Red Hat OpenStack Platform 15

Storage Guide

Understanding, using, and managing persistent storage in OpenStack

Last Updated: 2020-05-27

Red Hat OpenStack Platform 15 Storage Guide

Understanding, using, and managing persistent storage in OpenStack

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This 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

PREFACE

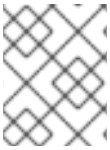
Red Hat OpenStack Platform (RHOSP) provides the foundation to build a private or public Infrastructure-as-a-Service (IaaS) cloud on top of Red Hat Enterprise Linux. It offers a massively scalable, fault-tolerant platform for the development of cloud-enabled workloads.

This guide discusses procedures for creating and managing persistent storage. Within OpenStack, this 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 using either the OpenStack dashboard or the command-line clients. Most procedures can be carried out using either method; some of the more advanced procedures can only be executed 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
```

CHAPTER 1. INTRODUCTION TO PERSISTENT STORAGE IN OPENSTACK

OpenStack 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, on the other hand, 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. OpenStack 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. For more information about volumes, see Chapter 2, *Block Storage and Volumes* .

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 through **containers**.

While a volume's contents can only be accessed 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. For example, the Data Processing service (**openstack-sahara**) can manage all of its binaries, data input, data output, and templates directly through the Object Storage service.

Shares

The Shared File System Service (**openstack-manila**) provides the means to easily provision remote, shareable file systems, or *shares*. Shares allow tenants 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

In general, a clustered storage solution provides greater back end scalability. For example, when using Red Hat Ceph as a Block Storage back end, you can scale storage capacity and redundancy by adding more Ceph OSD (Object Storage Daemon) nodes. Both Block Storage and Object Storage services support Red Hat Ceph 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 System 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 System Service can use both NetApp and CephFS, which can be configured 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. 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, “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 System 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. SECURITY

The Block Storage service provides basic data security through *volume encryption*. With this, you can configure a volume type to be encrypted through a static key; the key will then be used for encrypting all volumes created from the configured volume type. See Section 2.2.6, “Configure Volume Encryption” for more details.

Object and container security, on the other hand, is configured at the service and node level. The Object Storage service 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’s network security in order to protect object data.

The Shared File System service can secure shares through access restriction, whether by instance IP,

user/group, or TLS certificate. In addition, some Shared File System service deployments can feature a 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.

1.4. 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 2.2.9, “Configure and Use 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 System service provides no built-in backup features for shares, but it does allow you to create snapshots for cloning and restoration.

CHAPTER 2. BLOCK STORAGE AND VOLUMES

The Block Storage service (**openstack-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* .

2.1. 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). The director also has several integrated back end configurations.

Red Hat OpenStack Platform supports Red Hat Ceph and NFS as Block Storage 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*).

2.1.1. Third-Party Storage Providers

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.

2.2. BLOCK STORAGE SERVICE ADMINISTRATION

The following procedures explain how to configure the Block Storage service to suit your needs. All of these procedures require administrator privileges.

2.2.1. Active-active deployment for high availability

In active-passive mode, if the Block Storage service fails in a hyperconverged deployment, node fencing is undesirable, because it triggers 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.

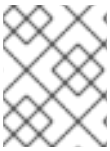
Active-active deployments improve scaling, performance, and reduce response time by balancing workloads across all available nodes. Deploying the Block Storage service in an active-active configuration creates a highly available environment that maintains the management layer during partial network outages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

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.

2.2.1.1. Enabling active-active configuration for high availability

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.



NOTE

Currently, active-active configuration for Block Storage works only with Ceph RADOS Block Device (RBD) back ends.

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
```

2.2.1.2. Maintenance commands for active-active configurations

After deploying an active-active configuration, there are several commands you can use to interact with the environment.

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.

2.2.1.3. 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.

2.2.1.4. 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.

