Red Hat OpenStack Platform 16.2

Storage Guide

Understanding, using, and managing persistent storage in OpenStack

Last Updated: 2021-10-19

Red Hat OpenStack Platform 16.2 Storage Guide

Understanding, using, and managing persistent storage in OpenStack

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide details the different procedures for using and managing persistent storage in a Red Hat OpenStack Platform environment. It also includes procedures for configuring and managing the respective OpenStack service of each persistent storage type.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

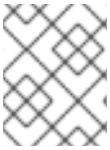
CHAPTER 1. INTRODUCTION TO PERSISTENT STORAGE IN RED HAT OPENSTACK PLATFORM (RHOSP)

Within Red Hat OpenStack Platform, storage is provided by three main services:

- Block Storage (**openstack-cinder**)
- Object Storage (**openstack-swift**)
- Shared File System Storage (**openstack-manila**)

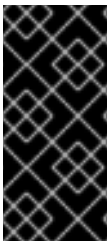
These services provide different types of persistent storage, each with its own set of advantages in different use cases. This guide discusses the suitability of each for general enterprise storage requirements.

You can manage cloud storage by using either the RHOSP dashboard or the command-line clients. You can perform most procedures by using either method. However, you can complete some of the more advanced procedures only on the command line. This guide provides procedures for the dashboard where possible.



NOTE

For the complete suite of documentation for Red Hat OpenStack Platform, see Red Hat OpenStack Platform Documentation.



IMPORTANT

This guide documents the use of **crudini** to apply some custom service settings. As such, you need to install the **crudini** package first:

```
# dnf install crudini -y
```

RHOSP recognizes two types of storage: *ephemeral* and *persistent*. Ephemeral storage is storage that is associated only to a specific Compute instance. Once that instance is terminated, so is its ephemeral storage. This type of storage is useful for basic runtime requirements, such as storing the instance's operating system.

Persistent storage, is designed to survive (persist) independent of any running instance. This storage is used for any data that needs to be reused, either by different instances or beyond the life of a specific instance. RHOSP uses the following types of persistent storage:

Volumes

The OpenStack Block Storage service (**openstack-cinder**) allows users to access block storage devices through *volumes*. Users can attach volumes to instances in order to augment their ephemeral storage with general-purpose persistent storage. Volumes can be detached and re-attached to instances at will, and can only be accessed through the instance they are attached to.
Volumes also provide inherent redundancy and disaster recovery through backups and snapshots. In addition, you can also encrypt volumes for added security.

NOTE

Instances can also be configured to use absolutely no ephemeral storage. In such cases, the Block Storage service can write images to a volume; in turn, the volume can be used as a bootable root volume for an instance.

Containers

The OpenStack Object Storage service (openstack-swift) provides a fully-distributed storage solution used to store any kind of static data or binary object, such as media files, large datasets, and disk images. The Object Storage service organizes these objects by using containers.
Although the content of a volume can be accessed only through instances, the objects inside a container can be accessed through the Object Storage REST API. As such, the Object Storage service can be used as a repository by nearly every service within the cloud.

Shares

The Shared File Systems service (**openstack-manila**) provides the means to easily provision remote, shareable file systems, or *shares*. Shares allow projects within the cloud to openly share storage, and can be consumed by multiple instances simultaneously.

Each storage type is designed to address specific storage requirements. Containers are designed for wide access, and as such feature the highest throughput, access, and fault tolerance among all storage types. Container usage is geared more towards services.

On the other hand, volumes are used primarily for instance consumption. They do not enjoy the same level of access and performance as containers, but they do have a larger feature set and have more native security features than containers. Shares are similar to volumes in this regard, except that they can be consumed by multiple instances.

The following sections discuss each storage type's architecture and feature set in detail, within the context of specific storage criteria.

1.1. SCALABILITY AND BACK END STORAGE

In general, a clustered storage solution provides greater back end scalability. For example, when you use Red Hat Ceph as a Block Storage (cinder) back end, you can scale storage capacity and redundancy by adding more Ceph OSD (Object Storage Daemon) nodes. Both Block Storage and Object Storage (swift) services support Red Hat Ceph Storage as a back end.

The Block Storage service can use multiple storage solutions as discrete back ends. At the back end level, you can scale capacity by adding more back ends and restarting the service. The Block Storage service also features a large list of supported back end solutions, some of which feature additional scalability features.

By default, the Object Storage service uses the file system on configured storage nodes, and can use as much space as is available. The Object Storage service supports the XFS and ext4 file systems, and both can be scaled up to consume as much available underlying block storage. You can also scale capacity by adding more storage devices to the storage node.

The Shared File Systems (manila) service provisions shares backed by storage from a separate storage pool. This pool, which is typically managed by a third-party back end service, provides the share with storage at the file system level. The Shared File Systems service can use both NetApp and CephFS, which you can configure to use a storage pool of pre-created volumes which provisioned shares can use for storage. In either deployment, scaling involves adding more volumes to the pool.

1.2. STORAGE ACCESSIBILITY AND ADMINISTRATION

Volumes are consumed only through instances, and can only be attached to and mounted within one instance at a time. Users can create snapshots of volumes, which can be used for cloning or restoring a volume to a previous state (see Section 1.4, "Storage redundancy and disaster recovery"). The Block Storage service also allows you to create *volume types*, which aggregate volume settings (for example, size and back end) that can be easily invoked by users when creating new volumes. These types can be further associated with *Quality-of-Service* specifications, which allow you to create different storage tiers for users.

Like volumes, shares are consumed through instances. However, shares can be directly mounted within an instance, and do not need to be attached through the dashboard or CLI. Shares can also be mounted by multiple instances simultaneously. The Shared File Systems service also supports share snapshots and cloning; you can also create *share types* to aggregate settings (similar to volume types).

Objects in a container are accessible via API, and can be made accessible to instances and services within the cloud. This makes them ideal as object repositories for services; for example, the Image service (**openstack-glance**) can store its images in containers managed by the Object Storage service.

1.3. STORAGE SECURITY

The Block Storage service (cinder) provides basic data security through volume encryption. With this, you can configure a volume type to be encrypted through a static key; the key is then used to encrypt all volumes that are created from the configured volume type. For more information, see Section 2.7, "Block Storage service (cinder) volume encryption".

Object and container security is configured at the service and node level. The Object Storage service (swift) provides no native encryption for containers and objects. Rather, the Object Storage service prioritizes accessibility within the cloud, and as such relies solely on the cloud network security to protect object data.

The Shared File Systems service (manila) can secure shares through access restriction, whether by instance IP, user or group, or TLS certificate. In addition, some Shared File Systems service deployments can feature separate share servers to manage the relationship between share networks and shares; some share servers support, or even require, additional network security. For example, a CIFS share server requires the deployment of an LDAP, Active Directory, or Kerberos authentication service.

For more information about how to secure the Image service (glance), such as image signing and verification and metadata definition (metadef) API restrictions, see Image service in the *Creating and Managing Images* guide.

1.4. STORAGE REDUNDANCY AND DISASTER RECOVERY

The Block Storage service features volume backup and restoration, providing basic disaster recovery for user storage. Backups allow you to protect volume contents. On top of this, the service also supports snapshots; aside from cloning, snapshots are also useful in restoring a volume to a previous state.

In a multi-backend environment, you can also migrate volumes between back ends. This is useful if you need to take a back end offline for maintenance. Backups are typically stored in a storage back end separate from their source volumes to help protect the data. This is not possible, however, with snapshots, as snapshots are dependent on their source volumes.

The Block Storage service also supports the creation of *consistency groups*, which allow you to group volumes together for simultaneous snapshot creation. This, in turn, allows for a greater level of data consistency across multiple volumes. See Section 2.9, "Block Storage service (cinder) consistency

groups" for more details.

The Object Storage service provides no built-in backup features. As such, all backups must be performed at the file system or node level. The service, however, features more robust redundancy and fault tolerance; even the most basic deployment of the Object Storage service replicates objects multiple times. You can use failover features like **dm-multipath** to enhance redundancy.

The Shared File Systems service provides no built-in backup features for shares, but it does allow you to create snapshots for cloning and restoration.

CHAPTER 2. CONFIGURING THE BLOCK STORAGE SERVICE (CINDER)

The Block Storage service (cinder) manages the administration, security, scheduling, and overall management of all volumes. Volumes are used as the primary form of persistent storage for Compute instances.

For more information about volume backups, refer to the *Block Storage Backup Guide* .

IMPORTANT

You must install host bus adapters (HBAs) on all Controller nodes and Compute nodes in any deployment that uses the Block Storage service (cinder) and a Fibre Channel (FC) back end.

2.1. BLOCK STORAGE SERVICE BACK ENDS

Red Hat OpenStack Platform is deployed using the OpenStack Platform director. Doing so helps ensure the proper configuration of each service, including the Block Storage service (and, by extension, its back end). Director also has several integrated back end configurations.

Red Hat OpenStack Platform supports Red Hat Ceph and NFS as Block Storage (cinder) back ends. By default, the Block Storage service uses an LVM back end as a repository for volumes. While this back end is suitable for test environments, LVM is not supported in production environments.

For instructions on how to deploy Ceph with OpenStack, see *Deploying an Overcloud with Containerized Red Hat Ceph*.

For instructions on how to set up NFS storage in the overcloud, see Configuring NFS Storage (from the *Advanced Overcloud Customization Guide*).

You can also configure the Block Storage service to use supported third-party storage appliances. The director includes the necessary components for easily deploying different backend solutions.

For a complete list of supported back end appliances and drivers, see Component, Plug-In, and Driver Support in RHEL OpenStack Platform. Some back ends have individual guides, which are available on the Red Hat OpenStack Storage documentation site.

Optional. Delete if not used.

2.2. ACTIVE-ACTIVE BLOCK STORAGE FOR HIGH AVAILABILITY

In active-passive mode, if the Block Storage service fails in a hyperconverged deployment, node fencing is undesirable. This is because node fencing can trigger storage to be rebalanced unnecessarily. Edge sites do not deploy Pacemaker, although Pacemaker is still present at the control site. Instead, edge sites deploy the Block Storage service in an active-active configuration to support highly available hyperconverged deployments.

Active-active deployments improve scaling, performance, and reduce response time by balancing workloads across all available nodes. Deploying the Block Storage service in an active-active configuration creates a highly available environment that maintains the management layer during partial network outages and single- or multi-node hardware failures. Active-active deployments allow a cluster to continue providing Block Storage services during a node outage.

Active-active deployments do not, however, enable workflows to resume automatically. If a service stops, individual operations running on the failed node will also fail during the outage. In this situation, confirm that the service is down and initiate a cleanup of resources that had in-flight operations.

2.2.1. Enabling active-active Block Storage

The **cinder-volume-active-active.yaml** file enables you to deploy the Block Storage service in an active-active configuration. This file ensures director uses the non-Pacemaker cinder-volume heat template and adds the **etcd** service to the deployment as a distributed lock manager (DLM).

The **cinder-volume-active-active.yaml** file also defines the active-active cluster name by assigning a value to the **CinderVolumeCluster** parameter. **CinderVolumeCluster** is a global Block Storage parameter. Therefore, you cannot include clustered (active-active) and non-clustered back ends in the same deployment.

IMPORTANT

Currently, active-active configuration for Block Storage works only with Ceph RADOS Block Device (RBD) back ends. If you plan to use multiple back ends, all back ends must support the active-active configuration. If a back end that does not support the active-active configuration is included in the deployment, that back end will not be available for storage. In an active-active deployment, you risk data loss if you save data on a back end that does not support the active-active configuration.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.

Procedure

To enable active-active Block Storage service volumes, include the following environment file in your overcloud deployment:

```
-e /usr/share/openstack-tripleo-heat-templates/environments/cinder-volume-active-active.yaml
```

2.2.2. Maintenance commands for active-active Block Storage configurations

After deploying an active-active configuration, there are several commands you can use to interact with the environment when using API version 3.17 and later.

User goal	Command
See the service listing, including details such as cluster name, host, zone, status, state, disabled reason, and back end state. **NOTE**: When deployed by director for the Ceph back end, the default cluster name is **tripleo@tripleo_ceph**.	**cinder service-list**

See detailed and summary information about clusters as a whole as opposed to individual services.	**cinder cluster-list** **NOTE**: This command requires a cinder API microversion of 3.7 or later.
See detailed information about a specific cluster.	**cinder cluster-show <cluster_name>** **NOTE**: This command requires a cinder API microversion of 3.7 or later.
Enable a disabled service.	**cinder cluster-enable <cluster_name>** **NOTE**: This command requires a cinder API microversion of 3.7 or later.
Disable a clustered service.	**cinder cluster-disable <cluster_name>** **NOTE**: This command requires a cinder API microversion of 3.7 or later.

2.2.3. Volume manage and unmanage

The unmanage and manage mechanisms facilitate moving volumes from one service using version X to another service using version X+1. Both services remain running during this process.

In API version 3.17 or later, you can see lists of volumes and snapshots that are available for management in Block Storage clusters. To see these lists, use the **--cluster** argument with **cinder manageable-list** or **cinder snapshot-manageable-list**.

In API version 3.16 and later, the **cinder manage** command also accepts the optional **--cluster** argument so that you can add previously unmanaged volumes to a Block Storage cluster.

2.2.4. Volume migration on a clustered service

With API version 3.16 and later, the **cinder migrate** and **cinder-manage** commands accept the **--cluster** argument to define the destination for active-active deployments.

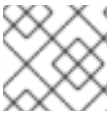
When you migrate a volume on a Block Storage clustered service, pass the optional **--cluster** argument and omit the **host** positional argument, because the arguments are mutually exclusive.

2.2.5. Initiating Block Storage service maintenance

All Block Storage volume services perform their own maintenance when they start. In an environment with multiple volume services grouped in a cluster, you can clean up services that are not currently running.

The command **work-cleanup** triggers server cleanups. The command returns:

- A list of the services that the command can clean.
- A list of the services that the command cannot clean because they are not currently running in the cluster.



NOTE

The **work-cleanup** command works only on servers running API version 3.24 or later.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.

Procedure

1. Run the following command to verify whether all of the services for a cluster are running:

```
$ cinder cluster-list --detailed
```

   Alternatively, run the **cluster show** command.

2. If any services are not running, run the following command to identify those specific services:

```
$ cinder service-list
```

3. Run the following command to trigger the server cleanup:

```
$ cinder work-cleanup [--cluster <cluster-name>] [--host <hostname>] [--binary <binary>] [--is-up <True|true|False|false>] [--disabled <True|true|False|false>] [--resource-id <resource-id>] [--resource-type <Volume|Snapshot>]
```

NOTE

Filters, such as **--cluster**, **--host**, and **--binary**, define what the command cleans. You can filter on cluster name, host name, type of service, and resource type, including a specific resource. If you do not apply filtering, the command attempts to clean everything that can be cleaned.

The following example filters by cluster name:

```
$ cinder work-cleanup --cluster tripleo@tripleo_ceph
```

2.3. GROUP VOLUME CONFIGURATION WITH VOLUME TYPES

With Red Hat OpenStack Platform you can create volume types so that you can apply associated settings to the volume type. You can apply settings during volume creation, see Section 3.1, “Creating Block Storage volumes”. You can also apply settings after you create a volume, see Section 4.5, “Block Storage volume retyping”. The following list shows some of the associated setting that you can apply to a volume type:

- The encryption of a volume. For more information, see Section 2.7.2, “Configuring Block Storage service volume encryption with the CLI”.
- The back end that a volume uses. For more information, see Section 2.10, “Specifying back ends for volume creation” and Section 4.8, “Migrating a volume between back ends with the CLI” .
- Quality-of-Service (QoS) Specs

Settings are associated with volume types using key-value pairs called Extra Specs. When you specify a volume type during volume creation, the Block Storage scheduler applies these key-value pairs as settings. You can associate multiple key-value pairs to the same volume type.

Volume types provide the capability to provide different users with storage tiers. By associating specific performance, resilience, and other settings as key-value pairs to a volume type, you can map tier-specific settings to different volume types. You can then apply tier settings when creating a volume by specifying the corresponding volume type.

2.3.1. Listing back end driver capabilities

Available and supported Extra Specs vary per back end driver. Consult the driver documentation for a list of valid Extra Specs.

Alternatively, you can query the Block Storage host directly to determine which well-defined standard Extra Specs are supported by its driver. Start by logging in (through the command line) to the node hosting the Block Storage service.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.

Procedure

```
# cinder service-list
```

This command will return a list containing the host of each Block Storage service (**cinder-backup**, **cinder-scheduler**, and **cinder-volume**). For example:

```
+------------------+--------------------------+------+---------
|      Binary      |           Host           | Zone |  Status ...
+------------------+--------------------------+------+---------
|  cinder-backup   |   localhost.localdomain  | nova | enabled ...
| cinder-scheduler |   localhost.localdomain  | nova | enabled ...
|  cinder-volume   | *localhost.localdomain@lvm* | nova | enabled ...
+------------------+--------------------------+------+---------
```

To display the driver capabilities (and, in turn, determine the supported Extra Specs) of a Block Storage service, run:

```
# cinder get-capabilities _VOLSVCHOST_
```

Where *VOLSVCHOST* is the complete name of the **cinder-volume**'s host. For example:

```
# cinder get-capabilities localhost.localdomain@lvm
   +---------------------+-----------------------------------------+
   |    Volume stats     |                  Value                  |
   +---------------------+-----------------------------------------+
   |     description     |                  None                   |
   |     display_name    |                  None                   |
   |    driver_version   |                  3.0.0                  |
   |      namespace      | OS::Storage::Capabilities::localhost.loc...
   |      pool_name      |                  None                   |
```

```
|   storage_protocol  |                  iSCSI           |
|     vendor_name     |              Open Source         |
|      visibility     |                None              |
| volume_backend_name |                 lvm              |
+---------------------+------------------------------------+
+--------------------+-------------------------------------+
| Backend properties |                Value                |
+--------------------+-------------------------------------+
|    compression     |     {u'type': u'boolean', u'description'...
|        qos         |          {u'type': u'boolean', u'des ...
|    replication     |     {u'type': u'boolean', u'description'...
| thin_provisioning  | {u'type': u'boolean', u'description': u'S...
+--------------------+-------------------------------------+
```

The **Backend properties** column shows a list of Extra Spec Keys that you can set, while the **Value** column provides information on valid corresponding values.

NOTE

Available and supported Extra Specs vary per back end driver. Consult the driver documentation for a list of valid Extra Specs.

Alternatively, you can query the Block Storage host directly to determine which well-defined standard Extra Specs are supported by its driver. Start by logging in (through the command line) to the node hosting the Block Storage service. Then:

```
# cinder service-list
```

This command will return a list containing the host of each Block Storage service (**cinder-backup**, **cinder-scheduler**, and **cinder-volume**). For example:

```
+------------------+--------------------------+------+---------
|      Binary      |           Host           | Zone |  Status ...
+------------------+--------------------------+------+---------
|  cinder-backup   |   localhost.localdomain  | nova | enabled ...
| cinder-scheduler |   localhost.localdomain  | nova | enabled ...
|  cinder-volume   | *localhost.localdomain@lvm* | nova | enabled ...
+------------------+--------------------------+------+---------
```

To display the driver capabilities (and, in turn, determine the supported Extra Specs) of a Block Storage service, run:

```
# cinder get-capabilities _VOLSVCHOST_
```

Where *VOLSVCHOST* is the complete name of the **cinder-volume**'s host. For example:

```
# cinder get-capabilities localhost.localdomain@lvm
   +---------------------+-----------------------------------------+
   |     Volume stats    |                  Value                  |
   +---------------------+-----------------------------------------+
   |     description     |                  None                   |
   |     display_name    |                  None                   |
   |    driver_version   |                 3.0.0                   |
   |      namespace      | OS::Storage::Capabilities::localhost.loc...
```

```
|     pool_name      |                  None           |
|   storage_protocol  |                  iSCSI          |
|     vendor_name     |                Open Source      |
|     visibility      |                None             |
| volume_backend_name |                  lvm            |
+---------------------+-----------------------------------+
+---------------------+-----------------------------------+
| Backend properties |                  Value            |
+---------------------+-----------------------------------+
|    compression     |     {u'type': u'boolean', u'description'...
|        qos         |          {u'type': u'boolean', u'des ...
|    replication     |     {u'type': u'boolean', u'description'...
| thin_provisioning  | {u'type': u'boolean', u'description': u'S...
+---------------------+-----------------------------------+
```

The **Backend properties** column shows a list of Extra Spec Keys that you can set, while the **Value** column provides information on valid corresponding values.

2.3.2. Creating and configuring a volume type

Create volume types so that you can apply associated settings to the volume type. Volume types provide the capability to provide different users with storage tiers. By associating specific performance, resilience, and other settings as key-value pairs to a volume type, you can map tier-specific settings to different volume types. You can then apply tier settings when creating a volume by specifying the corresponding volume type.

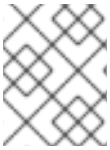
Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools in *Director Installation and Usage*.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output in *Director Installation and Usage*.

