Red Hat OpenStack Services on OpenShift 18.0

Performing storage operations

Performing operations with the Block Storage service, Image service, Object Storage service, and Shared File Systems service

Last Updated: 2024-08-19

Red Hat OpenStack Services on OpenShift 18.0 Performing storage operations

Performing operations with the Block Storage service, Image service, Object Storage service, and Shared File Systems service

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

Perform operations with volumes, images, objects, and file shares 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. PERFORMING STORAGE OPERATIONS IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

Red Hat OpenStack Services on OpenShift (RHOSO) provides the following storage services:

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

You can manage cloud storage by using either the RHOSO Dashboard (horizon) or the OpenStack command-line interface (CLI). You can perform most procedures by using either method, but you can only complete some of the more advanced procedures by using the OpenStack CLI.

1.1. BLOCK STORAGE (CINDER)

The Block Storage service (cinder) allows users to provision block storage volumes on back ends. Users can attach volumes to instances to augment their ephemeral storage with general-purpose persistent storage. You can detach and re-attach volumes to instances, but you can only access these volumes through the attached instance.

You can also configure instances so that they do not use ephemeral storage. Instead of using ephemeral storage, you can configure the Block Storage service to write images to a volume. You can then use the volume as a bootable root volume for an instance. Volumes also provide inherent redundancy and disaster recovery through backups and snapshots. However, backups are only provided if you deploy the optional Block Storage backup service. In addition, you can encrypt volumes for added security.

1.2. IMAGES (GLANCE)

The Image service (glance) provides discovery, registration, and delivery services for instance images. It also provides the ability to store snapshots of instances ephemeral disks for cloning or restore purposes. You can use stored images as templates to commission new servers quickly and more consistently than installing a server operating system and individually configuring services.

1.3. OBJECT STORAGE (SWIFT)

The Object Storage service (swift) provides a fully-distributed storage solution that you can use to store any kind of static data or binary object; such as media files, large datasets, and disk images. The Object Storage service organizes objects by using object containers, which are similar to directories in a file system, but they cannot be nested. You can use the Object Storage service as a repository for nearly every service in the cloud.

Red Hat Ceph Storage RGW can be used as an alternative to the Object Storage service.

1.4. SHARED FILE SYSTEMS (MANILA)

The Shared File Systems service (manila) provides the means to provision remote, shareable file systems. These are known as shares. Shares allow projects in the cloud to share POSIX compliant storage, and they can be consumed by multiple instances simultaneously.

Shares are used for instance consumption, and they can be consumed by multiple instances at the same time with read/write access mode.

1.5. 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 2. PERFORMING OPERATIONS WITH THE BLOCK STORAGE BACKUP SERVICE

You can use the Block Storage service (cinder) backup service to perform, protect, restore, and troubleshoot backups.

NOTE

To execute **openstack** client commands on the cloud, you must specify the name of the cloud detailed in your **clouds.yaml** file. You can specify the name of the cloud by using one of the following methods:

- Use the **--os-cloud** option with each command:

```
$ openstack flavor list --os-cloud <cloud_name>
```

Use this option if you access more than one cloud.

- Create an environment variable for the cloud name in your **bashrc** file:

```
`export OS_CLOUD=<cloud_name>`
```

Prerequisites

- The administrator has created a project for you, and they have provided you with a **clouds.yaml** file for you to access the cloud.
- You have installed the **python-openstackclient** package.
- Only administrators and volume owners with access can perform and access backups.

2.1. USING THE BLOCK STORAGE BACKUP SERVICE

You can use the Block Storage backup service to perform full or incremental backups, to protect your backups, and to restore a backup to a volume.

2.1.1. 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.

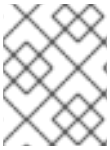
Procedure

- 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.

2.1.2. Creating backups

Create a backup of your Block Storage volume to protect your data from being lost if there is an issue with the volume. For more information, see Creating a full volume backup . You can also create a backup directly from a snapshot of a volume. In addition to the volume data, a backup stores the volume metadata, such as the name and description.

If data compression is enabled for the storage back end for your backups, then your backups are compressed, which can reduce the performance of backup operations.

Full backups are easier to manage but they can become resource intensive when the size of the volume increases over time. With incremental backups, you can capture periodic changes to volumes and minimize resource usage.

When you create a backup of a Block Storage volume, the metadata for this backup is stored in the Block Storage service database. The metadata is used if you restore the volume backup. To ensure that a backup survives a catastrophic loss of the Block Storage service database, you can manually export and store the metadata of this backup.

2.1.2.1. Creating a full volume backup

You can create one or more full backups of a volume.

Prerequisites

- Only volume owners and administrators can backup volumes.
- The storage back end for backups must have the necessary space.
- The backup quotas specified for your project have not been exceeded.

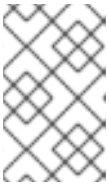
Procedure

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

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

2. List the volumes to obtain the ID or name of the volume you want to back up:

```
$ openstack volume list
```



NOTE

Usually, you can only back up a volume that has an **available** status, but you can backup a volume with an **in-use** status if required. For more information, see Creating a backup of an in-use volume .

3. Back up the volume:

```
$ openstack volume backup create [--name <backup_name>] <volume>
```

- Replace **<volume>** with the ID or name of the volume you want to back up.
- Optional: replace **<backup_name>** with the name of this backup.
  This command immediately provides the ID of this backup but the volume is backed up asynchronously, in the background. For example:

```
$ openstack volume backup create --name vol1bu2 vol_1
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb |
| name  | vol1bu2                              |
+-------+--------------------------------------+
```

Verification

- List the backups:

```
$ openstack volume backup list
```

The volume backup is created when this backup has an **available** status. For example:

```
+--------------------------------------+---------+-------------+-----------+------+
| ID                                   | Name    | Description | Status    | Size |
+--------------------------------------+---------+-------------+-----------+------+
| 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb | vol1bu2 | None        | available |    1 |
| b604a932-94a5-468e-bf6b-841ac16f69a8 | None    | None        | available |    1 |
+--------------------------------------+---------+-------------+-----------+------+
```

1. Exit the **openstackclient** pod:

```
$ exit
```

Additional resources

- Creating backups

2.1.2.2. Creating a full backup of a snapshot

You can create a full backup from a snapshot by using the ID of the volume associated with the snapshot.

The backup is created by directly attaching to the snapshot, which is faster than cloning the snapshot into a volume and then backing up this volume. But this feature can affect the backup performance because of the extra step of creating the volume from a snapshot.

Prerequisites:

- Your backup repository must have the necessary space.

- The backup quotas specified for your project have not been exceeded.

Procedure

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

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

2. List the snapshots to obtain the name or ID of the snapshot you want to backup:

```
$ openstack volume snapshot list
```

3. List the details of this snapshot to obtain the ID of the volume associated with this snapshot:

```
$ openstack volume snapshot show <snapshot>
```

   - Replace **<snapshot>** with the name or ID of the snapshot you want to backup.
     The value of the **volume_id** field is the ID of the volume associated with this snapshot. For example:

```
$ openstack volume snapshot show snap_1
+-------------------------------------------+--------------------------------------+
| Field                                     | Value                                |
+-------------------------------------------+--------------------------------------+
| created_at                                | 2023-07-18T08:14:16.000000           |
| description                               | None                                 |
| id                                        | 2d95b707-6657-49af-ac8f-f9a7417d4650 |
| name                                      | snap_1                               |
| os-extended-snapshot-attributes:progress  | 100%                                 |
| os-extended-snapshot-attributes:project_id | c2c1da89ed1648fc8b4f35a045f8d34c     |
| properties                                |                                      |
| size                                      | 1                                    |
| status                                    | available                            |
| updated_at                                | 2023-07-18T08:14:17.000000           |
| volume_id                                 | 6841e3d1-8a1a-4496-bc51-f7c04b787e8f |
+-------------------------------------------+--------------------------------------+
```

4. Backup the snapshot:

```
$ openstack volume backup create [--name <backup_name>] --snapshot <snapshot>
<volume_id>
```

   - Replace **<volume_id>** with the ID of the volume associated with this snapshot.
   - Optional: Replace **<backup_name>** with the name of this backup.
     This command immediately provides the ID of this backup but the snapshot is backed up asynchronously, in the background. For example:

```
$ openstack volume backup create --name snap1bu1 --snapshot snap_1 6841e3d1-
8a1a-4496-bc51-f7c04b787e8f
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
```

```
| id    | 867e6cfb-9be7-47fa-8a79-221b0e80c757 |
| name  | snap1bu1                             |
+-------+--------------------------------------+
```

Verification

- List the backups:

```
$ openstack volume backup list
```

The snapshot backup is created when this backup has an **available** status. For example:

```
+--------------------------------------+-----------+-------------+-----------+------+
| ID                                   | Name      | Description | Status    | Size |
+--------------------------------------+-----------+-------------+-----------+------+
| 867e6cfb-9be7-47fa-8a79-221b0e80c757 | snap1bu1  | None        | available |    1 |
+--------------------------------------+-----------+-------------+-----------+------+
```

1. Exit the **openstackclient** pod:

```
$ exit
```

2.1.2.3. Creating a backup of an in-use volume

Usually you can only backup a volume that has an **available** status. But you can use the **--force** option when creating a backup, to back up a volume that has an **in-use** status.

When you use the **--force** volume backup option, you create a crash-consistent, but not an application-consistent, backup because the volume is not quiesced before performing the backup. Therefore, the data is intact but the backup does not have an awareness of which applications were running when the backup was performed.

Prerequisites

- Only volume owners and project administrators can backup volumes.
- Your backup repository must have the necessary space.
- The backup quotas specified for your project have not been exceeded.

Procedure

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

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

2. List the volumes to obtain the ID or name of the volume you want to back up:

```
$ openstack volume list
```

For example:

```
+--------------------------------------+---------------+-----------+------+-------------------------------+
```

```
| ID                                   | Name          | Status    | Size | Attached to                  |
+--------------------------------------+---------------+-----------+------+------------------------------+
| 6841e3d1-8a1a-4496-bc51-f7c04b787e8f | vol_1         | available |    1 |                              |
| 92800cf6-82ae-448a-a2bb-872fa4d98099 | Pansible_vol_2 | in-use    |    1 | Attached to
inst1 on /dev/vdc  |
+--------------------------------------+---------------+-----------+------+------------------------------+
```

3. Force the back up, if the volume that you want to backup has an **in-use** status:

```
$ openstack volume backup create [--name <backup_name>] --force <volume>
```

- Replace **<volume>** with the ID or name of the volume you want to back up.
- Optional: replace **<backup_name>** with the name of this backup.
  This command immediately provides the ID of this backup but the volume is backed up asynchronously, in the background. For example:

```
$ openstack volume backup create --name panvol2bu1 --force Pansible_vol_2
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | 8c72bbf3-eb8e-4459-83e9-c7654ebe6343 |
| name  | panvol2bu1                           |
+-------+--------------------------------------+
```

Verification

- List the backups:

```
$ openstack volume backup list
```

The volume backup is created when this backup has an **available** status. For example:

```
+--------------------------------------+-----------+-------------+-----------+------+
| ID                                   | Name      | Description | Status    | Size |
+--------------------------------------+-----------+-------------+-----------+------+
| 8c72bbf3-eb8e-4459-83e9-c7654ebe6343 | panvol2bu1    | None        | available |    1 |
+--------------------------------------+-----------+-------------+-----------+------+
```

1. Exit the **openstackclient** pod:

```
$ exit
```

2.1.2.4. Incremental backups

If a volume has at least one full backup, you can use the Block Storage backup service to create an incremental backup. For more information, see Creating an incremental backup .

Full backups are easier to manage but they can become resource intensive when the size of the volume increases over time. With incremental backups, you can capture periodic changes to volumes and minimize your resource usage.

An incremental backup only stores the changes made to the volume since the last full or incremental backup.

Incremental backups increase the administrative overhead required for managing your backups. For instance, you cannot delete a full backup if it already has one or more incremental backups, you can only delete the latest incremental backup.

Incremental backups have a lower performance than full backups: When you create an incremental backup, all of the data in the volume must first be read and compared with the data in both the full backup and each subsequent incremental backup.

2.1.2.5. Creating an incremental backup

You can create an incremental backup to only store the changes made to the volume since the last full or incremental backup.

Prerequisites:

- At least one full backup of the volume. For more information, see Creating a full volume backup .
- Only volume owners and project administrators can backup volumes.
- Your backup repository must have the necessary space.
- The backup quotas specified for your project have not been exceeded.

Procedure

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

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

2. List the volumes to obtain the ID or name of the volume you want to back up:

```
$ openstack volume list
```

3. Back up the volume and use the **--incremental** option:

```
$ openstack volume backup create --incremental [--name <backup_name>] <volume>
```

   - Replace **<volume>** with the ID or name of the volume you want to back up.
   - Optional: replace **<backup_name>** with the name of this backup.
     This command immediately provides the ID of this backup but the volume is backed up asynchronously, in the background. For example:

```
$ openstack volume backup create --name vol3incbu1 --incremental vol_3
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | f1681313-b5ed-4520-9b63-5b533f7cdc11 |
| name  | vol3incbu1                           |
+-------+--------------------------------------+
```

Verification

- List the backups:

```
$ openstack volume backup list
```

The volume backup is created when this backup has an **available** status. For example:

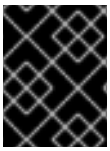
```
+--------------------------------------+---------+-------------+-----------+------+
| ID                                   | Name    | Description | Status    | Size |
+--------------------------------------+---------+-------------+-----------+------+
| f1681313-b5ed-4520-9b63-5b533f7cdc11 | vol3incbu1 | None        | available |    1 |
| f0e9ba67-67e1-4c2c-96ce-221df75bf2c2 | vol3bu1    | None        | available |    1 |
+--------------------------------------+---------+-------------+-----------+------+
```

1. Exit the **openstackclient** pod:

```
$ exit
```

2.1.2.6. Canceling a backup

You must request a force delete on the backup to cancel it.



IMPORTANT

You cannot cancel backups if you use the Red Hat Ceph Storage back end for your backup repository.

Procedure

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

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

2. List the backups to obtain the ID or name of the backup you want to cancel:

```
$ openstack volume backup list
```

3. Cancel the backup:

```
# openstack volume backup delete --force <backup>
```

- Replace **<backup>** with the ID or name of the volume backup that you want to cancel. There can be a slight delay for the backup to be successfully canceled.

Verification

- The backup is canceled when the backup record is not listed by this command:

```
$ openstack volume backup show <backup>
```

1. Exit the **openstackclient** pod:

```
$ exit
```

2.1.3. Protecting your backups

When you create a backup of a Block Storage volume, the metadata for this backup is stored in the Block Storage service database, which is used to restore this volume. To ensure that a backup survives a catastrophic loss of the Block Storage service database, you can manually export and store the metadata of this backup in a safe location, such as an offsite backup. For more information, see Exporting backup metadata.

When the Block Storage service database experiences a catastrophic loss, you cannot restore any of your backups because this database contains the backup metadata used when restoring backups. But if you manually exported and saved the metadata of a backup, then you can import this metadata into the new Block Storage database, so that you can use this backup to restore the volume. For more information see Importing backup metadata.