2.2.1.5. Initiating server 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.

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
```

2.2.2. Group Volume Settings 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 Create a Volume. You can also apply settings after you create a volume, see Changing the Type of a Volume (Volume Re-typing). 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 Configure Volume Type Encryption.
- The back end that a volume uses. For more information, see Specify Back End for Volume Creation and Migrate between Back Ends.
- 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.

2.2.2.1. List the Capabilities of a Host Driver

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.

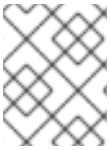
2.2.2.2. Create and Configure a Volume Type

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 tenants. If you need to create volume types with restricted access, you will need to do so through the CLI. For instructions, see Section 2.2.2.5, "Create and Configure Private Volume Types".



NOTE

You can also associate a QoS Spec to the volume type. For more information, see Section 2.2.5.4, "Associate a QOS Spec with a Volume Type" .

2.2.2.3. Edit a Volume Type

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.

2.2.2.4. Delete a Volume Type

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

2.2.2.5. Create and Configure Private Volume Types

By default, all volume types are available to all tenants. 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 tenants; examples include new back ends or ultra-high performance configurations that are being tested.

To create a private volume type, run:

```
$ 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 tenant level. To grant a tenant access to a private volume type, run:

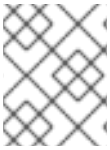
```
$ cinder  type-access-add --volume-type <TYPE-ID> --project-id <TENANT-ID>
```

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

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

To remove a tenant 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.

2.2.3. Create and Configure an Internal Tenant for the Block Storage Service

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.

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 Tenant" 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          |
   +----------+----------------------------------+
```

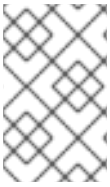
The procedure for adding Extra Config options creates an internal tenant. Refer to Section 2.2.4, "Configure and Enable the Image-Volume Cache".

2.2.4. Configure and Enable 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 2.3.1, "Create a volume".

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 2.2.3, "Create and Configure an Internal Tenant for the Block Storage Service".

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, refer to Creating the Overcloud with the CLI Tools section in the *Director Installation and Usage Guide*.

2.2.5. Use Quality-of-Service Specifications

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.

2.2.5.1. Basic volume Quality of Service

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**

2.2.5.2. Create and Configure a QOS Spec

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.

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.

2.2.5.3. Set Capacity-Derived QoS 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 2.2.2, "Group Volume Settings with Volume Types" and Section 2.2.2.2, "Create and Configure a Volume Type" . For instructions on how to set QoS specs, Section 2.2.5, "Use Quality-of-Service Specifications".

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 2.3.12, "Changing the volume type (volume re-typing)" for information about volume re-typing.

2.2.5.4. Associate a QOS Spec with a Volume Type

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

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.
4. Click **Associate**. The selected QOS Spec now appears in the **Associated QOS Spec** column of the edited volume type.

2.2.5.5. Disassociate a QOS Spec from a Volume Type

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 **None** from the QOS Spec to be associated list.
4. Click **Associate**. The selected QOS Spec is no longer in the **Associated QOS Spec** column of the edited volume type.

2.2.6. Configure 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

At present, volume encryption is not supported on file-based volumes (such as NFS).

Volume encryption is applied through volume type. See Section 2.2.6.1, "Configure Volume Type Encryption Through the Dashboard" for information on encrypted volume types.

2.2.6.1. Configure Volume Type Encryption Through 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:

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 2.2.2.2, "Create and Configure a Volume Type". Specifically, select the encrypted volume type from the Type drop-down list in the **Create Volume** window (see Section 2.3, "Basic volume usage and configuration").

To configure an encrypted volume type through the CLI, see Section 2.2.6.2, "Configure Volume Type Encryption Through 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.

2.2.6.2. Configure Volume Type Encryption Through the CLI

To configure Block Storage volume encryption, do the following:

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.

2.2.7. Configure How Volumes are Allocated 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 2.3.2, "Specify back end 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 (when the OpenStack Data Processing service is enabled)

To configure the filter scheduler, add an environment file to your deployment containing:

```
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.

2.2.8. Deploying availability zones

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.

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.

Procedure

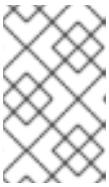
To deploy different availability zones for Block Storage volume back ends:

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.

2.2.9. Configure and Use Consistency Groups

The Block Storage service allows you to set *consistency groups*. With this, you can group multiple volumes together as a single entity. This, in turn, allows you to perform operations on multiple volumes at once, rather than individually. Specifically, this release allows you to use consistency groups to create snapshots for multiple volumes simultaneously. By extension, this will also allow you to restore or clone those volumes simultaneously.