Procedure

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**
2. Click **Create Volume Type**.
3. Enter the volume type name in the **Name** field.
4. Click **Create Volume Type**. The new type appears in the **Volume Types** table.
5. Select the volume type's **View Extra Specs** action.
6. Click **Create** and specify the **Key** and **Value**. The key-value pair must be valid; otherwise, specifying the volume type during volume creation will result in an error.
7. Click **Create**. The associated setting (key-value pair) now appears in the **Extra Specs** table.

By default, all volume types are accessible to all OpenStack projects. If you need to create volume types with restricted access, you will need to do so through the CLI. For instructions, see Section 2.3.4, "Creating and configuring private volume types".



NOTE

You can also associate a QoS Spec to the volume type. For more information, see Section 2.6.3, "Associating a Quality-of-Service specification with a volume type" .

2.3.3. Editing a volume type

Edit a volume type in the Dashboard to modify the **Extra Specs** configuration of the volume type.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**
2. In the **Volume Types** table, select the volume type's **View Extra Specs** action.
3. On the **Extra Specs** table of this page, you can:
    - Add a new setting to the volume type. To do this, click **Create** and specify the key/value pair of the new setting you want to associate to the volume type.
    - Edit an existing setting associated with the volume type by selecting the setting's **Edit** action.
    - Delete existing settings associated with the volume type by selecting the extra specs' check box and clicking **Delete Extra Specs** in this and the next dialog screen.

To delete a volume type, select its corresponding check boxes from the **Volume Types** table and click **Delete Volume Types**

2.3.4. Creating and configuring private volume types

By default, all volume types are available to all projects. You can create a restricted volume type by marking it **private**. To do so, set the type's **is-public** flag to **false**.

Private volume types are useful for restricting access to volumes with certain attributes. Typically, these are settings that should only be usable by specific projects; examples include new back ends or ultra-high performance configurations that are being tested.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools in *Director Installation and Usage*.

Procedure

```
$ cinder type-create --is-public false  <TYPE-NAME>
```

By default, private volume types are only accessible to their creators. However, admin users can find and view private volume types using the following command:

```
$ cinder type-list --all
```

This command lists both public and private volume types, and it also includes the name and ID of each one. You need the volume type's ID to provide access to it.

Access to a private volume type is granted at the project level. To grant a project access to a private volume type, run:

```
$ cinder  type-access-add --volume-type <TYPE-ID> --project-id <TENANT-ID>
```

To view which projects have access to a private volume type, run:

```
$ cinder  type-access-list --volume-type <TYPE-ID>
```

To remove a project from the access list of a private volume type, run:

```
$ cinder  type-access-remove --volume-type <TYPE-ID> --project-id <TENANT-ID>
```

NOTE

By default, only users with administrative privileges can create, view, or configure access for private volume types.

2.4. CREATING AND CONFIGURING AN INTERNAL PROJECT FOR THE BLOCK STORAGE SERVICE (CINDER)

Some Block Storage features (for example, the Image-Volume cache) require the configuration of an *internal tenant*. The Block Storage service uses this tenant/project to manage block storage items that do not necessarily need to be exposed to normal users. Examples of such items are images cached for frequent volume cloning or temporary copies of volumes being migrated.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools in *Director Installation and Usage*.

Procedure

1. To configure an internal project, first create a generic project and user, both named **cinder-internal**. To do so, log in to the Controller node and run:

```
# openstack project create --enable --description "Block Storage Internal Project" cinder-internal
  +-------------+----------------------------------+
  |  Property   |              Value               |
  +-------------+----------------------------------+
  | description |  Block Storage Internal Tenant   |
  |   enabled   |               True               |
  |      id     | cb91e1fe446a45628bb2b139d7dccaef |
  |     name    |         cinder-internal          |
  +-------------+----------------------------------+
# openstack user create --project cinder-internal cinder-internal
  +----------+----------------------------------+
  | Property |              Value               |
  +----------+----------------------------------+
  |  email   |               None               |
  | enabled  |               True               |
  |    id    | 84e9672c64f041d6bfa7a930f558d946 |
  |   name   |         cinder-internal          |
  |project_id| cb91e1fe446a45628bb2b139d7dccaef |
  | username |         cinder-internal          |
  +----------+----------------------------------+
```

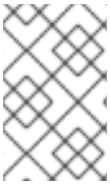
The procedure for adding Extra Config options creates an internal project. Refer to Section 2.5, "Configuring the image-volume cache".

2.5. CONFIGURING THE IMAGE-VOLUME CACHE

The Block Storage service features an optional *Image-Volume cache* which can be used when creating volumes from images. This cache is designed to improve the speed of volume creation from frequently-used images. For information on how to create volumes from images, see Section 3.1, "Creating Block Storage volumes".

When enabled, the Image-Volume cache stores a copy of an image the first time a volume is created from it. This stored image is cached locally to the Block Storage back end to help improve performance the next time the image is used to create a volume. The Image-Volume cache's limit can be set to a size (in GB), number of images, or both.

The Image-Volume cache is supported by several back ends. If you are using a third-party back end, refer to its documentation for information on Image-Volume cache support.



NOTE

The Image-Volume cache requires that an *internal tenant* be configured for the Block Storage service. For instructions, see Section 2.4, "Creating and configuring an internal project for the Block Storage service (cinder)".

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.

Procedure

To enable and configure the Image-Volume cache on a back end (*BACKEND*), add the values to an **ExtraConfig** section of an environment file on the undercloud. For example:

```
parameter_defaults:
  ExtraConfig:
    cinder::config::cinder_config:
      DEFAULT/cinder_internal_tenant_project_id:
        value: TENANTID
      DEFAULT/cinder_internal_tenant_user_id:
        value: USERID
      BACKEND/image_volume_cache_enabled: 1
        value: True
      BACKEND/image_volume_cache_max_size_gb:
        value: MAXSIZE 2
      BACKEND/image_volume_cache_max_count:
        value: MAXNUMBER 3
```

1. Replace *BACKEND* with the name of the target back end (specifically, its **volume_backend_name** value).
2. By default, the Image-Volume cache size is only limited by the back end. Change *MAXSIZE* to a number in GB.
3. You can also set a maximum number of images using *MAXNUMBER*.

The Block Storage service database uses a time stamp to track when each cached image was last used to create an image. If either or both *MAXSIZE* and *MAXNUMBER* are set, the Block Storage service will delete cached images as needed to make way for new ones. Cached images with the oldest time stamp are deleted first whenever the Image-Volume cache limits are met.

After you create the environment file in **/home/stack/templates/**, log in as the stack user and deploy the configuration by running:

```
$ openstack overcloud deploy --templates \
-e /home/stack/templates/<ENV_FILE>.yaml
```

Where **ENV_FILE.yaml** is the name of the file with the **ExtraConfig** settings added earlier.

IMPORTANT

If you passed any extra environment files when you created the overcloud, pass them again here using the **-e** option to avoid making undesired changes to the overcloud.

For additional information on the **openstack overcloud deploy** command, see Deployment command in *Director Installation and Usage*.

2.6. BLOCK STORAGE SERVICE (CINDER) QUALITY-OF-SERVICE

You can map multiple performance settings to a single Quality-of-Service specification (QOS Specs). Doing so allows you to provide performance tiers for different user types.

Performance settings are mapped as key-value pairs to QOS Specs, similar to the way volume settings are associated to a volume type. However, QOS Specs are different from volume types in the following respects:

- QOS Specs are used to apply performance settings, which include limiting read/write operations to disks. Available and supported performance settings vary per storage driver.
  To determine which QOS Specs are supported by your back end, consult the documentation of your back end device's volume driver.

- Volume types are directly applied to volumes, whereas QOS Specs are not. Rather, QOS Specs are associated to volume types. During volume creation, specifying a volume type also applies the performance settings mapped to the volume type's associated QOS Specs.

You can define performance limits for volumes on a per-volume basis using basic volume QOS values. The Block Storage service supports the following options:

- **read_iops_sec**
- **write_iops_sec**
- **total_iops_sec**
- **read_bytes_sec**
- **write_bytes_sec**
- **total_bytes_sec**
- **read_iops_sec_max**
- **write_iops_sec_max**
- **total_iops_sec_max**
- **read_bytes_sec_max**
- **write_bytes_sec_max**
- **total_bytes_sec_max**
- **size_iops_sec**

2.6.1. Creating and configuring a Quality-of-Service specification

As an administrator, you can create and configure a QOS Spec through the QOS Specs table. You can associate more than one key/value pair to the same QOS Spec.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**
2. On the **QOS Specs** table, click **Create QOS Spec**.
3. Enter a name for the **QOS Spec**.
4. In the **Consumer** field, specify where the QOS policy should be enforced:

   Table 2.1. Consumer Types

Type	Description
back-end	QOS policy will be applied to the Block Storage back end.
front-end	QOS policy will be applied to Compute.
both	QOS policy will be applied to both Block Storage and Compute.

5. Click **Create**. The new QOS Spec should now appear in the **QOS Specs** table.
6. In the **QOS Specs** table, select the new spec's **Manage Specs** action.
7. Click **Create**, and specify the **Key** and **Value**. The key-value pair must be valid; otherwise, specifying a volume type associated with this QOS Spec during volume creation will fail. For example, to set read limit IOPS to **500**, use the following Key/Value pair:

   ```
   read_iops_sec=500
   ```

8. Click **Create**. The associated setting (key-value pair) now appears in the **Key-Value Pairs** table.

2.6.2. Setting capacity-derived Quality-of-Service limits

You can use volume types to implement capacity-derived Quality-of-Service (QoS) limits on volumes. This will allow you to set a deterministic IOPS throughput based on the size of provisioned volumes. Doing this simplifies how storage resources are provided to users – namely, providing a user with pre-determined (and, ultimately, highly predictable) throughput rates based on the volume size they provision.

In particular, the Block Storage service allows you to set how much IOPS to allocate to a volume based on the actual provisioned size. This throughput is set on an IOPS per GB basis through the following QoS keys:

```
read_iops_sec_per_gb
write_iops_sec_per_gb
total_iops_sec_per_gb
```

These keys allow you to set read, write, or total IOPS to scale with the size of provisioned volumes. For example, if the volume type uses **read_iops_sec_per_gb=500**, then a provisioned 3GB volume would automatically have a read IOPS of 1500.

Capacity-derived QoS limits are set per volume type, and configured like any normal QoS spec. In addition, these limits are supported by the underlying Block Storage service directly, and is not dependent on any particular driver.

For more information about volume types, see Section 2.3, "Group volume configuration with volume types" and Section 2.3.2, "Creating and configuring a volume type" . For instructions on how to set QoS specs, Section 2.6, "Block Storage service (cinder) Quality-of-Service" .

WARNING

When you apply a volume type (or perform a volume re-type) with capacity-derived QoS limits to an attached volume, the limits will not be applied. The limits will only be applied once you detach the volume from its instance.

See Section 4.5, "Block Storage volume retyping" for information about volume re-typing.

2.6.3. Associating a Quality-of-Service specification with a volume type

As an administrator, you can associate a QOS Spec to an existing volume type using the **Volume Types** table.

Prerequisites

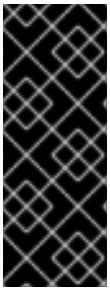
- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. As an administrator in the dashboard, select **Admin > Volumes > Volume Types**
2. In the **Volume Types** table, select the type's **Manage QOS Spec Association** action.
3. Select a QOS Spec from the **QOS Spec to be associated** list. To disassociate a QOS specification from an existing volume type, select **None**.
4. Click **Associate**. The selected QOS Spec now appears in the **Associated QOS Spec** column of the edited volume type.

2.7. BLOCK STORAGE SERVICE (CINDER) VOLUME ENCRYPTION

Volume encryption helps provide basic data protection in case the volume back-end is either compromised or outright stolen. Both Compute and Block Storage services are integrated to allow instances to read access and use encrypted volumes. You must deploy Barbican to take advantage of volume encryption.



IMPORTANT

- Volume encryption is not supported on file-based volumes (such as NFS).
- Retyping an unencrypted volume to an encrypted volume of the same size is not supported, because encrypted volumes require additional space to store encryption data. For more information about encrypting unencrypted volumes, see Encrypting unencrypted volumes.

Volume encryption is applied through volume type. See Section 2.7.2, "Configuring Block Storage service volume encryption with the CLI" for information on encrypted volume types.

2.7.1. Configuring Block Storage service volume encryption with the Dashboard

To create encrypted volumes, you first need an *encrypted volume type*. Encrypting a volume type involves setting what provider class, cipher, and key size it should use.

Prerequisites

- A successful undercloud installation. For more information, see Installing director *Director Installation and Usage*.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools *Director Installation and Usage*.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output *Director Installation and Usage*.

Procedure

1. As an admin user in the dashboard, select **Admin > Volumes > Volume Types**
2. In the **Actions** column of the volume to be encrypted, select **Create Encryption** to launch the **Create Volume Type Encryption** wizard.
3. From there, configure the **Provider**, **Control Location**, **Cipher**, and **Key Size** settings of the volume type's encryption. The **Description** column describes each setting.

IMPORTANT

The values listed below are the only supported options for **Provider**, **Cipher**, and **Key Size**.

   a. Enter **luks** for **Provider**.
   b. Enter **aes-xts-plain64** for **Cipher**.
   c. Enter **256** for **Key Size**.

4. Click **Create Volume Type Encryption**

Once you have an encrypted volume type, you can invoke it to automatically create encrypted volumes. For more information on creating a volume type, see Section 2.3.2, "Creating and configuring a volume type". Specifically, select the encrypted volume type from the Type drop-down list in the **Create Volume** window.

To configure an encrypted volume type through the CLI, see Section 2.7.2, "Configuring Block Storage service volume encryption with the CLI".

You can also re-configure the encryption settings of an encrypted volume type.

1. Select **Update Encryption** from the **Actions** column of the volume type to launch the **Update Volume Type Encryption** wizard.
2. In **Project > Compute > Volumes**, check the **Encrypted** column in the **Volumes** table to determine whether the volume is encrypted.
3. If the volume is encrypted, click **Yes** in that column to view the encryption settings.

2.7.2. Configuring Block Storage service volume encryption with the CLI

To create encrypted volumes, you first need an *encrypted volume type*. Encrypting a volume type involves setting what provider class, cipher, and key size it should use.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Create a volume type:

```
$ cinder type-create encrypt-type
```

2. Configure the cipher, key size, control location, and provider settings:

```
$ cinder encryption-type-create --cipher aes-xts-plain64 --key-size 256 --control-location
front-end encrypt-type luks
```

3. Create an encrypted volume:

```
$ cinder --debug create 1 --volume-type encrypt-type --name DemoEncVol
```

For additional information, refer to the Manage secrets with the OpenStack Key Manager guide.

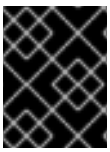
2.7.3. Automatic deletion of volume image encryption key

The Block Storage service (cinder) creates an encryption key in the Key Management service (barbican) when it uploads an encrypted volume to the Image service (glance). This creates a 1:1 relationship between an encryption key and a stored image.

Encryption key deletion prevents unlimited resource consumption of the Key Management service. The Block Storage, Key Management, and Image services automatically manage the key for an encrypted volume, including the deletion of the key.

The Block Storage service automatically adds two properties to a volume image:

- **cinder_encryption_key_id** - The identifier of the encryption key that the Key Management service stores for a specific image.
- **cinder_encryption_key_deletion_policy** - The policy that tells the Image service to tell the Key Management service whether to delete the key associated with this image.



IMPORTANT

The values of these properties are automatically assigned. **To avoid unintentional data loss, do not adjust these values**.

When you create a volume image, the Block Storage service sets the **cinder_encryption_key_deletion_policy** property to **on_image_deletion**. When you delete a volume image, the Image service deletes the corresponding encryption key if the **cinder_encryption_key_deletion_policy** equals **on_image_deletion**.

IMPORTANT

Red Hat does not recommend manual manipulation of the **cinder_encryption_key_id** or **cinder_encryption_key_deletion_policy** properties. If you use the encryption key that is identified by the value of **cinder_encryption_key_id** for any other purpose, you risk data loss.

2.8. DEPLOYING AVAILABILITY ZONES FOR BLOCK STORAGE VOLUME BACK ENDS

An availability zone is a provider-specific method of grouping cloud instances and services. Director uses **CinderXXXAvailabilityZone** parameters (where **XXX** is associated with a specific back end) to configure different availability zones for Block Storage volume back ends.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

1. Add the following parameters to the environment file to create two availability zones:

```
parameter_defaults:
 CinderXXXAvailabilityZone: zone1
 CinderYYYAvailabilityZone: zone2
```

Replace **XXX** and **YYY** with supported back-end values, such as:

```
CinderISCSIAvailabilityZone
CinderNfsAvailabilityZone
CinderRbdAvailabilityZone
```

NOTE

Search the **/usr/share/openstack-tripleo-heat-templates/deployment/cinder/** directory for the heat template associated with your back end for the correct back end value.

The following example deploys two back ends where **rbd** is zone 1 and **iSCSI** is zone 2:

```
parameter_defaults:
 CinderRbdAvailabilityZone: zone1
 CinderISCSIAvailabilityZone: zone2
```

2. Deploy the overcloud and include the updated environment file.

2.9. BLOCK STORAGE SERVICE (CINDER) CONSISTENCY GROUPS

You can use the Block Storage (cinder) service to set consistency groups to group multiple volumes together as a single entity. This means that you can perform operations on multiple volumes at the same time instead of individually. You can use consistency groups to create snapshots for multiple volumes simultaneously. This also means that you can restore or clone those volumes simultaneously.

A volume can be a member of multiple consistency groups. However, you cannot delete, retype, or migrate volumes after you add them to a consistency group.

2.9.1. Configuring Block Storage service consistency groups

By default, Block Storage security policy disables consistency groups APIs. You must enable it here before you use the feature. The related consistency group entries in the **/etc/cinder/policy.json** file of the node that hosts the Block Storage API service, **openstack-cinder-api** list the default settings:

```
"consistencygroup:create" : "group:nobody",
"consistencygroup:delete": "group:nobody",
"consistencygroup:update": "group:nobody",
"consistencygroup:get": "group:nobody",
"consistencygroup:get_all": "group:nobody",
"consistencygroup:create_cgsnapshot" : "group:nobody",
"consistencygroup:delete_cgsnapshot": "group:nobody",
"consistencygroup:get_cgsnapshot": "group:nobody",
"consistencygroup:get_all_cgsnapshots": "group:nobody",
```

You must change these settings in an environment file and then deploy them to the overcloud by using the **openstack overcloud deploy** command. Do not edit the JSON file directly because the changes are overwritten next time the overcloud is deployed.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

1. Edit an environment file and add a new entry to the **parameter_defaults** section. This ensures that the entries are updated in the containers and are retained whenever the environment is re-deployed by director with the **openstack overcloud deploy** command.

2. Add a new section to an environment file using **CinderApiPolicies** to set the consistency group settings. The equivalent **parameter_defaults** section with the default settings from the JSON file appear in the following way:

```
parameter_defaults:
  CinderApiPolicies: { \
     cinder-consistencygroup_create: { key: 'consistencygroup:create', value: 'group:nobody' },
\
     cinder-consistencygroup_delete: { key: 'consistencygroup:delete', value: 'group:nobody' },
\
     cinder-consistencygroup_update: { key: 'consistencygroup:update', value: 'group:nobody'
}, \
     cinder-consistencygroup_get: { key: 'consistencygroup:get', value: 'group:nobody' }, \
     cinder-consistencygroup_get_all: { key: 'consistencygroup:get_all', value: 'group:nobody' },
\
     cinder-consistencygroup_create_cgsnapshot: { key:
'consistencygroup:create_cgsnapshot', value: 'group:nobody' }, \
     cinder-consistencygroup_delete_cgsnapshot: { key:
'consistencygroup:delete_cgsnapshot', value: 'group:nobody' }, \
     cinder-consistencygroup_get_cgsnapshot: { key: 'consistencygroup:get_cgsnapshot',
value: 'group:nobody' }, \
     cinder-consistencygroup_get_all_cgsnapshots: { key:
'consistencygroup:get_all_cgsnapshots', value: 'group:nobody' }, \
  }
```

3. The value **'group:nobody'** determines that no group can use this feature so it is effectively disabled. To enable it, change the group to another value.