NOTE

For incremental backups, you must import the metadata of all the preceding backups before you can use it to restore the volume.

2.1.3.1. Exporting backup metadata

You can export the metadata of a backup and store it in a file so that you can restore the volume backup even if the Block Storage database suffers a catastrophic loss.

NOTE

For an incremental backup, you must export the metadata of all the preceding backups.

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

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

2. List the backups to obtain the ID or name of the backup:

```
$ openstack volume backup list
```

3. Export the metadata of the backup and store it in an appropriately named YAML file:

```
$ openstack volume backup record export -f yaml <backup> > <filename>.yaml
```

   - Replace **<backup>** with the ID or name of the volume backup.
   - Replace **<filename>** with the name of the YAML file to save the exported **backup_service** and **backup_url** values for this backup.
     For example:

```
$ openstack volume backup record export -f yaml vol1bu2 > vol1bu2.yaml
```

4. Copy the file to a safe location, such as an offsite backup.

Verification

- Edit the file to see that the values of the **backup_service** and **backup_url**, match the values provided by this command:

```
$ openstack volume backup record export -f yaml <backup>
```

For example:

```
$ openstack volume backup record export -f yaml vol1bu2
backup_service: cinder.backup.drivers.ceph.CephBackupDriver
backup_url: eyJkcml2 … YWxzZX0=
```

1. Exit the **openstackclient** pod:

```
$ exit
```

2.1.3.2. Importing backup metadata

If you export and save the metadata of a volume backup, then you can import this metadata and use the backup if there is a loss of the Block Storage service database.

You can also use this procedure to recreate a backup that was deleted.

NOTE

For incremental backups, you must also import the metadata of all the preceding backups.

Prerequisites

- You must provide the **backup_service** and **backup_url** metadata values of this backup. For more information see Exporting backup metadata.
- A Block Storage database that does not already contain this backup.

Procedure

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

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

2. Locate the file in which you stored the exported **backup_service** and **backup_url** metadata values of this backup.
3. Import the metadata values of this volume backup to the Block Storage database:

```
$ openstack volume backup record import <backup_service> <backup_url>
```

- Replace **<backup_service>** with the **backup_service** metadata value of this volume backup.
- Replace **<backup_url>** with the **backup_url** metadata value of this volume backup.
This command provides the name and the ID of this backup. For example:

```
$ openstack volume backup record import cinder.backup.drivers.ceph.CephBackupDriver
eyJkcml2 … YWxzZX0=
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb |
| name  | vol1bu2                              |
+-------+--------------------------------------+
```

4. Exit the **openstackclient** pod:

```
$ exit
```

Next steps

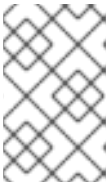
- Restoring backups

2.1.4. Restoring backups

After you create a Block Storage volume backup, you can restore this backed up data if needed.

You can use one of the following methods to restore your backups:

- Restore the backup to a volume that you specify. For more information, see Restoring a backup to a specific volume.



NOTE

If you choose Red Hat Ceph Storage as the back end for your backup repository, then you can only restore backed up volumes to a RBD-based Block Storage back end.

- Restore the backup to a new volume. For more information, see Restoring a backup to a new volume.

IMPORTANT

When the Block Storage service database experiences a catastrophic loss you cannot restore any of your backups, unless you have exported and saved their metadata.

Only administrators can cancel restoring a volume backup.

2.1.4.1. Restoring a backup to a specific volume

You can restore a volume backup to an **available** volume that you have already created.

If you restore a volume from an encrypted volume backup, then the destination volume type must also be encrypted.

IMPORTANT

If you choose Red Hat Ceph Storage as the back end for your backup repository, then you can only restore backed up volumes to a RBD-based Block Storage back end.

Procedure

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

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

2. List the backups to obtain the name or ID of the backup you want to restore:

```
$ openstack volume backup list
```

For example:

```
+--------------------------------------+---------+-------------+-----------+------+
| ID                                   | Name    | Description | Status    | Size |
+--------------------------------------+---------+-------------+-----------+------+
| 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb | vol1bu2 | None        | available |    1 |
```

3. List the volumes:

```
$ openstack volume list
```

Ensure that the status of the required volume is **available** and then obtain the name or ID of this volume. For example:

```
+--------------------------------------+--------------+-----------+------+--------------------------------+
| ID                                   | Name         | Status    | Size | Attached to                    |
+--------------------------------------+--------------+-----------+------+--------------------------------+
| 654e2be8-bc79-4528-96a7-5f773d31c201 | vol_3        | available |    1 |
|
```

4. Restore the backup to the volume:

```
$ openstack volume backup restore <backup> <volume>
```

- Replace **<backup>** with the name or ID of the Block Storage volume backup.
- Replace **<volume>** with the name or ID of the **available** Block Storage volume. For example:

```
$ openstack volume backup restore vol1bu2 vol_3
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| backup_id   | 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb |
| volume_id   | 654e2be8-bc79-4528-96a7-5f773d31c201 |
| volume_name | vol_3                                |
+-------------+--------------------------------------+
```

5. Verify that the **backup_id** provided by this command corresponds to the ID of the backup that was restored and that the **volume_name** and **volume_id** values correspond to the name and ID of the specified volume.

6. Delete the backup if you no longer need it:

```
$ openstack volume backup delete <backup>
```

7. Exit the **openstackclient** pod:

```
$ exit
```

2.1.4.2. Restoring a backup to a new volume

You can create a new volume when you restore a backup of a Block Storage volume.

Procedure

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

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

2. List the backups to obtain the name or ID of the backup you want to restore:

```
$ openstack volume backup list
```

For example:

```
+--------------------------------------+---------+-------------+-----------+------+
| ID                                   | Name    | Description | Status    | Size |
+--------------------------------------+---------+-------------+-----------+------+
| 83dadc43-2aa9-4c0b-bc05-a12203a8f4cb | vol1bu2 | None        | available |    1 |
```

3. Restore the backup to a new volume:

```
$ openstack volume backup restore <backup> [<volume>]
```

- Replace **<backup>** with the ID of the Block Storage volume backup.
- Optional: Replace **<volume>** with the ID of the Block Storage volume.

4. Verify that the **backup_id** provided by this command corresponds to the ID of the backup that was restored.
The **volume_id** value is the ID of the created volume. But the **volume_name** can be a temporary name that is replaced with the name of the backed up volume.

5. List the volumes to verify that the volume with an ID of **volume_id** has been created and to obtain this volume name:

```
$ openstack volume list
```

For example:

```
+--------------------------------------+---------------+-----------+------+-------------------------------+
| ID                                   | Name          | Status    | Size | Attached to                   |
+--------------------------------------+---------------+-----------+------+-------------------------------+
| 296c853c-c749-4eb6-857a-57ec182232a6 | vol_1         | available |    1 |
|
```

6. Delete the backup if you no longer need it:

```
$ openstack volume backup delete <backup>
```

7. Exit the **openstackclient** pod:

```
$ exit
```

2.1.4.3. Canceling restoring a backup

You can cancel restoring a volume backup by changing the status of the backup to **error**. But you cannot cancel restoring a backup when Red Hat Ceph Storage is the back end of the backup repository.

WARNING

If you cancel restoring a backup after it starts, the destination volume is useless, because there is no way of knowing how much data, if any, was actually restored.

Prerequisites

- Ensure that the back end of your backup repository is not Red Hat Ceph Storage.
- Only administrators can cancel restoring a volume backup.

Procedure

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

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

2. List the backups to obtain the name or ID of the backup that you want to stop restoring:

```
$ openstack volume backup list
```

3. Change the status of this backup to **error** to cancel its restore operation:

```
$ openstack volume backup set --state error <backup>
```

- Replace **<backup>** with the name or ID of the volume backup that you do not want to restore.

Canceling a restore is an asynchronous action, because the back end of the backup repository must detect the change in the backup status before it cancels the restore.

Verification

- List the volume backups to verify that the restore is canceled:

```
$ openstack volume backup list
```

When the status of the backup changes to **available**, then the restore is canceled.

1. Exit the **openstackclient** pod:

```
$ exit
```

2.2. TROUBLESHOOTING THE BLOCK STORAGE BACKUP SERVICE

You can diagnose many issues by verifying that the Block Storage services are running correctly and then by examining the log files for error messages.

2.2.1. Troubleshooting backups

The Block Storage backup service performs static checks when receiving a request to back up a Block Storage (cinder) volume. If these checks fail then you will immediately be notified:

- Check for an invalid volume reference (**missing**).
- Check if the volume is **in-use** or attached to an instance. The **in-use** case requires you to use the **--force** option to perform a backup. For more information, see Creating a backup of an in-use volume.
  When you use the **--force** volume backup option, you create a crash-consistent, but not an application-consistent, backup because the volume is not quiesced before performing the backup. Therefore, the data is intact but the backup does not have an awareness of which applications were running when the backup was performed.

When these checks succeed: the Block Storage backup service accepts the request to backup this volume, the CLI backup command returns immediately, and the volume is backed up in the background.

Therefore the CLI backup command returns even if the backup fails. You can use the **openstack volume backup list** command to verify that the volume backup is successful, when the **Status** of the backup entry is **available**.

If a backup fails, examine the Block Storage backup service log file for error messages to discover the cause.

2.2.2. Examining the Block Storage backup service log file

When a backup or restore does not succeed, you can examine the Block Storage backup service log file for error messages that can help you to determine the reason.

Procedure

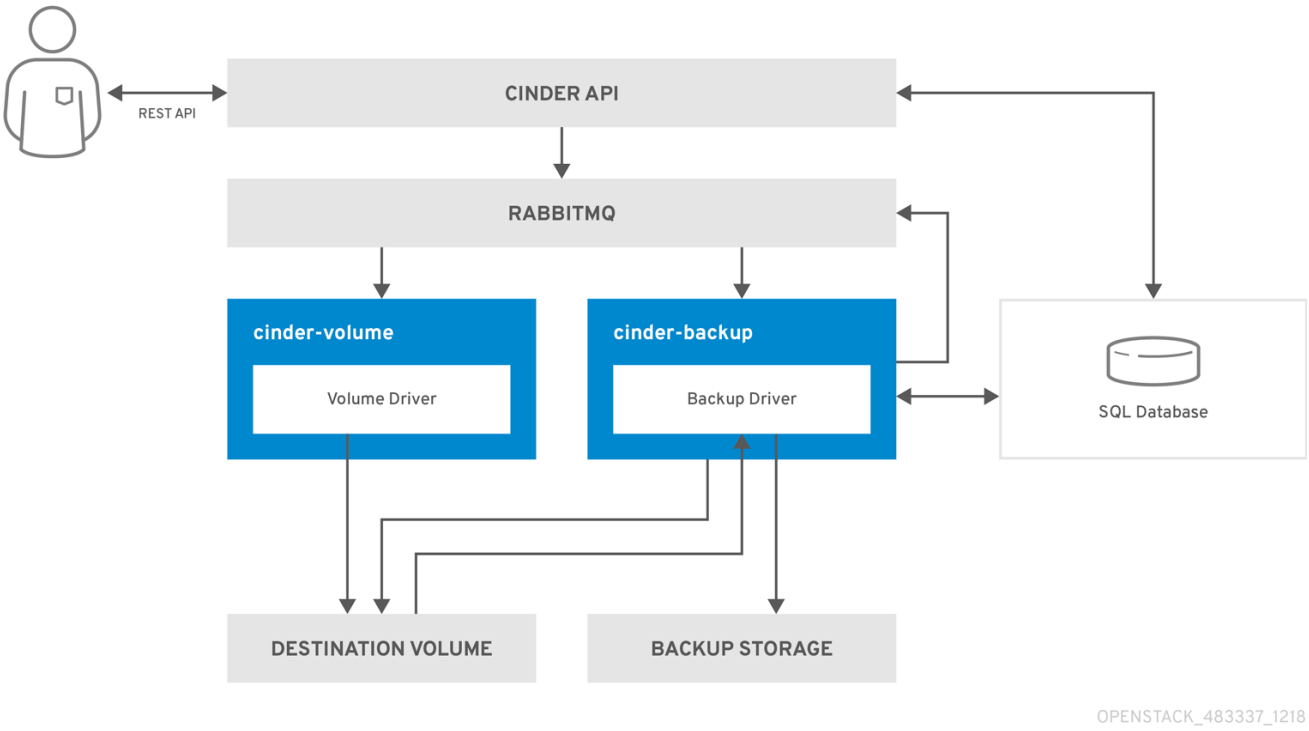
- Find the Block Storage backup service log file on the Controller node where the backup service is running.

This log file is located in the following path: **/var/log/containers/cinder/cinder-backup.log**.

2.2.3. Volume backup workflow

The following diagram and explanation describe the steps that occur when the user requests the cinder API to back up a Block Storage (cinder) volume.

Figure 2.1. Creating a backup of a Block Storage volume



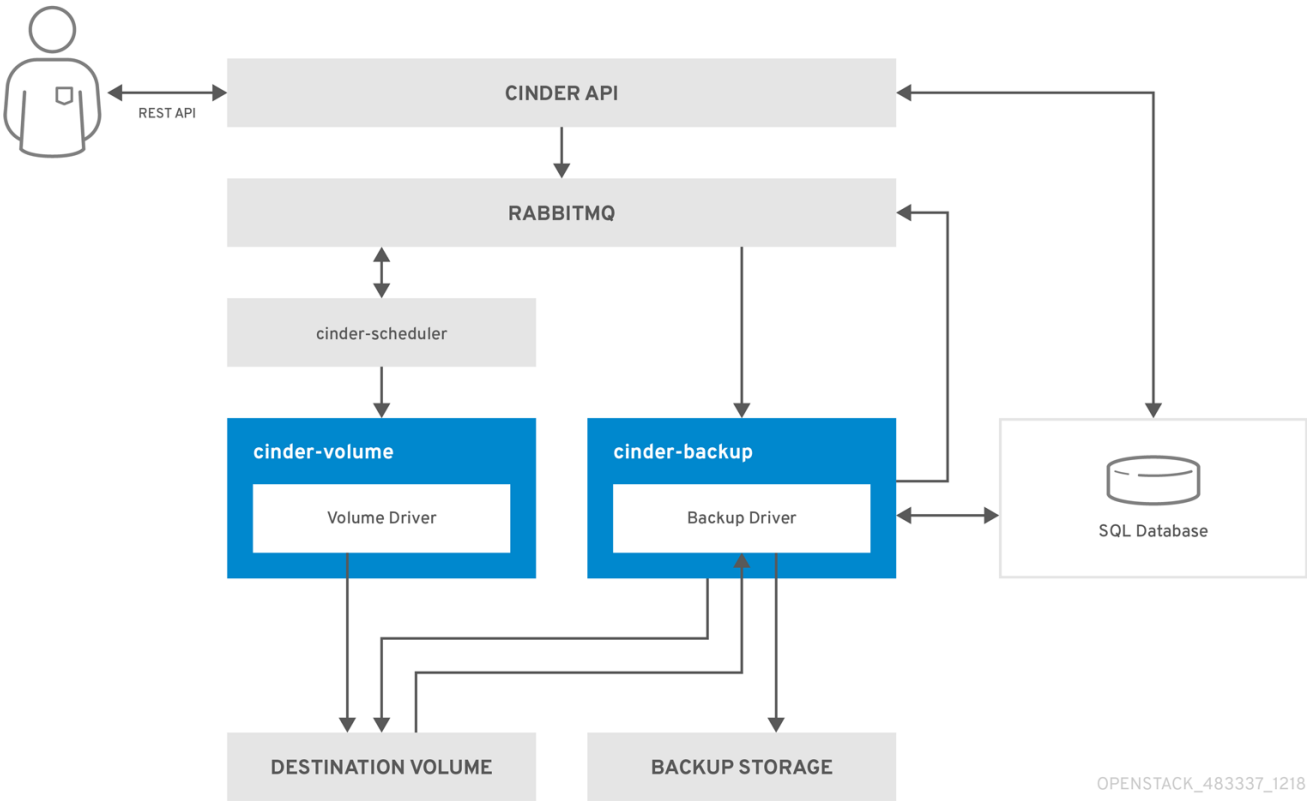
1. The user issues a request to the cinder API, which is a REST API, to back up a Block Storage volume.
2. The cinder API receives the request from HAProxy and validates the request, the user credentials, and other information.
3. The cinder API creates the backup record in the SQL database.
4. The cinder API makes an RPC call to the **cinder-scheduler**.
5. The **cinder-scheduler** makes an asynchronous RPC call to the **cinder-backup** service via AMQP to back up the volume.
6. The cinder API returns the current backup record, with an ID, to the API caller.
7. An RPC create message arrives on one of the backup services.
8. The **cinder-backup** service performs a synchronous RPC call to **get_backup_device**.
9. The **cinder-volume** service ensures that the correct device is returned to the caller. Normally, it is the same volume, but if the volume is in use, the service returns a temporary cloned volume or a temporary snapshot, depending on the configuration.
10. The **cinder-backup** service issues another synchronous RPC to **cinder-volume** to expose the source device.