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

2.2.9.1. Set Up Consistency Groups

By default, Block Storage security policy disables consistency groups APIs. You need to enable it here before using the feature. The related consistency group entries in */etc/cinder/policy.json* of the node hosting the Block Storage API service (namely, **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",
```

These settings need to be changed in an environment file and then deployed to the overcloud using the **openstack overcloud deploy** command. If you edit the JSON file directly, the changes will be overwritten next time the overcloud is deployed.

To enable the consistency groups, edit an environment file and add a new entry to the **parameter_defaults** section. This will ensure that the entries are updated in the containers and are retained whenever the environment is re-deployed using director with the **openstack overcloud deploy** command.

Add a new section to an environment file using **CinderApiPolicies** to set the consistency group settings. The equivalent parameter_defaults section showing the default settings from the JSON file would look like this:

```
parameter_defauts:
  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' }, \
  }
```

The value **'group:nobody'** determines that no group can use this feature, effectively disabling it. To enable it, you will need to change the group to another value.

For increased security, set the permissions for both consistency group API and volume type management API 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",
```

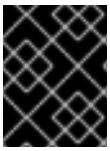
You can make the consistency groups feature available to all users by setting the API policy entries to allow users to create, use, and manage their own consistency groups. To do so, use **rule:admin_or_owner**:

```
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' }, \
 }
```

Once you have created the environment file file in **/home/stack/templates/**, log in as the stack user. Then, 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 more information on the **openstack overcloud deploy** command, see Deployment command options in the *Director Installation and Usage Guide*.

2.2.9.2. Create and Manage Consistency Groups

After enabling the consistency groups API, you can then start creating consistency groups. To do so:

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 should appear next in the **Volume Consistency Groups** table.

You can change the name or description of a consistency group by selecting **Edit Consistency Group** from its **Action** column.

In addition, you can also add or remove volumes from a consistency group directly. To do so:

1. As an admin user in the dashboard, select **Project > Compute > Volumes > Volume Consistency Groups**.

2. Find the consistency group you want to configure. In the **Actions** column of that consistency group, select **Manage Volumes**. Doing so will launch 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.

3. Click **Edit Consistency Group**.

2.2.9.3. Create and Manage Consistency Group Snapshots

After adding volumes to a consistency group, you can now create snapshots from it. Before doing so, first log in as **admin** user from the command line on the node hosting the **openstack-cinder-api** and run:

```
# export OS_VOLUME_API_VERSION=2
```

Doing so will configure the client to use version **2** of **openstack-cinder-api**.

To list all available consistency groups (along with their respective IDs, which you will need later):

```
# cinder consisgroup-list
```

To create snapshots using the consistency group, run:

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

Where:

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

To display a list of all available consistency group snapshots, run:

```
` # cinder cgsnapshot-list `
```

2.2.9.4. Clone Consistency Groups

Consistency groups can also be used 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.

To clone an existing consistency group:

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

Where: - *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).

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.

2.3. BASIC VOLUME USAGE AND CONFIGURATION

The following procedures describe how to perform basic end-user volume management. These procedures do not require administrative privileges.

2.3.1. Create a volume

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 2.2.2, "Group Volume Settings with Volume Types"). If you have multiple Block Storage back ends, you can use this to select a specific back end. See Section 2.3.2, "Specify back end for volume creation" for details.
Size (GB)	Volume size (in gigabytes).

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 on availability zones and host aggregates, see Manage Host Aggregates.

3. Specify a **Volume Source**:

Source	Description
No source, empty volume	The volume will be empty, and will 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 appears; you can then choose a snapshot from the list. For more information about volume snapshots, refer to Section 2.3.10, "Create, use, or delete volume snapshots".
Image	Use an existing image as a volume source. If you select this option, a new Use image as a source lists appears; 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 volume as a source list appears; 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's type later on. For details, see Section 2.3.12, "Changing the volume type (volume re-typing)".

2.3.2. Specify back end for volume creation

Whenever multiple Block Storage back ends are configured, you will also need to create a volume type for each back end. You can then use the type to specify which back end should be used for a created volume. For more information about volume types, see Section 2.2.2, "Group Volume Settings with Volume Types".

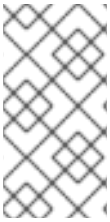
To specify a back end when creating a volume, select its corresponding volume type from the Type drop-down list (see Section 2.3.1, "Create a volume").

If you do not specify a back end during volume creation, the Block Storage service will automatically choose one for you. By default, the service will choose 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 2.2.7, "Configure How Volumes are Allocated to Multiple Back Ends".

