Red Hat

Red Hat OpenStack Services on OpenShift 18.0

Customizing persistent storage

Customizing storage services for Red Hat OpenStack Services on OpenShift

Last Updated: 2024-08-28

Red Hat OpenStack Services on OpenShift 18.0 Customizing persistent storage

Customizing storage services for Red Hat OpenStack Services on OpenShift

OpenStack Team
rhos-docs@redhat.com

Abstract

Customize the services for block, image, object, and file storage in your Red Hat OpenStack Services on OpenShift environment.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation for Red Hat OpenStack Services on OpenShift (RHOSO) or earlier releases of Red Hat OpenStack Platform (RHOSP). When you create an issue for RHOSO or RHOSP documents, the issue is recorded in the RHOSO Jira project, where you can track the progress of your feedback.

To complete the Create Issue form, ensure that you are logged in to Jira. If you do not have a Red Hat Jira account, you can create an account at https://issues.redhat.com.

1. Click the following link to open a **Create Issue** page: Create Issue
2. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
3. Click **Create**.

CHAPTER 1. CUSTOMIZING STORAGE IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

When you have planned and configured the storage solution for your Red Hat OpenStack Services on OpenShift (RHOSO) deployment, you can make post-deployment configurations and customizations for functionality, scaling, and security in the following storage services:

- Block Storage service (cinder)
- Image service (glance)
- Object Storage service (swift)
- Shared File Systems service (manila)

For information about planning the storage solution and related requirements for your RHOSO deployment, for example, networking and security, see Planning storage and shared file systems in *Planning your deployment*.

For information about configuring storage services, see *Configuring persistent storage*. For information about performing operations with the storage services, see *Performing storage operations*.

CHAPTER 2. CUSTOMIZING AND MANAGING RED HAT CEPH STORAGE

Red Hat OpenStack Services on OpenShift (RHOSO) 18.0 supports Red Hat Ceph Storage 7. For information on the customization and management of Red Hat Ceph Storage 7, refer to the Red Hat Ceph Storage documentation. The following guides contain key information and procedures for these tasks:

- Administration Guide
- Configuration Guide
- Operations Guide
- Data Security and Hardening Guide
- Dashboard Guide
- Troubleshooting Guide

CHAPTER 3. CUSTOMIZING THE BLOCK STORAGE BACKUP SERVICE

When you have deployed the backup service for the Block Storage service (cinder), you can change the default parameters.

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.

3.1. AUTHENTICATING VOLUME OWNERS FOR ACCESS TO VOLUME BACKUPS

Administrators can back up any volume belonging to the project. To ensure that the volume owner can also access the volume backup, administrators must provide arguments to authenticate the volume owner when backing up the volume.

Procedure

- Provide the following arguments to authenticate a volume owner for access to volume backups:

  ```
  $ openstack --os-project-name <projectname> \
  --os-username <username> \
  --os-password <password> \
  volume backup create [--name <backup_name>] <volume>
  ```

  - Replace **<projectname>** with the name of the project (tenant) of the owner of the volume.
  - Replace **<username>** and **<password>** with the username and password credentials of the user that is the owner of the volume within this project.

NOTE

[--name <backup_name>] <volume> are the typical arguments when creating a volume backup.

3.2. VIEWING AND MODIFYING PROJECT BACKUP QUOTAS

You can change or view the maximum number of backups and the maximum total size of all backups, in gigabytes, that can be created for a specific project (tenant) and see the usage of these backup quotas for this project.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

   ```
   $ oc rsh -n openstack openstackclient
   ```

2. List the projects to obtain the ID or name of the required project:

```
$ openstack project list
```

3. View the backup quotas for a specific project:

```
$ openstack quota show <project>
```

- Replace **<project>** with the ID or name of the required project.

The value of the **backup-gigabytes** field in the table is the maximum total size of all backups that can be created in this project. The value of the **backups** field in the table is the maximum number of backups that can be created in this project. For example:

+

```
$ openstack quota show c2c1da89ed1648fc8b4f35a045f8d34c
+-----------------------+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
| Field                 | Value
|
+-----------------------+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
| backup-gigabytes      | 1000
|
| backups               | 10
```

4. Modify the maximum total size of all backups created for a project:

```
$ openstack quota set --backup-gigabytes <maxgb> <project>
```

- Replace **<maxgb>** with the maximum total size, in gigabytes, of the backups that can be created for this project.

5. Modify the maximum number of backups that can be created for a project:

```
$ openstack quota set --backups <maxnum> <project>
```

- Replace **<maxnum>** with the maximum number of backups that can be created for this project.