4. For increased security, set the permissions for both consistency group API and volume type management API to be identical. The volume type management API is set to **"rule:admin_or_owner"** by default in the same **/etc/cinder/policy.json_ file**:

```
"volume_extension:types_manage": "rule:admin_or_owner",
```

5. To make the consistency groups feature available to all users, set the API policy entries to allow users to create, use, and manage their own consistency groups. To do so, use **rule:admin_or_owner**:

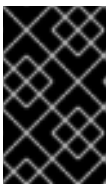
```
CinderApiPolicies: { \
     cinder-consistencygroup_create: { key: 'consistencygroup:create', value:
'rule:admin_or_owner' }, \
     cinder-consistencygroup_delete: { key: 'consistencygroup:delete', value:
'rule:admin_or_owner' },  \
     cinder-consistencygroup_update: { key: 'consistencygroup:update', value:
'rule:admin_or_owner' }, \
     cinder-consistencygroup_get: { key: 'consistencygroup:get', value: 'rule:admin_or_owner'
}, \
     cinder-consistencygroup_get_all: { key: 'consistencygroup:get_all', value:
'rule:admin_or_owner' }, \
     cinder-consistencygroup_create_cgsnapshot: { key:
'consistencygroup:create_cgsnapshot', value: 'rule:admin_or_owner' }, \
     cinder-consistencygroup_delete_cgsnapshot: { key:
'consistencygroup:delete_cgsnapshot', value: 'rule:admin_or_owner' }, \
     cinder-consistencygroup_get_cgsnapshot: { key: 'consistencygroup:get_cgsnapshot',
value: 'rule:admin_or_owner' }, \
```

```
    cinder-consistencygroup_get_all_cgsnapshots: { key:
'consistencygroup:get_all_cgsnapshots', value: 'rule:admin_or_owner' }, \
}
```

6. When you have created the environment file in **/home/stack/templates/**, log in as the stack user and deploy the configuration:

```
$ openstack overcloud deploy --templates \
-e /home/stack/templates/<ENV_FILE>.yaml
```

Replace **<ENV_FILE.yaml>** with the name of the file with the **ExtraConfig** settings you added.



IMPORTANT

If you passed any extra environment files when you created the overcloud, pass them again here by using the **-e** option to avoid making undesired changes to the overcloud.

For more information about the **openstack overcloud deploy** command, see Creating the Overcloud with the CLI Tools in the *Director Installation and Usage* guide.

2.9.2. Creating Block Storage consistency groups with the Dashboard

After you enable the consistency groups API, you can start creating consistency groups.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. As an admin user in the dashboard, select **Project > Compute > Volumes > Volume Consistency Groups**.
2. Click **Create Consistency Group**.
3. In the **Consistency Group Information** tab of the wizard, enter a name and description for your consistency group. Then, specify its **Availability Zone**.
4. You can also add volume types to your consistency group. When you create volumes within the consistency group, the Block Storage service will apply compatible settings from those volume types. To add a volume type, click its **+** button from the **All available volume types** list.
5. Click **Create Consistency Group**. It appears next in the **Volume Consistency Groups** table.

2.9.3. Managing Block Storage service consistency groups with the Dashboard

Use the Red Hat OpenStack Platform (RHOSP) Dashboard to manage consistency groups for Block Storage volumes.

Prerequisites

- A successful undercloud installation. For more information, see Installing director in *Director Installation and Usage*.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools in *Director Installation and Usage*.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output in *Director Installation and Usage*.

Procedure

1. Optional: You can change the name or description of a consistency group by selecting **Edit Consistency Group** from its **Action** column.
2. To add or remove volumes from a consistency group directly, as an admin user in the dashboard, select **Project > Compute > Volumes > Volume Consistency Groups**
3. Find the consistency group you want to configure. In the **Actions** column of that consistency group, select **Manage Volumes**. This launches the **Add/Remove Consistency Group Volumes** wizard.
   a. To add a volume to the consistency group, click its **+** button from the **All available volumes** list.
   b. To remove a volume from the consistency group, click its **-** button from the **Selected volumes** list.
4. Click **Edit Consistency Group**.

2.9.4. Creating and managing consistency group snapshots for the Block Storage service

After you add volumes to a consistency group, you can now create snapshots from it.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

1. Log in as **admin** user from the command line on the node that hosts the **openstack-cinder-api** and enter:

   ```
   # export OS_VOLUME_API_VERSION=2
   ```

   This configures the client to use version **2** of the **openstack-cinder-api**.
2. List all available consistency groups and their respective IDs:

   ```
   # cinder consisgroup-list
   ```

3. Create snapshots using the consistency group:

```
# cinder cgsnapshot-create --name <CGSNAPNAME> --description "<DESCRIPTION>"
<CGNAMEID>
```

Replace:

- **<CGSNAPNAME>** with the name of the snapshot (optional).
- **<DESCRIPTION>** with a description of the snapshot (optional).
- **<CGNAMEID>** with the name or ID of the consistency group.

4. Display a list of all available consistency group snapshots:

```
# cinder cgsnapshot-list
```

2.9.5. Cloning Block Storage service consistency groups

You can also use consistency groups to create a whole batch of pre-configured volumes simultaneously. You can do this by cloning an existing consistency group or restoring a consistency group snapshot. Both processes use the same command.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. To clone an existing consistency group:

```
# cinder consisgroup-create-from-src --source-cg <CGNAMEID> --name <CGNAME> --
description "<DESCRIPTION>"
```

Replace:

- **<CGNAMEID>** is the name or ID of the consistency group you want to clone.
- **<CGNAME>** is the name of your consistency group (optional).
- **<DESCRIPTION>** is a description of your consistency group (optional).

2. To create a consistency group from a consistency group snapshot:

```
# cinder consisgroup-create-from-src --cgsnapshot <CGSNAPNAME> --name <CGNAME> -
-description "<DESCRIPTION>
```

Replace **<CGSNAPNAME>** with the name or ID of the snapshot you are using to create the consistency group.

2.10. SPECIFYING BACK ENDS FOR VOLUME CREATION

Whenever multiple Block Storage (cinder) back ends are configured, you must also create a volume type for each back end. You can then use the type to specify which back end to use for a created volume. For more information about volume types, see Section 2.3, "Group volume configuration with volume types" .

To specify a back end when creating a volume, select its corresponding volume type from the Type list (see Section 3.1, "Creating Block Storage volumes").

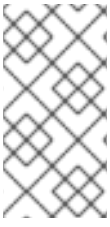
If you do not specify a back end during volume creation, the Block Storage service automatically chooses one for you. By default, the service chooses the back end with the most available free space. You can also configure the Block Storage service to choose randomly among all available back ends instead. For more information, see Section 3.5, "Allocating volumes to multiple back ends" .

2.11. ENABLING LVM2 FILTERING ON OVERCLOUD NODES

If you use LVM2 (Logical Volume Management) volumes with certain Block Storage service (cinder) back ends, the volumes that you create inside Red Hat OpenStack Platform (RHOSP) guests might become visible on the overcloud nodes that host **cinder-volume** or **nova-compute** containers. In this case, the LVM2 tools on the host scan the LVM2 volumes that the OpenStack guest creates, which can result in one or more of the following problems on Compute or Controller nodes:

- LVM appears to see volume groups from guests
- LVM reports duplicate volume group names
- Volume detachments fail because LVM is accessing the storage
- Guests fail to boot due to problems with LVM
- The LVM on the guest machine is in a partial state due to a missing disk that actually exists
- Block Storage service (cinder) actions fail on devices that have LVM
- Block Storage service (cinder) snapshots fail to remove correctly
- Errors during live migration: **/etc/multipath.conf** does not exist

To prevent this erroneous scanning, and to segregate guest LVM2 volumes from the host node, you can enable and configure a filter with the **LVMFilterEnabled** heat parameter when you deploy or update the overcloud. This filter is computed from the list of physical devices that host active LVM2 volumes. You can also allow and deny block devices explicitly with the **LVMFilterAllowlist** and **LVMFilterDenylist** parameters. You can apply this filtering globally, to specific node roles, or to specific devices.



NOTE

This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

Prerequisites

- A successful undercloud installation. For more information, see Installing the undercloud.

Procedure

1. Log in to the undercloud host as the **stack** user.

2. Source the undercloud credentials file:

```
$ source ~/stackrc
```

3. Create a new environment file, or modify an existing environment file. In this example, create a new file **lvm2-filtering.yaml**:

```
$ touch ~/lvm2-filtering.yaml
```

4. Include the following parameter in the environment file:

```
parameter_defaults:
  LVMFilterEnabled: true
```

You can further customize the implementation of the LVM2 filter. For example, to enable filtering only on Compute nodes, use the following configuration:

```
parameter_defaults:
  ComputeParameters:
    LVMFilterEnabled: true
```

These parameters also support regular expression. To enable filtering only on Compute nodes, and ignore all devices that start with **/dev/sd**, use the following configuration:

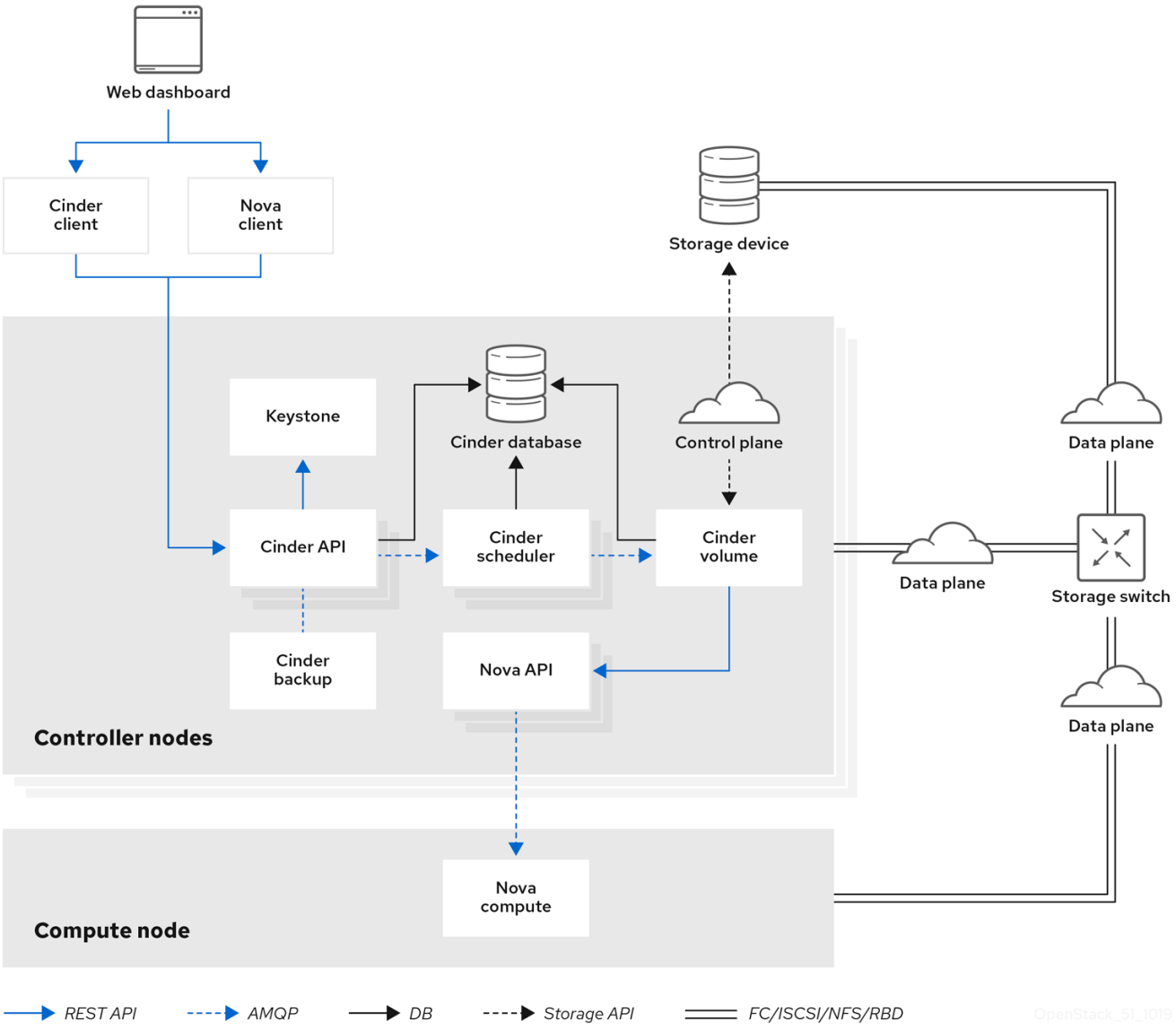
```
parameter_defaults:
  ComputeParameters:
    LVMFilterEnabled: true
    LVMFilterDenylist:
      - /dev/sd.*
```

5. Run the **openstack overcloud deploy** command and include the environment file that contains the LVM2 filtering configuration, as well as any other environment files that are relevant to your overcloud deployment:

```
$ openstack overcloud deploy --templates \
<environment-files> \
-e lvm2-filtering.yaml
```

2.12. MULTIPATH CONFIGURATION

Use multipath to configure multiple I/O paths between server nodes and storage arrays into a single device to create redundancy and improve performance. You can configure multipath on new and existing overcloud deployments.



2.12.1. Configuring multipath on new deployments

Complete this procedure to configure multipath on a new overcloud deployment.

For information about how to configure multipath on existing overcloud deployments, see Section 2.12.2, "Configuring multipath on existing deployments".

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. The overcloud Controller and Compute nodes must have access to the Red Hat Enterprise Linux server repository. For more information, see Downloading the base cloud image in the *Director Installation and Usage* guide.

Procedure

1. Configure the overcloud.

NOTE

For more information, see Configuring a basic overcloud with CLI tools in the *Director Installation and Usage* guide.

2. Update the heat template to enable multipath:

```
parameter_defaults:
  NovaLibvirtVolumeUseMultipath:  true
  NovaComputeOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
  CinderVolumeOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
```

3. Optional: If you are using Block Storage (cinder) as an Image service (glance) back end, you must also complete the following steps:

   a. Add the following **GlanceApiOptVolumes** configuration to the heat template:

```
parameter_defaults:
  GlanceApiOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
```

   b. Set the **ControllerExtraConfig** parameter in the following way:

```
parameter_defaults:
  ControllerExtraConfig:
    glance::config::api_config:
      default_backend/cinder_use_multipath:
        value: true
```

   Note

   Ensure that both **default_backend** and the **GlanceBackendID** heat template default value match.

4. For every configured back end, set **use_multipath_for_image_xfer** to **true**:

```
parameter_defaults:
  ExtraConfig:
    cinder::config::cinder_config:
      <backend>/use_multipath_for_image_xfer:
        value: true
```

5. Deploy the overcloud:

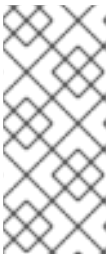
```
$ openstack overcloud deploy
```

NOTE

For information about creating the overcloud using overcloud parameters, see Creating the Overcloud with the CLI Tools in the *Director Installation and Usage* guide.

6. Before containers are running, install multipath on all Controller and Compute nodes:

```
$ sudo dnf install -y device-mapper-multipath
```



NOTE

Director provides a set of hooks to support custom configuration for specific node roles after the first boot completes and before the core configuration begins. For more information about custom overcloud configuration, see Pre-Configuration: Customizing Specific Overcloud Roles in the *Advanced Overcloud Customization* guide.

7. Configure the multipath daemon on all Controller and Compute nodes:

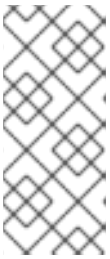
```
$ mpathconf --enable --with_multipathd y --user_friendly_names n --find_multipaths y
```

NOTE

The example code creates a basic multipath configuration that works for most environments. However, check with your storage vendor for recommendations, because some vendors have optimized configurations that are specific to their hardware. For more information about multipath, see the Configuring device mapper multipath guide.

8. Run the following command on all Controller and Compute nodes to prevent partition creation:

```
$ sed -i "s/^defaults {/defaults {\n\tskip_kpartx yes/" /etc/multipath.conf
```



NOTE

Setting **skip_kpartx** to **yes** prevents kpartx on the Compute node from automatically creating partitions on the device, which prevents unnecessary device mapper entries. For more information about configuration attributes, see Modifying the DM-Multipath configuration file in the *Configuring device mapper multipath* guide.

9. Start the multipath daemon on all Controller and Compute nodes:

```
$ systemctl enable --now multipathd
```

2.12.2. Configuring multipath on existing deployments

Configure multipath on existing deployments so that your workloads can use multipath functionality.

NOTE

Any new workloads that you create after you configure multipath on existing deployments are multipath-aware by default. If you have any pre-existing workloads, you must shelve and unshelve the instances to enable multipath on these instances.

For more information about how to configure multipath on new overcloud deployments, see Section 2.12.1, "Configuring multipath on new deployments".

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. The overcloud Controller and Compute nodes must have access to the Red Hat Enterprise Linux server repository. For more information, see Downloading the base cloud image in the *Director Installation and Usage* guide.

Procedure

1. Verify that multipath is installed on all Controller and Compute nodes:

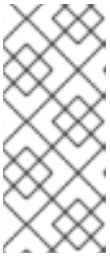
```
$ rpm -qa | grep device-mapper-multipath

device-mapper-multipath-0.4.9-127.el8.x86_64
device-mapper-multipath-libs-0.4.9-127.el8.x86_64
```

   If multipath is not installed, install it on all Controller and Compute nodes:

```
$ sudo dnf install -y device-mapper-multipath
```

2. Configure the multipath daemon on all Controller and Compute nodes:

```
$ mpathconf --enable --with_multipathd y --user_friendly_names n --find_multipaths y
```

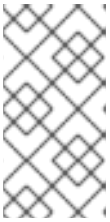


NOTE

The example code creates a basic multipath configuration that works for most environments. However, check with your storage vendor for recommendations, because some vendors have optimized configurations specific to their hardware. For more information about multipath, see the Configuring device mapper multipath guide.

3. Run the following command on all Controller and Compute nodes to prevent partition creation:

```
$ sed -i "s/^defaults {/defaults {\n\tskip_kpartx yes/" /etc/multipath.conf
```



NOTE

Setting **skip_kpartx** to **yes** prevents kpartx on the Compute node from automatically creating partitions on the device, which prevents unnecessary device mapper entries. For more information about configuration attributes, see the Configuring device mapper multipath guide.

4. Start the multipath daemon on all Controller and Compute nodes:

```
$ systemctl enable --now multipathd
```

5. Update the heat template to enable multipath:

```
parameter_defaults:
  NovaLibvirtVolumeUseMultipath:  true
  NovaComputeOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
  CinderVolumeOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
```

6. Optional: If you are using Block Storage (cinder) as an Image service (glance) back end, you must also complete the following steps:

   a. Add the following **GlanceApiOptVolumes** configuration to the heat template:

```
parameter_defaults:
  GlanceApiOptVolumes:
    - /etc/multipath.conf:/etc/multipath.conf:ro
    - /etc/multipath/:/etc/multipath/:rw
```

   b. Set the **ControllerExtraConfig** parameter in the following way:

```
parameter_defaults:
  ControllerExtraConfig:
    glance::config::api_config:
      default_backend/cinder_use_multipath:
        value: true
```

   Note

   Ensure that both **default_backend** and the **GlanceBackendID** heat template default value match.

7. For every configured back end, set **use_multipath_for_image_xfer** to **true**:

```
parameter_defaults:
  ExtraConfig:
    cinder::config::cinder_config:
      <backend>/use_multipath_for_image_xfer:
        value: true
```

8. Run the following command to update the overcloud:

```
$ openstack overcloud deploy
```

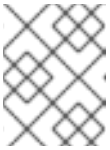
NOTE

When you run the **openstack overcloud deploy** command to install and configure multipath, you must pass all previous roles and environment files that you used to deploy the overcloud, such as **--templates**, **--roles-file**, **-e** for all environment files, and **--timeout**. Failure to pass all previous roles and environment files can cause problems with your overcloud deployment. For more information about using overcloud parameters, see Creating the Overcloud with the CLI Tools in the *Director Installation and Usage* guide.

9. Optional: If you have pre-existing workloads on instances in your deployment that you want to use multipath functionality, you must shelve and unshelve these instances:

```
$ nova shelve <instance>
$ nova unshelve <instance>
```

- Replace **<instance>** with the ID of the instance that you want to shelve and unshelve.



NOTE

Restarting the instance does not enable multipath. You must shelve and unshelve the instance.

2.12.3. Verifying multipath configuration

This procedure describes how to verify multipath configuration on new or existing overcloud deployments.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Create a VM.
2. Attach a non-encrypted volume to the VM.
3. Get the name of the Compute node that contains the instance:

```
$ nova show INSTANCE | grep OS-EXT-SRV-ATTR:host
```

Replace *INSTANCE* with the name of the VM that you booted.

4. Retrieve the virsh name of the instance:

```
$ nova show INSTANCE | grep instance_name
```

Replace *INSTANCE* with the name of the VM that you booted.

5. Get the IP address of the Compute node:

```
$ . stackrc
$ nova list | grep compute_name
```

Replace *compute_name* with the name from the output of the **nova show *INSTANCE*** command.

6. SSH into the Compute node that runs the VM:

```
$ ssh heat-admin@COMPUTE_NODE_IP
```

Replace *COMPUTE_NODE_IP* with the IP address of the Compute node.

7. Log in to the container that runs virsh:

```
$ podman exec -it nova_libvirt /bin/bash
```

8. Enter the following command on a Compute node instance to verify that it is using multipath in the cinder volume host location:

```
virsh domblklist VIRSH_INSTANCE_NAME | grep /dev/dm
```

Replace *VIRSH_INSTANCE_NAME* with the output of the **nova show *INSTANCE* | grep instance_name** command.

If the instance shows a value other than **/dev/dm-**, the connection is non-multipath and you must refresh the connection info with the **nova shelve** and **nova unshelve** commands:

```
$ nova shelve <instance>
$ nova unshelve <instance>
```

NOTE

If you have more than one type of back end, you must verify the instances and volumes on all back ends, because connection info that each back end returns might vary.