2.3.3. Edit a volume name or description

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, both Compute and Block Storage services must be configured to use the same static key. For information on how to set up the requirements for volume encryption, refer to Section 2.2.6, "Configure Volume Encryption" .

2.3.4. Resize (extend) a volume

NOTE

The ability to resize a volume depends on back end support. Contact Red Hat Support for more information.

1. List the volumes to get the volume ID.

```
$ cinder list
```

2. Resize the volume by passing the volume ID and the new size (a value greater than the old one) as parameters:

```
$ cinder extend <volume ID> <size>
```

For example:

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

2.3.5. Delete a volume

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

NOTE

A volume cannot be deleted if it has existing snapshots. For instructions on how to delete snapshots, see Section 2.3.10, "Create, use, or delete volume snapshots" .

2.3.6. Attach and detach 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 on instances, see **Manage Instances** in the **Instances and Images Guide** available at Red Hat OpenStack Platform.

2.3.6.1. Attaching a volume to an instance

1. In the dashboard, select **Project > Compute > Volumes**
2. Select the volume's **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 wish to attach the volume.
4. Click **Attach Volume**.

2.3.6.2. Detaching a volume from an instance

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.

2.3.7. Attach a volume to multiple instances

Volume multi-attach gives multiple instances simultaneous read/write access to a Block Storage volume.

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. Also, the cinder driver must support multi-attach. The Ceph RBD driver is not supported.

WARNING

Encryption is not supported with multi-attach volumes.

2.3.7.1. 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.

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>
```

2.3.7.2. 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.

2.3.7.3. Creating a multi-attach volume

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

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 about how to attach a volume to an instance, see Attach a volume to an instance .

2.3.7.4. Supported back ends

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

2.3.8. Read-only volumes

A volume can be marked read-only to protect its data from being accidentally overwritten or deleted. To do so, set the volume to read-only using the following command:

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

To set a read-only volume back to read-write, run:

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

2.3.9. Change a volume owner

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.

2.3.9.1. Transfer a volume from the command line

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
```

2.3.9.2. Transfer a volume using the dashboard

Create a volume transfer from the dashboard

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.

2.3.10. Create, use, or delete volume snapshots

You can preserve a volume's state 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. In addition, you cannot delete a volume if it has existing snapshots. Volume backups are used to prevent data loss, whereas snapshots are used to facilitate cloning.

For this reason, snapshot back ends are typically co-located with volume back ends in order to minimize latency during cloning. By contrast, a backup repository is usually located in a different location (eg. 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, refer to the *Block Storage Backup Guide* .

To create a volume snapshot:

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 once 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 2.3.1, "Create a volume".

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

If your OpenStack deployment uses a Red Hat Ceph back end, see Section 2.3.10.1, "Protected and unprotected snapshots in a Red Hat Ceph Storage back end" for more information on snapshot security and troubleshooting.

2.3.10.1. Protected and unprotected snapshots in a Red Hat Ceph Storage back end

When using Red Hat Ceph Storage as a back end for your OpenStack deployment, you can set a snapshot to *protected* in the back end. Attempting to delete protected snapshots through OpenStack (as in, through the dashboard or the **cinder snapshot-delete** command) will fail.

When this occurs, set the snapshot to *unprotected* in the Red Hat Ceph back end first. Afterwards, you should be able to delete the snapshot through OpenStack as normal.

For related instructions, see Protecting a Snapshot and Unprotecting a Snapshot.

2.3.11. Upload a volume to the Image Service

You can upload an existing volume as an image to the Image service directly. To do so:

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 **Manage Images** in the **Instances and Images Guide** available at Red Hat OpenStack Platform.

2.3.12. Changing the volume type (volume re-typing)

Volume re-typing is the process of applying a volume type (and, in turn, its settings) to an already existing volume. For more information about volume types, see Section 2.2.2, "Group Volume Settings with Volume Types".

A volume can be re-typed whether or not it has an existing volume type. In either case, a re-type will only be successful if the Extra Specs of the volume type can be applied to the volume. Volume re-typing is useful for applying pre-defined settings or storage attributes to an existing volume, such as when you want to:

- Migrate the volume to a different back end (Section 2.4.1.2, "Migrate between Back Ends").
- Change the volume's storage class/tier.

Users with no administrative privileges can only re-type volumes they own. To perform a volume re-type:

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** drop-down list.
4. If you are migrating the volume to another back end, select **On Demand** from the **Migration Policy** drop-down list. See Section 2.4.1.2, "Migrate between Back Ends" for more information.