11. The **cinder-volume** service exports and maps the source device (volume or snapshot) and returns the appropriate connection information.
12. The **cinder-backup** service attaches the source device by using the connection information.
13. The **cinder-backup** service calls the backup back end driver, with the device already attached, which begins the data transfer to the backup repository.
14. The source device is detached from the Backup host.
15. The **cinder-backup** service issues a synchronous RPC to **cinder-volume** to disconnect the source device.
16. The **cinder-volume** service unmaps and removes the export for the device.
17. If a temporary volume or temporary snapshot was created, **cinder-backup** calls **cinder-volume** to remove it.
18. The **cinder-volume** service removes the temporary volume.
19. When the backup is completed, the backup record is updated in the database.

2.2.4. Volume restore workflow

The following diagram and explanation describe the steps that occur when the user requests the cinder API to restore a Block Storage service (cinder) backup.

Figure 2.2. Restoring a Block Storage backup



1. The user issues a request to the cinder API, which is a REST API, to restore a Block Storage backup.

2. The cinder API receives the request from HAProxy and validates the request, the user credentials, and other information.
3. If the request does not contain an existing volume as the destination, the cinder API makes an asynchronous RPC call to create a new volume and polls the status of the volume until it becomes available.
4. The **cinder-scheduler** selects a volume service and makes the RPC call to create the volume.
5. The selected **cinder-volume** service creates the volume.
6. When the cinder API detects that the volume is available, the backup record is updated in the database.
7. The cinder API makes an asynchronous RPC call to the backup service via AMQP to restore the backup.
8. The cinder API returns the current volume ID, backup ID, and volume name to the API caller.
9. An RPC create message arrives on one of the backup services.
10. The **cinder-backup** service performs a synchronous RPC call to **cinder-volume** to expose the volume.
11. The **cinder-volume** service exports and maps the volume returning the appropriate connection information.
12. The **cinder-backup** service attaches the volume by using the connection information.
13. The **cinder-backup** service calls the back end driver with the volume already attached, which begins the data restoration to the volume.
14. The volume is detached from the backup host.
15. The **cinder-backup** service issues a synchronous RPC to **cinder-volume** to disconnect the volume.
16. The **cinder-volume** service unmaps and removes the export for the volume.
17. When the volume is restored, the backup record is updated in the database.

CHAPTER 3. PERFORMING OPERATIONS WITH THE IMAGE SERVICE (GLANCE)

You can create and manage images in the Red Hat OpenStack Services on OpenShift (RHOSO) Image service (glance).

NOTE

To execute **openstack** client commands on the cloud, you must specify the name of the cloud detailed in your **clouds.yaml** file. You can specify the name of the cloud by using one of the following methods:

- Use the **--os-cloud** option with each command:

```
$ openstack flavor list --os-cloud <cloud_name>
```

Use this option if you access more than one cloud.

- Create an environment variable for the cloud name in your **bashrc** file:

```
`export OS_CLOUD=<cloud_name>`
```

Prerequisites

- The administrator has created a project for you, and they have provided you with a **clouds.yaml** file for you to access the cloud.
- You have installed the **python-openstackclient** package.

3.1. CREATING OS IMAGES

To create OS images that you can manage in the Image service (glance), you can use Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance images, or you can manually create RHOSO-compatible images in the QCOW2 format by using RHEL ISO files or Windows ISO files.

3.1.1. Virtual machine image formats

A virtual machine (VM) image is a file that contains a virtual disk with a bootable OS installed. Red Hat OpenStack Services on OpenShift (RHOSO) supports VM images in different formats.

The disk format of a VM image is the format of the underlying disk image. The container format indicates if the VM image is in a file format that also contains metadata about the VM.

When you add an image to the Image service (glance), you can set the disk or container format for your image to any of the values in the following tables by using the **--disk-format** and **--container-format** command options with the **openstack image create**, **glance image-create-via-import**, and **openstack image set** commands. If you are not sure of the container format of your VM image, you can set it to **bare**.

Table 3.1. Disk image formats

Format	Description
aki	Indicates an Amazon kernel image that is stored in the Image service.
ami	Indicates an Amazon machine image that is stored in the Image service.
ari	Indicates an Amazon ramdisk image that is stored in the Image service.
iso	Sector-by-sector copy of the data on a disk, stored in a binary file. Although an ISO file is not normally considered a VM image format, these files contain bootable file systems with an installed operating system, and you use them in the same way as other VM image files.
ploop	A disk format supported and used by Virtuozzo to run OS containers.
qcow2	Supported by QEMU emulator. This format includes QCOW2v3 (sometimes referred to as QCOW3), which requires QEMU 1.1 or higher.
raw	Unstructured disk image format.
vdi	Supported by VirtualBox VM monitor and QEMU emulator.
vhd	Virtual Hard Disk. Used by VM monitors from VMware, VirtualBox, and others.
vhdx	Virtual Hard Disk v2. Disk image format with a larger storage capacity than VHD.
vmdk	Virtual Machine Disk. Disk image format that allows incremental backups of data changes from the time of the last backup.

Table 3.2. Container image formats

Format	Description
aki	Indicates an Amazon kernel image that is stored in the Image service.
ami	Indicates an Amazon machine image that is stored in the Image service.
ari	Indicates an Amazon ramdisk image that is stored in the Image service.
bare	Indicates there is no container or metadata envelope for the image.
docker	Indicates a TAR archive of the file system of a Docker container that is stored in the Image service.
ova	Indicates an Open Virtual Appliance (OVA) TAR archive file that is stored in the Image service. This file is stored in the Open Virtualization Format (OVF) container file.

Format	Description
ovf	OVF container file format. Open standard for packaging and distributing virtual appliances or software to be run on virtual machines.

3.1.2. Creating RHEL KVM images

Use Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance images to create images that you can manage in the Red Hat OpenStack Services on OpenShift (RHOSO) Image service (glance).

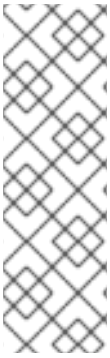
3.1.2.1. Using a RHEL KVM instance image

You can use the following Red Hat Enterprise Linux (RHEL) Kernel-based Virtual Machine (KVM) instance image with Red Hat OpenStack Services on OpenShift (RHOSO):

- Red Hat Enterprise Linux 9 KVM Guest Image

QCOW2 images are configured with **cloud-init** and must have EC2-compatible metadata services for provisioning Secure Shell (SSH) keys to function correctly.

Ready Windows KVM instance images in QCOW2 format are not available.



NOTE

For KVM instance images:

- The **root** account in the image is deactivated, but **sudo** access is granted to a special user named **cloud-user**.
- There is no **root** password set for this image.

The **root** password is locked in **/etc/shadow** by placing **!!** in the second field.

For a RHOSO instance, generate an SSH keypair from the RHOSO dashboard or command line, and use that key combination to perform an SSH public authentication to the instance as root user.

When you launch the instance, this public key is injected to it. You can then authenticate by using the private key that you download when you create the keypair.

3.1.2.2. Creating a RHEL-based root partition image for bare-metal instances

To create a custom root partition image for bare-metal instances, download the base Red Hat Enterprise Linux KVM instance image, and then upload the image to the Image service (glance).

Procedure

1. Download the base Red Hat Enterprise Linux KVM instance image from the Customer Portal.
2. Define **DIB_LOCAL_IMAGE** as the downloaded image:

```
$ export DIB_LOCAL_IMAGE=rhel-<ver>-x86_64-kvm.qcow2
```

- Replace **<ver>** with the RHEL version number of the image.

3. Set your registration information depending on your method of registration:

   - Red Hat Customer Portal:

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_AUTO_ATTACH=true
$ export REG_METHOD=portal
$ export https_proxy='<IP_address:port>' (if applicable)
$ export http_proxy='<IP_address:port>' (if applicable)
```

   - Red Hat Satellite:

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_SAT_URL='<satellite-url>'
$ export REG_ORG='<satellite-org>'
$ export REG_ENV='<satellite-env>'
$ export REG_METHOD=<method>
```

   - Replace values in angle brackets **<>** with the correct values for your Red Hat Customer Portal or Red Hat Satellite registration.

4. Optional: If you have any offline repositories, you can define **DIB_YUM_REPO_CONF** as a local repository configuration:

```
$ export DIB_YUM_REPO_CONF=<file-path>
```

   - Replace **<file-path>** with the path to your local repository configuration file.

5. Use the **diskimage-builder** tool to extract the kernel as **rhel-image.vmlinuz** and the initial RAM disk as **rhel-image.initrd**:

```
$ export DIB_RELEASE=<ver>
$ disk-image-create rhel baremetal \
  -o rhel-image
```

6. Upload the images to the Image service:

```
$ KERNEL_ID=$(openstack image create \
  --file rhel-image.vmlinuz --public \
  --container-format aki --disk-format aki \
  -f value -c id rhel-image.vmlinuz)
$ RAMDISK_ID=$(openstack image create \
  --file rhel-image.initrd --public \
  --container-format ari --disk-format ari \
  -f value -c id rhel-image.initrd)
$ openstack image create \
  --file rhel-image.qcow2   --public \
  --container-format bare \
  --disk-format qcow2 \
```

```
--property kernel_id=$KERNEL_ID \
--property ramdisk_id=$RAMDISK_ID \
rhel-root-partition-bare-metal-image
```

3.1.2.3. Creating a RHEL-based whole-disk user image for bare-metal instances

To create a whole-disk user image for bare-metal instances, download the base Red Hat Enterprise Linux KVM instance image, and then upload the image to the Image service (glance).

Procedure

1. Download the base Red Hat Enterprise Linux KVM instance image from the Customer Portal.
2. Define **DIB_LOCAL_IMAGE** as the downloaded image:

```
$ export DIB_LOCAL_IMAGE=rhel-<ver>-x86_64-kvm.qcow2
```

   - Replace **<ver>** with the RHEL version number of the image.

3. Set your registration information depending on your method of registration:
   - Red Hat Customer Portal:

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_AUTO_ATTACH=true
$ export REG_METHOD=portal
$ export https_proxy='<IP_address:port>' (if applicable)
$ export http_proxy='<IP_address:port>' (if applicable)
```

   - Red Hat Satellite:

```
$ export REG_USER='<username>'
$ export REG_PASSWORD='<password>'
$ export REG_SAT_URL='<satellite-url>'
$ export REG_ORG='<satellite-org>'
$ export REG_ENV='<satellite-env>'
$ export REG_METHOD=<method>
```

   - Replace values in angle brackets **<>** with the correct values for your Red Hat Customer Portal or Red Hat Satellite registration.

4. Optional: If you have any offline repositories, you can define **DIB_YUM_REPO_CONF** as a local repository configuration:

```
$ export DIB_YUM_REPO_CONF=<file-path>
```

   - Replace **<file-path>** with the path to your local repository configuration file.

5. Upload the image to the Image service:

```
$ openstack image create \
  --file rhel-image.qcow2 --public \
  --container-format bare \
```

```
--disk-format qcow2 \
rhel-whole-disk-bare-metal-image
```

3.1.3. Creating instance images with RHEL or Windows ISO files

You can create custom Red Hat Enterprise Linux (RHEL) or Windows images in QCOW2 format from ISO files, and upload these images to the Red Hat OpenStack Services on OpenShift (RHOSO) Image service (glance) for use when creating instances.

3.1.3.1. Prerequisites

- A Linux host machine to create an image. This can be any machine on which you can install and run the Linux packages, except for the undercloud or the overcloud.
- The **advanced-virt** repository is enabled:

```
$ sudo subscription-manager repos --enable=advanced-virt-for-rhel-<ver>-x86_64-rpms
```

- The **virt-manager** application is installed to have all packages necessary to create a guest operating system:

```
$ sudo dnf module install -y virt
```

- The **libguestfs-tools** package is installed to have a set of tools to access and modify virtual machine images:

```
$ sudo dnf install -y libguestfs-tools-c
```

- A RHEL 9 ISO file or a Windows ISO file. For more information about RHEL ISO files, see RHEL 9.0 Binary DVD. If you do not have a Windows ISO file, see the Microsoft Evaluation Center to download an evaluation image.
- A text editor, if you want to change the **kickstart** files (RHEL only).

IMPORTANT

If you install the **libguestfs-tools** package on the undercloud, deactivate **iscsid.socket** to avoid port conflicts with the **tripleo_iscsid** service on the undercloud:

```
$ sudo systemctl disable --now iscsid.socket
```

When you have the prerequisites in place, you can proceed to create a RHEL or Windows image:

- Create a Red Hat Enterprise Linux 9 image
- Create a Windows image

3.1.3.2. Creating a Red Hat Enterprise Linux 9 image

You can create a Red Hat OpenStack Services on OpenShift (RHOSO) image in QCOW2 format by using a Red Hat Enterprise Linux (RHEL) 9 ISO file.

Procedure

1. Log on to your host machine as the **root** user.
2. Start the installation by using **virt-install**:

```
[root@host]# virt-install \
 --virt-type kvm \
 --name <rhel9-cloud-image> \
 --ram <2048> \
 --cdrom </var/lib/libvirt/images/rhel-9.0-x86_64-dvd.iso> \
 --disk <rhel9.qcow2>,format=qcow2,size=<10> \
 --network=bridge:virbr0 \
 --graphics vnc,listen=127.0.0.1 \
 --noautoconsole \
 --os-variant=<rhel9.0>
```

- Replace the values in angle brackets **<>** with the correct values for your RHEL 9 image. This command launches an instance and starts the installation process.

NOTE

If the instance does not launch automatically, run the **virt-viewer** command to view the console:

```
[root@host]# virt-viewer <rhel9-cloud-image>
```

3. Configure the instance:
   a. At the initial Installer boot menu, select **Install Red Hat Enterprise Linux 9**.
   b. Choose the appropriate **Language** and **Keyboard** options.
   c. When prompted about which type of devices your installation uses, select **Auto-detected installation media**.
   d. When prompted about which type of installation destination, select **Local Standard Disks**. For other storage options, select **Automatically configure partitioning**.
   e. In the **Which type of installation would you like?** window, choose the **Basic Server** install, which installs an SSH server.
   f. For network and host name, select **eth0** for network and choose a host name for your device. The default host name is **localhost.localdomain**.
   g. Enter a password in the **Root Password** field and enter the same password again in the **Confirm** field.
4. When the on-screen message confirms that the installation is complete, reboot the instance and log in as the root user.
5. Update the **/etc/sysconfig/network-scripts/ifcfg-eth0** file so that it contains only the following values:

```
TYPE=Ethernet
```

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

6. Reboot the machine.
7. Register the machine with the Content Delivery Network.