6. View the usage of these backup quotas for a specific project:

```
$ cinder quota-usage <project_id>
```

- Replace **<project_id>** with the ID of the project.
  For example:

```
$ cinder quota-usage c2c1da89ed1648fc8b4f35a045f8d34c
+-----------------------+--------+----------+-------+-----------+
| Type                  | In_use | Reserved | Limit | Allocated |
+-----------------------+--------+----------+-------+-----------+
| backup_gigabytes      | 7      | 0        | 1000  |           |
| backups               | 7      | 0        | 10    |           |
| gigabytes             | 6      | 0        | 1000  |           |
| gigabytes_multiattach | 0      | 0        | -1    |           |
| gigabytes_tripleo    | 6     | 0        | -1    |           |
| groups               | 0     | 0        | 10    |           |
| per_volume_gigabytes | 0     | 0        | -1    |           |
| snapshots            | 1     | 0        | 10    |           |
| snapshots_multiattach | 0    | 0        | -1    |           |
| snapshots_tripleo    | 1     | 0        | -1    |           |
| volumes              | 5     | 0        | 10    |           |
| volumes_multiattach  | 0     | 0        | -1    |           |
| volumes_tripleo      | 5     | 0        | -1    |           |
+----------------------+--------+----------+-------+-----------+
```

Verification

- If you have changed either of these quotas then review these changes:

  ```
  $ openstack quota show <project>
  ```

  Ensure that the modified values are specified by the **backup-gigabytes** and **backups** fields in the table. For example:

  ```
  +----------------------+-------------------------------------------------------------------------------------
  ----------------------------------------------------------------------------------------------------------+
  | Field                | Value
  |
  +----------------------+-------------------------------------------------------------------------------------
  ----------------------------------------------------------------------------------------------------------+
  | backup-gigabytes     | 500
  |
  | backups              | 12
  ```

1. Exit the **openstackclient** pod:

   ```
   $ exit
   ```

CHAPTER 4. CUSTOMIZING THE IMAGE SERVICE (GLANCE)

When you have deployed the Image service (glance), you can customize the image import workflow and configure scalability and security features.

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.

4.1. CONFIGURING THE IMAGE IMPORT WORKFLOW

Users can upload their own images to the Image service (glance) by using the default **glance-direct** or **web-download** import methods. If you have multiple Red Hat Ceph Storage back ends for the Image service, you can also enable the **copy-image** import method. You can monitor uploaded images in a staging area before they go active in a storage back end, and you can configure the import workflow to run plugins to make user images discoverable, for example, the Inject Image Metadata plugin for metadata or the Image Conversion plugin for image formats.

4.1.1. Distributed image import

Distributed image import works with the **glance-direct** image import method, and it is enabled by default in Red Hat OpenStack Services on OpenShift (RHOSO).

When using the **glance-direct** image import method, users can upload local images to the Image service (glance) without any requirement for a shared storage area where API worker nodes (or replicas) stage images. Instead, staging is distributed because individual API workers have their own local and unshared staging directory. The API worker that owns the image data is the same API worker that performs the image import.

When the image is created and staged, the Image service records the URL of the staging API worker in a database. With this URL, other API workers can proxy image import requests from clients to the worker that has the image data and can perform the import operation. This workflow allows API worker nodes to be isolated for High Availability (HA) and distributed geographically for a distributed compute node (DCN) environment.

4.1.2. URI allowlist and blocklist for web-download sources

You can limit the sources of web-download image imports by specifying a URI allowlist and blocklist in the **glance** template in your **OpenStackControlPlane** custom resource (CR) file.

You can allow or block image source URIs at three levels:

- scheme (allowed_schemes, disallowed_schemes)
- host (allowed_hosts, disallowed_hosts)
- port (allowed_ports, disallowed_ports)

If you specify both an allowlist and a blocklist at any level, the allowlist is honored and the blocklist is ignored. For an example of a URI allowlist, see Configuring a URI allowlist for web-import sources .

Decision logic for URI validation

The Image service applies the following decision logic to validate image source URIs:

1. The scheme is checked.
    a. Missing scheme: reject
    b. If there is an allowlist, and the scheme is not present in the allowlist: reject. Otherwise, skip C and continue on to 2.
    c. If there is a blocklist, and the scheme is present in the blocklist: reject.
2. The host name is checked.
    a. Missing host name: reject
    b. If there is an allowlist, and the host name is not present in the allowlist: reject. Otherwise, skip C and continue on to 3.
    c. If there is a blocklist, and the host name is present in the blocklist: reject.
3. If there is a port in the URI, the port is checked.
    a. If there is a allowlist, and the port is not present in the allowlist: reject. Otherwise, skip B and continue on to 4.
    b. If there is a blocklist, and the port is present in the blocklist: reject.
4. The URI is accepted as valid.

If you allow a scheme, either by adding it to an allowlist or by not adding it to a blocklist, any URI that uses the default port for that scheme by not including a port is allowed. If the URI does include a port, the URI is validated according to the default decision logic.

4.1.2.1. Default settings for the URI allowlist and blocklist

The following allowlist and blocklist values are the default settings for the **web-download** image import method in your Red Hat OpenStack Services on OpenShift (RHOSO) deployment.