NOTE

Retyping a volume between two different types of back ends is not supported in this release.

5. Click **Change Volume Type** to start the migration.

2.4. ADVANCED VOLUME CONFIGURATION

The following procedures describe how to perform advanced volume management procedures.

2.4.1. Migrate a Volume

The Block Storage service allows you to migrate volumes between hosts or back ends within and across availability zones (AZ). Volume migration has some limitations:

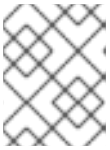
- In-use volume migration depends upon driver support.
- The volume cannot have snapshots.
- The target of the in-use volume migration requires ISCSI or fibre channel (FC) block-backed devices and cannot use non-block devices, such as Ceph RADOS Block Device (RBD).

The speed of any migration depends upon your host setup and configuration. With driver-assisted migration, the data movement takes place at the storage backplane instead of inside of the OpenStack Block Storage service. Optimized driver-assisted copying is available for not-in-use RBD volumes if volume re-typing is not required. Otherwise, data is copied from one host to another through the Block Storage service.

2.4.1.1. Migrate between Hosts

When migrating a volume between hosts, both hosts must reside on the same back end. Use the dashboard to migrate a volume between hosts:

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** checkbox.

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

2.4.1.2. Migrate between Back Ends

Migrating a volume between back ends, on the other hand, involves **volume re-typing**. This means that in order to migrate to a new back end:

1. The new back end must be specified as an **Extra Spec** in a separate *target volume type*.
2. All other Extra Specs defined in the target volume type must be compatible with the volume's original volume type.

See] and xref:section-specify-backend[for details.

When defining the back end as an Extra Spec, use **volume_backend_name** as the **Key**. Its corresponding value will be the back end's name, as defined in the Block Storage configuration file (/etc/cinder/cinder.conf). In this file, each back end is defined in its own section, and its corresponding name is set in the **volume_backend_name** parameter.

After you have a back end defined in a target volume type, you can migrate a volume to that back end through **re-typing**. This involves re-applying the target volume type to a volume, thereby applying the new back end settings. See Section 2.3.12, "Changing the volume type (volume re-typing)" for instructions.

NOTE

Retyping a volume between two different types of back ends is not supported in this release.

CHAPTER 3. OBJECT STORAGE SERVICE

OpenStack Object Storage (**swift**) stores its objects (data) in containers, which 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 might include photos, text files, or images. Stored objects are not compressed.

3.1. OBJECT STORAGE RINGS

Object Storage 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. It allows the Object Storage service to 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 different types of data: one for account information, another for containers (to facilitate organizing objects under an account), and another for object replicas.

3.1.1. Rebalancing rings

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

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

3.1.2. 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.

To use **swift-recon**, log in to one of the controller nodes and 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.
==============================================================
```

NOTE

As an alternative, 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 can't update database files fast enough, some updates will occur asynchronously. These numbers should decrease over time.
- Replication metrics: Notice the replication timestamps; full replication passes should happen frequently and there should be 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 ring files are consistent on all nodes.
- Quarantined files: There should be no (or very few) quarantined files for all nodes.
- Time-sync: All nodes must be synchronized.

3.1.3. Increasing ring partition power

The ring power determines the partition to which a resource (account, container, or object) is mapped. The partition is included in the path under which the resource is stored in a back end filesystem. Therefore, changing the partition power requires relocating resources to new paths in the back end filesystems.

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.

3.1.4. Creating 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 may differ from those of the original nodes. Without custom adjustments, the larger capacity of the new nodes may be underutilized. Or, if weights change in the rings, data dispersion can become uneven, which reduces safety.

Automation may 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.

3.2. OBJECT STORAGE SERVICE ADMINISTRATION

The following procedures explain how to customize the Object Storage service.

3.2.1. 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 by doing the following:

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
```

3.2.2. Enabling at-rest encryption

By default, objects uploaded to Object Storage are kept 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.

3.2.3. Deploying a standalone Object Storage cluster

You can use the composable role concept to deploy a standalone Object Storage (openstack-swift) cluster with the bare minimum of additional services (for example Keystone, HAProxy). For more information about roles, see Creating a **roles_data** File.