```
# sudo subscription-manager register
# sudo subscription-manager attach \
  --pool=<pool-id>
# sudo subscription-manager repos \
  --enable rhel-9-for-x86_64-baseos-rpms \
  --enable rhel-9-for-x86_64-appstream-rpms
```

   - Replace **pool-id** with a valid pool ID. You can see a list of available pool IDs by running the **subscription-manager list --available** command.

8. Update the system:

```
# dnf -y update
```

9. Install the **cloud-init** packages:

```
# dnf install -y cloud-utils-growpart cloud-init
```

10. Edit the **/etc/cloud/cloud.cfg** configuration file and add the following content under **cloud_init_modules**:

```
- resolv-conf
```

    The **resolv-conf** option automatically configures the **resolv.conf** file when an instance boots for the first time. This file contains information related to the instance such as **nameservers**, **domain**, and other options.

11. Add the following line to **/etc/sysconfig/network** to avoid issues when accessing the EC2 metadata service:

```
NOZEROCONF=yes
```

12. To ensure that the console messages appear in the **Log** tab on the dashboard and the **nova console-log** output, add the following boot option to the **/etc/default/grub** file:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8"
```

13. Run the **grub2-mkconfig** command:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

    The output is as follows:

```
Generating grub configuration file ...
```

```
Found linux image: /boot/vmlinuz-3.10.0-229.9.2.el9.x86_64
Found initrd image: /boot/initramfs-3.10.0-229.9.2.el9.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-121.el9.x86_64
Found initrd image: /boot/initramfs-3.10.0-121.el9.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-b82a3044fb384a3f9aeacf883474428b
Found initrd image: /boot/initramfs-0-rescue-b82a3044fb384a3f9aeacf883474428b.img
done
```

14. Deregister the instance so that the resulting image does not contain the subscription details for this instance:

```
# subscription-manager repos --disable=*
# subscription-manager unregister
# dnf clean all
```

15. Power off the instance:

```
# poweroff
```

16. Reset and clean the image by using the **virt-sysprep** command so that it can be used to create instances without issues:

```
[root@host]# virt-sysprep -d <rhel9-cloud-image>
```

17. Reduce the image size by converting any free space in the disk image back to free space in the host:

```
[root@host]# virt-sparsify \
  --compress <rhel9.qcow2> <rhel9-cloud.qcow2>
```

This command creates a new **<rhel9-cloud.qcow2>** file in the location from where the command is run.

NOTE

You must manually resize the partitions of instances based on the image in accordance with the disk space in the flavor that is applied to the instance.

The **<rhel9-cloud.qcow2>** image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSO deployment, see Uploading images to the Image service.

3.1.3.3. Creating a Windows image

You can create a Red Hat OpenStack Services on OpenShift (RHOSO) image in QCOW2 format by using a Windows ISO file.

Procedure

1. Log on to your host machine as the **root** user.
2. Start the installation by using **virt-install**:

```
[root@host]# virt-install \
  --name=<windows-image> \
  --disk size=<size> \
  --cdrom=<file-path-to-windows-iso-file> \
  --os-type=windows \
  --network=bridge:virbr0 \
  --graphics spice \
  --ram=<ram>
```

- Replace the values in angle brackets **<>** withe the correct values for your Windows image.

NOTE

The **--os-type=windows** parameter ensures that the clock is configured correctly for the Windows instance and enables its Hyper-V enlightenment features. You must also set **os_type=windows** in the image metadata before uploading the image to the Image service (glance).

3. The **virt-install** command saves the instance image as **/var/lib/libvirt/images/<windows-image>.qcow2** by default. If you want to keep the instance image elsewhere, change the parameter of the **--disk** option:

```
--disk path=<file-name>,size=<size>
```

- Replace **<file-name>** with the name of the file that stores the instance image, and optionally its path. For example, **path=win8.qcow2,size=8** creates an 8 GB file named **win8.qcow2** in the current working directory.

NOTE

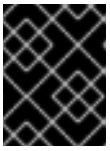
If the instance does not launch automatically, run the **virt-viewer** command to view the console:

```
[root@host]# virt-viewer <windows-image>
```

For more information about how to install Windows, see the Microsoft documentation.

4. To allow the newly-installed Windows system to use the virtualized hardware, you might need to install VirtIO drivers. For more information, see Installing KVM paravirtualized drivers for Windows virtual machines in *Configuring and managing virtualization*.

5. To complete the configuration, download and run Cloudbase-Init on the Windows system. At the end of the installation of Cloudbase-Init, select the **Run Sysprep** and **Shutdown** checkboxes. The **Sysprep** tool makes the instance unique by generating an OS ID, which is used by certain Microsoft services.



IMPORTANT

Red Hat does not provide technical support for Cloudbase-Init. If you encounter an issue, see Contact Cloudbase Solutions.

When the Windows system shuts down, the **<windows-image.qcow2>** image file is ready to be uploaded to the Image service. For more information about uploading this image to your RHOSO deployment, see Uploading images to the Image service .

3.1.4. Creating an image for UEFI Secure Boot

If your Red Hat OpenStack Services on OpenShift (RHOSO) deployment contains UEFI Secure Boot Compute nodes, you can create a Secure Boot image that cloud users can use to launch Secure Boot instances.

Procedure

1. Create a new image for UEFI Secure Boot:

```
$ openstack image create \
--file <base_image_file> \
--container-format <container_format> \
--disk-format <disk_format> \
uefi_secure_boot_image
```

   - Replace **<base_image_file>** with an image file that supports UEFI and the GUID Partition Table (GPT) standard, and includes an EFI system partition.
   - Replace **<container_format>** with one of the following container formats: none, ami, ari, aki, bare, ovf, ova, docker
   - Replace **<disk_format>** with one of the following disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.

2. If the default machine type is not **q35**, then set the machine type to **q35**:

```
$ openstack image set --property hw_machine_type=q35 uefi_secure_boot_image
```

3. Specify that the instance must be scheduled on a UEFI Secure Boot host:

```
$ openstack image set \
 --property hw_firmware_type=uefi \
 --property os_secure_boot=required \
 uefi_secure_boot_image
```

3.1.5. Metadata properties for virtual hardware

The Compute service (nova) has deprecated support for using **libosinfo** data to set default device models. Instead, use the following image metadata properties to configure the optimal virtual hardware for an instance:

- **os_distro**
- **os_version**
- **hw_cdrom_bus**
- **hw_disk_bus**
- **hw_scsi_model**

- **hw_vif_model**
- **hw_video_model**
- **hypervisor_type**

3.2. UPLOADING, IMPORTING, AND MANAGING IMAGES

Manage images and the properties and formats of images that you upload, import, or store in the Red Hat OpenStack Services on OpenShift (RHOSO) Image service (glance).

3.2.1. Uploading images to the Image service

You can upload an image to the OpenStack Image service (glance) by using the **openstack image create** command with the **--property** option.

Procedure

- Use the **openstack image create** command with the **property** option to upload an image. For example:

```
$ openstack image create --name <name> \
   --is-public true --disk-format <qcow2> \
   --container-format <bare> \
   --file </path/to/image> \
   --property <os_version>=<11.10>
```

  - Replace **<name>** with a descriptive name for your image.
  - Replace **<disk-format>** with one of the following disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
  - Replace **<container-format>** with one of the following container formats: none, ami, ari, aki, bare, ovf, ova, docker.
  - Replace **</path/to/image>** with the file path to your image file.
  - Replace **<os_version>** and **<11.10>** with the key-value pair of the property you want to associate to your image. You can use the **--property** option multiple times with different key-value pairs you want to associate to your image.

3.2.2. Image service image import methods

You can import images to the Image service (glance) by using the following methods:

- Use the **web-download** (default) method to import images from a URI.
- Use the **copy-image** method to copy an existing image to other Image service back ends that are in your deployment. Use this import method only if multiple Image service back ends are enabled in your deployment.

The **web-download** method is enabled by default, but the administrator configures other import methods. You can run the **openstack image import info** command to list available import options.

3.2.2.1. Importing an image from a remote URI

You can use the **web-download** image import method to copy an image from a remote URI to the OpenStack Image service (glance).

The Image service **web-download** method uses a two-stage process to perform the import:

1. The **web-download** method creates an image record.
2. The **web-download** method retrieves the image from the specified URI.

The URI is subject to optional **allowlist** and **blocklist** filtering.

If the Inject Image Metadata plugin is enabled in your Red Hat OpenStack Services on OpenShift (RHOSO) deployment, the plugin might inject metadata properties to the image. These metadata properties determine which Compute nodes the image instances are launched on.

Procedure

- Create an image and specify the URI of the image to import:

```
$ glance image-create-via-import \
   --container-format <container_format> \
   --disk-format <disk_format> \
   --name <name> \
   --import-method web-download \
   --uri <uri>
```

  - Replace **<container_format>** with one of the following container formats: none, ami, ari, aki, bare, ovf, ova, docker
  - Replace **<disk_format>** with one of the following disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
  - Replace **<name>** with a descriptive name for your image.
  - Replace **<uri>** with the URI of your image.

Verification

- Check the availability of the image:

```
$ openstack image show <image-id>
```

  - Replace **<image-id>** with the image ID you provided during image creation.

3.2.2.2. Importing an image from a local volume

The **glance-direct** image import method creates an image record, which generates an image ID. When you upload an image to the Image service (glance) from a local volume, the image is stored in a staging area and becomes active when it passes any configured checks.

NOTE

The **glance-direct** method requires a shared staging area when used in a highly available (HA) configuration. If you upload images by using the **glance-direct** import method, the upload can fail in a HA environment if a shared staging area is not present. In a HA active-active environment, API calls are distributed to the Image service controllers. The download API call can be sent to a different controller than the API call to upload the image.

The **glance-direct** image import method uses three different calls to import an image:

- **openstack image create**
- **openstack image stage**
- **openstack image import**

You can use the **glance image-create-via-import** command to perform all three of the **glance-direct** calls in one command.

Procedure

1. Use the **glance image-create-via-import** command to import a local image:

```
$ glance image-create-via-import \
   --container-format <container-format> \
   --disk-format <disk-format> \
   --name <name> \
   --file </path/to/image>
```

   - Replace **<container-format>** with one of the following container formats: none, ami, ari, aki, bare, ovf, ova, docker
   - Replace **<disk-format>** with one of the following disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
   - Replace **<name>** with a descriptive name for your image.
   - Replace **</path/to/image>** with the file path to your image file.
     When the image moves from the staging area to the back-end storage location, the image is listed. However, it might take some time for the image to become active.

Verification

- Check the availability of the image:

```
$ openstack image show <image-id>
```

  - Replace **<image-id>** with the image ID you provided during image creation.

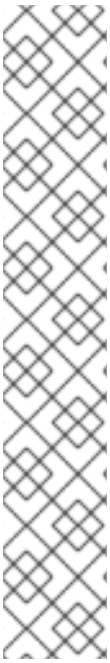
3.2.3. Converting the format of an image

When you import an image to the Image service (glance), you can convert the image to a different format if your administrator has configured the Image Conversion plugin with a preferred format for images in your Red Hat OpenStack Services on OpenShift (RHOSO) deployment.

For example,if you import a QCOW2 image to the Image service and the Image Conversion plugin is configured to the preferred format of RAW, your QCOW2 image is converted to the RAW format when you import it.

You can trigger image conversion only when you import an image. It does not run when you upload an image.

When you import an image to the Image service, the bits of the image are stored in a particular format in a temporary location. When you activate the Image Conversion plugin, the image is converted to the target format and moved to a final storage destination. When the task is finished, the Image service deletes the temporary location. The Image service does not retain the format that you initially uploaded.



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

- Convert the format of an image by using the **web-download** or **glance-direct** import method:
  - Convert the format by using the **glance image-create-via-import** command with **web-download**:

```
$ glance image-create-via-import \
   --disk-format <qcow2> \
   --container-format <bare> \
   --name <name> \
   --visibility public \
   --import-method web-download \
   --uri __<http://server/image.qcow2>__
```

    - Replace **<disk-format>** with one of the following disk formats: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop.
    - Replace **<container-format>** with one of the following container formats: none, ami, ari, aki, bare, ovf, ova, docker
    - Replace **<name>** with a descriptive name for your image.
    - Replace **<http://server/image.qcow2>** with the URI of your image.

- Convert the format by using the **glance-direct** image import method:

```
$ glance image-create-via-import \
   --disk-format <qcow2> \
   --container-format <bare>
   --name <name>
   --visibility public
   --file <local_file.qcow2>
```

  - Replace **<local_file.qcow2>** with your image file.

3.2.3.1. Converting an image to RAW format manually

To launch instances from images that are stored in Red Hat Ceph Storage more efficiently, the image format must be RAW. If your administrator has enabled the Image Conversion plugin for your Red Hat OpenStack Services on OpenShift (RHOSO) deployment, your QCOW2 images are automatically converted to RAW format when you import them to the Image service. Alternatively, you can convert the image manually.

Procedure

1. When you convert an image to RAW format, the RAW image is larger in size than the original QCOW2 image file. Run the following command before the conversion to determine the final RAW image size:

```
$ qemu-img info <image_id>.qcow2
```

   - Replace **<image_id>** with the ID of your QCOW2 image.