- allowed_schemes - [*http*, *https*]
- disallowed_schemes - empty list
- allowed_hosts - empty list
- disallowed_hosts - empty list
- allowed_ports - [80, 443]
- disallowed_ports - empty list

If you use the default values, users can only access URIs by using the **http** or **https** scheme, and they can only specify ports **80** and **443**. Users do not have to specify a port, but if they do, it must be either **80** or **443**.

4.1.2.2. Configuring a URI allowlist for web-import sources

You configure the sources of web-import image downloads by specifying URI allowlists and blocklists in the **glance** template in your **OpenStackControlPlane** custom resource (CR) file.

In this example, you are using an FTP server for image upload. The default port for FTP is 21.

Procedure

1. Open your **OpenStackControlPlane** custom resource CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template:

   #### Example

   ```
   glance:
     template:
       customServiceConfig: |
         [DEFAULT]
         allowed_schemes = [http,https,ftp]
         disallowed_schemes = []
         allowed_hosts = []
         disallowed_hosts = []
         allowed_ports = [80,443]
         disallowed_ports = []
       glanceAPIs:
         ...
   ```

   - Because **ftp** is in the list for **allowed_schemes**, this URL to the image resource is allowed: ftp://example.org/some/resource.
   - Because 21 is not in the list for **allowed_ports**, this URL to the same image resource is rejected: ftp://example.org:21/some/resource.

2. Update the control plane:

   ```
   $ oc apply -f openstack_control_plane.yaml -n openstack
   ```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

   ```
   $ oc get openstackcontrolplane -n openstack
   ```

   #### TIP

   Append the **-w** option to the end of the **get** command to track deployment progress.

4.1.3. Configuring the copy-image import method

You can configure the Image service (glance) to copy existing images to multiple Red Hat Ceph Storage stores.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **customServiceConfig** in the **glance** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  glance:
    enabled: true
    template:
      customServiceConfig: |
        [DEFAULT]
        enabled_import_methods=web-download,glance-direct,copy-image
      glanceAPIs:
        ...
```

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

4.1.4. Enabling or rejecting disk formats

You can configure the Image service (glance) to enable or reject disk formats. For example, you can enable only RAW and ISO disk formats or reject images in QCOW2 disk format.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  glance:
    enabled: true
    template:
      databaseInstance: openstack
      databaseUser: glance
      customServiceConfig: |
         [image_format]
        disk_formats=<raw>,<iso>
      glanceAPIs:
        ...
```

- Replace **<raw>** and **<iso>** with the formats you want to enable from the following supported supported disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

The **OpenStackControlPlane** resources are created when the status is "Setup complete".

TIP

Append the **-w** option to the end of the **get** command to track deployment progress.

4.1.5. Enabling plugins for the image import workflow

You can enable plugins for the image import workflow by configuring the **image_import_plugins** option in the **glance-image-import.conf** file. The plugins do not run in parallel; they run in the order in which they appear in the **image_import_plugins** list.

Image conversion is enabled by default when you use Red Hat Ceph Storage as the back end for the Image service.

NOTE

When you use image conversion with the ISO image format, the image import operation remains in an **importing** state. If your deployment supports uploading images in ISO format, you can use the **image-create** command to upload ISO images instead of using image conversion with the **image-create-via-import** command:

Example:

```
glance image-create \
--name <iso_image> \
--disk-format iso \
--container-format bare \
--file <my_file.iso>
```

- Replace **<iso_image>** with the name of your image.
- Replace **<my_file.iso>** with the file name for your image.

Procedure

- Configure plugins in the **glance-image-import.conf** file. In this example, you convert the images to RAW format before you inject metadata properties.

```
[image_import_opts]
image_import_plugins = ['image_conversion','inject_image_metadata']
```

```
[image_conversion]
output_format = raw

[inject_metadata_properties]
ignore_user_roles = admin,...
inject = "property1":"value1","property2":"value2",...
```

4.1.5.1. Injecting metadata on image import to control where instances launch

You can enable the Inject Image Metadata plugin to apply metadata properties to images that are imported by cloud users so that instances that are launched from the images are located on specific Compute nodes.