CHAPTER 3. PERFORMING BASIC OPERATIONS WITH THE BLOCK STORAGE SERVICE (CINDER)

Create and configure Block Storage volumes as the primary form of persistent storage for Compute instances in your overcloud. Create volumes, attach your volumes to instances, edit and resize your volumes, and modify volume ownership.

3.1. CREATING BLOCK STORAGE VOLUMES

Create volumes to provide persistent storage for instances that you launch with the Compute service (nova) in the overcloud.

IMPORTANT

The default maximum number of volumes you can create for a project is 10.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Click **Create Volume**, and edit the following fields:

Field	Description
Volume name	Name of the volume.
Description	Optional, short description of the volume.
Type	Optional volume type (see Section 2.3, "Group volume configuration with volume types"). If you have multiple Block Storage back ends, you can use this to select a specific back end. See Section 2.10, "Specifying back ends for volume creation".
Size (GB)	Volume size (in gigabytes). If you want to create an encrypted volume from an unencrypted image, you must ensure that the volume size is larger than the image size so that the encryption data does not truncate the volume data.

Field	Description
Availability Zone	Availability zones (logical server groups), along with host aggregates, are a common method for segregating resources within OpenStack. Availability zones are defined during installation. For more information about availability zones and host aggregates, see Creating and managing host aggregates in the *Configuring the Compute Service for Instance Creation* guide.

3. Specify a **Volume Source**:

Source	Description
No source, empty volume	The volume is empty and does not contain a file system or partition table.
Snapshot	Use an existing snapshot as a volume source. If you select this option, a new **Use snapshot as a source list** opens; you can then choose a snapshot from the list. If you want to create a new volume from a snapshot of an encrypted volume, you must ensure that the new volume is at least 1GB larger than the old volume. For more information about volume snapshots, see Section 4.1, "Creating, using, and deleting volume snapshots".
Image	Use an existing image as a volume source. If you select this option, a new **Use snapshot as a source list** opens; you can then choose an image from the list.
Volume	Use an existing volume as a volume source. If you select this option, a new **Use snapshot as a source list** opens; you can then choose a volume from the list.

4. Click **Create Volume**. After the volume is created, its name appears in the **Volumes** table.

You can also change the volume type later on. For more information, see Section 4.5, "Block Storage volume retyping".

3.2. EDITING A VOLUME NAME OR DESCRIPTION

Edit volume names and descriptions in the Dashboard (horizon).

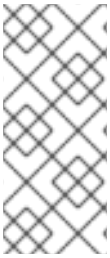
Prerequisites

- A successful undercloud installation. For more information, see Installing director.

- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the volume's **Edit Volume** button.
3. Edit the volume name or description as required.
4. Click **Edit Volume** to save your changes.



NOTE

To create an encrypted volume, you must first have a volume type configured specifically for volume encryption. In addition, you must configure both Compute and Block Storage services to use the same static key. For information about how to set up the requirements for volume encryption, see Section 2.7, "Block Storage service (cinder) volume encryption".

3.3. RESIZING (EXTENDING) A BLOCK STORAGE SERVICE VOLUME

Resize volumes to increase the storage capacity of the volumes.

NOTE

The ability to resize a volume in use is supported but is driver dependent. RBD is supported. You cannot extend in-use multi-attach volumes. For more information about support for this feature, contact Red Hat Support.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

1. List the volumes to retrieve the ID of the volume you want to extend:

```
$ cinder list
```

2. To resize the volume, run the following commands to specify the correct API microversion, then pass the volume ID and the new size (a value greater than the old one) as parameters:

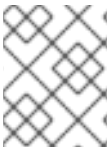
```
$ OS_VOLUME_API_VERSION=<API microversion>
$ cinder extend <volume ID> <size>
```

Replace <API microversion>, <volume ID>, and <size> with appropriate values. Use the following example as a guide:

```
$ OS_VOLUME_API_VERSION=3.42
$ cinder extend 573e024d-5235-49ce-8332-be1576d323f8 10
```

3.4. DELETING A BLOCK STORAGE SERVICE VOLUME

Use the Dashboard to delete volumes that you no longer require.



NOTE

A volume cannot be deleted if it has existing snapshots. For instructions on how to delete snapshots, see Section 4.1, "Creating, using, and deleting volume snapshots".

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. In the **Volumes** table, select the volume to delete.
3. Click **Delete Volumes**.

3.5. ALLOCATING VOLUMES TO MULTIPLE BACK ENDS

If the Block Storage service is configured to use multiple back ends, you can use configured volume types to specify where a volume should be created. For details, see Section 2.10, "Specifying back ends for volume creation".

The Block Storage service will automatically choose a back end if you do not specify one during volume creation. Block Storage sets the first defined back end as a default; this back end will be used until it runs out of space. At that point, Block Storage will set the second defined back end as a default, and so on.

If this is not suitable for your needs, you can use the filter scheduler to control how Block Storage should select back ends. This scheduler can use different filters to triage suitable back ends, such as:

AvailabilityZoneFilter
: Filters out all back ends that do not meet the availability zone requirements of the requested volume.

CapacityFilter
: Selects only back ends with enough space to accommodate the volume.

CapabilitiesFilter
: Selects only back ends that can support any specified settings in the volume.

InstanceLocality
: Configures clusters to use volumes local to the same node.

Prerequisites

- A successful undercloud installation. For more information, see the Director Installation and Usage guide.

Procedure

1. Add an environment file to your deployment command that contains the following parameters:

```
parameter_defaults:
  ControllerExtraConfig: # 1
    cinder::config::cinder_config:
      DEFAULT/scheduler_default_filters:
        value: 'AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter,InstanceLocality'
```

1 You can also add the **ControllerExtraConfig:** hook and its nested sections to the **parameter_defaults:** section of an existing environment file.

3.6. ATTACHING A VOLUME TO AN INSTANCE

Instances can use a volume for persistent storage. A volume can only be attached to one instance at a time. For more information about instances, see Image service in the *Creating and Managing Images* guide.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the **Edit Attachments** action. If the volume is not attached to an instance, the **Attach To Instance** drop-down list is visible.
3. From the **Attach To Instance** list, select the instance to which you want to attach the volume.
4. Click **Attach Volume**.

3.7. DETACHING A VOLUME FROM AN INSTANCE

Instances can use a volume for persistent storage. A volume can only be attached to one instance at a time. For more information about instances, see Image service in the *Creating and Managing Images* guide.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the volume's **Manage Attachments** action. If the volume is attached to an instance, the instance's name is displayed in the **Attachments** table.
3. Click **Detach Volume** in this and the next dialog screen.

3.8. CONFIGURING READ-ONLY VOLUMES

A volume can be marked read-only to protect its data from being accidentally overwritten or deleted.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

- Set the volume to read-only:

```
# cinder readonly-mode-update <VOLUME-ID> true
```

- Set a read-only volume back to read-write:

```
# cinder readonly-mode-update <VOLUME-ID> false
```

3.9. CHANGING A VOLUME OWNER WITH THE CLI

To change a volume's owner, you will have to perform a volume transfer. A volume transfer is initiated by the volume's owner, and the volume's change in ownership is complete after the transfer is accepted by the volume's new owner.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Log in as the volume's current owner.
2. List the available volumes:

```
# cinder list
```

3. Initiate the volume transfer:

```
# cinder transfer-create VOLUME
```

Where **VOLUME** is the name or **ID** of the volume you wish to transfer. For example,

```
+------------+--------------------------------------+
|  Property  |                Value                 |
+------------+--------------------------------------+
|  auth_key  |           f03bf51ce7ead189           |
| created_at |      2014-12-08T03:46:31.884066      |
|     id     | 3f5dc551-c675-4205-a13a-d30f88527490 |
|    name    |                None                  |
| volume_id  | bcf7d015-4843-464c-880d-7376851ca728 |
+------------+--------------------------------------+
```

The **cinder transfer-create** command clears the ownership of the volume and creates an **id** and **auth_key** for the transfer. These values can be given to, and used by, another user to accept the transfer and become the new owner of the volume.

4. The new user can now claim ownership of the volume. To do so, the user should first log in from the command line and run:

```
# cinder transfer-accept TRANSFERID TRANSFERKEY
```

Where **TRANSFERID** and **TRANSFERKEY** are the **id** and **auth_key** values returned by the **cinder transfer-create** command, respectively. For example,

```
# cinder transfer-accept 3f5dc551-c675-4205-a13a-d30f88527490 f03bf51ce7ead189
```

NOTE

You can view all available volume transfers using:

```
# cinder transfer-list
```

3.10. CHANGING A VOLUME OWNER WITH THE DASHBOARD

To change a volume's owner, you will have to perform a volume transfer. A volume transfer is initiated by the volume's owner, and the volume's change in ownership is complete after the transfer is accepted by the volume's new owner.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. As the volume owner in the dashboard, select **Projects > Volumes**.
2. In the **Actions** column of the volume to transfer, select **Create Transfer**.
3. In the **Create Transfer** dialog box, enter a name for the transfer and click **Create Volume Transfer**.
   The volume transfer is created, and in the **Volume Transfer** screen you can capture the **transfer ID** and the **authorization key** to send to the recipient project.

   Click the **Download transfer credentials** button to download a **.txt** file containing the **transfer name**, **transfer ID**, and **authorization key**.

   NOTE

   The authorization key is available only in the **Volume Transfer** screen. If you lose the authorization key, you must cancel the transfer and create another transfer to generate a new authorization key.

4. Close the **Volume Transfer** screen to return to the volume list.
   The volume status changes to **awaiting-transfer** until the recipient project accepts the transfer

Accept a volume transfer from the dashboard

1. As the recipient project owner in the dashboard, select **Projects > Volumes**.
2. Click **Accept Transfer**.
3. In the **Accept Volume Transfer** dialog box, enter the **transfer ID** and the **authorization key** that you received from the volume owner and click **Accept Volume Transfer**.
   The volume now appears in the volume list for the active project.

CHAPTER 4. PERFORMING ADVANCED OPERATIONS WITH THE BLOCK STORAGE SERVICE (CINDER)

Block Storage volumes form persistent storage for Compute instances in your overcloud. Configure advanced features of your volumes, for example, using volume snapshots, migrating volumes, retyping volumes, and configuring multipath.

4.1. CREATING, USING, AND DELETING VOLUME SNAPSHOTS

You can preserve the state of a volume at a specific point in time by creating a volume snapshot. You can then use the snapshot to clone new volumes.

NOTE

Volume backups are different from snapshots. Backups preserve the data contained in the volume, whereas snapshots preserve the state of a volume at a specific point in time. You cannot delete a volume if it has existing snapshots. Volume backups prevent data loss, whereas snapshots facilitate cloning.

For this reason, snapshot back ends are typically colocated with volume back ends to minimize latency during cloning. By contrast, a backup repository is usually located in a different location, for example, on a different node, physical storage, or even geographical location in a typical enterprise deployment. This is to protect the backup repository from any damage that might occur to the volume back end.

For more information about volume backups, see the *Block Storage Backup Guide* .

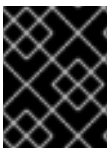
Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the target volume's **Create Snapshot** action.
3. Provide a **Snapshot Name** for the snapshot and click **Create a Volume Snapshot** The **Volume Snapshots** tab displays all snapshots.

You can clone new volumes from a snapshot when it appears in the **Volume Snapshots** table. To do so, select the snapshot's **Create Volume** action. For more information about volume creation, see Section 3.1, "Creating Block Storage volumes" .



IMPORTANT

If you want to create a new volume from a snapshot of an encrypted volume, ensure that the new volume is at least 1GB larger than the old volume.

To delete a snapshot, select its **Delete Volume Snapshot** action.

If your OpenStack deployment uses a Red Hat Ceph back end, see Section 4.10, "Protected and unprotected snapshots in a Red Hat Ceph Storage back end" for more information about snapshot security and troubleshooting.

NOTE

For RADOS block device (RBD) volumes for the Block Storage service (cinder) that are created from snapshots, you can use the **CinderRbdFlattenVolumeFromSnapshot** heat parameter to flatten and remove the dependency on the snapshot. When you set **CinderRbdFlattenVolumeFromSnapshot** to **true**, the Block Storage service flattens RBD volumes and removes the dependency on the snapshot and also flattens all future snapshots. The default value is false, which is also the default value for the cinder RBD driver.

Be aware that flattening a snapshot removes any potential block sharing with the parent and results in larger snapshot sizes on the back end and increases the time for snapshot creation.

4.2. RESTORING A VOLUME FROM A SNAPSHOT

You can recover the most recent snapshot of a volume. This means that you can perform an in-place revert of the volume data to its most recent snapshot.

WARNING

The ability to recover the most recent snapshot of a volume is supported but is driver dependent. The correct implementation of this feature is driver assisted. For more information about support for this feature, contact your driver vendor.

Limitations

- There might be limitations to using the revert-to-snapshot feature with multi-attach volumes. Check whether such limitations apply before you use this feature.
- You cannot revert a volume that you resize (extend) after you take a snapshot.
- You cannot use the revert-to-snapshot feature on an attached or in-use volume.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

- Block Storage (cinder) API microversion 3.40 or later.
- You must have created at least one snapshot for the volume.

Procedure

1. Log in to the undercloud as the **stack** user.
2. Source the **overcloudrc** file:

```
[stack@undercloud ~] $ source overcloudrc
```

3. Detach your volume:

```
$ nova volume-detach <instance_id> <vol_id>
```

Replace <instance_id> and <vol_id> with the IDs of the instance and volume that you want to revert.

4. Locate the ID or name of the snapshot that you want to revert. You can only revert the latest snapshot.

```
$ cinder snapshot-list
```

5. Revert the snapshot:

```
$ cinder --os-volume-api-version=3.40 revert-to-snapshot  <snapshot_id or snapshot_name>
```

Replace <snapshot_id or snapshot_name> with the ID or the name of the snapshot.

6. Optional: You can use the **cinder snapshot-list** command to check that the volume you are reverting is in a reverting state.

```
$  cinder snapshot-list
```

7. Reattach the volume:

```
$  nova volume-attach <instance_id> <vol_id>
```

Replace <instance_id> and <vol_id> with the IDs of the instance and volume that you reverted.

Verification

- To check that the procedure is successful, you can use the **cinder list** command to verify that the volume you reverted is now in the available state.

```
$ cinder list
```

Note

If you used Block Storage (cinder) as a bootable root volume, you cannot use the revert-to-snapshot feature on that volume because it is not in the available state. To use this feature, the instance must have been booted with the **delete_on_termination=false** (default)

property to preserve the boot volume if the instance is terminated. When you want to revert to a snapshot, you must first delete the initial instance so that the volume is available. You can then revert it and create a new instance from the volume.

4.3. UPLOADING A VOLUME TO THE IMAGE SERVICE (GLANCE)

You can upload an existing volume as an image to the Image service directly.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the target volume's **Upload to Image** action.
3. Provide an **Image Name** for the volume and select a **Disk Format** from the list.
4. Click **Upload**.

To view the uploaded image, select **Project > Compute > Images** The new image appears in the **Images** table. For information on how to use and configure images, see Image service in the *Creating and Managing Images* guide.

4.4. MOVING VOLUMES BETWEEN BACK ENDS

There are many reasons to move volumes from one storage back end to another, such as:

- To retire storage systems that are no longer supported.
- To change the storage class or tier of a volume.
- To change the availability zone of a volume.

With the Block Storage service (cinder), you can move volumes between back ends in the following ways:

- Retype: The default policy allows volume owners and administrators to retype a volume. The retype operation is the most common way to move volumes between back ends.
- Migrate: The default policy allows only administrators to migrate a volume. Volume migration is reserved for specific use cases, because it is restrictive and requires a clear understanding about how deployments work. For more information, see Section 4.8, "Migrating a volume between back ends with the CLI".

Restrictions

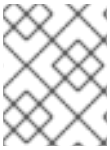
Red Hat supports moving volumes between back ends within and across availability zones (AZs), but with the following restrictions:

- Volumes must have either available or in-use status to move.
- Support for in-use volumes is driver dependent.
- Volumes cannot have snapshots.
- Volumes cannot belong to a group or consistency group.

4.4.1. Moving available volumes

You can move available volumes between all back ends, but performance depends on the back ends that you use. Many back ends support assisted migration. For more information about back end support for assisted migration, contact the vendor.

Assisted migration works with both volume retype and volume migration. With assisted migration, the back end optimizes the movement of the data from the source back end to the destination back end, but both back ends must be from the same vendor.

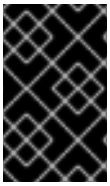


NOTE

Red Hat supports back-end assisted migrations only with multi-pool back ends or when you use the cinder migrate operation for single-pool back ends, such as RBD.

When assisted migrations between back ends are not possible, the Block Storage service performs a generic volume migration.

Generic volume migration requires volumes on both back ends to be connected before the Block Storage (cinder) service moves data from the source volume to the Controller node and from the Controller node to the destination volume. The Block Storage service seamlessly performs the process regardless of the type of storage from the source and destination back ends.



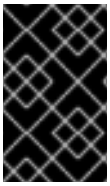
IMPORTANT

Ensure that you have adequate bandwidth before you perform a generic volume migration. The duration of a generic volume migration is directly proportional to the size of the volume, which makes the operation slower than assisted migration.

4.4.2. Moving in-use volumes

There is no optimized or assisted option for moving in-use volumes. When you move in-use volumes, the Compute service (nova) must use the hypervisor to transfer data from a volume in the source back end to a volume in the destination back end. This requires coordination with the hypervisor that runs the instance where the volume is in use.

The Block Storage service (cinder) and the Compute service work together to perform this operation. The Compute service manages most of the work, because the data is copied from one volume to another through the Compute node.



IMPORTANT

Ensure that you have adequate bandwidth before you move in-use volumes. The duration of this operation is directly proportional to the size of the volume, which makes the operation slower than assisted migration.

Restrictions

- In-use multi-attach volumes cannot be moved while they are attached to more than one nova instance.
- Non block devices are not supported, which limits storage protocols on the target back end to be iSCSI, Fibre Channel (FC), and RBD.

4.5. BLOCK STORAGE VOLUME RETYPING

Volume retyping is the standard way to move volumes from one back end to another. The operation requires the administrator to define the appropriate volume types for the different back ends. The default policy allows volume owners and administrators to retype volumes.

When you retype a volume, you apply a volume type and its settings to an already existing volume. For more information about volume types, see Section 2.3, “Group volume configuration with volume types” .

You can retype a volume provided that the extra specs of the new volume type can be applied to the existing volume. You can retype a volume to apply pre-defined settings or storage attributes to an existing volume, such as:

- To move the volume to a different back end.
- To change the storage class or tier of a volume.
- To enable or disable features such as replication.

Retyping a volume does not necessarily mean that you must move the volume from one back end to another. However, there are circumstances in which you must move a volume to complete a retype:

- The new volume type has a different **volume_backend_name** defined.
- The **volume_backend_name** of the current volume type is undefined, and the volume is stored in a different back end than the one specified by the **volume_backend_name** of the new volume type.

Moving a volume from one back end to another can require extensive time and resources. Therefore, when a retype requires moving data, the Block Storage service does not move data by default. The operation fails unless it is explicitly allowed by specifying a migration policy as part of the retype request. For more information, see Section 4.5.2, “Retyping a volume from the command line” .

Restrictions

- You cannot retype all volumes. For more information about moving volumes between back ends, see Section 4.4, “Moving volumes between back ends” .
- Unencrypted volumes cannot be retyped to encrypted volume types, but encrypted volumes can be retyped to unencrypted ones.

- Retyping an unencrypted volume to an encrypted volume of the same size is not supported, because encrypted volumes require additional space to store encryption data. For more information about encrypting unencrypted volumes, see Encrypting unencrypted volumes.
- Users with no administrative privileges can only retype volumes that they own.

4.5.1. Retyping a volume from the dashboard UI

Use the dashboard UI to retype a volume.

IMPORTANT

Retyping an unencrypted volume to an encrypted volume of the same size is not supported, because encrypted volumes require additional space to store encryption data. For more information about encrypting unencrypted volumes, see Encrypting unencrypted volumes.

Prerequisites

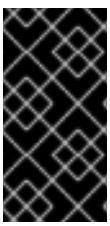
- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Project > Compute > Volumes**
2. In the **Actions** column of the volume you want to migrate, select **Change Volume Type**.
3. In the **Change Volume Type** dialog, select the target volume type and define the new back end from the **Type** list.
4. If you are migrating the volume to another back end, select **On Demand** from the **Migration Policy** list. For more information, see Section 4.4, “Moving volumes between back ends” .
5. Click **Change Volume Type** to start the migration.

4.5.2. Retyping a volume from the command line

Similar to the dashboard UI procedure, you can retype a volume from the command line.



IMPORTANT

Retyping an unencrypted volume to an encrypted volume of the same size is not supported, because encrypted volumes require additional space to store encryption data. For more information about encrypting unencrypted volumes, see Encrypting unencrypted volumes.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Enter the following command to retype a volume:

```
$ cinder retype <volume name or id> <new volume type name>
```

2. If the retype operation requires moving the volume from one back end to another, the Block Storage service requires a specific flag:

```
$ cinder retype --migration-policy on-demand <volume name or id> <new volume type name>
```

4.6. ATTACH A VOLUME TO MULTIPLE INSTANCES

Volume multi-attach gives multiple instances simultaneous read/write access to a Block Storage volume. This feature requires support in the Block Storage (cinder) back end. The Ceph RBD driver is supported.

