly**.
- Ensure that the minimum interval between each operation is at least **24 hours**.
- Schedule the reclaim space operation during off-peak, maintenance window, or when the workload input/output is expected to be low.

## Procedure

1. Create and apply the following custom resource for reclaim space operation:

```
apiVersion: csiaddons.openshift.io/v1alpha1
kind: ReclaimSpaceJob
metadata:
  name: sample-1
spec:
  target:
    persistentVolumeClaim: pvc-1
  timeout: 360
```

where,

### target

Indicates the volume target on which the operation is performed.

### persistentVolumeClaim

Name of the **PersistentVolumeClaim**.

### backOfflimit

Specifies the maximum number of retries before marking the reclaim space operation as **failed**. The default value is **6**. The allowed maximum and minimum values are **60** and **0** respectively.

### retryDeadlineSeconds

Specifies the duration in which the operation might retire in seconds and it is relative to the start time. The value must be a positive integer. The default value is **600** seconds and the allowed maximum value is **1800** seconds.

### timeout

Specifies the timeout in seconds for the **grpc** request sent to the CSI driver. If the timeout value is not specified, it defaults to the value of global `reclaimspace` timeout. Minimum allowed value for timeout is 60.

2. Delete the custom resource after completion of the operation.

## 10.3. ENABLING RECLAIM SPACE OPERATION USING RECLAIMSPACECRONJOB

**ReclaimSpaceCronJob** invokes the reclaim space operation based on the given schedule such as daily, weekly, and so on. You have to create **ReclaimSpaceCronJob** only once for a persistent volume claim. The **CSI-addons** controller creates a **ReclaimSpaceJob** at the requested time and interval with the `schedule` attribute.



## NOTE

- Recommended schedule interval is **@weekly**.
- Minimum interval between each scheduled operation should be at least 24 hours. For example, **@daily** (At 00:00 every day) or "0 3 \* \* \*" (At 3:00 every day).
- Schedule the ReclaimSpace operation during off-peak, maintenance window, or the interval when workload input/output is expected to be low.

## Procedure

1. Create and apply the following custom resource for reclaim space operation

```

apiVersion: csiaddons.openshift.io/v1alpha1
kind: ReclaimSpaceCronJob
metadata:
  name: reclaimspacecronjob-sample
spec:
  jobTemplate:
    spec:
      target:
        persistentVolumeClaim: data-pvc
      timeout: 360
    schedule: '@weekly'
  concurrencyPolicy: Forbid

```

where,

### concurrencyPolicy

Describes the changes when a new **ReclaimSpaceJob** is scheduled by the **ReclaimSpaceCronJob**, while a previous **ReclaimSpaceJob** is still running. The default **Forbid** prevents starting a new job whereas **Replace** can be used to delete the running job potentially in a failure state and create a new one.

### failedJobsHistoryLimit

Specifies the number of failed **ReclaimSpaceJobs** that are kept for troubleshooting.

### jobTemplate

Specifies the **ReclaimSpaceJob.spec** structure that describes the details of the requested **ReclaimSpaceJob** operation.

### successfulJobsHistoryLimit

Specifies the number of successful **ReclaimSpaceJob** operations.

### schedule

Specifies the and/or interval of the recurring operation request and it is in the same format as the [Kubernetes CronJobs](#).

2. Delete the **ReclaimSpaceCronJob** custom resource when execution of reclaim space operation is no longer needed or when the target PVC is deleted.

## 10.4. CUSTOMISING TIMEOUTS REQUIRED FOR RECLAIM SPACE OPERATION



Depending on the RBD volume size and its data pattern, Reclaim Space Operation might fail with the **context deadline exceeded** error. You can avoid this by increasing the timeout value.

The following example shows the failed status by inspecting **-o yaml** of the corresponding **ReclaimSpaceJob**:

### Example

```
Status:
Completion Time: 2023-03-08T18:56:18Z
Conditions:
  Last Transition Time: 2023-03-08T18:56:18Z
  Message:             Failed to make controller request: context deadline exceeded
  Observed Generation: 1
  Reason:              failed
  Status:              True
  Type:                Failed
Message:              Maximum retry limit reached
Result:               Failed
Retries:              6
Start Time:           2023-03-08T18:33:55Z
```

You can also set custom timeouts at global level by creating the following **configmap**:

### Example

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: csi-addons-config
  namespace: openshift-storage
data:
  "reclaim-space-timeout": "6m"
```

Restart the **csi-addons** operator pod.