2. Convert the image from QCOW2 to RAW format:

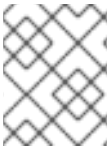
```
$ qemu-img convert -p -f qcow2 -O raw <image_id>.qcow2 <image_id>.raw
```

3.2.3.2. Storing an image in RAW format

With the **GlanceImageImportPlugins** parameter enabled, run the following command to store a previously created image in RAW format:

```
$ glance image-create-via-import \
   --disk-format qcow2 \
   --container-format bare \
   --name <name> \
   --visibility public \
   --import-method web-download \
   --uri <http://server/image.qcow2>
```

- Replace **<name>** with the name of the image; this is the name that will appear in **openstack image list**.
- Replace **<http://server/image.qcow2>** with the location and file name of the QCOW2 image.



NOTE

This command example creates the image record and imports it by using the **web-download** method.

3.2.4. Updating image properties

You can update the properties of an image you have stored in the Image service (glance) by using the **openstack image set** command with the **--property** option.

Procedure

- Use the **openstack image set** command with the **--property** option to update an image. For example:

```
$ openstack image set <image-id> \
    --property <architecture>=<x86_64>
```

  - Replace **<image-id>** with the ID of the image you want to update.
  - Replace **<architecture>** and **<x86_64>** with the key-value pair of the property you want to update for your image. You can use the **--property** option multiple times with different key-value pairs you want to associate to your image.

3.2.5. Hiding or unhiding images

You can hide public images from normal listings presented to cloud users. For example, you can hide obsolete CentOS 7 images and show only the latest version to simplify the user experience. By default, project administrators and project members can delete images. Cloud users can discover and use hidden images.

To create a hidden image, add the **--hidden** argument to the **openstack image create** command.

Procedure

- Hide an image:

```
$ openstack image set <image_id> --hidden 'true'
```

- Unhide an image:

```
$ openstack image set <image_id> --hidden 'false'
```

- List hidden images:

```
$ openstack image list --hidden 'true'
```

3.2.6. Deleting images from the Image service

Use the **openstack image delete** command to delete one or more images that you do not need to store in the Image service (glance). By default, project administrators and project members can delete images.

Procedure

- Delete one or more images:

```
$ openstack image delete <image-id> [<image-id> ...]
```

  - Replace **<image-id>** with the ID of the image you want to delete.

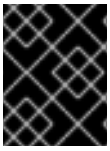
WARNING

The **openstack image delete** command permanently deletes the image and all copies of the image, as well as the image instance and metadata.

3.3. IMPORTING AND COPYING IMAGES TO SINGLE OR MULTIPLE STORES

When you configure the Image service (glance) to use Red Hat Ceph Storage as a back end, you can import image data from a local file system or a web server to multiple Ceph Storage clusters.

You can import an image from a web server to multiple stores at once. If the image is not available on a web server, you can import the image from a local file system to the central store, and then copy it to other stores.



IMPORTANT

Always store an image copy on the central site, even if there are no instances using the image at the central location.

3.3.1. Importing image data to a single store

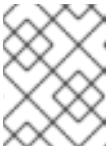
You can use the Image service (glance) to import image data to a single store.

Procedure

1. Import image data to a single store:

```
$ glance image-create-via-import \
--container-format bare \
--name <image-name> \
--import-method web-download \
--uri <uri> \
--store <store>
```

   - Replace **<image-name>** with the name of the image you want to import.
   - Replace **<uri>** with the URI of the image.
   - Replace **<store>** with the name of the store to which you want to copy the image data.



NOTE

If you do not include the options of **--stores**, **--all-stores**, or **--store** in the command, the Image service creates the image in the central store.

2. Verify that the image data was added to specific stores:

```
$ openstack image show <image-id> | grep stores
```

- Replace **<image-id>** with the ID of the original image.
  The output displays a comma-delimited list of stores.

3.3.2. Importing image data to multiple stores

Because the default setting of the **--allow-failure** parameter is **true**, you do not need to include the parameter in the command if it is acceptable for some stores to fail to import the image data.

NOTE

This procedure does not require all stores to successfully import the image data.

Procedure

- Import image data to multiple, specified stores:

```
$ glance image-create-via-import \
--container-format bare \
--name <image-name> \
--import-method web-download \
--uri <uri> \
--stores <store-1>,<store-2>,<store-3>
```

  - Replace **<image-name>** with the name of the image you want to import.
  - Replace **<uri>** with the URI of the image.
  - Replace **<store-1>**, **<store-2>**, and **<store-3>** with the names of the stores to which you want to import the image data.
  - Alternatively, replace **--stores** with **--all-stores true** to upload the image to all the stores.

3.3.3. Importing image data to all stores without failure

This procedure requires all stores to successfully import the image data.

Procedure

- Import image data to multiple, specified stores:

```
$ glance image-create-via-import \
--container-format bare \
--name <image-name> \
```

```
--import-method web-download \
--uri <uri> \
--stores <store-1>,<store-2>,<store-3>
```

- Replace **<image-name>** with the name of the image you want to import.
- Replace **<uri>** with the URI of the image.
- Replace **<store-1>**, **<store-2>**, and **<store-3>** with the names of stores to which you want to copy the image data.
- Alternatively, replace **--stores** with **--all-stores true** to upload the image to all the stores.

NOTE

With the **--allow-failure** parameter set to **false**, the Image service (glance) does not ignore stores that fail to import the image data. You can view the list of failed stores with the image property **os_glance_failed_import**. For more information, see Section 3.3.4, "Checking the progress of the image import operation".

Verification

- Verify that the image data was added to specific stores:

```
$ openstack image show <image-id> | grep stores
```

Replace **<image-id>** with the ID of the original existing image.

The output displays a comma-delimited list of stores.

3.3.4. Checking the progress of the image import operation

The image import workflow sequentially imports image data into stores. The size of the image, the number of stores, and the network speed between the central site and the edge sites impact how long it takes for the image import operation to complete.

You can follow the progress of the image import by looking at two image properties, which appear in notifications sent during the image import operation:

- The **os_glance_importing_to_stores** property lists the stores that have not imported the image data. At the beginning of the import, all requested stores show up in the list. Each time a store successfully imports the image data, the Image service removes the store from the list.
- The **os_glance_failed_import** property lists the stores that fail to import the image data. This list is empty at the beginning of the image import operation.

NOTE

In the following procedure, the environment has three Red Hat Ceph Storage clusters: the **central** store and two stores at the edge, **dcn0** and **dcn1**.

Procedure

1. Verify that the image data was added to specific stores:

```
$ openstack image show <image-id>
```

- Replace **<image-id>** with the ID of the original image.
  The output displays a comma-delimited list of stores similar to the following example snippet:

```
| os_glance_failed_import       |
| os_glance_importing_to_stores | central,dcn0,dcn1
| status                        | importing
```

2. Monitor the status of the image import operation. When you precede a command with **watch**, the command output refreshes every two seconds.

```
$ watch openstack image show <image-id>
```

- Replace **<image-id>** with the ID of the original image.
  The status of the operation changes as the image import operation progresses:

```
| os_glance_failed_import       |
| os_glance_importing_to_stores | dcn0,dcn1
| status                        | importing
```

Output that shows that an image failed to import resembles the following example:

```
| os_glance_failed_import       | dcn0
| os_glance_importing_to_stores | dcn1
| status                        | importing
```

After the operation completes, the status changes to active:

```
| os_glance_failed_import       | dcn0
| os_glance_importing_to_stores |
| status                        | active
```

3.3.5. Managing image import failures

You can manage failures of the image import operation by using the **--allow-failure** parameter:

- If the value of the **--allow-failure** parameter to **true**, the image status becomes **active** after the first store successfully imports the data. This is the default setting. You can view a list of stores that failed to import the image data by using the **os_glance_failed_import** image property.
- If you set the value of the **--allow-failure** parameter to **false**, the image status only becomes **active** after all specified stores successfully import the data. Failure of any store to import the image data results in an image status of **failed**. The image is not imported into any of the specified stores.

3.3.6. Copying an image to specific stores

Use the following procedure to copy image data to one or more specific stores.

Procedure

1. Copy image data to one or more specific stores.
   - Copy image data to a single store:

     ```
     $ openstack image import <image_id> \
     --store <store_id>\
     --import-method copy-image
     ```

     - Replace **<image_id>** with the name of the image you want to copy.
     - Replace **<store_id>** with the name of the stores to which you want to copy the image data.
   - Copy image data to specific stores:

     ```
     $ openstack image import <image-id> \
     --stores <store-1>,<store-2> \
     --import-method copy-image
     ```

     - Replace **<store-1>** and **<store-2>** with the names of the stores to which you want to copy the image data.
2. Confirm that the image data successfully replicated to the specified stores:

   ```
   $ openstack image list --include-stores
   ```

   For information about how to check the status of the image import operation, see Section 3.3.4, "Checking the progress of the image import operation".

3.3.7. Copying an image to multiple stores

You can use the Image service (glance) to copy image data to multiple Red Hat Ceph Storage stores at the edge by using the image import workflow.

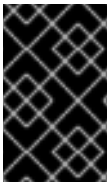
NOTE

The image must be present at the central site before you copy it to any edge sites. Only the image owner or project administrator can copy existing images to newly added stores.

You can copy existing image data either by setting **--all-stores** to **true** or by specifying specific stores to receive the image data.

- The default setting for the **--all-stores** option is **false**. If **--all-stores** is **false**, you must specify which stores receive the image data by using **--stores <store-1>,<store-2>**. If the image data is already present in any of the specified stores, the request fails.
- If you set **all-stores** to **true**, and the image data already exists in some of the stores, then those stores are excluded from the list.

After you specify which stores receive the image data, the Image service copies data from the central site to a staging area. Then, the Image service imports the image data by using the image import workflow.



IMPORTANT

Red Hat recommends that you avoid closely timed image copy requests. Closely timed **copy-image** operations for the same image cause race conditions and unexpected results. Existing image data remains as it is, but copying data to new stores fails.

3.3.8. Copying an image to all stores

Use the following procedure to copy image data to all available stores.

Procedure

1. Copy image data to all available stores:

```
$ openstack image import <image-id> \
--all-stores true \
--import-method copy-image
```

   - Replace **<image-id>** with the ID of the image you want to copy.

2. Confirm that the image data successfully replicated to all available stores:

```
$ openstack image list --include-stores
```

   For information about how to check the status of the image import operation, see Section 3.3.4, "Checking the progress of the image import operation".

3.3.9. Deleting an image from a specific store

Delete an existing image copy on a specific store by using the Red Hat OpenStack Services on OpenShift (RHOSO) Image service (glance).

Procedure

- Delete an image from a specific store:

```
$ openstack image delete --store <store-id> <image-id>
```

- Replace **<store-id>** with the name of the store on which the image copy should be deleted.
- Replace **<image-id>** with the ID of the image you want to delete.

WARNING

The **openstack image delete --store <store-id>** command permanently deletes the image across all the sites. All image copies are deleted, as well as the image instance and metadata.

3.3.10. Listing image locations and location properties

Although an image can be present on multiple sites, there is only a single Universal Unique Identifier (UUID) for a given image. The image metadata contains the locations of each copy. For example, an image present on two edge sites is exposed as a single UUID with three locations: the central site and the two edge sites.

Procedure

1. Show the sites on which a copy of the image exists:

```
$ openstack image show ID | grep "stores"

| stores |  default_backend,dcn1,dcn2
```

   In the example, the image is present on the central site, the **default_backend**, and on the two edge sites **dcn1** and **dcn2**.

2. Alternatively, you can run the **openstack image list** command with the **--include-stores** option to see the sites where the images exist:

```
$ openstack image list --include-stores

| ID                                   | Name    | Stores

| 2bd882e7-1da0-4078-97fe-f1bb81f61b00 | cirros | default_backend,dcn1,dcn2
```

3. List the image location properties to show the details of each location:

```
$ openstack image show ID -c properties

| properties |

(--- cut ---)
locations='[{'url': 'rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-
4078-97fe-f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}, {'url': 'rbd://63df2767-
8ddb-4e06-8186-8c155334f487/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap',
'metadata': {'store': 'dcn1'}}, {'url': 'rbd://1b324138-2ef9-4ef9-bd9e-
aa7e6d6ead78/images/2bd882e7-1da0-4078-97fe-f1bb81f61b00/snap', 'metadata': {'store':
'dcn2'}}]',
(--- cut --)
```

   The image properties show the different Ceph RBD URIs for the location of each image.

   In the example, the central image location URI is:

```
rbd://79b70c32-df46-4741-93c0-8118ae2ae284/images/2bd882e7-1da0-4078-97fe-
f1bb81f61b00/snap', 'metadata': {'store': 'default_backend'}}
```

   The URI is composed of the following data:

   - **79b70c32-df46-4741-93c0-8118ae2ae284** corresponds to the central Ceph FSID. Each Ceph cluster has a unique FSID.
   - The default value for all sites is **images**, which corresponds to the Ceph pool on which the images are stored.

- **2bd882e7-1da0-4078-97fe-f1bb81f61b00** corresponds to the image UUID. The UUID is the same for a given image regardless of its location.
- The metadata shows the glance store to which this location maps. In this example, it maps to the **default_backend**, which is the central hub site.

3.4. IMAGE SERVICE COMMAND OPTIONS AND PROPERTIES

You can use optional arguments, properties, and property keys with the **openstack image create**, **glance image-create-via-import**, and **openstack image set** commands.

3.4.1. Image service command options

You can use the following optional arguments with the **openstack image create**, **glance image-create-via-import**, and **openstack image set** commands.

Table 3.3. Command options

Specific to	Option	Description
All	**--architecture <ARCHITECTURE>**	Operating system architecture as specified in https://docs.openstack.org/glance/latest/user/common-image-properties.html#architecture
All	**--protected [True_False]**	If true, image will not be deletable.
All	**--name <NAME>**	Descriptive name for the image
All	**--instance-uuid <INSTANCE_UUID>**	Metadata that can be used to record which instance this image is associated with. (Informational only, does not create an instance snapshot.)
All	**--min-disk <MIN_DISK>**	Amount of disk space (in GB) required to boot image.
All	**--visibility <VISIBILITY>**	Scope of image accessibility. Valid values: public, private, community, shared
All	**--kernel-id <KERNEL_ID>**	ID of image stored in the Image service (glance) that should be used as the kernel when booting an AMI-style image.
All	**--os-version <OS_VERSION>**	Operating system version as specified by the distributor
All	**--disk-format <DISK_FORMAT>**	Format of the disk. Valid values: none, ami, ari, aki, vhd, vhdx, vmdk, raw, qcow2, vdi, iso, ploop

Specific to	Option	Description
All	**--os-distro <OS_DISTRO>**	Common name of operating system distribution as specified in https://docs.openstack.org/glance/latest/user/common-image-properties.html#os-distro
All	**--owner <OWNER>**	Owner of the image
All	**--ramdisk-id <RAMDISK_ID>**	ID of image stored in the Image service that should be used as the ramdisk when booting an AMI-style image.
All	**--min-ram <MIN_RAM>**	Amount of RAM (in MB) required to boot image.
All	**--container-format <CONTAINER_FORMAT>**	Format of the container. Valid values: none, ami, ari, aki, bare, ovf, ova, docker
All	**--property <key=value>**	Arbitrary property to associate with image. May be used multiple times.
openstack image create	**--tags <TAGS> [<TAGS> ...]**	List of strings related to the image
openstack image create	**--id <ID>**	An identifier for the image
openstack image set	**--remove-property**	Key name of arbitrary property to remove from the image.