The Block Storage back end must support multi-attach. For information about supported back ends, contact Red Hat Support.

WARNING

You must use a multi-attach or cluster-aware file system to manage write operations from multiple instances. Failure to do so causes data corruption.

WARNING

To use the optional multi-attach feature, your cinder driver must support it. For more information about the certification for your vendor plugin, see the following locations:

- https://access.redhat.com/articles/1535373#Cinder
- https://access.redhat.com/ecosystem/search/#/category/Software?sort=sortTitle%20asc&softwareCategories=Storage&ecosystem=Red%20Hat%20C

Contact Red Hat support to verify that multi-attach is supported for your vendor plugin.

WARNING

Restrictions for multi-attach volumes:

- Live migration of multi-attach volumes is not available.
- Volume encryption is not supported with multi-attach volumes.
- Retyping an attached volume from a multi-attach type to non-multi-attach type or non-multi-attach to multi-attach is not possible.
- Read-only multi-attach is not supported.

4.6.1. Creating a multi-attach volume type

To attach a volume to multiple instances, set the **multiattach** flag to **<is> True** in the volume extra specs. When you create a multi-attach volume type, the volume inherits the flag and becomes a multi-attach volume.

NOTE

By default, creating a new volume type is an admin-only operation.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Run the following commands to create a multi-attach volume type:

```
$ cinder type-create multiattach
$ cinder type-key multiattach set multiattach="<is> True"
```

NOTE

This procedure creates a volume on any back end that supports multiattach. Therefore, if there are two back ends that support multiattach, the scheduler decides which back end to use based on the available space at the time of creation.

2. Run the following command to specify the back end:

```
$ cinder type-key multiattach set volume_backend_name=<backend_name>
```

4.6.2. Multi-attach volume retyping

You can retype a volume to be multi-attach capable or retype a multi-attach capable volume to make it incapable of attaching to multiple instances. However, you can retype a volume only when it is not in use and its status is **available**.

When you attach a multi-attach volume, some hypervisors require special considerations, such as when you disable caching. Currently, there is no way to safely update an attached volume while keeping it attached the entire time. Retyping fails if you attempt to retype a volume that is attached to multiple instances.

4.6.3. Creating a multi-attach volume

After you create a multi-attach volume type, create a multi-attach volume.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

Procedure

1. Run the following command to create a multi-attach volume:

```
$ cinder create <volume_size> --name <volume_name> --volume-type multiattach
```

2. Run the following command to verify that a volume is multi-attach capable. If the volume is multi-attach capable, the **multiattach** field equals **True**.

```
$ cinder show <vol_id> | grep multiattach

| multiattach | True |
```

You can now attach the volume to multiple instances. For information, see Attaching a volume to an instance.

4.7. MIGRATING A VOLUME BETWEEN BACK ENDS WITH THE DASHBOARD

With the Block Storage service (cinder) you can migrate volumes between back ends within and across availability zones (AZs). This is the least common way to move volumes from one back end to another. The default policy allows only administrators to migrate volumes. Do not change the default policy.

In highly customized deployments or in situations in which you must retire a storage system, an administrator can migrate volumes. In both use cases, multiple storage systems share the same **volume_backend_name**, or it is undefined.

Restrictions

- The volume cannot be replicated.
- The destination back end must be different from the current back end of the volume.

- The existing volume type must be valid for the new back end, which means the following must be true:
  - Volume type must not have the **backend_volume_name** defined in its extra specs, or both Block Storage back ends must be configured with the same **backend_volume_name**.
  - Both back ends must support the same features configured in the volume type, such as support for thin provisioning, support for thick provisioning, or other feature configurations.

NOTE

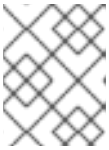
Moving volumes from one back end to another might require extensive time and resources. For more information, see Section 4.4, "Moving volumes between back ends" .

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Access to the Red Hat OpenStack Platform (RHOSP) Dashboard (horizon). For more information, see Overcloud deployment output.

Procedure

1. In the dashboard, select **Admin > Volumes**.
2. In the **Actions** column of the volume you want to migrate, select **Migrate Volume**.
3. In the **Migrate Volume** dialog, select the target host from the **Destination Host** drop-down list.



NOTE

To bypass any driver optimizations for the host migration, select the **Force Host Copy** check box.

4. Click **Migrate** to start the migration.

4.8. MIGRATING A VOLUME BETWEEN BACK ENDS WITH THE CLI

With the Block Storage service (cinder) you can migrate volumes between back ends within and across availability zones (AZs). This is the least common way to move volumes from one back end to another. The default policy allows only administrators to migrate volumes. Do not change the default policy.

In highly customized deployments or in situations in which you must retire a storage system, an administrator can migrate volumes. In both use cases, multiple storage systems share the same **volume_backend_name**, or it is undefined.

Restrictions

- The volume cannot be replicated.
- The destination back end must be different from the current back end of the volume.

- The existing volume type must be valid for the new back end, which means the following must be true:
  - Volume type must not have the **backend_volume_name** defined in its extra specs, or both Block Storage back ends must be configured with the same **backend_volume_name**.
  - Both back ends must support the same features configured in the volume type, such as support for thin provisioning, support for thick provisioning, or other feature configurations.

NOTE

Moving volumes from one back end to another might require extensive time and resources. For more information, see Section 4.4, “Moving volumes between back ends” .

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

1. Enter the following command to retrieve the name of the destination back end:

```
$ cinder get-pools --detail

Property                | Value

...

| name                  | localdomain@lvmdriver-1#lvmdriver-1
| pool_name             | lvmdriver-1

...

| volume_backend_name   | lvmdriver-1

...

Property                | Value

...
                                    |
| name                  | localdomain@lvmdriver-2#lvmdriver-1
| pool_name             | lvmdriver-1

...

| volume_backend_name   | lvmdriver-1

...
```

The back end names take the form **host@volume_backend_name#pool**.

In the example output, there are two LVM back ends exposed in the Block Storage service: **localdomain@lvmdriver-1#lvmdriver-1** and **localdomain@lvmdriver-2#lvmdriver-1**. Notice that both back ends share the same **volume_backend_name**, **lvmdriver-1**.

2. Enter the following command to migrate a volume from one back end to another:

```
$ cinder migrate <volume id or name> <new host>
```

4.9. ENCRYPTING UNENCRYPTED VOLUMES

To encrypt an unencrypted volume, you must either back up the unencrypted volume and then restore it to a new encrypted volume, or create a glance image from the unencrypted volume and then create a new encrypted volume from the image.

Prerequisites

- An unencrypted volume that you want to encrypt.

Procedure

1. If the **cinder-backup** service is available, back up the current unencrypted volume:

```
cinder backup-create <unencrypted_volume>
```

   - Replace **<unencrypted_volume>** with the name or ID of the unencrypted volume.

2. Create a new encrypted volume:

```
cinder create <encrypted_volume_size> --volume-type <encrypted_volume_type>
```

   - Replace **<encrypted_volume_size>** with the size of the new volume in GB. This value must be larger than the size of the unencrypted volume by 1GB to accommodate the encryption metadata.
   - Replace **<encrypted_volume_type>** with the encryption type that you require.

3. Restore the backup of the unencrypted volume to the new encrypted volume:

```
cinder backup-restore <backup> --volume <encrypted_volume>
```

   - Replace **<backup>** with the name or ID of the unencrypted volume backup.
   - Replace **<encrypted_volume>** with the ID of the new encrypted volume.

If the **cinder-backup** service is unavailable, use the **upload-to-image** command to create an image of the unencrypted volume, and then create a new encrypted volume from the image.

1. Create a glance image of the unencrypted volume:

```
cinder upload-to-image <unencrypted_volume> <new_image>
```

   - Replace **<unencrypted_volume>** with the name or ID of the unencrypted volume.
   - Replace **<new_image>** with a name for the new image.

2. Create a new volume from the image that is 1GB larger than the image:

```
cinder volume create --size <size> --volume-type luks --image <new_image>
<encrypted_volume_name>
```

- Replace **<size>** with the size of the new volume. This value must be 1GB larger than the size of the old unencrypted volume.
- Replace **<new_image>** with the name of the image that you created from the unencrypted volume.
- Replace **<encrypted_volume_name>** with a name for the new encrypted volume.

4.10. PROTECTED AND UNPROTECTED SNAPSHOTS IN A RED HAT CEPH STORAGE BACK END

When you use Red Hat Ceph Storage (RHCS) as a back end for your Red Hat OpenStack Platform (RHOSP) deployment, you can set a snapshot to **protected** in the back end. Do not delete protected snapshots through the RHOSP dashboard or the **cinder snapshot-delete** command because deletion fails.

When this occurs, set the snapshot to **unprotected** in the RHCS back end first. You can then delete the snapshot through RHOSP as normal.

Additional resources

- For more information about protecting snapshots, see Protecting a block device snapshot and Unprotecting a block device snapshot.

CHAPTER 5. CONFIGURING THE OBJECT STORAGE SERVICE (SWIFT)

Red Hat OpenStack Platform Object Storage (swift) stores its objects, or data, in containers. Containers are similar to directories in a file system although they cannot be nested. Containers provide an easy way for users to store any kind of unstructured data. For example, objects can include photos, text files, or images. Stored objects are not compressed.

5.1. OBJECT STORAGE RINGS

The Object Storage service (swift) uses a data structure called the ring to distribute partition space across the cluster. This partition space is core to the data durability engine in the Object Storage service. With rings, the Object Storage service can quickly and easily synchronize each partition across the cluster.

Rings contain information about Object Storage partitions and how partitions are distributed among the different nodes and disks. When any Object Storage component interacts with data, a quick lookup is performed locally in the ring to determine the possible partitions for each object.

The Object Storage service has three rings to store the following types of data:

- Account information
- Containers, to facilitate organizing objects under an account
- Object replicas

5.1.1. Rebalancing Object Storage rings

When you change the Object Storage (swift) environment by adding or removing storage capacity, nodes, or disks, you must rebalance the rings. You can run the **openstack overcloud deploy** command to rebalance the rings, but this method redeploys the entire overcloud. This can be cumbersome, especially if you have a large overcloud. Instead, rebalance the rings manually.

Procedure

1. Log in to the undercloud node as the **stack** user.
2. Source the **stackrc** credentials file.
3. Run the **swift_ring_rebalance.yaml** Ansible playbook:

```
ansible-playbook -i /usr/bin/tripleo-ansible-inventory
/usr/share/openstack-tripleo-common/playbooks/swift_ring_rebalance.yaml
```

5.1.2. Checking cluster health

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

- Auditors constantly re-read database and object files and compare them using checksums to make sure there is no silent bit-rot. Any database or object file that no longer matches its checksum is quarantined and becomes unreadable on that node. The replicators then copy one of the other replicas to make the local copy available again.

- Objects and files can disappear when you replace disks or nodes or when objects are quarantined. When this happens, replicators copy missing objects or database files to one of the other nodes.

The Object Storage service includes a tool called **swift-recon** that collects data from all nodes and checks the overall cluster health.

Procedure

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

```
[root@overcloud-controller-2 ~]# sudo podman exec -it -u swift swift_object_server
/usr/bin/swift-recon -arqlT --md5

===============================================================================-
-> Starting reconnaissance on 3 hosts (object)
===============================================================================
[2018-12-14 14:55:47] Checking async pendings
[async_pending] - No hosts returned valid data.
===============================================================================
[2018-12-14 14:55:47] Checking on replication
[replication_failure] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported: 3
[replication_success] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported: 3
[replication_time] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported: 3
[replication_attempted] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported:
3
Oldest completion was 2018-12-14 14:55:39 (7 seconds ago) by 172.16.3.186:6000.
Most recent completion was 2018-12-14 14:55:42 (4 seconds ago) by 172.16.3.174:6000.
===============================================================================
[2018-12-14 14:55:47] Checking load averages
[5m_load_avg] low: 1, high: 2, avg: 2.1, total: 6, Failed: 0.0%, no_result: 0, reported: 3
[15m_load_avg] low: 2, high: 2, avg: 2.6, total: 7, Failed: 0.0%, no_result: 0, reported: 3
[1m_load_avg] low: 0, high: 0, avg: 0.8, total: 2, Failed: 0.0%, no_result: 0, reported: 3
===============================================================================
[2018-12-14 14:55:47] Checking ring md5sums
3/3 hosts matched, 0 error[s] while checking hosts.
===============================================================================
[2018-12-14 14:55:47] Checking swift.conf md5sum
3/3 hosts matched, 0 error[s] while checking hosts.
===============================================================================
[2018-12-14 14:55:47] Checking quarantine
[quarantined_objects] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported: 3
[quarantined_accounts] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0, reported:
3
[quarantined_containers] low: 0, high: 0, avg: 0.0, total: 0, Failed: 0.0%, no_result: 0,
reported: 3
===============================================================================
[2018-12-14 14:55:47] Checking time-sync
3/3 hosts matched, 0 error[s] while checking hosts.
===============================================================================
```

3. Optional: Use the **--all** option to return additional output.
   This command queries all servers on the ring for the following data:

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

5.1.3. Increasing ring partition power

The ring power determines the partition to which a resource, such as an account, container, or object, is mapped. The partition is included in the path under which the resource is stored in a back end file system. Therefore, changing the partition power requires relocating resources to new paths in the back end file systems.

In a heavily populated cluster, a relocation process is time consuming. To avoid downtime, relocate resources while the cluster is still operating. You must do this without temporarily losing access to data or compromising the performance of processes, such as replication and auditing. For assistance with increasing ring partition power, contact Red Hat support.

5.1.4. Custom rings

As technology advances and demands for storage capacity increase, creating custom rings is a way to update existing Object Storage clusters.

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

Automation might not keep pace with future technology trends. For example, some older Object Storage clusters in use today originated before SSDs were available.

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

5.2. CUSTOMIZE THE OBJECT STORAGE SERVICE

Depending on the requirements of your Red Hat OpenStack Platform (RHOSP) environment, you might want to customize some of the default settings of the Object Storage service (swift) to optimize your deployment performance.

5.2.1. Configuring fast-post

By default, the Object Storage service copies an object whole whenever any part of its metadata changes. You can prevent this by using the *fast-post* feature. The fast-post feature saves time when you change the content types of multiple large objects.

To enable the fast-post feature, disable the **object_post_as_copy** option on the Object Storage proxy service.

Procedure

1. Edit **swift_params.yaml**:

```
cat > swift_params.yaml << EOF
parameter_defaults:
    ExtraConfig:
      swift::proxy::copy::object_post_as_copy: False
EOF
```

2. Include the parameter file when you deploy or update the overcloud:

```
openstack overcloud deploy [... previous args ...] -e swift_params.yaml
```

5.2.2. Enabling at-rest encryption

By default, objects uploaded to the Object Storage service (swift) are unencrypted. Because of this, it is possible to access objects directly from the file system. This can present a security risk if disks are not properly erased before they are discarded.

You can use OpenStack Key Manager (barbican) to encrypt at-rest swift objects. For more information, see Encrypt at-rest swift objects.

5.2.3. Deploying a standalone Object Storage cluster

You can use the composable role concept to deploy a standalone Object Storage service (swift) cluster with the bare minimum of additional services (for example Keystone, HAProxy). The Creating a **roles_data** File section has information on roles.

Procedure

1. Copy the **roles_data.yaml** from **/usr/share/openstack-tripleo-heat-templates**.
2. Edit the new file.
3. Remove unneeded controller roles, for example Aodh*, Ceilometer*, Ceph*, Cinder*, Glance*, Heat*, Ironic*, Manila*, Mistral*, Nova*, Octavia*, Swift*.
4. Locate the ObjectStorage role within **roles_data.yaml**.
5. Copy this role to a new role within that same file and name it **ObjectProxy**.
6. Replace **SwiftStorage** with **SwiftProxy** in this role.
   The example **roles_data.yaml** file below shows sample roles.

```
- name: Controller
  description: |
 Controller role that has all the controller services loaded and handles
 Database, Messaging and Network functions.
  CountDefault: 1
  tags:
```

```
  - primary
  - controller
   networks:
  - External
  - InternalApi
  - Storage
  - StorageMgmt
  - Tenant
   HostnameFormatDefault: '%stackname%-controller-%index%'
   ServicesDefault:
  - OS::TripleO::Services::AuditD
  - OS::TripleO::Services::CACerts
  - OS::TripleO::Services::CertmongerUser
  - OS::TripleO::Services::Clustercheck
  - OS::TripleO::Services::Docker
  - OS::TripleO::Services::Ec2Api
  - OS::TripleO::Services::Etcd
  - OS::TripleO::Services::HAproxy
  - OS::TripleO::Services::Keepalived
  - OS::TripleO::Services::Kernel
  - OS::TripleO::Services::Keystone
  - OS::TripleO::Services::Memcached
  - OS::TripleO::Services::MySQL
  - OS::TripleO::Services::MySQLClient
  - OS::TripleO::Services::Ntp
  - OS::TripleO::Services::Pacemaker
  - OS::TripleO::Services::RabbitMQ
  - OS::TripleO::Services::Securetty
  - OS::TripleO::Services::Snmp
  - OS::TripleO::Services::Sshd
  - OS::TripleO::Services::Timezone
  - OS::TripleO::Services::TripleoFirewall
  - OS::TripleO::Services::TripleoPackages
  - OS::TripleO::Services::Vpp

- name: ObjectStorage
   CountDefault: 1
   description: |
  Swift Object Storage node role
   networks:
  - InternalApi
  - Storage
  - StorageMgmt
   disable_upgrade_deployment: True
   ServicesDefault:
  - OS::TripleO::Services::AuditD
  - OS::TripleO::Services::CACerts
  - OS::TripleO::Services::CertmongerUser
  - OS::TripleO::Services::Collectd
  - OS::TripleO::Services::Docker
  - OS::TripleO::Services::FluentdClient
  - OS::TripleO::Services::Kernel
  - OS::TripleO::Services::MySQLClient
  - OS::TripleO::Services::Ntp
  - OS::TripleO::Services::Securetty
  - OS::TripleO::Services::SensuClient
```

```
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages

  - name: ObjectProxy
    CountDefault: 1
    description: |
   Swift Object proxy node role
    networks:
   - InternalApi
   - Storage
   - StorageMgmt
    disable_upgrade_deployment: True
    ServicesDefault:
   - OS::TripleO::Services::AuditD
   - OS::TripleO::Services::CACerts
   - OS::TripleO::Services::CertmongerUser
   - OS::TripleO::Services::Collectd
   - OS::TripleO::Services::Docker
   - OS::TripleO::Services::FluentdClient
   - OS::TripleO::Services::Kernel
   - OS::TripleO::Services::MySQLClient
   - OS::TripleO::Services::Ntp
   - OS::TripleO::Services::Securetty
   - OS::TripleO::Services::SensuClient
   - OS::TripleO::Services::Snmp
   - OS::TripleO::Services::Sshd
   - OS::TripleO::Services::SwiftRingBuilder
   - OS::TripleO::Services::SwiftProxy
   - OS::TripleO::Services::Timezone
   - OS::TripleO::Services::TripleoFirewall
   - OS::TripleO::Services::TripleoPackages
```

7. Deploy the overcloud with your regular **openstack deploy** command, including the new roles.

```
openstack overcloud deploy --templates -r roles_data.yaml -e [...]
```

5.2.4. Using external SAN disks

By default, when the Red Hat OpenStack Platform (RHOSP) director deploys the Object Storage service (swift), Object Storage is configured and optimized to use independent local disks. This configuration ensures that the workload is distributed across all disks, which helps minimize performance impacts during node failure or other system issues.

In similar performance-impacting events, an environment that uses a single SAN can experience decreased performance across all LUNs. The Object Storage service cannot mitigate performance issues in an environment that uses SAN disks.

Therefore, Red Hat strongly recommends that you use additional local disks for Object Storage instead to meet performance and disk space requirements. For more information, see Configuration recommendations for the Object Storage service (swift) in the *Deployment Recommendations for*

Specific Red Hat OpenStack Platform Services guide.

Using an external SAN for Object Storage requires evaluation on a per-case basis. For more information, contact Red Hat Support.

IMPORTANT

If you choose to use an external SAN for Object Storage, be aware of the following conditions:

- The Object Storage service stores telemetry data and Image service (glance) images by default. Glance images require more disk space, but from a performance perspective, the impact of storing glance images impacts performance less than storing telemetry data. Storing and processing telemetry data requires increased performance. Red Hat does not provide support for issues related to performance that result from using an external SAN for Object Storage.
- Red Hat does not provide support for issues that arise outside of the core Object Storage service offering. For support with high availability and performance, contact your storage vendor.
- Red Hat does not test SAN solutions with the Object Storage service. For more information about compatibility, guidance, and support for third-party products, contact your storage vendor.
- Red Hat recommends that you evaluate and test performance demands with your deployment. To confirm that your SAN deployment is tested, supported, and meets your performance requirements, contact your storage vendor.