3.2.3.1. Creating the roles_data.yaml File

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
```

3.2.3.2. Deploying the New Roles

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

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

3.2.4. Using external SAN disks

By default, when the Red Hat OpenStack Platform 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 Object Storage 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.

3.2.4.1. SAN disk deployment configuration

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

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

3.3. BASIC CONTAINER MANAGEMENT

To help with organization, pseudo-folders are logical devices that can contain objects (and can 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.

3.3.1. Creating a container

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**.

New containers use the default storage policy. If you have multiple storage policies defined (for example, a default and another that enables erasure coding), you can configure a container to use a non-default storage policy through the command line. To do so, run:

```
# swift post -H "X-Storage-Policy:POLICY" CONTAINERNAME
```

Where:

- *POLICY* is the name or alias of the policy you want the container to use.
- *CONTAINERNAME* is the name of the container.

3.3.2. Creating a pseudo folder for a container

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**.

3.3.3. Deleting a container

1. In the dashboard, select **Project > Object Store > Containers**
2. Browse for the container in the **Containers** section, and ensure all objects have been deleted (see Section 3.3.6, "Deleting an object").
3. Select **Delete Container** in the container's arrow menu.
4. Click **Delete Container** to confirm the container's removal.

3.3.4. Uploading an object

If you do not upload an actual file, the object is still created (as placeholder) and can later be used to upload the file.

1. In the dashboard, select **Project > Object Store > Containers**
2. Click the name of the container in which the uploaded object will be placed; 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:
   - Pseudo-folders can be specified in the name 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 object's contents.
5. Click **Upload Object**.

3.3.5. Copying an object

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**.

3.3.6. Deleting an object

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's removal.

CHAPTER 4. SHARED FILE SYSTEM SERVICE

With the OpenStack Shared File Systems service (manila) you can provision shared file systems that can be consumed by multiple compute instances.

4.1. BACK ENDS

When cloud administrators use OpenStack director to deploy the Shared File System service, they may choose either of the two supported back ends:

- CephFS via NFS Back End Guide for the Shared File System Service
- NetApp

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

4.2. CREATING AND MANAGING SHARE TYPES

When creating a share, share types are used to select an appropriate storage back end. OpenStack director configures the Shared File System service with a default share type named **default**, but does not itself create the share type.

1. After deploying the overcloud, as the cloud administrator, create this share type by running the following command:

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

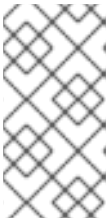
   The **<spec_driver_handles_share_servers>** parameter is a boolean value:

   - For CephFS via NFS, the value is **false**.
   - For NetApp back ends, the value can be **true** or **false**; set **<spec_driver_handles_share_servers>** to match the value of the **ManilaNetappDriverHandlesShareServers** parameter, as described in the NetApp Back End Guide for the Shared File System Service guide.
     The cloud administrator can add additional specifications to the default share type and create additional share types, if that is useful for multiple configured back ends.

     For example:

2. Set up the **default** share type to select a CephFS back end and an additional share type that picks a NetApp **driver_handles_share_servers=True** back end using the following commands:

   ```
   (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 tenants. It is also possible to create private share types for use within specific projects. To make private share types, or to set additional share-type options, see the Security and Hardening Guide .

4.2.1. Creating a share

Create a share by using a command similar to the following:

```
$ manila create [--share-type <SHARETYPE>] [--name <SHARENAME>] PROTO GB
```

Where:

- **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 NetApp, PROTO is **nfs** or **cifs**.
- **GB** is the size of the share in gigabytes.

For example, in Section 4.2, “Creating and Managing Share Types” , 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.

1. Using these share types, create a 10 GB NFS share named **share-01** on the CephFS back end by running the following command:

```
(user) [stack@undercloud-0 ~]$ manila create --name share-01 nfs 10

+---------------------------------------+--------------------------
| Property                              | Value                                +---------------------------------------+----
-----------------------
| status                                | creating
| share_type_name                       | default                              | 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   | False
| is_public                             | False                                | task_state                           | None
| snapshot_support                      | False                                | id                                   |
8c3bedd8-bc82-4100-a65d-53ec51b5fe81

| size                                  | 10                                   |
```

```
source_share_group_snapshot_member_id | None                         | user_id
| 19f014d7b5fd43519363c5bd75da864c

| name                          | share-01                      | share_type                    |
89415974-3f82-4a73-8efc-9a4f9970dc00

| has_replicas                  | False                         | replication_type              |
None                          | created_at                    | 2018-09-17T16:00:07.000000
| share_proto                   | NFS                           | mount_snapshot_support
| False                         | project_id                    |
b0434b7f2c5943e797a24edd958d95e6

| metadata                      | {}                            |+--------------------------------------+-----
----------------------
```