3.4.2. Image properties and property keys

You can use the following keys with the **property** option for with the **openstack image create**, **glance image-create-via-import**, and **openstack image set** commands.

Table 3.4. Property keys

Specific to	Key	Description	Supported values
All	**architecture**	The CPU architecture that must be supported by the hypervisor. For example, **x86_64**, **arm**, or **ppc64**. Run **uname -m** to get the architecture of a machine.	• **aarch** - ARM 64-bit • **alpha** - DEC 64-bit RISC • **armv7l** - ARM Cortex-A7 MPCore • **cris** - Ethernet, Token Ring, AXis-Code Reduced Instruction Set • **i686** - Intel sixth-generation x86 (P6 micro architecture)

Specific to	Key	Description	Supported values
			• **ia64** - Itanium • **lm32** - Lattice Micro32 • **m68k** - Motorola 68000 • **microblaze** - Xilinx 32-bit FPGA (Big Endian) • **microblazeel** - Xilinx 32-bit FPGA (Little Endian) • **mips** - MIPS 32-bit RISC (Big Endian) • **mipsel** - MIPS 32-bit RISC (Little Endian) • **mips64** - MIPS 64-bit RISC (Big Endian) • **mips64el** - MIPS 64-bit RISC (Little Endian) • **openrisc** - OpenCores RISC • **parisc** - HP Precision Architecture RISC • **parisc64** - HP Precision Architecture 64-bit RISC • **ppc** - PowerPC 32-bit • **ppc64** - PowerPC 64-bit • **ppcemb** - PowerPC (Embedded 32-bit) • **s390** - IBM Enterprise Systems Architecture/390 • **s390x** - S/390 64-bit • **sh4** - SuperH SH-4 (Little Endian) • **sh4eb** - SuperH SH-4 (Big Endian) • **sparc** - Scalable Processor Architecture, 32-bit • **sparc64** - Scalable Processor Architecture, 64-bit • **unicore32** - Microprocessor Research and Development Center RISC Unicore32 • **x86_64** - 64-bit extension of IA-32

Specific to	Key	Description	Supported values
			• **xtensa** - Tensilica Xtensa configurable microprocessor core • **xtensaeb** - Tensilica Xtensa configurable microprocessor core (Big Endian)
All	**hypervisor_type**	The hypervisor type.	**kvm**, **vmware**
All	**instance_uuid**	For snapshot images, this is the UUID of the server used to create this image.	Valid server UUID
All	**kernel_id**	The ID of an image stored in the Image Service that should be used as the kernel when booting an AMI-style image.	Valid image ID
All	**os_distro**	The common name of the operating system distribution in lowercase.	• **arch** - Arch Linux. Do not use **archlinux** or **org.archlinux**. • **centos** - Community Enterprise Operating System. Do not use **org.centos** or **CentOS**. • **debian** - Debian. Do not use **Debian** or **org.debian**. • **fedora** - Fedora. Do not use **Fedora**, **org.fedora**, or **org.fedoraproject**. • **freebsd** - FreeBSD. Do not use **org.freebsd**, **freeBSD**, or **FreeBSD**. • **gentoo** - Gentoo Linux. Do not use **Gentoo** or **org.gentoo**. • **mandrake** - Mandrakelinux (MandrakeSoft) distribution. Do not use **mandrakelinux** or **MandrakeLinux**. • **mandriva** - Mandriva Linux. Do not use **mandrivalinux**. • **mes** - Mandriva Enterprise Server. Do not use **mandrivaent** or

Specific to	Key	Description	Supported values
			mandrivaES. • **msdos** - Microsoft Disc Operating System. Do not use **ms-dos**. • **netbsd** - NetBSD. Do not use **NetBSD** or **org.netbsd**. • **netware** - Novell NetWare. Do not use **novell** or **NetWare**. • **openbsd** - OpenBSD. Do not use **OpenBSD** or **org.openbsd**. • **opensolaris** - OpenSolaris. Do not use **OpenSolaris** or **org.opensolaris**. • **opensuse** - openSUSE. Do not use **suse**, **SuSE**, or **org.opensuse**. • **rhel** - Red Hat Enterprise Linux. Do not use **redhat**, **RedHat**, or **com.redhat**. • **sled** - SUSE Linux Enterprise Desktop. Do not use **com.suse**. • **ubuntu** - Ubuntu. Do not use **Ubuntu**, **com.ubuntu**, **org.ubuntu**, or **canonical**. • **windows** - Microsoft Windows. Do not use **com.microsoft.server**.
All	**os_version**	The operating system version as specified by the distributor.	Version number (for example, "11.10")
All	**ramdisk_id**	The ID of image stored in the Image Service that should be used as the ramdisk when booting an AMI-style image.	Valid image ID
All	**vm_mode**	The virtual machine mode. This represents the host/guest ABI (application binary interface) used for the virtual machine.	**hvm**-Fully virtualized. This is the mode used by QEMU and KVM.

Specific to	Key	Description	Supported values
libvirt API driver	**hw_cdrom_bus**	Specifies the type of disk controller to attach CD-ROM devices to.	**scsi**, **virtio**, **ide**, or **usb**. If you specify **iscsi**, you must set the **hw_scsi_model** parameter to **virtio-scsi**.
libvirt API driver	**hw_disk_bus**	Specifies the type of disk controller to attach disk devices to.	**scsi**, **virtio**, **ide**, or **usb**. Note that if using **iscsi**, the **hw_scsi_model** needs to be set to **virtio-scsi**.
libvirt API driver	**hw_firmware_type**	Specifies the type of firmware to use to boot the instance.	Set to one of the following valid values: • **bios** • **uefi**
libvirt API driver	**hw_machine_type**	Enables booting an ARM system using the specified machine type. If an ARM image is used and its machine type is not explicitly specified, then Compute uses the virt machine type as the default for ARMv7 and AArch64.	Valid types can be viewed by using the **virsh capabilities** command. The machine types are displayed in the machine tag.
libvirt API driver	**hw_numa_nodes**	Number of NUMA nodes to expose to the instance (does not override flavor definition).	Integer.
libvirt API driver	**hw_numa_cpus.0**	Mapping of vCPUs N-M to NUMA node 0 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	**hw_numa_cpus.1**	Mapping of vCPUs N-M to NUMA node 1 (does not override flavor definition).	Comma-separated list of integers.
libvirt API driver	**hw_numa_mem.0**	Mapping N MB of RAM to NUMA node 0 (does not override flavor definition).	Integer
libvirt API driver	**hw_numa_mem.1**	Mapping N MB of RAM to NUMA node 1 (does not override flavor definition).	Integer

Specific to	Key	Description	Supported values

libvirt API driver	**hw_pci_numa_affinity_policy**	Specifies the NUMA affinity policy for PCI passthrough devices and SR-IOV interfaces.	Set to one of the following valid values:
			• **required**: The Compute service creates an instance that requests a PCI device only when at least one of the NUMA nodes of the instance has affinity with the PCI device. This option provides the best performance.
			• **preferred**: The Compute service attempts a best effort selection of PCI devices based on NUMA affinity. If affinity is not possible, then the Compute service schedules the instance on a NUMA node that has no affinity with the PCI device.
			• **legacy**: (Default) The Compute service creates instances that request a PCI device in one of the following cases:
			  ◦ The PCI device has affinity with at least one of the NUMA nodes.
			  ◦ The PCI devices do not provide information about their NUMA affinities.

libvirt API driver	**hw_qemu_guest_agent**	Guest agent support. If set to **yes**, and if **qemu-ga** is also installed, file systems can be quiesced (frozen) and snapshots created automatically.	**yes** / **no**

Specific to	Key	Description	Supported values
libvirt API driver	**hw_rng_model**	Adds a random number generator (RNG) device to instances launched with this image. The instance flavor enables the RNG device by default. To disable the RNG device, the administrator must set **hw_rng:allowed** to **False** on the flavor. The default entropy source is **/dev/random**. To specify a hardware RNG device, set **rng_dev_path** to **/dev/hwrng** in your Compute environment file.	**virtio**, or other supported device.
libvirt API driver	**hw_scsi_model**	Enables the use of VirtIO SCSI (virtio-scsi) to provide block device access for compute instances; by default, instances use VirtIO Block (virtio-blk). VirtIO SCSI is a para-virtualized SCSI controller device that provides improved scalability and performance, and supports advanced SCSI hardware.	**virtio-scsi**
libvirt API driver	**hw_tpm_model**	Set to the model of TPM device to use. Ignored if **hw:tpm_version** is not configured.	• **tpm-tis**: (Default) TPM Interface Specification. • **tpm-crb**: Command-Response Buffer. Compatible only with TPM version 2.0.
libvirt API driver	**hw_tpm_version**	Set to the version of TPM to use. TPM version **2.0** is the only supported version.	**2.0**

Specific to	Key	Description	Supported values
libvirt API driver	**hw_video_model**	The video device driver for the display device to use in virtual machine instances.	Set to one of the following values to specify the supported driver to use: • **virtio** - (Default) **Recommended** Driver for the virtual machine display device, supported by most architectures. The VirtIO GPU driver is included in RHEL-7 and later, and Linux kernel versions 4.4 and later. If an instance kernel has the VirtIO GPU driver, then the instance can use all the VirtIO GPU features. If an instance kernel does not have the VirtIO GPU driver, the VirtIO GPU device gracefully falls back to VGA compatibility mode, which provides a working display for the instance. • **qxl** - **Deprecated** Driver for Spice or noVNC environments that is no longer maintained. • **cirrus** - Legacy driver, supported only for backward compatibility. Do not use for new instances. • **vga** - Use this driver for IBM Power environments. • **gop** - Not supported for QEMU/KVM environments. • **xen** - Not supported for KVM environments. • **vmvga** - Legacy driver, do not use. • **none** - Use this value to disable emulated graphics or video in virtual GPU (vGPU) instances where the driver is configured separately.
libvirt API driver	**hw_video_ram**	Maximum RAM for the video image. Used only if a **hw_video:ram_max_mb** value has been set in the flavor's **extra_specs** and that value is higher than the value set in **hw_video_ram**.	Integer in MB (for example, *64*)

Specific to	Key	Description	Supported values
libvirt API driver	**hw_watchdog_action**	Enables a virtual hardware watchdog device that carries out the specified action if the server hangs. The watchdog uses the i6300esb device (emulating a PCI Intel 6300ESB). If **hw_watchdog_action** is not specified, the watchdog is disabled.	• disabled-The device is not attached. Allows the user to disable the watchdog for the image, even if it has been enabled using the image's flavor. The default value for this parameter is disabled. • reset-Forcefully reset the guest. • poweroff-Forcefully power off the guest. • pause-Pause the guest. • none-Only enable the watchdog; do nothing if the server hangs.
libvirt API driver	**os_command_line**	The kernel command line to be used by the **libvirt** driver, instead of the default. For Linux Containers(LXC), the value is used as arguments for initialization. This key is valid only for Amazon kernel, ramdisk, or machine images (aki, ari, or ami).	

Specific to	Key	Description	Supported values
libvirt API driver	**os_secure_boot**	Use to create an instance that is protected with UEFI Secure Boot.	Set to one of the following valid values: • **required**: Enables Secure Boot for instances launched with this image. The instance is only launched if the Compute service locates a host that can support Secure Boot. If no host is found, the Compute service returns a "No valid host" error. • **disabled**: Disables Secure Boot for instances launched with this image. Disabled by default. • **optional**: Enables Secure Boot for instances launched with this image only when the Compute service determines that the host can support Secure Boot.
libvirt API driver and VMware API driver	**hw_vif_model**	Specifies the model of virtual network interface device to use.	The valid options depend on the configured hypervisor. • KVM and QEMU: e1000, ne2k_pci, pcnet, rtl8139, and virtio. • VMware: e1000, e1000e, VirtualE1000, VirtualE1000e, VirtualPCNet32, VirtualSriovEthernetCard, and VirtualVmxnet. • Xen: e1000, netfront, ne2k_pci, pcnet, and rtl8139.
VMware API driver	**vmware_adaptertype**	The virtual SCSI or IDE controller used by the hypervisor.	**lsiLogic**, **busLogic**, or **ide**

Specific to	Key	Description	Supported values
VMware API driver	**vmware_ostype**	A VMware GuestID which describes the operating system installed in the image. This value is passed to the hypervisor when creating a virtual machine. If not specified, the key defaults to **otherGuest**.	For more information, see Images with VMware vSphere.
VMware API driver	**vmware_image_version**	Currently unused.	**1**
XenAPI driver	**auto_disk_config**	If true, the root partition on the disk is automatically resized before the instance boots. This value is only taken into account by the Compute service when using a Xen-based hypervisor with the XenAPI driver. The Compute service will only attempt to resize if there is a single partition on the image, and only if the partition is in **ext3** or **ext4** format.	**true** / **false**
libvirt API driver and XenAPI driver	**os_type**	The operating system installed on the image. The XenAPI driver contains logic that takes different actions depending on the value of the **os_type** parameter of the image. For example, for **os_type=windows** images, it creates a FAT32-based swap partition instead of a Linux swap partition, and it limits the injected host name to less than 16 characters.	**linux** or **windows**

CHAPTER 4. PERFORMING OPERATIONS WITH THE OBJECT STORAGE SERVICE (SWIFT)

The Object Storage service (swift) stores its objects, or data, in containers. Containers are similar to directories in a file system although you cannot nest them. You can store any kind of unstructured data in containers. For example, objects can include photos, text files, or images. Stored objects are not compressed.

You can create pseudo-folders in containers to organize data. Pseudo-folders are logical devices for containing objects and creating a nested structure in containers. For example, you might create an *Images* folder in which to store pictures and a *Media* folder in which to store videos.

You can create one or more containers in each project, and one or more objects or pseudo-folders in each container.

NOTE

To execute **openstack** client commands on the cloud, you must specify the name of the cloud detailed in your **clouds.yaml** file. You can specify the name of the cloud by using one of the following methods:

- Use the **--os-cloud** option with each command:

```
$ openstack flavor list --os-cloud <cloud_name>
```

Use this option if you access more than one cloud.

- Create an environment variable for the cloud name in your **bashrc** file:

```
`export OS_CLOUD=<cloud_name>`
```

Prerequisites

- The administrator has created a project for you, and they have provided you with a **clouds.yaml** file for you to access the cloud.
- You have installed the **python-openstackclient** package.

4.1. CREATING PRIVATE AND PUBLIC CONTAINERS

You can create private or public containers to store data in the Object Storage service (swift):

- Private: Limits access to a member of a project.
- Public: Permits access to anyone with the public URL.

New containers use the default storage policy. If your Red Hat OpenStack Services on OpenShift (RHOSO) deployment has multiple storage policies defined, for example, a default policy and another policy that enables erasure coding, you can configure a container to use a non-default storage policy.