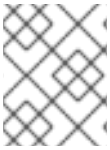
Procedure

- This template is an example of how to use two devices (**/dev/mapper/vdb** and **/dev/mapper/vdc**) for Object Storage:

```
parameter_defaults:
  SwiftMountCheck: true
  SwiftUseLocalDir: false
  SwiftRawDisks: {"vdb": {"base_dir":"/dev/mapper/"}, "vdc": {"base_dir":"/dev/mapper/"}}
```

5.3. INSTALL AND CONFIGURE STORAGE NODES FOR AN OBJECT STORAGE SERVICE DEPLOYMENT

To use Red Hat OpenStack Platform (RHOSP) Object Storage service (swift) on external storage nodes, you must install and configure the storage nodes that operate the account, container, and object services processes. This configuration references two storage nodes, each of which contains two empty local block storage devices.



NOTE

The internal network for the Object Storage service is not authenticated. For security purposes, keep the storage nodes on a dedicated network or VLAN.

5.3.1. Preparing storage devices for Object Storage service installation

Before you install and configure the Object Storage service (swift) on the storage nodes, you must prepare the storage devices.

NOTE

The following procedure uses **/dev/sdb** and **/dev/sdc** as device names, but you can substitute the values for specific nodes in your environment.

NOTE

Perform all the following steps on each storage node.

Procedure

1. Install the supporting utility packages:

```
# yum install xfsprogs rsync
```

2. Format the **/dev/sdb** and **/dev/sdc** devices as XFS:

```
# mkfs.xfs /dev/sdb
# mkfs.xfs /dev/sdc
```

3. Create the mount point directory structure:

```
# mkdir -p /srv/node/sdb
# mkdir -p /srv/node/sdc
```

4. Edit the **/etc/fstab** file and add the following data to it:

```
/dev/sdb /srv/node/sdb xfs defaults 0 2
/dev/sdc /srv/node/sdc xfs defaults 0 2
```

5. Mount the devices:

```
# mount /srv/node/sdb
# mount /srv/node/sdc
```

6. Create or edit the **/etc/rsyncd.conf** file to contain the following data:

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = <management_interface_IP_address>

[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock

[container]
```

```
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

Replace **<management_interface_IP_address>** with the IP address of the management network on the storage node.

7. Start the **rsyncd** service and configure it to start when the system boots:

```
# systemctl enable rsyncd.service
# systemctl start rsyncd.service
```

5.3.2. Configuring Object Storage service components on external Storage nodes

Configure the account, container, and object storage servers on the external nodes that you want to use with the Object Storage service (swift).

Procedure

1. Install the packages:

```
# yum install openstack-swift-account openstack-swift-container \
  openstack-swift-object
```

2. Edit the **/etc/swift/account-server.conf** file and complete the following actions:

   a. In the **[DEFAULT]** section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

```
[DEFAULT]
...
bind_ip = <management_interface_IP_address>
bind_port = 6202
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

   Replace **<management_interface_IP_address>** with the IP address of the management network on the storage node.

   b. In the **[pipeline:main]** section, enable the **healthcheck** and **recon** modules:

```
[pipeline:main]
pipeline = healthcheck recon account-server
```

   c. In the **[filter:recon]** section, configure the recon cache directory:

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

3. Open the default firewall port for the account service:

```
# firewall-cmd --permanent --add-port=6202/tcp
```

4. Edit the **/etc/swift/container-server.conf** file and complete the following actions:

   a. In the **[DEFAULT]** section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

```
[DEFAULT]
...
bind_ip = <management_interface_IP_address>
bind_port = 6201
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

   Replace **<management_interface_IP_address>** with the IP address of the management network on the storage node.

   b. In the **[pipeline:main]** section, enable the **healthcheck** and **recon** modules:

```
[pipeline:main]
pipeline = healthcheck recon container-server
```

   c. In the **[filter:recon]** section, configure the recon cache directory:

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
```

5. Open the default firewall port for the container service:

```
# firewall-cmd --permanent --add-port=6201/tcp
```

6. Edit the **/etc/swift/object-server.conf** file and complete the following actions:

   a. In the **[DEFAULT]** section, configure the bind IP address, bind port, user, configuration directory, and mount point directory:

```
[DEFAULT]
...
bind_ip = <management_interface_IP_address>
bind_port = 6200
user = swift
```

```
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

Replace **<management_interface_IP_address>** with the IP address of the management network on the storage node.

b. In the **[pipeline:main]** section, enable the **healthcheck** and **recon** modules:

```
[pipeline:main]
pipeline = healthcheck recon object-server
```

c. In the **[filter:recon]** section, configure the **recon_cache_path** and the **recon_lock_path** directories:

```
[filter:recon]
use = egg:swift#recon
...
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

7. Open the default firewall port for the object service:

```
# firewall-cmd --permanent --add-port=6200/tcp
```

8. Ensure that the ownership of the mount point directory structure is correct:

```
# chown -R swift:swift /srv/node
```

9. Create the **recon** directory and ensure proper ownership of it:

```
# mkdir -p /var/cache/swift
# chown -R root:swift /var/cache/swift
# chmod -R 775 /var/cache/swift
```

5.4. ADDING NEW OBJECT STORAGE NODES

To add new Object Storage (swift) nodes to your cluster, you must increase the node count, update the rings, and synchronize the changes. The following example procedure increases three nodes to four.

Procedure

1. Log in to the undercloud node as the **stack** user and source the **stackrc** credentials file:

```
$ source ~/stackrc
```

2. Use the **ObjectStorageCount** parameter to increase the Object Storage count by 1. This parameter is usually located in the **node-info.yaml** file, which is the environment file that contains your node counts:

```
parameter_defaults:
  ObjectStorageCount: 4
```

3. Add the IP address of the new node to the **ObjectStorageIPs** parameter in the **node-info.yaml** file:

```
ObjectStorageIPs: <ip_address>
```

4. Create an environment file, for example **hostname-map.yaml**, and add the host name of the new storage node, for example, **overcloud-objectstorage-4** to the **HostnameMap** parameter:

```
parameter_defaults:
  HostnameMap:
    overcloud-objectstorage-4: overcloud-objectstorage-4
```

5. Add the hardware and power management information about the new node to the node definition template. The node definition template is a file that was manually created to register overcloud nodes during initial overcloud configuration. For example, the template might be: **/home/stack/nodes.json**. The following example uses JSON format, although if your template is a YAML file, add the information according to YAML formatting and attributes. For more information about the node definition template, see Registering nodes for the overcloud in the *Director Installation and Usage* guide.

```
"ports":[
"bb:bb:bb:bb:bb:bb"
],
"name":"node01",
"cpu":"4",
"memory":"6144",
"disk":"40",
"arch":"x86_64",
"pm_type":"pxe_ipmitool",
"pm_user":"admin",
"pm_password":"p@55w0rd!",
"pm_addr":"192.168.24.205"
```

6. Add the capabilities information about the new node to the node definition template:

```
"capabilities": "profile:swift-storage,boot_option:local,boot_mode:uefi,node:objectstorage-4",
```

7. Import the node to the overcloud and perform introspection on the node:

```
$ openstack overcloud node import ~/nodes.yaml
$ openstack overcloud node introspect objectstorage-4 --provide
```

8. Add the root disk serial number to the new node:

```
$ openstack baremetal introspection data save objectstorage-4 | jq ".inventory.disks"
$ openstack baremetal node set --property root_device='{"serial": "<disk_serial_num>"}'
<node_UUID>
```

   Replace **<disk_serial_num>** and **<node_UUID>** according to your new node.

9. Include the files you created that contain your new node configurations in the deployment command with any other environment files that are relevant to your environment:

```
$ openstack overcloud deploy \
```

```
--templates \
…
-e <existing_overcloud_environment_files> \
-e node-info.yaml \
-e hostname-map.yaml \
```

5.4.1. Updating and rebalancing the Object Storage rings

The Object Storage service (swift) stores a copy of the ring files on the undercloud to deploy new Storage nodes and replace existing Controller and Object Storage nodes. Use this copy to modify overcloud ring files and distribute them across your nodes.

Procedure

1. Log in to the undercloud as the **stack** user and source the **stackrc** credentials file:

```
$ source ~/stackrc
```

2. Create the following directory:

```
$ mkdir temp && cd temp/
```

3. Download the overcloud rings builder file to the new directory:

```
$ openstack object save overcloud-swift-rings swift-rings.tar.gz
```

4. Extract the rings:

```
$ tar xzvf swift-rings.tar.gz
```

5. Create a directory to back up the original version of the rings:

```
$ mkdir backup && cp swift-rings.tar.gz backup/
```

   Object Storage ring files are located in the **etc/swift** folder.

6. Collect the values for the following variables according to your device details:

   - **<device_name>**:

     ```
     $ openstack baremetal introspection data save <node_name> | jq ".inventory.disks"
     ```

   - **<node_ip>**:

     ```
     openstack server show <node_name>
     ```

   - **<port>** The default port is **600x**. If you altered the default, use the applicable port.
   - **<builder_file>** The builder file name from step 3.
   - **<weight>** and **<zone>** variables are user-defined.

7. Use **swift-ring-builder** to add and update the new node to the existing rings. Replace the variables according to the device details.

```
$ swift-ring-builder etc/swift/<builder_file> add <zone>-<node_ip>:<port>/<device_name>
<weight>
```

8. Upload the modified ring files to the Object Storage service on the undercloud:

```
$ tar cvzf swift-rings.tar.gz etc/
$ openstack object create overcloud-swift-rings swift-rings.tar.gz
```

5.4.2. Syncing node changes and migrating data

You must deliver the changed ring files to the Object Storage (swift) containers after you copy new ring files to their correct folders.

Important

Do not migrate all of the data at the same time. Migrate the data in 10% increments. For example, configure the weight of the source device to equal 90.0 and the target device to equal 10.0. Then configure the weight of the source device to equal 80.0 and 20.0. Continue to incrementally migrate the data until you complete the process. During the migration, if you move all of the data at the same time, the old data is on the source device but the ring points to the new target device for all replicas. Until the replicators have moved all of the data to the target device, the data is inaccessible.

Important

During migration, the Object Storage rings reassign the location of data, and then the replicator moves the data to the new location. As cluster activity increases, the process of replication slows down due to increased load. The larger the cluster, the longer a replication pass takes to complete. This is the expected behavior, but it can result in 404 errors in the log files if a client accesses the data that is currently being relocated. When a proxy attempts to retrieve data from a new location, but the data is not yet in the new location, **swift-proxy** reports a 404 error in the log files.
When the migration is gradual, the proxy accesses replicas that are not being moved and no error occurs. And when the proxy attempts to retrieve the data from an alternative replica, 404 errors in log files resolve. To confirm that the replication process is progressing, refer to the replication logs. The Object Storage service issues replication logs every 5 minutes.

Procedure

1. Use a script similar to the following example to distribute ring files from the undercloud node to a specified Controller node and restart the Object Storage service containers on those nodes:

```
cd etc/swift/
for j in 8 11 23; do
  ssh heat-admin@192.168.24."$j" "rm *.ring.gz"
  for i in account.ring.gz container.ring.gz object.ring.gz; do
    scp $i heat-admin@192.168.24."$j":~/
    ssh heat-admin@192.168.24."$j" "sudo cp -f "$i" /var/lib/config-data/puppet-
generated/swift/etc/swift/"
    ssh heat-admin@192.168.24."$j" "sudo chown root:root /var/lib/config-data/puppet-
generated/swift/etc/swift/"$i""
    ssh heat-admin@192.168.24."$j" "sudo restorecon /var/lib/config-data/puppet-
generated/swift/etc/swift/"$i""
  done
  ssh heat-admin@192.168.24."$j" "rm *.builder"
```

```
  for i in account.builder container.builder object.builder; do
    scp $i heat-admin@192.168.24."$j":~/
    ssh heat-admin@192.168.24."$j" "cat "$i" | sudo tee  /var/lib/config-data/puppet-
generated/swift/etc/swift/"$i" >/dev/null"
  done
  ssh heat-admin@192.168.24."$j" 'for k in `sudo podman ps --format "{{.Names}}" | grep
swift`; do sudo podman restart $k; done'
done
cd ../../
```

2. To confirm that the data is being moved to the new disk, run the following command on the new storage node:

```
$  sudo grep -i replication  /var/log/container/swift/swift.log
```

5.5. REMOVING OBJECT STORAGE NODES

There are two ways to remove an Object Storage (swift) node. Choose one of the following options depending on the complexity of your cluster or the quantity of data that it contains:

- Simple removal: This method removes the node in one action and is appropriate for an efficiently-powered cluster with smaller quantities of data. See Removing an Object Storage node in one action.
- Incremental removal: Alter the rings to decrease the weight of the disks on the node that you want to remove. This method is appropriate if you want to minimize impact on storage network usage or if your cluster contains larger quantities of data. See Altering rings to incrementally remove an Object Storage node.

5.5.1. Removing an Object Storage node in one action

Use this method to remove a node in an efficiently-powered cluster with small quantities of data. The following example procedure decreases four nodes to three.

Procedure

1. Log in to the undercloud as the **stack** user and source the **stackrc** credentials file:

```
$ source ~/stackrc
```

2. Use the **ObjectStorageCount** parameter to decrease the Object Storage count by 1. This parameter is usually located in **node-info.yaml**, which is the environment file that contains your node counts:

```
parameter_defaults:
  ObjectStorageCount: 3
```

3. Delete the IP address of the node that you removed from the **ObjectStorageIPs** parameter in the **node-info.yaml** file:

```
ObjectStorageIPs: <ip_address>
```

4. Create an environment file, for example, **remove-object-node.yaml**. This file identifies and removes the Object Storage node that you specify. The following example content specifies the removal of **overcloud-objectstorage-1**:

```
parameter_defaults:
  ObjectStorageRemovalPolicies:
    [{'resource_list': ['1']}]
```

5. Include both the **node-info.yaml** and **remove-object-node.yaml** files in the deployment command:

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e node-info.yaml \
-e remove-object-node.yaml \
…
```

   Director deletes the Object Storage node from the overcloud and updates the rest of the nodes on the overcloud to accommodate the node removal.

6. List active nodes to verify that you removed the correct node:

```
$ openstack server list
```

7. On the undercloud node, delete the **remove-object-node.yaml** file so that it is not included in future redeployments:

```
$ rm <file_path>/remove-object-node.yaml
```

5.5.2. Altering rings to incrementally remove an Object Storage node

Use this method if you need to minimize the impact on the storage network while you remove a node, or if your cluster contains large quantities of data. To alter the storage rings to decrease the weight of the disks on the node that you want to remove, first complete the procedures listed in **Prerequisites**. The following example decreases the nodes from four to three.

Prerequisites

- Updated and rebalanced Object Storage rings. See Updating and rebalancing the Object Storage rings.
- Changes in the Object Storage rings are synchronized. See Syncing the changes and migrating data.

Procedure

1. Log in to the undercloud as the **stack** user and source the **stackrc** credentials file:

```
$ source ~/stackrc
```

2. Use the **ObjectStorageCount** parameter to decrease the Object Storage count by 1. This parameter is usually located in **node-info.yaml**, which is the environment file that contains your node counts:

```
parameter_defaults:
  ObjectStorageCount: 3
```

3. Delete the IP address of the node that you removed from the **ObjectStorageIPs** parameter in the **node-info.yaml** file:

```
ObjectStorageIPs: <ip_address>
```

4. Create an environment file, for example, **remove-object-node.yaml**. This file identifies and removes the Object Storage node that you specify. The following example content specifies the removal of **overcloud-objectstorage-1**:

```
parameter_defaults:
  ObjectStorageRemovalPolicies:
    [{'resource_list': ['1']}]
```

5. Include both the **node-info.yaml** and **remove-object-node.yaml** files in the deployment command:

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e node-info.yaml \
-e remove-object-node.yaml \
…
```

Director deletes the Object Storage node from the overcloud and updates the rest of the nodes on the overcloud to accommodate the node removal.

6. List active nodes to verify that you removed the correct node:

```
$ openstack server list
```

7. On the undercloud node, delete the **remove-object-node.yaml** file so that it is not included in future redeployments:

```
$ rm <file_path>/remove-object-node.yaml
```

5.6. REPLACING OBJECT STORAGE NODES

Follow the instructions in this section to understand how to replace Object Storage nodes without impact to the integrity of the cluster. This example involves a three-node Object Storage cluster in which you want to replace the node **overcloud-objectstorage-1** node. The goal of the procedure is to add one more node and then remove the **overcloud-objectstorage-1** node. The new node replaces the **overcloud-objectstorage-1** node.

Procedure

1. Increase the Object Storage count using the **ObjectStorageCount** parameter. This parameter is usually located in **node-info.yaml**, which is the environment file that contains your node counts:

```
parameter_defaults:
  ObjectStorageCount: 4
```

   The **ObjectStorageCount** parameter defines the quantity of Object Storage nodes in your environment. In this example, scale the quantity of Object Storage nodes from **3** to **4**.

2. Run the deployment command with the updated **ObjectStorageCount** parameter:

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates -e node-info.yaml
<environment_files>
```

   After the deployment command completes, the overcloud contains an additional Object Storage node.

3. Replicate data to the new node. Before you remove a node, in this case, **overcloud-objectstorage-1**, wait for a replication pass to finish on the new node. Check the replication pass progress in the **/var/log/swift/swift.log** file. When the pass finishes, the Object Storage service should log entries similar to the following example:

```
Mar 29 08:49:05 localhost object-server: Object replication complete.
Mar 29 08:49:11 localhost container-server: Replication run OVER
Mar 29 08:49:13 localhost account-server: Replication run OVER
```

4. To remove the old node from the ring, reduce the **ObjectStorageCount** parameter to omit the old node. In this example, reduce the **ObjectStorageCount** parameter to **3**:

```
parameter_defaults:
  ObjectStorageCount: 3
```

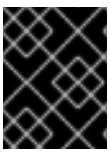
5. Create a new environment file named **remove-object-node.yaml**. This file identifies and removes the specified Object Storage node. The following content specifies the removal of **overcloud-objectstorage-1**:

```
parameter_defaults:
  ObjectStorageRemovalPolicies:
    [{'resource_list': ['1']}]
```

6. Include both the **node-info.yaml** and **remove-object-node.yaml** files in the deployment command:

```
(undercloud) $ openstack overcloud deploy --templates -e node-info.yaml
<environment_files> -e remove-object-node.yaml
```

Director deletes the Object Storage node from the overcloud and updates the rest of the nodes on the overcloud to accommodate the node removal.



IMPORTANT

Include all environment files and options from your initial overcloud creation. This includes the same scale parameters for non-Compute nodes.

5.7. BASIC CONTAINER MANAGEMENT IN THE OBJECT STORAGE SERVICE

To help with organization in the Object Storage service (swift), you can use pseudo folders. These folders are logical devices that can contain objects and be nested. 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.

5.7.1. Creating a container in the Object Storage service

Use the dashboard to create a container.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Click **Create Container**.
3. Specify the **Container Name**, and select one of the following in the **Container Access** field.

Type	Description
Private	Limits access to a user in the current project.
Public	Permits API access to anyone with the public URL. However, in the dashboard, project users cannot see public containers and data from other projects.

4. Click **Create Container**.
5. Optional: New containers use the default storage policy. If you have 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:

```
# swift post -H "X-Storage-Policy:<policy>" <container_name>
```

Replace **<policy>** with the name or alias of the policy that you want the container to use, and replace **<container_name>** with the name of the container.

5.7.2. Creating pseudo folders for containers in the Object Storage service

Use the dashboard to create a pseudo folder for a container.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the container to which you want to add the pseudo folder.
3. Click **Create Pseudo-folder**.

4. Specify the name in the **Pseudo-folder Name** field, and click **Create**.

5.7.3. Deleting a container in the Object Storage service

Use the dashboard to delete a container.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Browse for the container in the **Containers** section, and ensure that all objects are deleted. For more information, see Deleting an object.
3. Select **Delete Container** in the container arrow menu.
4. Click **Delete Container** to confirm the container removal.

5.7.4. Uploading an object to the Object Storage service

If you do not upload an actual file, the object is still created as a placeholder that you can use later to upload the file.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the container in which you want to place the uploaded the object. If a pseudo folder already exists in the container, you can click its name.
3. Browse for your file, and click **Upload Object**.
4. Specify a name in the **Object Name** field:
   - You can specify pseudo folders in the name by using a **/** character,for example, *Images/myImage.jpg*. If the specified folder does not already exist, it is created when the object is uploaded.
   - A name that is not unique to the location, that is, the object already exists, overwrites the contents of the object.
5. Click **Upload Object**.

5.7.5. Copying an object within the Object Storage service

Use the dashboard to copy an object.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the object's container or folder (to display the object).
3. Click **Upload Object**.
4. Browse for the file to be copied, and select **Copy** in its arrow menu.

5. Specify the following:

Field	Description
Destination container	Target container for the new object.
Path	Pseudo-folder in the destination container; if the folder does not already exist, it is created.
Destination object name	New object's name. If you use a name that is not unique to the location (that is, the object already exists), it overwrites the object's previous contents.

6. Click **Copy Object**.

5.7.6. Deleting an object from the Object Storage service

Use the dashboard to delete an object.