The Inject Image Metadata plugin contains two parameters:

- **ignore_user_roles** is a comma-separated list of Identity service (keystone) roles that the plugin will ignore. If the user making the image import call has any of these roles, the plugin will not inject any properties into the image.
- **inject** is a comma-separated list of properties and values that will be injected into the image record for the imported image.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  glance:
    enabled: true
    template:
      databaseInstance: openstack
      databaseUser: glance
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:rbd
        enabled_import_methods=[web-download,glance-direct]
        [image_import_opts]
        image_import_plugins = ['inject_image_metadata']
        [inject_metadata_properties]
        ignore_user_roles = <admin>,...
        inject = "<property1>":"<value1>","<property2>":"<value2>",...
  ...
```

   - Replace **<admin>** with the user roles you want the plugin to ignore.
   - Replace **<property1>**, **<value1>**, **<property2>**, **<value2>**, and so on with the properties and values that you want to inject to the image.

2. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

   The **OpenStackControlPlane** resources are created when the status is "Setup complete".

   TIP

   Append the **-w** option to the end of the **get** command to track deployment progress.

4.2. CONFIGURING SCALABILITY AND SECURITY

You can improve the scalability of Image service (glance) operations by configuring sparse image upload. You can configure security features, such as image signing and verification and metadata definition (metadef) APIs.

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.

4.2.1. Enabling sparse image upload

When you use Red Hat Ceph Storage RADOS Block Device (RBD) as the storage back end for the Image service (glance), you can use sparse image upload to reduce network traffic and save storage space. This feature is useful in distributed compute node (DCN) environments. With a sparse image file, the Image service does not write null byte sequences. The Image service writes data with a given offset. Storage back ends interpret these offsets as null bytes that do not consume storage space.

Prerequisites

- Your Red Hat OpenStack Services on OpenShift (RHOSO) deployment uses Ceph Storage RBD as the storage back end for the Image service.

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and add the following parameters to the **glance** template. Enable sparse image upload by setting the **rbd_thin_provisioning** parameter to **True**:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
spec:
    ...
  glance:
    template:
      databaseInstance: openstack
      databaseUser: glance
      customServiceConfig: |
        [DEFAULT]
        enabled_backends = default_backend:rbd
        enabled_import_methods=[web-download]
        [glance_store]
        default_backend = default_backend
        [default_backend]
        rbd_store_ceph_conf = /etc/ceph/ceph.conf
        store_description = "RBD backend"
        rbd_store_pool = images
        rbd_store_user = openstack
        rbd_thin_provisioning = True
  ...
```

2. Update the control plane:

   ```
   $ oc apply -f openstack_control_plane.yaml -n openstack
   ```

3. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

   ```
   $ oc get openstackcontrolplane -n openstack
   ```

   The **OpenStackControlPlane** resources are created when the status is "Setup complete".

   ### TIP

   Append the **-w** option to the end of the **get** command to track deployment progress.

Verification

You can import an image and check its size to verify sparse image upload.

1. Download the image file locally:

   ```
   $ wget <file_location>/<file_name>
   ```

   - Replace **<file_location>** with the location of the file.
   - Replace **<file_name>** with the name of the file.
     For example:

     ```
     $ wget https://cloud.centos.org/centos/6/images/CentOS-6-x86_64-GenericCloud-
     1508.qcow2
     ```

2. Check the disk size and the virtual size of the image to be uploaded:

   ```
   $ qemu-img info <file_name>
   ```

   For example:

   ```
   $ qemu-img info CentOS-6-x86_64-GenericCloud-1508.qcow2
   ```

```
image: CentOS-6-x86_64-GenericCloud-1508.qcow2
file format: qcow2
virtual size: 8 GiB (8589934592 bytes)
disk size: 1.09 GiB
cluster_size: 65536
Format specific information:
compat: 0.10
refcount bits: 1
```

3. Import the image:

```
$ glance image-create-via-import --disk-format qcow2 --container-format bare --name
centos_1 --file <file_name>
```

4. Record the image ID. It is required in a subsequent step.

5. Verify that the image is imported and in an active state:

   - Access the remote shell for the **OpenStackClient** pod from your workstation:

     ```
     $ oc rsh -n openstack openstackclient
     ```

   - View the image:

     ```
     $ openstack image show <image_id>
     ```

   - Exit the openstackclient pod:

     ```
     $ exit
     ```

6. From a Ceph Storage node, verify that the size of the image is less than the virtual size from the output of step 2:

```
$ sudo rbd -p images diff <image_id> | awk '{ SUM += $2 } END { print SUM/1024/1024/1024
" GB" }'

1.03906 GB
```

4.2.2. Image signature verification

You can use image signature verification to validate images that are uploaded to the Image service (glance) before storing the images in the configured back end. If validation fails for an image, then the upload is stopped and the image is deleted.

To protect image integrity and authenticity, you can save the signatures and public key certificates as image properties.

You store the secret for signature verification in the Key Manager service (barbican), and the Image service interacts with the Key Manager service through the internal endpoint provided by the Identity service (keystone):

```
[key_manager]
backend = barbican
```

```
[barbican]
auth_endpoint={{ .KeystoneInternalURL }}
barbican_endpoint_type=internal
```

Other services, such as the Compute service (nova) can use the image properties to perform data validation when a user downloads the image from the Image service.

NOTE

Image signing and verification is not supported if the Compute service (nova) is using Ceph RADOS Block Device (RBD) to store virtual machines disks.

For information about creating signed images, see Signing Image service (glance) images in *Performing security operations*.