Procedure

1. Create a private or public container:

- Create a private container to allow members of a project to list the objects in the container, upload, and download objects. Project members include an Identity service (keystone) token for the project in their requests:

```
$ openstack container create <container> \
    --read-acl "<project_id>":*" \
    --write-acl "<project_id>:*"
```

  - Replace **<container>** with the name of your container.
  - Replace **<project_id>** with the ID of the project.

- Create a public container to allow anyone with the public URL to list objects in the container and download objects from the container:

```
$ openstack container create <container> \
    --read-acl ".r:*,.rlistings"
```

2. Configure the container to use a non-default storage policy:

```
$ openstack container set -H "X-Storage-Policy:<policy>" <container>
```

- Replace **<policy>** with the name or alias of the policy you want to use for the container.

4.2. CREATING PSEUDO-FOLDERS IN CONTAINERS

You can create pseudo-folders to organize data in a container in the OpenStack Object Storage service (swift). You create a pseudo-folder by prefixing the names of the objects with the name of the pseudo-folder and a forward slash character (/).

For example, if you have a container called **container**, and you want to organize objects in a pseudo-folder called **folder**, you add **folder/** at the beginning of the name of the object data file: **folder/object.ext**. You can create nested pseudo-folders in the same way, by including the name of the nested folder and a forward slash at the beginning of the object name, for example, **folder/nested_folder/object.ext**.

The URL of the object will end with **container/folder/object.ext** or **container/folder/nested_folder/object.ext**. You can use the **GET** method with **prefix** and **delimiter** parameters to navigate pseudo-folders.

Procedure

1. Upload an object and create a pseudo-folder in a container:

```
$ openstack object create <container> <pseudo_folder>/<object_filename>
```

- Replace **<container>** with the name of your container.
- Replace **<pseudo_folder>** with the name of the pseudo-folder you want to create.
- Replace **<object_filename>** with the name of your object data file.

2. Upload an object and create a nested pseudo-folder:

```
$ openstack object create <container> <pseudo_folder>/<nested_folder>/<object_filename>
```

- Replace **<nested_folder>** with the name of your nested pseudo-folder.

3. View a list of objects, including nested pseudo-folders, in a pseudo-folder:

```
$ curl -X GET -i -H "X-Auth-Token: $token" \
 $publicurl/v1/<account>/<container>?prefix=<folder>&delimiter=/
```

- Replace **<account>** with your namespace for containers, for example, your Red Hat OpenStack Services on OpenShift (RHOSO) project or tenant.

4.3. DELETING CONTAINERS FROM THE OBJECT STORAGE SERVICE

If you want to delete a container from the Object Storage service (swift), ensure that you delete all objects in the container first. For more information, see Deleting objects from the Object Storage service.

Procedure

- Delete a container:

```
$ openstack container delete <container>
```

  - Replace **<container>** with the name of the container you want to delete.

4.4. UPLOADING OBJECTS TO CONTAINERS

You can upload object data files to a container or pseudo-folder in the Object Storage service (swift). Alternatively, you can create an object as a placeholder in a container or pseudo-folder, and upload the file to the object later.

Procedure

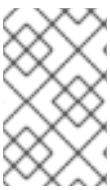
- Upload an object to a container:

```
$ openstack object create <container> <object_filename>
```

  - Replace **<container>** with the name of the container.
  - Replace **<object_filename>** with the name of the object data file.

4.5. COPYING OBJECTS BETWEEN CONTAINERS

You can copy an object from a source container or pseudo-folder to a destination container or pseudo-folder in the Object Storage service (swift).



NOTE

If you do not specify a unique name for the destination object, it keeps the same name as the source object. If you use a name that already exists in the destination, the new object overwrites the contents of the previous object.

Procedure

- Copy an object from one container to a destination container:

```
$ openstack copy --destination </container/object> \
                <container> <object> \
                [<object>] [...]
```

  - Replace **</container/object>** with the container and name of the destination object.
  - Replace **<container>** with the name of the container you want to copy the object from.
  - Replace **<object>** with the name of the object you want to copy. You can specify multiple objects to copy.

4.6. DELETING OBJECTS FROM THE OBJECT STORAGE SERVICE

Delete an object from a container in the Object Storage service (swift).

Procedure

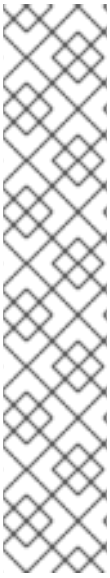
- Delete an object from a container:

```
$ openstack object delete [--all] <container> <object> [...]
```

  - Replace **<container>** with the name of the container you are deleting the object from.
  - Replace **<object>** with the name of the object you are deleting. You can specify multiple objects to delete.
  - Optional: To delete all objects in the container, use the **--all** command option.

CHAPTER 5. PERFORMING OPERATIONS WITH THE SHARED FILE SYSTEMS SERVICE (MANILA)

You can create and manage shares from the available share types in the Shared File Systems service (manila).



NOTE

To execute **openstack** client commands on the cloud, you must specify the name of the cloud detailed in your **clouds.yaml** file. You can specify the name of the cloud by using one of the following methods:

- Use the **--os-cloud** option with each command:

```
$ openstack flavor list --os-cloud <cloud_name>
```

Use this option if you access more than one cloud.

- Create an environment variable for the cloud name in your **bashrc** file:

```
`export OS_CLOUD=<cloud_name>`
```

Prerequisites

- The administrator has created a project for you, and they have provided you with a **clouds.yaml** file for you to access the cloud.
- You have installed the **python-openstackclient** package.

5.1. LISTING SHARE TYPES

You must specify a share type when you create a share, and you can only create shares that match the available share types. The configured share types define the type of service that the Shared File Systems service scheduler uses to make scheduling decisions and that drivers use to control share creation.

Procedure

- List the available share types:

```
$ openstack share type list
```

The command output lists the name and ID of the available share types.

5.2. CREATING NFS, CEPHFS, OR CIFS SHARES

You can create CephFS-NFS, native CephFS, or CIFS shares to read and write data.

When you create a share, you must specify the share protocol and the size of the share in gigabytes. You can also include the **share-type**, **share-network** and **name** command options:

```
$ openstack share create [--share-type <share_type>] \
      [--share-network <share_network>]  \
      [--name <share_name>] <share_protocol> <GB>
```

In the command example, replace the following values:

Value	Description	Required or optional
<share_type>	Applies settings associated with the specified share type	Optional. If you do not specify a share type, the **default** share type is used.
<share_network>	The name of the share network	• Required if the share type has **driver_handles_share_servers** set to **true**. • Unsupported if the share type has **driver_handles_share_servers** set to **false**. • Unsupported for CephFS-NFS and native CephFS. These protocols do not support share types that have **driver_handles_share_servers** set to **true**.
<share_name>	The name of the share	Optional. Shares are not required to have a name, and the name does not need to be unique.
<share_protocol>	The share protocol you want to use	• For CephFS-NFS, replace **<share_protocol>** with **nfs**. • For native CephFS, replace **<share_protocol>** with **cephfs**. • For other storage back ends that support NFS or CIFS protocols, for example, NetApp or Dell EMC storage back ends, replace **<share_protocol>** with **nfs** or **cifs**.
<GB>	The size of the share in gigabytes	Required.

5.2.1. Creating NFS or CIFS shares with DHSS=true

When the share type extra specification, **driver_handles_share_servers** is set to **true**, you can add your own security services to a share network to create and export NFS or CIFS shares. The native CephFS protocol does not support share networks.

To add a security service, you must create a share network first. If you are creating CIFS shares, you must also create a security service resource to represent your Active Directory server. You then associate the security service to the share network.

If you are creating NFS shares, you do not require a security service unless you want to use Kerberos or LDAP authorization on your shares.

Procedure

1. Create a share network:

```
$ openstack share network create --name <network_name> \
   --neutron-net-id <25d1e65c-d961-4f22-9476-1190f55f118f> \
   --neutron-subnet-id <8ba20dce-0ca5-4efd-bf1c-608d6bceffe1>
```

   - Replace **<network_name>** with the share network name that you want to use for your NFS or CIFS shares.
   - Replace the **neutron-net-id** and **neutron-subnet-id** with the correct values for your share network.

2. Create a security service resource to represent your Active Directory server:

```
$ openstack share security service create <active_directory> \
      --dns-ip <192.02.12.10> \
      --domain <domain_name.com> \
      --user <administrator> \
      --password <password> \
      --name <AD_service>
```

   - Replace the values in angle brackets **<>** with the correct details for your security service resource.

3. Associate the security service resource to the share network:

```
$ openstack share network set --new-security-service \
 <AD_service> <network_name>
```

4. Create an NFS or CIFS share:

   - 10 GB NFS example:

```
$ openstack share create --name <nfs_share> --share-type <netapp> \
   --share-network <nfs_network> nfs 10
```

   - 20 GB CIFS example:

```
$ openstack share create --name <cifs_share> --share-type dhss_true \
   --share-network <cifs_network> cifs 20
```

     - Replace the values in angle brackets **<>** with the correct details for your NFS or CIFS share.

5.2.2. Creating NFS, CephFS, or CIFS shares with DHSS=false

When the share type extra specification, **driver_handles_share_servers**, is set to`false`, you cannot use custom security services because security services have been configured directly on the storage system. Because CIFS shares require an Active Directory server along with the storage system to

manage access control, your administrator must pre-create an Active Directory server and associate it with the storage system to use CIFS shares.

When DHSS=false, you can create shares without using the **share-network** command option because the shared storage network is pre-configured.

Procedure

- Create an NFS, native CephFS, or CIFS share when DHSS=false. These examples specify a **name**, but they do not specify the **share-type** or **share-network**. They use the **default** share type and the configured shared storage network:
  - Create a 10 GB NFS share named **share-01**.

```
$ openstack share create --name share-01 nfs 10
```

  - Create a 15 GB native CephFS share named **share-02**:

```
$ openstack share create --name share-02 cephfs 15
```

  - Create a 20 GB CIFS share named **share-03**:

```
$ openstack share create --name share-03 cifs 20
```

5.3. LISTING SHARES AND EXPORTING INFORMATION

To verify that you have successfully created NFS, CephFS, or CIFS shares in the Shared File Systems service (manila), you can list the shares and view their export locations and parameters.

Procedure

1. List the shares:

```
$ openstack share list
```

2. View the export locations of the share:

```
$ openstack share export location list <share>
```

   - Replace **<share>** with either the share name or the share ID.

3. View the parameters for the share:

```
$ openstack share export location show <share_id>
```

   - Replace **<share_id>** with the share ID.

NOTE

You use the export location to mount the share, as described in Section 5.8.2, “Mounting NFS, native CephFS, or CIFS shares” .

5.4. CREATING A SNAPSHOT OF DATA ON A SHARED FILE SYSTEM

A snapshot is a read-only, point-in-time copy of data on a share. You can use a snapshot to recover data lost through accidental data deletion or file system corruption. Snapshots are more space efficient than backups, and they do not impact the performance of the Shared File Systems service (manila).

Prerequisites

- The **snapshot_support** parameter must equal **true** on the parent share. You can run the following command to verify:

```
$ openstack share show | grep snapshot_support
```

Procedure

1. Create a snapshot of a share:

```
$ openstack share snapshot create [--name <snapshot_name>] <share>
```

   - Replace **<share>** with the name or ID of the share for which you want to create a snapshot.
   - Optional: Replace **<snapshot_name>** with the name of the snapshot.

2. Confirm that you created the snapshot:

```
$ openstack share snapshot list --share <share>
```

   Replace **<share>** with the ID of the share from which you created the snapshot.

5.4.1. Creating a share from a snapshot

You can create a share from a snapshot. If the parent share that the snapshot was created from has a share type of **driver_handles_share_servers** set to **true**, the new share is created on the same share network as the parent, and you cannot change this share network for the new share.

Prerequisites

- The **create_share_from_snapshot_support** share attribute is set to **true**.
- The **status** attribute of the snapshot is set to **available**.

Procedure

1. Retrieve the ID of the share snapshot that contains the data that you require for your new share:

```
$ openstack share snapshot list
```

2. A share created from a snapshot can be larger, but not smaller, than the snapshot. Retrieve the size of the snapshot:

```
$ openstack share snapshot show <snapshot_id>
```

   - Replace **<snapshot_id>** with the ID of the snapshot you want to use to create a share.

3. Create a share from a snapshot:

```
$ openstack share create <share_protocol> <size> \
  --snapshot-id <snapshot_id> \
  --name <name>
```

   - Replace **<share_protocol>** with the protocol, such as NFS.
   - Replace **<size>** with the size of the share to be created, in GiB.
   - Replace **<name>** with the name of the new share.

4. List the shares to confirm that the share was created successfully:

```
$ openstack share list
```

5. View the properties of the new share:

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

Verification

After you create a snapshot, confirm that the snapshot is available.

- List the snapshots to confirm that they are available:

```
$ openstack share snapshot list
```

5.4.2. Deleting a snapshot

When you create snapshots of a share, you cannot delete the share until you delete all of the snapshots created from that share.

Procedure

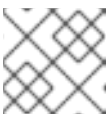
1. Identify the snapshot you want to delete and retrieve its ID:

```
$ openstack share snapshot list
```

2. Delete the snapshot:

```
$ share snapshot delete <snapshot>
```

   - Replace **<snapshot>** with the name or ID of the snapshot you want to delete.



NOTE

Repeat this step for each snapshot you want to delete.

3. After you delete the snapshot, run the following command to confirm that you deleted the snapshot:

```
$ share snapshot list
```

5.5. CONNECTING TO A SHARED NETWORK TO ACCESS SHARES

When the **driver_handles_share_servers** parameter (DHSS) equals **false**, shares are exported to the shared provider network that your administrator has made available. You must connect your client, such as a Compute instance, to the shared provider network to access your shares.

In this example procedure, the shared provider network is called StorageNFS. StorageNFS is configured when the Shared File Systems service (manila) is deployed with a CephFS-NFS back end. Follow similar steps to connect to the available network in your Red Hat OpenStack Services on OpenShift (RHOSO) deployment.

NOTE

The steps in the example procedure use IPv4 addressing, but the steps are identical for IPv6.