Procedure

1. In the dashboard, select **Project > Object Store > Containers**
2. Browse for the object, and select **Delete Object** in its arrow menu.
3. Click **Delete Object** to confirm the object is removed.

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

With the Shared File Systems service (manila), you can provision shared file systems that multiple compute instances, bare metal nodes, or containers can consume. Cloud administrators create share types to prepare the share service and enable end users to create and manage shares.

Prerequisites

- An end user requires at least one share type to use the Shared File Systems service.
- For back ends where **driver_handles_share_servers=False**, a cloud administrator configures the requisite networking in advance rather than dynamically in the Shared File System back end.
- For a CephFS through NFS back end, a cloud administrator deploys Red Hat OpenStack Platform (RHOSP) director with isolated networks and environment arguments and a custom **network_data** file to create an isolated StorageNFS network for NFS exports. After deployment, before the overcloud is used, the administrator creates a corresponding Networking service (neutron) StorageNFS shared provider network that maps to the isolated StorageNFS network of the data center.
- For a Compute instance to connect to this shared provider network, the user must add an additional neutron port.

6.1. SHARED FILE SYSTEMS SERVICE BACK ENDS

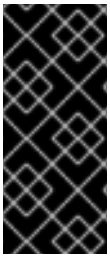
When cloud administrators use Red Hat OpenStack Platform (RHOSP) director to deploy the Shared File Systems service (manila), they can choose one or more of the following supported back ends:

- CephFS through NFS. For more information, see Deploying the Shared File Systems service with CephFS through NFS.
- Native CephFS. For more information, see CephFS Back End Guide for the Shared File System Service.
- NetApp. For more information, see the NetApp documentation: Shared File Systems Service (Manila).
- Dell EMC Unity, Dell VNX, or Dell PowerMax. For more information, see the Dell documentation: Dell EMC Manila Backend Deployment Guide for Red Hat OpenStack Platform 16.

For a complete list of supported back end appliances and drivers, see Component, Plug-In, and Driver Support in RHEL OpenStack Platform.

6.1.1. Using multiple back ends

A back end is a storage system or technology that is paired with the Shared File Systems service (manila) driver to export file systems. The Shared File Systems service requires at least one back end to operate. In many cases, one back end is sufficient. However, you can also use multiple back ends in a single Shared File Systems service installation.



IMPORTANT

Currently, Red Hat OpenStack Platform (RHOSP) does not support multiple instances of the same back end to a Shared File Systems service deployment. For example, you cannot add two Ceph Storage clusters as back ends within the same deployment. The back ends that you choose must be heterogeneous. CephFS native and CephFS through NFS are considered one back end with different protocols.

The scheduler for the Shared File Systems service determines the destination back end for share creation requests. A single back end in the Shared File Systems service can expose multiple storage pools.

When you configure multiple back ends, the scheduler chooses one storage pool to create a resource from all the pools exposed by all configured back ends. This is abstracted from the end user. End users see only the capabilities that are exposed by the cloud administrator.

6.1.2. Deploying multiple back ends

By default, a standard Shared File Systems deployment environment file has a single back end. Use the following example procedure to add multiple back ends to the Shared File Systems service (manila) and deploy an environment with a CephFS-Ganesha and a NetApp back end.

Prerequisites

- At least two back ends.
- If a back end requires a custom container, you must use one from the Red Hat Ecosystem Catalog instead of the standard Shared File Systems service container. For example, if you want to use a Dell EMC Unity storage system back end with Ceph, choose the Dell EMC Unity container from the catalog.

Procedure

1. Create a storage customization YAML file. This file can be used to provide any values or overrides that suit your environment:

```
$ vi /home/stack/templates/storage_customizations.yaml
```

2. Configure the storage customization YAML file to include any overrides, including enabling multiple back ends:

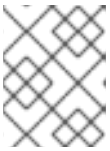
```
parameter_defaults:
  ManilaEnabledShareProtocols:
    - NFS
  ManilaNetappLogin: '<login_name>'
  ManilaNetappPassword: '<password>'
  ManilaNetappServerHostname: '<netapp-hostname>'
  ManilaNetappVserver: '<netapp-vserver>'
  ManilaNetappDriverHandlesShareServers: 'false'
```

NOTE

For more information about the **ManilaEnabledShareProtocols** parameter, see Section 6.1.4, "Overriding allowed NAS protocols".

3. Specify the back end templates in the **openstack overcloud deploy** command. The example configuration enables the Shared File System service with a NetApp back end and a CephFS through NFS back end.

```
$ openstack overcloud deploy \
  --timeout 100 \
  --stack overcloud \
  --templates /usr/share/openstack-tripleo-heat-templates \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_ganesha.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -r /home/stack/templates/roles/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsganesha-
config.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/manila-netapp-config.yaml \
-e /home/stack/templates/storage_customizations.yaml \

...
```



NOTE

For more information about the deployment command, see Director Installation and Usage.

6.1.3. Confirming deployment of multiple back ends

Use the **manila service-list** command to verify that your back ends deployed successfully. If you use a health check on multiple back ends, a ping test returns a response even if one of the back ends is unresponsive, so this is not a reliable way to verify your deployment.

Procedure

1. Log in to the undercloud host as the **stack** user.
2. Source the **overcloudrc** credentials file:

```
$ source ~/overcloudrc
```

3. Confirm the list of Shared File Systems service back ends:

```
$ manila service-list

+----+--------+--------+------+---------+-------+----------------------------+
| Id | Binary | Host | Zone | Status | State | Updated_at |
+----+--------+--------+------+---------+-------+----------------------------+
| 2 | manila-scheduler | hostgroup | nova | enabled | up | 2021-03-24T16:49:09.000000 |
| 5 | manila-share | hostgroup@cephfs | nova | enabled | up | 2021-03-24T16:49:12.000000 |
| 8 | manila-share | hostgroup@tripleo_netapp | nova | enabled | up | 2021-03-
24T16:49:06.000000 |
```

The status of each successfully deployed back end shows **enabled** and the state shows **up**.

6.1.4. Overriding allowed NAS protocols

The Shared File Systems service can export shares in one of many network attached storage (NAS) protocols, such as NFS, CIFS, or CEPHFS. By default, the Shared File Systems service enables all of the NAS protocols supported by the back ends in a deployment.

As a Red Hat OpenStack Platform (RHOSP) administrator, you can override the **ManilaEnabledShareProtocols** parameter and list only the protocols that you want to allow in your cloud.

For example, if back ends in your deployment support both NFS and CIFS, you can override the default value and enable only one protocol.

Procedure

1. Log in to the undercloud host as the **stack** user.
2. Source the **overcloudrc** credentials file:
3. Create a storage customization YAML file. This file can be used to provide any values or overrides that suit your environment:

```
$ vi /home/stack/templates/storage_customizations.yaml
```

4. Configure the **ManilaEnabledShareProtocols** parameter with the values that you want:

```
parameter_defaults:
  ManilaEnabledShareProtocols:
    - NFS
    - CEPHFS
```

5. Include the environment file that contains your new content in the **openstack overcloud deploy** by using the **-e** option. Ensure that you include all other environment files that are relevant to your deployment.

```
openstack overcloud deploy \
...
-e /home/stack/templates/storage_customizations.yaml
```

NOTE

The deployment does not validate the settings. The NAS protocols that you assign must be supported by the back ends in your Shared File Systems service deployment.

6.1.5. Viewing back end capabilities

The scheduler component of the Shared File Systems service (manila) makes intelligent placement decisions based on several factors such as capacity, provisioning configuration, placement hints, and the capabilities that the back end storage system driver detects and exposes.

Procedure

- Run the following command to view the available capabilities:

```
$ manila pool-list --detail
+----------------------------------+--------------------------+
| Property                         | Value                    |
+----------------------------------+--------------------------+
| name                             | hostgroup@cephfs#cephfs  |
| pool_name                        | cephfs                   |
| total_capacity_gb                | 1978                     |
| free_capacity_gb                 | 1812                     |
...
| driver_handles_share_servers     | False                    |
| snapshot_support                 | True                     |
| create_share_from_snapshot_support | False                  |
| revert_to_snapshot_support       | False                    |
| mount_snapshot_support           | False                    |
...
+----------------------------------+--------------------------+
+----------------------------------+----------------------------------+
| Property                         | Value                            |
+----------------------------------+----------------------------------+
| name                             | hostgroup@tripleo_netapp#aggr1_n1 |
| pool_name                        | aggr1_n1                         |
| total_capacity_gb                | 6342.1                           |
| free_capacity_gb                 | 6161.99                          |
...
| driver_handles_share_servers     | False                            |
| mount_snapshot_support           | False                            |
| replication_type                 | None                             |
| replication_domain               | None                             |
| sg_consistent_snapshot_support   | host                             |
| ipv4_support                     | True                             |
| ipv6_support                     | False                            |
+----------------------------------+----------------------------------+
+----------------------------------+----------------------------------+
| Property                         | Value                            |
+----------------------------------+----------------------------------+
| name                             | hostgroup@tripleo_netapp#aggr1_n2 |
| pool_name                        | aggr1_n2                         |
| total_capacity_gb                | 6342.1                           |
| free_capacity_gb                 | 6209.26                          |
...
| snapshot_support                 | True                             |
| create_share_from_snapshot_support | True                           |
| revert_to_snapshot_support       | True                             |
| driver_handles_share_servers     | False                            |
| mount_snapshot_support           | False                            |
| replication_type                 | None                             |
| replication_domain               | None                             |
| sg_consistent_snapshot_support   | host                             |
| ipv4_support                     | True                             |
| ipv6_support                     | False                            |
+----------------------------------+----------------------------------+
```

Related information

To influence placement decisions, as an administrator, you can use share types and extra specifications. For more information about share types, see Creating a share type .

6.2. CREATING A SHARE TYPE

Share types serve as hints to the Shared File Systems service scheduler to perform placement decisions. Red Hat OpenStack Platform (RHOSP) director configures the Shared File Systems service with a default share type named default, but does not create the share type.

IMPORTANT

An end user requires at least one share type to use the Shared File Systems service.

Procedure

1. After you deploy the overcloud, run the following command as the cloud administrator to create a share type:

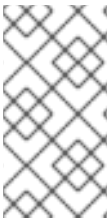
```
# manila type-create default <spec_driver_handles_share_servers>
```

   The **<spec_driver_handles_share_servers>** parameter is a Boolean value:

   - For CephFS through NFS or native CephFS, the value is false.
   - For other back ends, the value can be true or false; set **<spec_driver_handles_share_servers>** to match the value of the **Manila<backend>DriverHandlesShareServers** parameter. For example, if you use a NetApp back end, the parameter is called **ManilaNetappDriverHandlesShareServers**.

2. Add specifications to the default share type or create additional share types to use with multiple configured back ends. For example, configure the default share type to select a CephFS back end and an additional share type that uses a NetApp **driver_handles_share_servers=True** back end:

```
(overcloud) [stack@undercloud-0 ~]$ manila type-create default false --extra-specs
share_backend_name='cephfs'
(overcloud) [stack@undercloud-0 ~]$ manila type-create netapp true --extra-specs
share_backend_name='tripleo_netapp'
```



NOTE

By default, share types are public, which means they are visible to and usable by all Cloud projects, but you can create private share types for use within specific projects. For more information about how to make private share types, or to set additional share-type options, see the Security and Hardening Guide .

6.2.1. Common capabilities of share types

Share types define the common capabilities of shares. Review the common capabilities of share types to understand what you can do with your shares.

Table 6.1. Capabilities of share types

Capability	Values	Description
driver_handles_share_servers	true or false	Grants permission to use share networks to create shares.

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

6.2.2. Discovering share types

As a cloud user, you must specify a share type when you create a share.

Procedure

1. Discover the available share types:

```
$ manila type-list
```

The command output lists the name and ID of the share type.

6.3. CREATING A SHARE

Create a share to read and write data.

To create a share, use a command similar to the following:

-

```
$ manila create [--share-type <sharetype>] [--name <sharename>] proto GB
```

Replace the following values:

- **sharetype** applies settings associated with the specified share type.
  - Optional: if not supplied, the **default** share type is used.
- **sharename** is the name of the share.
  - Optional: shares are not required to have a name, nor is the name guaranteed to be unique.
- **proto** is the share protocol you want to use.
  - For CephFS with NFS, **proto** is **nfs**.
  - For CephFS native, **proto** is **cephfs**.
  - For NetApp and Dell EMC storage back ends, **proto** is **nfs** or **cifs**.
- **GB** is the size of the share in gigabytes.

For example, in Section 6.2, "Creating a share type", the cloud administrator created a **default** share type that selects a CephFS back end and another share type named **netapp** that selects a NetApp back end.

Procedure

1. Use the example share types to create a 10 GB NFS share named **share-01** on the CephFS NFS back end. This example uses CephFS with NFS:

```
(user) [stack@undercloud-0 ~]$ manila create --name share-01 nfs 10
```

2. Optional: Create a 20 GB NFS share named **share-02** on the NetApp back end:

```
(user) [stack@undercloud-0 ~]$ manila create --name share-02 --share-type netapp --share-network mynet nfs 20
```

6.3.1. Listing shares and exporting information

To verify that you successfully created the shares, complete the following steps.

Procedure

1. List the shares:

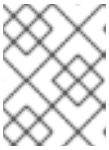
```
(user) [stack@undercloud-0 ~]$ manila list

+--------------------------------------+----------+-----+-----------+          | ID                                   |
Name     | ... | Status    ...
+--------------------------------------+----------+-----+-----------+
| 8c3bedd8-bc82-4100-a65d-53ec51b5fe81 | share-01 | ... | available ...
+--------------------------------------+----------+-----+-----------+
```

2. View the export locations of the share:

```
(user) [stack@undercloud-0 ~]$ manila share-export-location-list share-01

+--------------------------------------------------------------------
| Path
|  172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01
+--------------------------------------------------------------------
```

3. View the parameters for the share:

```
manila share-export-location-show <id>
```



NOTE

This information is used to mount the share in Section 6.6.2, “Mounting the share”.

6.4. NETWORKING FOR SHARED FILE SYSTEMS

Shared file systems are accessed over a network. It is important to plan the networking on your cloud to ensure that end user clients can connect their shares to workloads that run on Red Hat OpenStack Platform (RHOSP) virtual machines, bare-metal servers, and containers.

Depending on the level of security and isolation required for end users, as an administrator, you can set the **driver_handles_share_servers** parameter to true or false.

If you set the **driver_handles_share_servers** parameter to true, this enables the service to export shares to end user-defined share networks with the help of isolated share servers.

When the **driver_handles_share_servers** parameter equals true, users can provision their workloads on self-service share networks. This ensures that their shares are exported by completely isolated NAS file servers on dedicated network segments.

The share networks used by end users can be the same as the private project networks that they can create. As an administrator, you must ensure that the physical network to which you map these isolated networks extends to your storage infrastructure.

You must also ensure that the network segmentation style by project networks is supported by the storage system used. Storage systems, such as NetApp ONTAP and Dell EMC PowerMax, Unity, and VNX, do not support virtual overlay segmentation styles such as GENEVE or VXLAN.

As an alternative, you can terminate the overlay networking at top-of-rack switches and use a more primitive form of networking for your project networks, such as VLAN. Another alternative is to allow VLAN segments on shared provider networks or provide access to a pre-existing segmented network that is already connected to your storage system.

If you set the **driver_handles_share_servers** parameter to false, users cannot create shares on their own share networks. Instead, they must connect their clients to the network configured by the cloud administrator.

When the **driver_handles_share_servers** parameter equals false, director can create a dedicated shared storage network for you. For example, when you deploy the native CephFS back end with standard director templates, director creates a shared provider network called **Storage**. When you deploy CephFS through the NFS back end, the shared provider network is called **StorageNFS**. Your end users must connect their clients to the shared storage network to access their shares.

Not all shared file system storage drivers support both modes of operation. Regardless of which mode you choose, the service ensures hard data path multi-tenancy isolation guarantees.

If you want to offer hard network path multi-tenancy isolation guarantees to tenant workloads as part of a self-service model, you must deploy with back ends that support the **driver_handles_share_servers** driver mode.

For information about network connectivity to the share, see Section 6.4.1, "Ensuring network connectivity to the share"

6.4.1. Ensuring network connectivity to the share

Clients that need to connect to a file share must have network connectivity to one or more of the export locations for that share.

There are many ways to configure networking with the Shared File Systems service, including using network plugins.

When the **driver_handles_share_servers** parameter for a share type equals true, a cloud user can create a share network with the details of a network to which the compute instance attaches and then reference it when creating shares.

When the **driver_handles_share_servers** parameter for a share type equals false, a cloud user must connect their compute instance to the shared storage network.

For more information about how to configure and validate network connectivity to a shared network, see Section 6.4.2, "Connecting to a shared network to access shares" .

6.4.2. Connecting to a shared network to access shares

When the **driver_handles_share_servers** parameter equals false, shares are exported to the shared provider network that the administrator made available. As an end user, 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 director deploys the Shared File Systems service with the CephFS through NFS back end. Follow similar steps to connect to the network made available by your cloud administrator.

NOTE

In the example procedure, the IP address family version of the client is not important. The steps in this 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 which does not allow ingress packets from unestablished connections:

```
(user) [stack@undercloud-0 ~]$ 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"'
updated_at: '2018-09-19T08:19:58Z'
```

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

```
(user) [stack@undercloud-0 ~]$ 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='172.17.5.160', 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

StorageNFSSubnet assigned IP address 172.17.5.160 to **nfs-port0**.

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

```
(user) [stack@undercloud-0 ~]$ openstack server add port instance0 nfs-port0
```

```
(user) [stack@undercloud-0 ~]$ openstack server list -f yaml
- Flavor: m1.micro
  ID: 0b878c11-e791-434b-ab63-274ecfc957e8
  Image: manila-test
  Name: demo-instance0
  Networks: demo-network=172.20.0.4, 10.0.0.53; StorageNFS=172.17.5.160
  Status: ACTIVE
```

In addition to its private and floating addresses, the Compute instance is assigned a port with the IP address 172.17.5.160 on the StorageNFS network that you can use to mount NFS shares when access is granted to that address for the share in question.

NOTE

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

6.4.3. 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 issues with DHCPv6 on the secondary interface. Use this procedure to configure an IPv6 interface manually on the instance. For more information, see BZ#1859695.

Prerequisites

- Connect to a shared network to access shares.

Procedure

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

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

3. Activate the interface:

```
$ 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:

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

6.5. GRANT SHARE ACCESS

Before you can mount a share on a client, such as a compute instance, you must grant the client access to the share by using a command similar to the following:

```
# manila access-allow <share> <accesstype> --access-level <accesslevel>  <clientidentifier>
```

Replace the following values:

- **share** - the share name or ID of the share created in Section 6.3, "Creating a share".
- **accesstype** - the type of access to be requested on the share. Some types include:
  - **user**: use to authenticate by user or group name.
  - **ip**: use to authenticate an instance through its IP address.
  - **cephx**: use to authenticate by native CephFS client user name.

NOTE

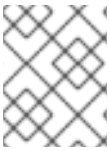
The type of access depends on the protocol of the share. For NFS shares, only **ip** access type is allowed. For CIFS, **user** access type is appropriate. For native CephFS shares, you must use **cephx**.

- **accesslevel** - optional, default is **rw**
  - **rw**: read-write access to shares.
  - **ro**: read-only access to shares.
- **clientidentifier** - varies depending on **accesstype**.
  - Use an IP address for **ip accesstype**.
  - Use CIFS user or group for **user accesstype**.
  - Use a user name string for **cephx accesstype**.

6.5.1. Granting access to a share

You must grant end user clients access to the share so that users can read data from and write data to the share.

Use this procedure to grant a client compute instance access to an NFS share through the IP address of the instance. The **user** rules for CIFS shares and **cephx** rules for CephFS shares follow a similar pattern. With **user** and **cephx** access types, you can use the same **clientidentifier** across multiple clients, if desired.



NOTE

In the example procedure, the IP address family version of the client is not important. The steps in this procedure use IPv4 addressing, but the steps are identical for IPv6.

Procedure

1. Retrieve the IP address of the client compute instance where you plan to mount the share. Make sure that you pick 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:

```
(user) [stack@undercloud-0 ~]$ openstack server list -f yaml
- Flavor: m1.micro
  ID: 0b878c11-e791-434b-ab63-274ecfc957e8
  Image: manila-test
  Name: demo-instance0
  Networks: demo-network=172.20.0.4, 10.0.0.53;
  StorageNFS=172.17.5.160
  Status: ACTIVE

(user) [stack@undercloud-0 ~]$ manila access-allow share-01 ip 172.17.5.160
```

NOTE

Access to the share has its own ID (**accessid**).

```
+-----------------+--------------------------------------+
| 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       | 172.17.5.160
| access_level    | rw
| state           | queued_to_apply
| id              | 875c6251-c17e-4c45-8516-fe0928004fff
+-----------------+--------------------------------------+
```

2. Verify that the access configuration was successful:

```
(user) [stack@undercloud-0 ~]$ manila access-list share-01

+--------------+-------------+--------------+--------------+--------+ ...
| id           | access_type | access_to    | access_level | state  | ...
+--------------+-------------+--------------+--------------+--------+
| 875c6251-... | ip          | 172.17.5.160 | rw           | active | ...
+--------------+-------------+--------------+--------------+--------+ ...
```

6.5.2. Revoking access to a share

The owner of a share can revoke access to the share for any reason. Complete the following steps to revoke previously-granted access to a share.