```
oc delete po -n openshift-storage -l "app.kubernetes.io/name=csi-addons"
```

All Reclaim Space Operations started after the above **configmap** creation use the customized timeout.

## CHAPTER 11. 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.

Volume snapshot class allows an administrator to specify different attributes belonging to a volume snapshot object. The OpenShift Data Foundation operator installs default volume snapshot classes depending on the platform in use. The operator owns and controls these default volume snapshot classes and they cannot be deleted or modified.

You can create many snapshots of the same persistent volume claim (PVC) but cannot schedule periodic creation of snapshots.

- 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

Persistent Volume encryption now supports volume snapshots.

### 11.1. 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.

## 11.2. 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.

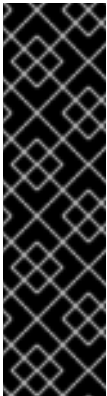
6. Select the **Storage Class** name.



#### 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.

## 11.3. 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 12. 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).

### 12.1. 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 13. MANAGING CONTAINER STORAGE INTERFACE (CSI) COMPONENT PLACEMENTS

Each cluster consists of a number of dedicated nodes such as **infra** and **storage** nodes. However, an **infra** node with a custom taint will not be able to use OpenShift Data Foundation Persistent Volume Claims (PVCs) on the node. So, if you want to use such nodes, you can set tolerations to bring up **csi-plugins** on the nodes.

### Procedure

1. Edit the configmap to add the toleration for the custom taint. Remember to save before exiting the editor.

```
$ oc edit configmap rook-ceph-operator-config -n openshift-storage
```

2. Display the **configmap** to check the added toleration.

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

Example output of the added toleration for the taint, **nodetype=infra:NoSchedule** :

```
apiVersion: v1
data:
[...]
```

CSI\_PLUGIN\_TOLERATIONS: |

```
- key: nodetype
  operator: Equal
  value: infra
  effect: NoSchedule
- key: node.ocs.openshift.io/storage
  operator: Equal
  value: "true"
  effect: NoSchedule
[...]
```

```
kind: ConfigMap
metadata:
[...]
```



### NOTE

Ensure that all non-string values in the Tolerations **value** field has double quotation marks. For example, the values **true** which is of type boolean, and **1** which is of type int must be input as "true" and "1".

3. Restart the **rook-ceph-operator** if the **csi-cephfsplugin-\*** and **csi-rbdplugin-\*** pods fail to come up on their own on the infra nodes.

```
$ oc delete -n openshift-storage pod <name of the rook_ceph_operator pod>
```

Example :



```
$ oc delete -n openshift-storage pod rook-ceph-operator-5446f9b95b-jrn2j  
pod "rook-ceph-operator-5446f9b95b-jrn2j" deleted
```

### Verification step

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

## CHAPTER 14. CREATING EXPORTS USING NFS

This section describes how to create exports using NFS that can then be accessed externally from the OpenShift cluster.

Follow the instructions below to create exports and access them externally from the OpenShift Cluster:

- [Section 14.1, “Enabling the NFS feature”](#)
- [Section 14.2, “Creating NFS exports”](#)
- [Section 14.3, “Consuming NFS exports in-cluster”](#)
- [Section 14.4, “Consuming NFS exports externally from the OpenShift cluster”](#)

### 14.1. ENABLING THE NFS FEATURE

To use NFS feature, you need to enable it in the storage cluster using the command-line interface (CLI) after the cluster is created. You can also enable the NFS feature while creating the storage cluster using the user interface.

#### Prerequisites

- OpenShift Data Foundation is installed and running in the **openshift-storage** namespace.
- The OpenShift Data Foundation installation includes a CephFilesystem.

#### Procedure

- Run the following command to enable the NFS feature from CLI:

```
$ oc --namespace openshift-storage patch storageclusters.ocs.openshift.io ocs-storagecluster --type merge --patch '{"spec": {"nfs":{"enable": true}}}'
```

#### Verification steps

NFS installation and configuration is complete when the following conditions are met:

- The CephNFS resource named **ocs-storagecluster-cephnfs** has a status of **Ready**.
- Check if all the **csi-nfsplugin-\*** pods are running:

```
oc -n openshift-storage describe cephnfs ocs-storagecluster-cephnfs
```

```
oc -n openshift-storage get pod | grep csi-nfsplugin
```

Output has multiple pods. For example:

```
csi-nfsplugin-47qwq           2/2   Running 0 10s
csi-nfsplugin-77947          2/2   Running 0 10s
csi-nfsplugin-ct2pm          2/2   Running 0 10s
csi-nfsplugin-provisioner-f85b75fbb-2rm2w      2/2   Running 0 10s
csi-nfsplugin-provisioner-f85b75fbb-8nj5h      2/2   Running 0 10s
```

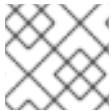
## 14.2. CREATING NFS EXPORTS

NFS exports are created by creating a Persistent Volume Claim (PVC) against the **ocs-storagecluster-ceph-nfs** StorageClass.

You can create NFS PVCs two ways:

### Create NFS PVC using a yaml.

The following is an example PVC.



#### NOTE

**volumeMode: Block** will not work for NFS volumes.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <desired_name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ocs-storagecluster-ceph-nfs
```

<desired\_name>

Specify a name for the PVC, for example, **my-nfs-export**.

The export is created once the PVC reaches the **Bound** state.

### Create NFS PVCs from the OpenShift Container Platform web console.

#### Prerequisites

- Ensure that you are logged into the OpenShift Container Platform web console and the NFS feature is enabled for the storage cluster.

#### Procedure

1. In the OpenShift Web Console, click **Storage** → **Persistent Volume Claims**
2. Set the **Project** to **openshift-storage**.
3. Click **Create PersistentVolumeClaim**.
  - a. Specify **Storage Class**, **ocs-storagecluster-ceph-nfs**.
  - b. Specify the **PVC Name**, for example, **my-nfs-export**.
  - c. Select the required **Access Mode**.
  - d. Specify a **Size** as per application requirement.

- e. Select **Volume mode** as **Filesystem**.  
Note: **Block** mode is not supported for NFS PVCs
- f. Click **Create** and wait until the PVC is in **Bound** status.

## 14.3. CONSUMING NFS EXPORTS IN-CLUSTER

Kubernetes application pods can consume NFS exports created by mounting a previously created PVC.

You can mount the PVC one of two ways:

### Using a YAML:

Below is an example pod that uses the example PVC created in [Section 14.2, "Creating NFS exports"](#):

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-export-example
spec:
  containers:
  - name: web-server
    image: nginx
    volumeMounts:
    - name: nfs-export-pvc
      mountPath: /var/lib/www/html
  volumes:
  - name: nfs-export-pvc
    persistentVolumeClaim:
      claimName: <pvc_name>
    readOnly: false
```

<pvc\_name>

Specify the PVC you have previously created, for example, **my-nfs-export**.

Using the OpenShift Container Platform web console.

### Procedure

1. On the OpenShift Container Platform web console, navigate to **Workloads** → **Pods**.
2. Click **Create Pod** to create a new application pod.
3. Under the **metadata** section add a name. For example, **nfs-export-example**, with **namespace** as **openshift-storage**.
4. Under the **spec:** section, add **containers:** section with **image** and **volumeMounts** sections:

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-export-example
  namespace: openshift-storage
spec:
  containers:
```

```
- name: web-server
  image: nginx
  volumeMounts:
    - name: <volume_name>
      mountPath: /var/lib/www/html
```

For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-export-example
  namespace: openshift-storage
spec:
  containers:
    - name: web-server
      image: nginx
      volumeMounts:
        - name: nfs-export-pvc
          mountPath: /var/lib/www/html
```

5. Under the **spec:** section, add **volumes:** section to add the NFS PVC as a volume for the application pod:

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

For example:

```
volumes:
  - name: nfs-export-pvc
    persistentVolumeClaim:
      claimName: my-nfs-export
```

## 14.4. CONSUMING NFS EXPORTS EXTERNALLY FROM THE OPENSIFT CLUSTER

NFS clients outside of the OpenShift cluster can mount NFS exports created by a previously-created PVC.

### Procedure

1. After the **nfs** flag is enabled, single-server CephNFS is deployed by Rook. You need to fetch the value of the **ceph\_nfs** field for the **nfs-ganesha** server to use in the next step:

```
$ oc get pods -n openshift-storage | grep rook-ceph-nfs
```

```
$ oc describe pod <name of the rook-ceph-nfs pod> | grep ceph_nfs
```

For example:

■

```
$ oc describe pod rook-ceph-nfs-ocs-storagecluster-cephnfs-a-7bb484b4bf-bbdhs | grep
ceph_nfs
ceph_nfs=my-nfs
```

2. Expose the NFS server outside of the OpenShift cluster by creating a Kubernetes LoadBalancer Service. The example below creates a LoadBalancer Service and references the NFS server created by OpenShift Data Foundation.

```
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-nfs-ocs-storagecluster-cephnfs-load-balancer
  namespace: openshift-storage
spec:
  ports:
    - name: nfs
      port: 2049
  type: LoadBalancer
  externalTrafficPolicy: Local
  selector:
    app: rook-ceph-nfs
    ceph_nfs: <my-nfs>
  instance: a
```

Replace **<my-nfs>** with the value you got in step 1.

3. Collect connection information. The information external clients need to connect to an export comes from the Persistent Volume (PV) created for the PVC, and the status of the LoadBalancer Service created in the previous step.
  - a. Get the share path from the PV.

- i. Get the name of the PV associated with the NFS export's PVC:

```
$ oc get pvc <pvc_name> --output jsonpath='{.spec.volumeName}'
pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d
```

Replace **<pvc\_name>** with your own PVC name. For example:

```
oc get pvc pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d --output
jsonpath='{.spec.volumeName}'
pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d
```

- ii. Use the PV name obtained previously to get the NFS export's share path:

```
$ oc get pv pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d --output
jsonpath='{.spec.csi.volumeAttributes.share}'
/0001-0011-openshift-storage-0000000000000001-ba9426ab-d61b-11ec-9ffd-
0a580a800215
```

- b. Get an ingress address for the NFS server. A service's ingress status may have multiple addresses. Choose the one desired to use for external clients. In the example below, there is only a single address: the host name **ingress-id.somedomain.com**.

```
$ oc -n openshift-storage get service rook-ceph-nfs-ocs-storagecluster-cephnfs-load-balancer --output jsonpath='{.status.loadBalancer.ingress}' [{"hostname":"ingress-id.somedomain.com"}]
```

4. Connect the external client using the share path and ingress address from the previous steps. The following example mounts the export to the client's directory path **/export/mount/path**:

```
$ mount -t nfs4 -o proto=tcp ingress-id.somedomain.com:/0001-0011-openshift-storage-0000000000000001-ba9426ab-d61b-11ec-9ffd-0a580a800215 /export/mount/path
```

If this does not work immediately, it could be that the Kubernetes environment is still taking time to configure the network resources to allow ingress to the NFS server.

## CHAPTER 15. ANNOTATING ENCRYPTED RBD STORAGE CLASSES

Starting with OpenShift Data Foundation 4.14, when the OpenShift console creates a RADOS block device (RBD) storage class with encryption enabled, the annotation is set automatically. However, you need to add the annotation, **cdi.kubevirt.io/clone-strategy=copy** for any of the encrypted RBD storage classes that were previously created before updating to the OpenShift Data Foundation version 4.14. This enables customer data integration (CDI) to use host-assisted cloning instead of the default smart cloning.

The keys used to access an encrypted volume are tied to the namespace where the volume was created. When cloning an encrypted volume to a new namespace, such as, provisioning a new OpenShift Virtualization virtual machine, a new volume must be created and the content of the source volume must then be copied into the new volume. This behavior is triggered automatically if the storage class is properly annotated.