Procedure

1. Create a security group for the StorageNFS port that allows packets to egress the port but does not allow ingress packets from unestablished connections:

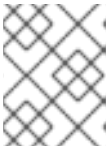
```
$ openstack security group create no-ingress -f yaml
created_at: '2018-09-19T08:19:58Z'
description: no-ingress
id: 66f67c24-cd8b-45e2-b60f-9eaedc79e3c5
name: no-ingress
project_id: 1e021e8b322a40968484e1af538b8b63
revision_number: 2
rules: 'created_at="2018-09-19T08:19:58Z", direction="egress", ethertype="IPv4",
 id="6c7f643f-3715-4df5-9fef-0850fb6eaaf2", updated_at="2018-09-19T08:19:58Z"
 created_at="2018-09-19T08:19:58Z", direction="egress", ethertype="IPv6",
 id="a8ca1ac2-fbe5-40e9-ab67-3e55b7a8632a", updated_at="2018-09-19T08:19:58Z"
```

2. Create a port on the StorageNFS network with security enforced by the **no-ingress** security group.

```
$ openstack port create nfs-port0 \
  --network StorageNFS \
  --security-group no-ingress -f yaml

admin_state_up: UP
allowed_address_pairs: ''
binding_host_id: null
binding_profile: null
binding_vif_details: null
binding_vif_type: null
binding_vnic_type: normal
created_at: '2018-09-19T08:03:02Z'
data_plane_status: null
description: ''
device_id: ''
device_owner: ''
dns_assignment: null
dns_name: null
extra_dhcp_opts: ''
```

```
fixed_ips: ip_address='198.51.100.160', subnet_id='7bc188ae-aab3-425b-a894-
863e4b664192'
id: 7a91cbbc-8821-4d20-a24c-99c07178e5f7
ip_address: null
mac_address: fa:16:3e:be:41:6f
name: nfs-port0
network_id: cb2cbc5f-ea92-4c2d-beb8-d9b10e10efae
option_name: null
option_value: null
port_security_enabled: true
project_id: 1e021e8b322a40968484e1af538b8b63
qos_policy_id: null
revision_number: 6
security_group_ids: 66f67c24-cd8b-45e2-b60f-9eaedc79e3c5
status: DOWN
subnet_id: null
tags: ''
trunk_details: null
updated_at: '2018-09-19T08:03:03Z'
```



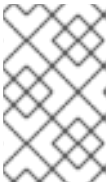
NOTE

In this example, the StorageNFS subnet on the StorageNFS network assigned IP address 198.51.100.160 to **nfs-port0**.

3. Add **nfs-port0** to a Compute instance.

```
$ openstack server add port instance0 nfs-port0
$ openstack server list -f yaml
- Flavor: m1.micro
  ID: 0b878c11-e791-434b-ab63-274ecfc957e8
  Image: manila-test
  Name: demo-instance0
  Networks: demo-network=198.51.100.4, 10.0.0.53; StorageNFS=198.51.100.160
  Status: ACTIVE
```

In addition to its private and floating addresses, the Compute instance is assigned a port with the IP address 198.51.100.160 on the StorageNFS network. You can use this IP address to mount NFS shares when access is granted to that address for the shares.



NOTE

You might need to adjust the networking configuration on the Compute instance, and then restart the services for the Compute instance to activate an interface with this address.

5.6. CONFIGURING AN IPV6 INTERFACE BETWEEN THE NETWORK AND AN INSTANCE

When the shared network to which shares are exported uses IPv6 addressing, you might experience an issue with DHCPv6 on the secondary interface. If this issue occurs, configure an IPv6 interface manually on the instance.

Prerequisites

Prerequisites

- Connection to a shared network to access shares

Procedure

1. Log in to the instance.
2. Configure the IPv6 interface address:

```
$ sudo ip address add fd00:fd00:fd00:7000::c/64 dev eth1
```

3. Activate the interface:

```
$ sudo ip link set dev eth1 up
```

4. Ping the IPv6 address in the export location of the share to test interface connectivity:

```
$ ping -6  fd00:fd00:fd00:7000::21
```

5. Alternatively, verify that you can reach the NFS server through Telnet:

```
$ sudo dnf install -y telnet
$ telnet fd00:fd00:fd00:7000::21 2049
```

5.7. GRANTING SHARE ACCESS FOR END-USER CLIENTS

Before you mount a share on a client, such as a Compute instance, you grant end-user clients access to the share so that users can read data from and write data to the share.

The type of access depends on the protocol of the share:

- For CIFS shares, use the CIFS user or group name.
- For NFS shares, use the IP address of the Compute instance where you plan to mount the share.
- For native CephFS shares, use Ceph client usernames for **cephx** authentication.

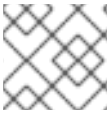
You can grant access to the share by using a command similar to the following command:

```
$ openstack share access create <share> <access_type> \
  --access-level <access_level>  <client_identifier>
```

- Replace **<share>** with the share name or ID of the share you created.
- Replace **<access_type>** with the type of access you want to grant to the share, for example, **user** for CIFS, **ip** for NFS, or **cephfx** for native CephFS.
- Optional: Replace **<access_level>** with **ro** for read-only access. The default value is **rw** for read-write access.
- Replace **client_identifier** with the IP address of the instance for NFS, user or group name for CIFS, or Ceph client username for native CephFS. For CIFS and native CephFS, you can use the same **client_identifier** across multiple clients.

5.7.1. Granting access to an NFS share

You can provide access to NFS shares by using the IP address of the client Compute instance where you plan to mount the share.



NOTE

You can use the following procedure with IPv4 or IPv6 addresses.

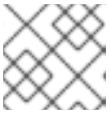
Procedure

- Retrieve the IP address of the client Compute instance where you plan to mount the share. Make sure that you select the IP address that corresponds to the network that can reach the shares. In this example, it is the IP address of the StorageNFS network:

```
$ openstack server list -f yaml
- Flavor: m1.micro
  ID: 0b878c11-e791-434b-ab63-274ecfc957e8
  Image: manila-test
  Name: demo-instance0
  Networks: demo-network=198.51.100.4, 10.0.0.53;
  StorageNFS=198.51.100.160
  Status: ACTIVE

$ openstack share access create <share> ip 198.51.100.160
```

- Replace **<share>** with the name or ID of the share you are granting access to.



NOTE

Access to the share has its own ID, **id**.

```
+-----------------+--------------------------------------+
| Property        | Value                                |
+-----------------+--------------------------------------+
| access_key      | None
| share_id        | db3bedd8-bc82-4100-a65d-53ec51b5cba3
| created_at      | 2018-09-17T21:57:42.000000
| updated_at      | None
| access_type     | ip
| access_to       | 198.51.100.160
| access_level    | rw
| state           | queued_to_apply
| id              | 875c6251-c17e-4c45-8516-fe0928004fff
+-----------------+--------------------------------------+
```

Verification

- Verify that the access configuration was successful:

```
$ openstack share access list <share>

+--------------+------------+--------------+--------------+--------+ ...
```

```
| id          | access_type | access_to    | access_level | state  | ...
+-------------+-------------+--------------+--------------+--------+
| 875c6251-... | ip          | 198.51.100.160 | rw         | active | ...
+-------------+-------------+--------------+--------------+---------+ ...
```

5.7.2. Granting access to a native CephFS share

You can provide access to native CephFS shares by using Ceph client usernames for cephx authentication. The Shared File Systems service (manila) prevents the use of pre-existing Ceph users so you must create unique Ceph client usernames.

To mount a share, you need a Ceph client username and an access key. You can retrieve access keys by using the Shared File Systems service API. By default, access keys are visible to all users in a project namespace. You can provide the same user with access to different shares in the project namespace. Users can then access the shares by using the CephFS kernel client on the client machine.

IMPORTANT

Use the native CephFS driver with trusted clients only.

Procedure

1. Grant users access to a native CephFS share:

```
$ openstack share access create <share> cephx <user>
```

   - Replace **<share>** with either the share name or share ID.
   - Replace **<user>** with the Ceph client username.

2. Collect the access key for the user:

```
$ openstack share access list <share>
```

5.7.3. Granting access to a CIFS share

You can grant access to CIFS shares by using the usernames in the Active Directory service. The Shared File Systems service (manila) does not create new users on the Active Directory server. It only validates usernames through the security service, and access rules with invalid usernames result in an **error** status.

If the value of the **driver_handles_share_servers** (DHSS) parameter is set to **true**, then you can configure the Active Directory service by adding a security service. If the DHSS parameter is set to **false**, then your administrator has already configured the Active Directory service and associated it with the storage network.

To mount a share, you must specify the user's Active Directory username and password. You cannot obtain this password through the Shared File Systems service.

Procedure

- Grant users access to a CIFS share:

```
$ openstack share access create <share> user <user>
```

-

- Replace **<share>** with either the share name or the share ID.
- Replace **<user>** with the username of the Active Directory user.

5.7.4. Revoking access to a share

The owner of a share can revoke access to the share. Complete the following steps to revoke access that was previously granted to a share.

Procedure

1. View the access list for the share to retrieve the access ID:

```
$ openstack share access list <share_01>
```

   - Replace **<share_01>** with either the share name or share ID.

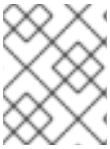
2. Revoke access to the share:

```
$ openstack share access delete <share_01> <875c6251-c17e-4c45-8516-fe0928004fff>
```

   - Replace **<875c6251-c17e-4c45-8516-fe0928004fff>** with the access ID of the share.

3. View the access list for the share again to verify the share has been deleted:

```
$ openstack share access list <share_01>
```



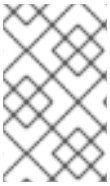
NOTE

If you have a client with read-write access to the share, you must revoke their access to the share, and then add a read-only rule if you want the client to have read-only access.

5.8. MOUNTING SHARES ON COMPUTE INSTANCES

When you grant share access to clients, then the clients can mount and use the shares. Any type of client can access shares as long as there is network connectivity to the client.

The steps used to mount an NFS share on a virtual Compute instance are similar to the steps to mount an NFS share on a bare-metal Compute instance. For more information about how to mount shares on OpenShift containers, see Product Documentation for Red Hat OpenShift Container Platform .



NOTE

Client packages for the different protocols must be installed on the Compute instance that mounts the shares. For example, for the Shared File Systems service with CephFS through NFS, the NFS client packages must support NFS 4.1.

5.8.1. Listing share export locations

Retrieve the export locations of shares so that you can mount a share.

Procedure

- Retrieve the export locations of a share:

```
$ openstack share export location list <share_01>
```

  - Replace **<share_01>** with either the share name or share ID.
    When multiple export locations exist, choose one for which the value of the **preferred** metadata field equals **True**. If no preferred locations exist, you can use any export location.

5.8.2. Mounting NFS, native CephFS, or CIFS shares

When you create NFS, native CephFS, or CIFS shares and grant share access to end-user clients, you can then mount the shares on the client to enable access to data, as long as there is network connectivity.

Prerequisites

- To mount NFS shares, the **nfs-utils** package must be installed on the client machine.
- To mount native CephFS shares, the **ceph-common** package must be installed on the client machine. Users access native CephFS shares by using the CephFS kernel client on the client machine.
- To mount CIFS shares, the **cifs-utils** package must be installed on the client machine.

Procedure

1. Log in to the instance:

```
$ openstack server ssh demo-instance0 --login user
```

2. Mount an NFS share. Refer to the following example for sample syntax:

```
$ mount -t nfs \
  -v <198.51.100.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01>  \
  /mnt
```

   - Replace **<198.51.100.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01>** with the export location of the share.
   - Retrieve the export location as described in Section 5.8.1, “Listing share export locations”.

3. Mount a native CephFS share. Refer to the following example for sample syntax:

```
$  mount -t ceph \
  <192.0.2.125:6789,192.0.2.126:6789,192.0.2.127:6789:/volumes/_nogroup/4c55ad20-9c55-
4a5e-9233-8ac64566b98c> \
  -o name=<user>,secret='<AQA8+ANW/<4ZWNRAAOtWJMFPEihBA1unFImJczA==>'
```

   - Replace **<192.0.2.125:6789,192.0.2.126:6789,192.0.2.127:6789:/volumes/_nogroup/4c55ad20-9c55-4a5e-9233-8ac64566b98c>** with the export location of the share.
   - Retrieve the export location as described in Section 5.8.1, “Listing share export locations”.

- Replace **<user>** with the cephx user who has access to the share.
- Replace the **secret** value with the access key that you collected in Section 5.7.2, "Granting access to a native CephFS share".

4. Mount a CIFS share. Refer to the following example for sample syntax:

```
$  mount -t cifs \
 -o user=<user>,pass=<password> \
 <\\192.0.2.128/share_11265e8a_200c_4e0a_a40f_b7a1117001ed>
```

- Replace **<user>** with the Active Directory user who has access to the share.
- Replace **<password>** with the user's Active Directory password.
- Replace **<\\192.0.2.128/share_11265e8a_200c_4e0a_a40f_b7a1117001ed>** with the export location of the share.
- Retrieve the export location as described in Section 5.8.1, "Listing share export locations".

Verification

- Verify that the mount command succeeded:

```
$ df -k
```

5.9. DELETING SHARES

The Shared File Systems service (manila) provides no protection to prevent you from deleting your data. The service does not check whether clients are connected or workloads are running. When you delete a share, you cannot retrieve it.

WARNING

Back up your data before you delete a share.

Prerequisites

- If you created snapshots from a share, you must delete all of the snapshots and replicas before you can delete the share. For more information, see Deleting a snapshot.

Procedure

- Delete a share:

```
$ openstack share delete <share>
```

  - Replace **<share>** with either the share name or the share ID.

5.10. LISTING RESOURCE LIMITS OF THE SHARED FILE SYSTEMS SERVICE

You can list the current resource limits for the Shared File Systems service (manila) in a project to plan workloads and prepare for any operations based on resource consumption.

Procedure

- List the resource limits and current resource consumption for the project:

```
$ openstack share limits show --absolute
```

5.11. TROUBLESHOOTING OPERATION FAILURES

In the event of an error when you create or mount shares, you can run queries from the command line for more information about the error.

5.11.1. Viewing error messages for shares

You can use the command line to retrieve user support messages if a share shows an error status.

Procedure

1. When you create a share, run the following command to view the status of the share:

```
$ openstack share list
```

2. If the status of your share shows an error, run the **share message list** command. You can use the **--resource-id** option to filter to the specific share you want to find out about:

```
$ openstack share message list [--resource-id]
```

3. Check the **User Message** column in the **share message list** command output for a summary of the error.

4. To view more details about the error, run the **message show** command, followed by the message ID from the **message list** command output:

```
$ openstack share message show <id>
```

   - Replace **<id>** with the message ID from the **message list** command output.

5.11.2. Debugging share mounting failures

You can use these verification steps to identify the root cause of an error when you mount shares.

Procedure

1. Verify the access control list of the share to ensure that the rule that corresponds to your client is correct and has been successfully applied:

```
$ openstack share access list <share_01>
```

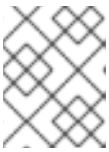
- Replace **<share_01>** with either the share name or share ID.
  In a successful rule, the **state** attribute equals **active**.

2. If the share type parameter is configured to **driver_handles_share_servers=false**, copy the hostname or IP address from the export location and ping it to confirm connectivity to the NAS server:
   Example:

```
$ ping -c 1 198.51.100.13
PING 198.51.100.13 (198.51.100.13) 56(84) bytes of data.
64 bytes from 198.51.100.13: icmp_seq=1 ttl=64 time=0.048 ms--- 198.51.100.13 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.851/7.851/7.851/0.000 ms
```

3. If you are using the NFS protocol, you can verify that the NFS server is ready to respond to NFS RPC calls on the correct port:

```
$ rpcinfo -T tcp -a 198.51.100.13.8.1 100003 4
program 100003 version 4 ready and waiting
```



NOTE

The IP address is written in universal address format (uaddr), which adds two extra octets (8.1) to represent the NFS service port, 2049.

If these verification steps fail, there might be a network connectivity issue or an issue with the back-end storage for the Shared File Systems service (manila). Collect the log files and contact Red Hat Support.