Procedure

1. Revoke access to a share:

```
# manila access-deny <share> <accessid>
```

NOTE

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

For example:

```
(user) [stack@undercloud-0 ~]$ manila access-list share-01
+--------------+-------------+--------------+--------------+--------+
| id           | access_type | access_to    | access_level | state  | ...
+--------------+-------------+--------------+--------------+--------+ ...
| 875c6251-... | ip          | 172.17.5.160 | rw           | active | ...
+--------------+-------------+--------------+--------------+--------+

(user) [stack@undercloud-0 ~]$ manila access-deny share-01 875c6251-c17e-4c45-8516-
fe0928004fff

(user) [stack@undercloud-0 ~]$ manila access-list share-01

+--------------+------------+--------------+--------------+--------+ ...
| id           | access_type| access_to    | access_level | state  | ...
+--------------+------------+--------------+--------------+--------+ ...
+--------------+------------+--------------+--------------+--------+ ...
```

NOTE

If you have an existing client that has read-write permissions, you must revoke their access to a share and add a read only rule if you want the client to have read-only permissions.

6.6. MOUNT SHARE ON COMPUTE INSTANCES

After you grant access to clients, shares can be mounted and used by them. 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 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.

6.6.1. Listing shares export locations

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

Procedure

1. Retrieve the export location of a share:

```
(user) [stack@undercloud-0 ~]$ manila share-export-location-list share-01
```

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.

6.6.2. Mounting the share

Mount a share on the client to enable access to data.

For information about creating and granting share access, see the following procedures:

- Section 6.3, "Creating a share"
- Section 6.5, "Grant share access"

Procedure

1. Log in to the instance and run the following command:

```
(user) [stack@undercloud-0 ~]$ openstack server ssh demo-instance0 --login root
# hostname
demo-instance0
```

2. Mount the share on an IPv4 network by using the export location:

```
# mount -t nfs -v  172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01
/mnt
```

6.7. DELETING A SHARE

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

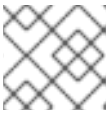
WARNING

Back up your data before you delete a share.

Procedure

1. Delete a share:

```
# manila delete <share>
```



NOTE

In the example command, <share> can be either the share name or the share ID.

For example:

```
# manila delete share-01
```

6.8. CHANGE THE DEFAULT QUOTAS IN THE SHARED FILE SYSTEMS SERVICE

To prevent system capacities from being exhausted without notification, cloud administrators can configure quotas. Quotas are operational limits.

6.8.1. Listing quotas

As a cloud administrator, you can list the quotas for a project or user by using the **manila quota-show** command. If you include the optional **--user** parameter, you can view the quota for this user in the specified project. If you omit this parameter, you get the quotas for the specified project.

You can update and delete quotas. You can update the shares, snapshots, gigabytes, snapshot-gigabytes, share-networks, share_groups,share_group_snapshots, and share-type quotas.

Procedure

1. To see the usage statements, run the following commands:

```
# manila help quota-show
# manila help quota-update
# manila help quota-delete
```

6.9. TROUBLESHOOTING FAILURES

In the event that Shared File Systems (manila) operations, such as create share or create share group, fail asynchronously, as an end user, you can run queries from the command line for more information about the errors.

6.9.1. Fixing create share or create share group failures

In this example, the goal of the end user is to create a share to host software libraries on several virtual machines. The example deliberately introduces two share creation failures to illustrate how to use the command line to retrieve user support messages.

Procedure

1. To create the share, you can use a share type that specifies some capabilities that you want the share to have. Cloud administrators can create share types. View the available share types:

```
clouduser1@client:~$ manila type-list
+--------------------------------------+-------------+------------+------------+------------------------------
--------+-----------------------------------------+-------------+
| ID                                   | Name        | visibility | is_default | required_extra_specs                |
optional_extra_specs                    | Description |
+--------------------------------------+-------------+------------+------------+------------------------------
--------+-----------------------------------------+-------------+
| 1cf5d45a-61b3-44d1-8ec7-89a21f51a4d4 | dhss_false  | public     | YES        |
driver_handles_share_servers : False | create_share_from_snapshot_support : True  | None
|
|                                      |             |            |            |                                |
mount_snapshot_support : False          |             |
|                                      |             |            |            |                                |
revert_to_snapshot_support : False      |             |
|                                      |             |            |            |                                | snapshot_support :
True                 |             |
| 277c1089-127f-426e-9b12-711845991ea1 | dhss_true   | public     | -          |
```

```
driver_handles_share_servers : True  | create_share_from_snapshot_support : True  | None
|
|                                    |           |           |           |                                |
mount_snapshot_support : False             |           |
|                                    |           |           |           |                                |
revert_to_snapshot_support : False         |           |
|                                    |           |           |           |                                | snapshot_support :
True                  |           |
+------------------------------------+------------+-----------+-----------+-----------------------------
--------+-------------------------------------------+------------+
```

In this example, two share types are available.

2. To use a share type that specifies the **driver_handles_share_servers=True** capability, you must create a share network on which to export the share. Create a share network from a private project network.

```
clouduser1@client:~$ openstack subnet list
+--------------------------------------+--------------------+--------------------------------------+----------
----------+
| ID                                   | Name               | Network                              | Subnet             |
+--------------------------------------+--------------------+--------------------------------------+----------
----------+
| 78c6ac57-bba7-4922-ab81-16cde31c2d06 | private-subnet      | 74d5cfb3-5dd0-43f7-b1b2-
5b544cb16212 | 10.0.0.0/26         |
| a344682c-718d-4825-a87a-3622b4d3a771 | ipv6-private-subnet | 74d5cfb3-5dd0-43f7-
b1b2-5b544cb16212 | fd36:18fc:a8e9::/64 |
+--------------------------------------+--------------------+--------------------------------------+----------
----------+

clouduser1@client:~$ manila share-network-create --name mynet --neutron-net-id 74d5cfb3-
5dd0-43f7-b1b2-5b544cb16212 --neutron-subnet-id 78c6ac57-bba7-4922-ab81-
16cde31c2d06
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| network_type      | None                                 |
| name              | mynet                                |
| segmentation_id   | None                                 |
| created_at        | 2018-10-09T21:32:22.485399           |
| neutron_subnet_id | 78c6ac57-bba7-4922-ab81-16cde31c2d06 |
| updated_at        | None                                 |
| mtu               | None                                 |
| gateway           | None                                 |
| neutron_net_id    | 74d5cfb3-5dd0-43f7-b1b2-5b544cb16212 |
| ip_version        | None                                 |
| cidr              | None                                 |
| project_id        | cadd7139bc3148b8973df097c0911016     |
| id                | 0b0fc320-d4b5-44a1-a1ae-800c56de550c |
| description       | None                                 |
+-------------------+--------------------------------------+

clouduser1@client:~$ manila share-network-list
+--------------------------------------+-------+
| id                                   | name  |
```

```
+--------------------------------------+-------+
| 6c7ef9ef-3591-48b6-b18a-71a03059edd5 | mynet |
+--------------------------------------+-------+
```

3. Create the share:

```
clouduser1@client:~$ manila create nfs 1 --name software_share --share-network mynet --
share-type dhss_true
+---------------------------------------+--------------------------------------+
| Property                              | Value                                |
+---------------------------------------+--------------------------------------+
| status                                | creating                             |
| share_type_name                       | dhss_true                            |
| description                           | None                                 |
| availability_zone                     | None                                 |
| share_network_id                      | 6c7ef9ef-3591-48b6-b18a-71a03059edd5 |
| share_server_id                       | None                                 |
| share_group_id                        | None                                 |
| host                                  |                                      |
| revert_to_snapshot_support            | False                                |
| access_rules_status                   | active                               |
| snapshot_id                           | None                                 |
| create_share_from_snapshot_support    | False                                |
| is_public                             | False                                |
| task_state                            | None                                 |
| snapshot_support                      | False                                |
| id                                    | 243f3a51-0624-4bdd-950e-7ed190b53b67 |
| size                                  | 1                                    |
| source_share_group_snapshot_member_id | None                                 |
| user_id                               | 61aef4895b0b41619e67ae83fba6defe     |
| name                                  | software_share                       |
| share_type                            | 277c1089-127f-426e-9b12-711845991ea1 |
| has_replicas                          | False                                |
| replication_type                      | None                                 |
| created_at                            | 2018-10-09T21:12:21.000000           |
| share_proto                           | NFS                                  |
| mount_snapshot_support                | False                                |
| project_id                            | cadd7139bc3148b8973df097c0911016     |
| metadata                              | {}                                   |
+---------------------------------------+--------------------------------------+
```

4. View the status of the share:

```
clouduser1@client:~$ manila list
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+-------------------+
| ID                                   | Name          | Size | Share Proto | Status | Is Public | Share Type
Name | Host | Availability Zone |
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+-------------------+
| 243f3a51-0624-4bdd-950e-7ed190b53b67 | software_share | 1    | NFS         | error  | False
| dhss_true       |      | None              |
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+-------------------+
```

In this example, an error occurred during the share creation.

5. To view the user support message, run the **message-list** command. Use the **--resource-id** to filter to the specific share you want to find out about.

```
clouduser1@client:~$ manila message-list
+--------------------------------------+-------------+-------------------------------------+-----------+---
----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
| ID                                   | Resource Type | Resource ID                         | Action ID | User
Message                                                                                            | Detail ID | Created At
|
+--------------------------------------+-------------+-------------------------------------+-----------+---
----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
| 7d411c3c-46d9-433f-9e21-c04ca30b209c | SHARE       | 243f3a51-0624-4bdd-950e-
7ed190b53b67 | 001       | allocate host: No storage could be allocated for this share request,
Capabilities filter didn't succeed. | 008       | 2018-10-09T21:12:21.000000 |
+--------------------------------------+-------------+-------------------------------------+-----------+---
----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
```

In the **User Message** column, notice that the Shared File Systems service failed to create the share because of a capabilities mismatch.

6. To view more message information, run the **message-show** command, followed by the ID of the message from the **message-list** command:

```
clouduser1@client:~$ manila message-show 7d411c3c-46d9-433f-9e21-c04ca30b209c
+---------------+-----------------------------------------------------------------------------------------------
-----------+
| Property      | Value                                                                                         |
+---------------+-----------------------------------------------------------------------------------------------
-----------+
| request_id    | req-0a875292-6c52-458b-87d4-1f945556feac
|
| detail_id     | 008                                                                                           |
| expires_at    | 2018-11-08T21:12:21.000000
|
| resource_id   | 243f3a51-0624-4bdd-950e-7ed190b53b67
|
| user_message  | allocate host: No storage could be allocated for this share request,
Capabilities filter didn't succeed. |
| created_at    | 2018-10-09T21:12:21.000000
|
| message_level | ERROR                                                                                           |
| id            | 7d411c3c-46d9-433f-9e21-c04ca30b209c
|
| resource_type | SHARE                                                                                         |
| action_id     | 001                                                                                           |
+---------------+-----------------------------------------------------------------------------------------------
-----------+
```

7. As the cloud user, you can check capabilities through the share type so you can review the share types available. The difference between the two share types is the value of **driver_handles_share_servers**:

```
clouduser1@client:~$ manila type-list
+--------------------------------------+------------+------------+------------+-------------------------------
--------+-------------------------------------------+-------------+
| ID                                   | Name       | visibility | is_default | required_extra_specs                |
optional_extra_specs                       | Description |
+--------------------------------------+------------+------------+------------+-------------------------------
--------+-------------------------------------------+-------------+
| 1cf5d45a-61b3-44d1-8ec7-89a21f51a4d4 | dhss_false | public     | YES        |
driver_handles_share_servers : False | create_share_from_snapshot_support : True  | None
|
|                                      |            |            |            |                                |
mount_snapshot_support : False             |             |
|                                      |            |            |            |                                |
revert_to_snapshot_support : False         |             |
|                                      |            |            |            |                                | snapshot_support :
True                  |             |
| 277c1089-127f-426e-9b12-711845991ea1 | dhss_true  | public     | -          |
driver_handles_share_servers : True  | create_share_from_snapshot_support : True  | None
|
|                                      |            |            |            |                                |
mount_snapshot_support : False             |             |
|                                      |            |            |            |                                |
revert_to_snapshot_support : False         |             |
|                                      |            |            |            |                                | snapshot_support :
True                  |             |
+--------------------------------------+------------+------------+------------+-------------------------------
--------+-------------------------------------------+-------------+
```

8. Create a share with the other available share type:

```
clouduser1@client:~$ manila create nfs 1 --name software_share --share-network mynet --
share-type dhss_false
+---------------------------------------+--------------------------------------+
| Property                              | Value                                |
+---------------------------------------+--------------------------------------+
| status                                | creating                             |
| share_type_name                       | dhss_false                           |
| description                           | None                                 |
| availability_zone                     | None                                 |
| share_network_id                      | 6c7ef9ef-3591-48b6-b18a-71a03059edd5 |
| share_group_id                        | None                                 |
| revert_to_snapshot_support            | False                                |
| access_rules_status                   | active                               |
| snapshot_id                           | None                                 |
| create_share_from_snapshot_support    | True                                 |
| is_public                             | False                                |
| task_state                            | None                                 |
| snapshot_support                      | True                                 |
| id                                    | 2d03d480-7cba-4122-ac9d-edc59c8df698 |
| size                                  | 1                                    |
| source_share_group_snapshot_member_id | None                                 |
| user_id                               | 5c7bdb6eb0504d54a619acf8375c08ce     |
| name                                  | software_share                       |
| share_type                            | 1cf5d45a-61b3-44d1-8ec7-89a21f51a4d4 |
| has_replicas                          | False                                |
```

```
| replication_type                | None                                 |
| created_at                      | 2018-10-09T21:24:40.000000           |
| share_proto                     | NFS                                  |
| mount_snapshot_support          | False                                |
| project_id                      | cadd7139bc3148b8973df097c0911016     |
| metadata                        | {}                                   |
+---------------------------------+--------------------------------------+
```

In this example, the second share creation attempt fails.

9. View the user support message:

```
clouduser1@client:~$ manila list
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+------------------+
| ID                                   | Name          | Size | Share Proto | Status | Is Public | Share Type
Name | Host | Availability Zone |
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+------------------+
| 2d03d480-7cba-4122-ac9d-edc59c8df698 | software_share | 1    | NFS         | error  | False
| dhss_false      |      | nova              |
| 243f3a51-0624-4bdd-950e-7ed190b53b67 | software_share | 1    | NFS         | error  | False
| dhss_true       |      | None              |
+--------------------------------------+---------------+------+-------------+--------+-----------+----------
-------+------+------------------+

clouduser1@client:~$ manila message-list
+--------------------------------------+--------------+--------------------------------------+-----------+---
-----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
| ID                                   | Resource Type | Resource ID                          | Action ID | User
Message                                                                                             | Detail ID | Created At
|
+--------------------------------------+--------------+--------------------------------------+-----------+---
-----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
| ed7e02a2-0cdb-4ff9-b64f-e4d2ec1ef069 | SHARE         | 2d03d480-7cba-4122-ac9d-
edc59c8df698 | 002       | create: Driver does not expect share-network to be provided with
current configuration.                  | 003       | 2018-10-09T21:24:40.000000 |
| 7d411c3c-46d9-433f-9e21-c04ca30b209c | SHARE         | 243f3a51-0624-4bdd-950e-
7ed190b53b67 | 001       | allocate host: No storage could be allocated for this share request,
Capabilities filter didn't succeed. | 008       | 2018-10-09T21:12:21.000000 |
+--------------------------------------+--------------+--------------------------------------+-----------+---
-----------------------------------------------------------------------------------------------------+--------
---+----------------------------+
```

The service does not expect a share network for the share type that you used.

10. Without consulting the administrator, you can discover that the administrator has not made available a storage back end that supports exporting shares directly on to your private neutron network. Create the share without the **share-network** parameter:

```
clouduser1@client:~$ manila create nfs 1 --name software_share --share-type dhss_false
+---------------------------------+--------------------------------------+
| Property                        | Value                                |
+---------------------------------+--------------------------------------+
```

```
| status                           | creating                             |
| share_type_name                  | dhss_false                           |
| description                      | None                                 |
| availability_zone                | None                                 |
| share_network_id                 | None                                 |
| share_group_id                   | None                                 |
| revert_to_snapshot_support       | False                                |
| access_rules_status              | active                               |
| snapshot_id                      | None                                 |
| create_share_from_snapshot_support | True                                 |
| is_public                        | False                                |
| task_state                       | None                                 |
| snapshot_support                 | True                                 |
| id                               | 4d3d7fcf-5fb7-4209-90eb-9e064659f46d |
| size                             | 1                                    |
| source_share_group_snapshot_member_id | None                                 |
| user_id                          | 5c7bdb6eb0504d54a619acf8375c08ce     |
| name                             | software_share                       |
| share_type                       | 1cf5d45a-61b3-44d1-8ec7-89a21f51a4d4 |
| has_replicas                     | False                                |
| replication_type                 | None                                 |
| created_at                       | 2018-10-09T21:25:40.000000           |
| share_proto                      | NFS                                  |
| mount_snapshot_support           | False                                |
| project_id                       | cadd7139bc3148b8973df097c0911016     |
| metadata                         | {}                                   |
+----------------------------------+--------------------------------------+
```

11. Ensure that the share was created successfully:

```
clouduser1@client:~$ manila list
+--------------------------------------+----------------+------+-------------+-----------+-----------+-------
----------+------+------------------+
| ID                                   | Name           | Size | Share Proto | Status    | Is Public | Share Type
Name | Host | Availability Zone |
+--------------------------------------+----------------+------+-------------+-----------+-----------+-------
----------+------+------------------+
| 4d3d7fcf-5fb7-4209-90eb-9e064659f46d | software_share | 1    | NFS         | available |
False     | dhss_false      |      | nova              |
| 2d03d480-7cba-4122-ac9d-edc59c8df698 | software_share | 1    | NFS         | error     | False
| dhss_false      |      | nova              |
| 243f3a51-0624-4bdd-950e-7ed190b53b67 | software_share | 1    | NFS         | error     |
False     | dhss_true       |      | None              |
+--------------------------------------+----------------+------+-------------+-----------+-----------+-------
----------+------+------------------+
```

12. Delete the shares and support messages:

```
clouduser1@client:~$ manila message-list
+--------------------------------------+-------------+-----------------------------------+-----------+---
-------------------------------------------------------------------------------------------+--------
---+----------------------------+
| ID                                   | Resource Type | Resource ID                       | Action ID | User
Message                                                                                    | Detail ID | Created At
|
```

```
+--------------------------------------+-------------+-----------------------------------+----------+---
-----------------------------------------------------------------------------------------------+-------
---+--------------------------+
| ed7e02a2-0cdb-4ff9-b64f-e4d2ec1ef069 | SHARE         | 2d03d480-7cba-4122-ac9d-
edc59c8df698 | 002       | create: Driver does not expect share-network to be provided with
current configuration.                  | 003       | 2018-10-09T21:24:40.000000 |
| 7d411c3c-46d9-433f-9e21-c04ca30b209c | SHARE         | 243f3a51-0624-4bdd-950e-
7ed190b53b67 | 001       | allocate host: No storage could be allocated for this share request,
Capabilities filter didn't succeed. | 008       | 2018-10-09T21:12:21.000000 |
+--------------------------------------+-------------+-----------------------------------+----------+---
-----------------------------------------------------------------------------------------------+-------
---+--------------------------+

clouduser1@client:~$ manila delete 2d03d480-7cba-4122-ac9d-edc59c8df698 243f3a51-
0624-4bdd-950e-7ed190b53b67
clouduser1@client:~$ manila message-delete ed7e02a2-0cdb-4ff9-b64f-e4d2ec1ef069
7d411c3c-46d9-433f-9e21-c04ca30b209c

clouduser1@client:~$ manila message-list
+----+--------------+-------------+-----------+-------------+-----------+------------+
| ID | Resource Type | Resource ID | Action ID | User Message | Detail ID | Created At |
+----+--------------+-------------+-----------+-------------+-----------+------------+
+----+--------------+-------------+-----------+-------------+-----------+------------+
```

6.9.2. Debugging share mounting failures

If you experience trouble when you mount shares, use these verification steps to identify the root cause of the issue.

Procedure

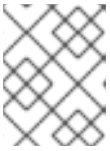
1. Verify the access control list of the share to ensure that the rule that corresponds to your client is correct and has been successfully applied.

```
$ manila access-list share-01
```

   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:

```
# ping -c 1 172.17.5.13
PING 172.17.5.13 (172.17.5.13) 56(84) bytes of data.
64 bytes from 172.17.5.13: icmp_seq=1 ttl=64 time=0.048 ms--- 172.17.5.13 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
If using the NFS protocol, you may verify that the NFS server is ready to respond to NFS rpcs
on the proper port:
# rpcinfo -T tcp -a 172.17.5.13.8.1 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 your shared file system back end storage has failed. Collect the logs and contact Red Hat Support.