4.2.3. Secure metadef APIs

In Red Hat OpenStack Services on OpenShift (RHOSO), you can define key value pairs and tag metadata with metadata definition (metadef) APIs. There is no limit on the number of metadef namespaces, objects, properties, resources, or tags that you can create.

Image service policies control metadef APIs. By default, only administrators can create, update, or delete (CUD) metadef APIs. This limitation prevents metadef APIs from exposing information to unauthorized users and mitigates the risk of a malicious user filling the Image service (glance) database with unlimited resources, which can create a Denial of Service (DoS) style attack.

CHAPTER 5. CUSTOMIZING THE OBJECT STORAGE SERVICE (SWIFT)

You can customize some of the default settings of the Object Storage service (swift) to optimize deployment performance.

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.

5.1. CHANGING DEFAULT PARAMETERS FOR OBJECT STORAGE

You can customize the following options for your Object Storage service (swift) deployment in the OpenStackControlPlane CR:

Table 5.1. OpenStackControlPlane CR options

Option	Description
swiftProxy/ceilometerEnabled	Enables the Ceilometer middleware in the proxy server.
swiftProxy/encryptionEnabled	Enables object encryption by using the Key Manager service (barbican).
swiftRing/minPartHours	Sets the minimum time in hours before a partition in a ring can be moved following a rebalance.
swiftRing/partPower	Sets the partition power to use when building Object Storage rings.
swiftRing/ringReplicas	Sets the number of object replicas to use in the Object Storage rings.

You can customize the following configuration files for Object Storage services by using the **defaultConfigOverwrite** parameter and keys in the OpenStackControlPlane CR:

Table 5.2. Configuration file options

Service	Key
account-server	**01-account-server.conf**
container-server	**01-container-server.conf**
object-server	**01-object-server.conf**

Service	Key
object-expirer	**01-object-expirer.conf**
proxy-server	**01-proxy-server.conf**

Procedure

1. Open your **OpenStackControlPlane** CR file, **openstack_control_plane.yaml**, and enable Ceilometer middleware and object encryption under the **swiftProxy** parameter in the **swift** template:

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: openstack-control-plane
  namespace: openstack
spec:
  ...
  swift:
    enabled: true
    template:
      swiftProxy:
        ceilometerEnabled: true
        encryptionEnabled: true
        replicas: 2
...
```

2. Add values for **minPartHours**, **partPower**, and **ringReplicas** under the **swiftRing** parameter:

```
...
spec:
  ...
  swift:
    enabled: true
    template:
      swiftProxy:
        ...
      swiftRing:
        minPartHours: <number_of_hours>
        partPower: <partition_power>
        ringReplicas: <number_of_copies>
...
```

- Replace **<number_of_hours>** with the minimum time in hours before you want a partition in a ring to be moved following a rebalance.
- Replace **<partition_power>** with the partition power you want to use when building Object Storage rings, for example, **12**.
- Replace **<number_of_copies>** with the number of object copies you want in your cluster.

3. Change the number of workers in the **object-server** service by adding the **defaultConfigOverwrite** parameter under the **swiftStorage** parameter:

```
...
spec:
  ...
  swift:
    enabled: true
    template:
      swiftProxy:
        ...
      swiftRing:
        ...
      swiftStorage:
        replicas: 3
        storageClass: local-storage
        storageRequest: 10Gi
        defaultConfigOverwrite:
          01-object-server.conf: |
            [DEFAULT]
            workers = <number_of_workers>
```

   - Replace **<number_of_workers>** with the number of workers you want in the **object-server** service.

4. Update the control plane:

```
$ oc apply -f openstack_control_plane.yaml -n openstack
```

5. Wait until RHOCP creates the resources related to the **OpenStackControlPlane** CR. Run the following command to check the status:

```
$ oc get openstackcontrolplane -n openstack
```

   The **OpenStackControlPlane** resources are created when the status is "Setup complete".

   TIP

   Append the **-w** option to the end of the **get** command to track deployment progress.

5.2. CUSTOM RINGS

You can create custom rings to update existing Object Storage service (swift) clusters.

When you add new nodes to a cluster, their characteristics might differ from those of the original nodes. Without custom adjustments, the larger capacity of the new nodes may be underused, or if weights change in the rings, data dispersion can become uneven, which reduces safety.

The ring builder helps manage Object Storage as clusters grow and technologies evolve. For assistance with creating custom rings, contact Red Hat Support.

5.3. CHECKING CLUSTER HEALTH

The Object Storage service (swift) runs many processes in the background to ensure long-term data availability, durability, and persistence. For example:

- Auditors constantly re-read database and object files and compare them by using checksums to make sure there is no silent bit-rot. Any database or object file that no longer matches its checksum is quarantined and becomes unreadable on that node. The replicators then copy one of the other replicas to make the local copy available again.
- Objects and files can disappear when you replace disks or nodes or when objects are quarantined. When this happens, replicators copy missing objects or database files to one of the other nodes.

The Object Storage service includes a tool called **swift-recon** that collects data from all nodes and checks the overall cluster health. You can use the **swift-recon** command line utility to obtain metrics from the account, container, and object servers.

Procedure

1. Log in to one of the Controller nodes.
2. Run the following command:

   ```
   $ oc debug --keep-labels=true job/swift-ring-rebalance -- /bin/sh -c 'swift-ring-tool get &&
   swift-recon -arqlT --md5'
   ```

   - Optional: Use the **--all** option to return additional output.
     This command queries all servers on the ring for the following data:
     - Async pendings: If the cluster load is too high and processes cannot update database files fast enough, some updates occur asynchronously. These numbers decrease over time.
     - Replication metrics: Review the replication timestamps; full replication passes happen frequently with few errors. An old entry, for example, an entry with a timestamp from six months ago, indicates that replication on the node has not completed in the last six months.
     - Ring md5sums: This ensures that all ring files are consistent on all nodes.
     - **swift.conf** md5sums: This ensures that all configuration files are consistent on all nodes.
     - Quarantined files: There must be no, or very few, quarantined files for all nodes.
     - Time-sync: All nodes must be synchronized.

CHAPTER 6. CUSTOMIZING THE SHARED FILE SYSTEMS SERVICE (MANILA)

With the Shared File Systems service (manila), you can provision shared file systems that multiple cloud user instances, bare-metal nodes, or containers can consume. You can create share types to prepare the share service and enable cloud users to create and manage shares. You use the OpenStack command-line interface (CLI) to manage shared file systems.

Prerequisites

- You have the **oc** command line tool installed on your workstation.
- You are logged on to a workstation that has access to the RHOSO control plane as a user with **cluster-admin** privileges.
- An end user requires at least one share type to use the Shared File Systems service.
- For a Compute instance to connect to the shared provider network, the user must add an additional Networking service (neutron) port.

6.1. CREATING SHARE TYPES

You can create share types to define the type of service that the Shared File Systems service (manila) scheduler uses to make scheduling decisions and that drivers use to control share creation.

Share types include a description and extra specifications, for example, **driver_handles_share_servers** and **snapshot_support**, to filter back ends.

Users require at least one share type to use the Shared File Systems service, and users can only create shares that match the available share types.

By default, share types are public, which means they are available to all cloud projects. However, you can create private share types for use in specific projects.

In the following example procedure, you use the **driver_handles_share_servers** parameter (DHSS), which you can set to **true** or **false**:

- For CephFS-NFS and native CephFS, you set DHSS to **false**.
- For other back ends, you can set DHSS to **true** or **false**, based on the custom resource (CR) configuration.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

   ```
   $ oc rsh -n openstack openstackclient
   ```

2. Run the following command to create a share type:

   ```
   $ openstack share type create default <driver_handles_share_servers>
   ```

   - Replace **<driver_handles_share_servers>** with **true** or **false**.

3. Add specifications to the default share type or create additional share types to use with different back ends. In this example, configure the default share type to select a CephFS back end and an additional share type that uses a NetApp **driver_handles_share_servers=true** back end:

   ```
   $ openstack share type create default false \
     --extra-specs share_backend_name='cephfs'
   $ openstack share type create netapp true \
     --extra-specs share_backend_name='netapp_ontap'
   ```

4. Exit the **openstackclient** pod:

   ```
   $ exit
   ```

6.2. COMPARING SHARE TYPES

You can create share types to define the capabilities of shares. The following table describes share types and their capabilities.

Table 6.1. Share types

Share type	Values	Description
driver_handles_share_servers	true or false	Grants permission to use share networks to create shares.
snapshot_support	true or false	Grants permission to create snapshots of shares.
create_share_from_snapshot_support	true or false	Grants permission to create clones of share snapshots.
revert_to_snapshot_support	true or false	Grants permission to revert your shares to the most recent snapshot.
mount_snapshot_support	true or false	Grants permission to export and mount your snapshots.
replication_type	dr	Grants permission to create replicas for disaster recovery. Only one active export is allowed at a time.
	readable	Grants permission to create read-only replicas. Only one writable, active export is allowed at a time.
	writable	Grants permission to create read/write replicas. Any number of active exports are allowed at a time per share.

Share type	Values	Description
availability_zones	a list of one or more availability zones	Grants permission to create shares only on the availability zones listed.

6.3. ADDING AND REMOVING SHARES BY USING MANAGE/UNMANAGE

You can manage file shares that already exist in storage by using the manage/unmanage feature of the Shared File Systems service (manila). Users can perform operations on managed shares, such as granting access, mounting, and resizing, in the same way that they perform these operations on Shared File Systems service shares.

You can manage the lifecycle of shares that have the **driver_handles_share_servers** (DHSS) parameter set to true and shares that have DHSS set to false. To manage DHSS=true shares, you must also manage the share server that contains the share.

When you unmanage a share, you remove the share from the management of the Shared File Systems service without deleting the share. If shares have dependent snapshots or share replicas, you can only remove the shares from the Shared File Systems service when the snapshots or share replicas have been removed.

Limitations

- The driver must support manage/unmanage functionality.
- The manage/unmanage feature does not support native CephFS or CephFS-NFS back ends. You can remove CephFS shares from the management of the Shared File Systems service. However, you cannot bring existing CephFS shares under the management of the Shared File Systems service.
- When you bring a share under the management of the Shared File Systems service, existing clients are disconnected. When you remove a share from the management of the Shared File Systems service, existing clients remain connected.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. Manage a share:

```
$ openstack share adopt
    --name <name>
    --description <description>
    --share-type <share_type>
    --driver-options [<key=value> [<key=value> ...]]
                [--public] [--share-server-id <share_server_id>] \
                [--wait] <service_host> <protocol> <export_path>
```

- Replace **<name>** with a descriptive name for the share.
- Replace **<description>** with a description of the share.
- Replace **<share_type>** with the share type of the share.
- Replace **<key=value>** with the key-value pair of the driver property you want to associate to your share. You can use multiple key-value pairs you want to associate to your image.
- Replace **<share_server_id>** with ID of your share server.
- Replace **<service_host>** with the host server.
- Replace **<protocol>** with the NAS protocol of the share, for example, **nfs**.
- Replace **<export_path>** with the export path of the share.

3. Verify that the share is available:

```
$ openstack share show <name>
```

4. Unmanage a share:

```
$ openstack share abandon [--wait] <name>
```

5. Exit the **openstackclient** pod:

```
$ exit
```

6.4. CHANGING THE DEFAULT QUOTAS FOR PROJECTS, USERS, AND SHARE TYPES

To prevent system capacities from being exhausted without notification, you can configure quotas for the Shared File Systems service (manila). The Shared File Systems service enforces some default quotas, but you can override the default quotas so that individual projects have different consumption limits.

You can update the following quotas for all users in a project, a specific project user, or a share type that is used by the project users. You can only set **share-type** quotas at the project level. You cannot set **share-type** quotas for specific project users.

Table 6.2. Quotas

Quota	Description
shares	Number of shares you can create
snapshots	Number of snapshots you can create

Quota	Description
share-groups	Total number of share groups you can create
share-group-snapshots	Total number of share group snapshots you can create
share-networks	Total number of share networks you can create
share-replicas	Total number of share replicas you can create
gigabytes	Total size in GB that you can allocate for all shares
snapshot-gigabytes	Total size in GB that you can allocate for all snapshots of shares
replica-gigabytes	Total size in GB that you can allocate across all share replicas

6.4.1. Viewing quotas for projects, users, and share types

You can view the quotas for a project, user, or share type in the Shared File Systems service (manila) by using the **openstack share quota show** command. The **--user** and **--share-type** command options are mutually exclusive.

- If you include the **--user** option, you can view the quota for a user in the project.
- If you omit the **--user** option, you can view the quotas that apply to all users in the project.
- If you include the **--share-type** option, you can view the quota for a specific share type in the project.

In the following procedure, enter the values carefully. The Shared File Systems service does not detect or report incorrect values.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

   ```
   $ oc rsh -n openstack openstackclient
   ```

2. Use the following commands to view quotas:
   - View the quotas for a project:

     ```
     $ openstack share quota show <af2838436f3f4cf6896399dd97c4c050>
     ```

     - Replace **<af2838436f3f4cf6896399dd97c4c050>** with the project ID.
   - View the quotas for a project user:

     ```
     $ openstack share quota show <af2838436f3f4cf6896399dd97c4c050> \
       --user <81ebb491dd0e4c2aae0775dd564e76d1>
     ```

- Replace **<81ebb491dd0e4c2aae0775dd564e76d1>** with the user ID.

- View the quotas for a specific share type in a project:

```
$ openstack share quota show <af2838436f3f4cf6896399dd97c4c050> \
  --share-type <dhss_false>
```

  - Replace **<dhss_false>** with the share type you want to check.

3. Exit the **openstackclient** pod:

```
$ exit
```

6.4.2. Updating quotas for projects, users, and share types

You can update quotas for all project users, a specific project user, or a share type in a project by using the **openstack share quota set** command. You can only set **share-type** quotas at the project level, not for specific project users.

In the following procedure, enter the values carefully. The Shared File Systems service does not detect or report incorrect values.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. Use the following commands to update quotas:

- Update quotas for all users in a project:

```
$ openstack share quota set <project_id> \
  [--shares <share_quota> \
  --gigabytes <gigabytes_quota> \
  ...]
```

  - Replace **<project_id>** with the project ID. This value must be the project ID, not the project name.
  - Replace **<share_quota>** with the total number of shares you want to set as the quota for the project.
  - Replace **<gigabytes_quota>** with the total size in GB that you want to allocate for all shares in the project.

- Update quotas for a specific user in a project:

```
$ openstack share quota set <project_id> \
  --user <user_id> \
  [--shares <share_quota> \
  --gigabytes <gigabytes_quota> \
  ...]
```

   - Replace **<user_id>** with the user ID. The value must be the user ID, not the username.
   - Replace **<share_quota>** with the total number of shares you want to set as the quota for the user in the project.
   - Replace **<gigabytes_quota>** with the total size in GB that you want to allocate for the user's shares in the project.

- Update quotas for all users who use a specific share type:

```
$ openstack share quota set <project_id> \
  --share-type <share_type> \
  [--shares <share_quota>
  --gigabytes <gigabytes_quota> \
  …]
```

   - Replace **<share_type>** with the share type you want to apply the quota to.
   - Replace **<share_quota>** with the total number of shares you want to set as the quota for the share type.
   - Replace **<gigabytes_quota>** with the total size in GB that you want to allocate for the shares that are that share type in the project.

Verification

1. The **openstack share quota set** command does not produce any output. Use the **openstack share quota show** command to verify that a quota was successfully updated.
2. Exit the **openstackclient** pod:

```
$ exit
```

6.4.3. Resetting quotas for projects, users, and share types

You can remove quota overrides for projects, users, and share types in the Shared File Systems service (manila) to return quotas to their default values. Reset quotas to their default values by using the **openstack share quota delete** command.

In the following procedure, enter the values carefully. The Shared File Systems service does not detect or report incorrect values.

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. Use the following commands to reset quotas:

- Reset project quotas:

```
$ openstack share quota delete <project_id>
```

  - Replace **<project_id>** with the project ID. This value must be the project ID, not the project name.

- Reset quotas for a specific user:

  ```
  $ openstack share quota delete <project_id> --user <user_id>
  ```

  - Replace **<user_id>** with the user ID. The value must be the user ID, not the user name.

- Reset quotas for a share type that is used by project users:

  ```
  $ openstack share quota delete <project_id> --share-type <share_type>
  ```

  - Replace **<share_type>** with the share type you want to reset.

Verification

1. The **openstack share quota delete** command does not produce any output. Use the **openstack share quota show** command to verify that a quota was successfully reset.

2. List the default quotas for all projects. Default quotas apply to projects that have no overrides.

   ```
   $ openstack share quota show <project> --defaults
   ```

3. Exit the **openstackclient** pod:

   ```
   $ exit
   ```

6.4.4. Updating the default quota values for projects

You can update the default value of quotas that apply to the Shared File System service (manila) in all projects that do not already have quota overrides.

You can update the default values for any of the following quota options:

Table 6.3. Quota options

Option	Description
--shares <shares>	Adds a new value for the **shares** quota
--snapshots <snapshots>	Adds a new value for the **snapshots** quota
--share-groups <share_groups>	Adds a new value for the **share-groups** quota
--share-group-snapshots <share_group_snapshots	Adds a new value for the **share-group-snapshots** quota
--share-networks <share_networks>	Adds a new value for the **share-networks** quota.
--share-replicas <share_replicas>	Adds a new value for the **share-replicas** quota

Option	Description
--gigabytes <gigabytes>	Adds a new value for the **gigabytes** quota
--snapshot-gigabytes <snapshot_gigabytes>	Adds a new value for the **snapshot-gigabytes** quota
--replica-gigabytes <replica_gigabytes>	Adds a new value for the **replica-gigabytes** quota

Procedure

1. Access the remote shell for the **OpenStackClient** pod from your workstation:

```
$ oc rsh -n openstack openstackclient
```

2. View the usage statement of the **openstack share quota update --class** command:

```
$ openstack share quota set --class
usage: openstack share quota update --class [--shares <shares>] [--snapshots <snapshots>]
                                    [--gigabytes <gigabytes>]
                                    [--snapshot-gigabytes <snapshot_gigabytes>]
                                    [--share-networks <share_networks>]
                                    [--share-replicas <share_replicas>]
                                    [--replica-gigabytes <replica_gigabytes>]
                                     <class_name>
```

NOTE

The parameter **<class_name>** is a positional argument. It identifies the quota class for which the quotas are set. Set the value of this parameter to **default**. No other quota classes are supported.

3. Use the information from the usage statement to update the default quotas. The following example updates the default quotas for **shares** and **gigabytes**:

```
$ openstack share quota set --class default \
  --shares 30 \
  --gigabytes 512
```

Verification

1. List the default quotas for all projects to verify that the default quota values have been reset:

```
$ openstack share quota show <project> --defaults
```

2. Exit the **openstackclient** pod:

```
$ exit
```