2. Optionally, create a 20 GB NFS share named **share-02** on the NetApp back end by running the following command:

```
(user) [stack@undercloud-0 ~]$ manila create --name share-02 --share-type netapp --share-
network mynet nfs 20

+---------------------------------------+--------------------------
| Property                      | Value                         +--------------------------------------+-----
----------------------
| status                        | creating
| share_type_name               | netapp
| description                   | None
| availability_zone             | None
| share_network_id              | mynet
| share_group_id                | None
| revert_to_snapshot_support    | True
| access_rules_status           | active
| snapshot_id                   | None
| create_share_from_snapshot_support   | True
| is_public                     | False
| task_state                    | None
| snapshot_support              | False
| id                            | db3bedd8-bc82-4100-a65d-53ec51b5cba3

| size                          | 20
| source_share_group_snapshot_member_id   | None
| user_id                       | 19f014d7b5fd43519363c5bd75da864c

| name                          | share-02
| share_type                    | abcde974-3f82-4a73-8efc-9a4f9970abab

| has_replicas                  | False
| replication_type              | None
| created_at                    | 2018-09-17T16:00:07.000000
| share_proto                   | NFS
| mount_snapshot_support        | False
| project_id                    | b0434b7f2c5943e797a24edd958d95e6

| metadata                      | {}  +--------------------------------------+--------------------------
```

4.2.2. Listing shares and exporting information

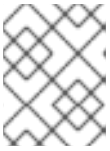
To verify that the shares were created successfully, complete the following steps:

1. Run the following command:

```
(user) [stack@undercloud-0 ~]$ manila list

+--------------------------------------+----------+-----+-----------+          | ID                                   |
Name     | ... | Status    ...
+--------------------------------------+----------+-----+-----------+
| 8c3bedd8-bc82-4100-a65d-53ec51b5fe81 | share-01 | ... | available ...
+--------------------------------------+----------+-----+-----------+
```

2. Run the **manila share-export-location-list** command to see the share's export locations:

```
(user) [stack@undercloud-0 ~]$ manila share-export-location-list share-01

 +-----------------------------------------------------------------
| Path
|  172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01
 +-----------------------------------------------------------------
```



NOTE

This information is used to mount the share in Section 4.3.1.1, "Mounting the share".

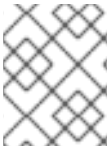
4.2.3. Ensuring network connectivity to the share

The Shared File System service serves storage over networks. Therefore, compute instances intended for mounting a file share must have network connectivity to one or more of the export locations for that share.

There are many ways to configure OpenStack networking with the Shared File System service, including using network plugins as described in Networking requirements for manila.

For back ends where **driver_handles_share_servers=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.

- For back ends where **driver_handles_share_servers=False**, a cloud administrator sets up the requisite networking in advance rather than dynamically in the Shared File System back end.
- For the CephFS via NFS back end, a cloud administrator deploys OpenStack director with isolated networks and environment arguments as documented in Installing OpenStack with CephFS via NFS 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 data center's isolated StorageNFS network.



NOTE

For a compute instance to connect to this shared provider network, the user must add an additional neutron port.

To ensure network connectivity to the share, complete the following steps:

1. Create a security group for the StorageNFS port that allows packets to egress the port (which is required to initiate an NFS mount) but that does not allow ingress packets for 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'
```

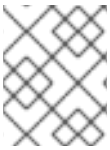
NOTE

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, notice that the compute instance is assigned a port with the IP address 172.17.5.160 on the StorageNFS network, which can be used to mount NFS shares when access is granted to that address for the share in question.



NOTE

Networking configuration on the compute instance may need to be adjusted and the services restarted for the compute instance to activate an interface with this address.

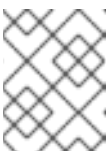
4.2.4. Granting share access

Before you can mount a share on an instance, you must grant the instance access to the share by using a command similar to the following:

```
# manila access-allow <SHARE> <ACCESSTYPE> --access-level <ACCESSLEVEL>
<CLIENTIDENTIFIER>
```

Where:

- **SHARE** - the share name or ID of the share created in Section 4.2.1, “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.



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.

- **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**

For example:

1. To grant read-write access to **share-01** to a compute instance with a StorageNFS network port with the IP address 172.17.5.160, run the following command:

```
(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
+-----------------+--------------------------------------+
| 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
+-----------------+--------------------------------------+
```

NOTE

Access to the share has its own ID (**ACCESSID**).

2. Enter the following command to verify that the access configuration was successful:

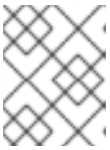
```
(user) [stack@undercloud-0 ~]$ manila access-list share-01

+--------------+-------------+--------------+--------------+--------+ ...
| id           | access_type | access_to    | access_level | state  | ...
+--------------+-------------+--------------+--------------+--------+
| 875c6251-... | ip          | 172.17.5.160 | rw           | active | ...
+--------------+-------------+--------------+--------------+---------+ ...
```

4.2.5. Revoking access to a share

Complete the following steps to revoke previously-granted access to a share:

1. Run the following command:

```
# manila access-deny <SHARE> <ACCESSID>
```



NOTE

In the example command, **<SHARE>** can be 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  | ...
+--------------+------------+--------------+--------------+--------+ ...
+--------------+------------+--------------+--------------+--------+ ...
```

4.3. MOUNTING A SHARE ON AN INSTANCE

After configuring the share to authenticate an instance, verify the functionality of the environment and then mount the share.

NOTE: NFS client packages supporting version 4.1 must be installed on the compute instance that mounts the shares.

4.3.1. Verifying the environment

To verify the functionality of the environment, complete the following steps:

1. Run the following command to get the virtual IP (VIP) for the NFS-Ganesha service:

```
(user) [stack@undercloud-0 ~]$ manila share-export-location-list share-01
  172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01
```

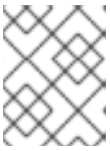
2. From the VM in which you will mount the share, ensure that the VIP is reachable via ping:

```
# ping 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
64 bytes from 172.17.5.13: icmp_seq=2 ttl=64 time=0.061 ms
^C
```

```
--- 172.17.5.13 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.048/0.054/0.061/0.009 ms
```

3. Verify the VIP is ready to respond to NFS rpcs on the proper port:

```
# rpcinfo -T tcp -a 172.17.5.13.8.1 100003 4
```



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.

4.3.1.1. Mounting the share

To mount the share from] on the instance from xref:shares-access[, complete the following steps:

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 using the export location from Section 4.2.2, "Listing shares and exporting information":

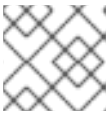
```
# mount.nfs -v  172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01
/mnt mount.nfs: timeout set for Wed Sep 19 09:14:46 2018                mount.nfs: trying
text-based options 'vers=4.2,addr=172.17.5.13,clientaddr=172.17.5.160'
172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-67d4999fbc01 on /mnt type nfs
# mount | grep mnt       172.17.5.13:/volumes/_nogroup/e840b4ae-6a04-49ee-9d6e-
67d4999fbc01 on /mnt type nfs4
(rw,relatime,vers=4.2,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo
=600,retrans=2,sec=sys,clientaddr=172.17.5.160,local_lock=none,addr=172.17.5.13)
```

4.3.2. Deleting a share

To delete a share, complete the following step:

1. Run the following command:

```
# 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
```

4.4. QUOTAS IN THE SHARED FILE SYSTEM SERVICE

To prevent system capacities from being exhausted without notification, you can set up quotas. Quotas are operational limits. To list the quotas for a project or user, use 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.

To see the usage statements, use the following commands:

```
# manila help quota-show
# manila help quota-update
# manila help quota-delete
```

4.5. TROUBLESHOOTING ASYNCHRONOUS FAILURES

If manila operations such as **create share** or **create share group** fail asynchronously, you can use the command line to query for more information about the error.

4.5.1. Scenario

In this example, the user wants 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.

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 **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 tenant 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, use 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, you can see that the Shared File System service failed to create the share because of a capabilities mismatch.

6. To view more message information, use 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 know about 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 |
+--------------------------------------+--------------+--------------------------------------+-----------+---
-----------------------------------------------------------------------------------------------+--------
---+----------------------------+
```

You can see that the service does not expect a share network for the share type 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. To ensure that the share was created successfully, use the **manila list** command:

```
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 |
+----+---------------+-------------+-----------+-------------+-----------+------------+
+----+---------------+-------------+-----------+-------------+-----------+------------+
```