Red Hat OpenStack Platform 16.1

CephFS Back End Guide for the Shared File System Service

Deploying the Shared File Systems service with native CephFS in a Red Hat OpenStack Platform Overcloud

Last Updated: 2023-08-28

Red Hat OpenStack Platform 16.1 CephFS Back End Guide for the Shared File System Service

Deploying the Shared File Systems service with native CephFS in a Red Hat OpenStack Platform Overcloud

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document describes the procedures to install, configure, and verify the Shared File Systems service (manila) with the native Red Hat Ceph File System (CephFS) back end for the Red Hat OpenStack Platform environment.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

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

CHAPTER 1. INTRODUCTION

IMPORTANT

This document pertains to the deployment and use of native CephFS to provide self-service Shared File Systems service (manila) in your Red Hat OpenStack Platform Cloud through the native CephFS NAS protocol. This type of deployment requires guest VM access to Ceph public network and infrastructure. Red Hat recommends that you deploy native CephFS with trusted OpenStack Platform tenants only, because it requires a permissive trust model that is not suitable for general purpose OpenStack Platform deployments.

For general purpose OpenStack Platform deployments that use a conventional tenant trust model, you can deploy CephFS through the NFS protocol. For more information about trust models, see Security considerations.

For more information about using CephFS through NFS, see Deploying the Shared File Systems service with CephFS through NFS.

With the Shared File Systems service (manila), you can provision shared file systems that multiple compute instances, bare metal nodes, or containers can consume.

CephFS is the highly scalable, open-source, distributed file system component of Ceph, a unified distributed storage platform. Ceph Storage implements object, block, and file storage using Reliable Autonomic Distributed Object Store (RADOS). CephFS, which is POSIX compatible, provides file access to a Ceph Storage cluster.

The Shared File Systems service enables users to create shares in CephFS and access them using the native Ceph File System protocol. The Shared File Systems service manages the life cycle of these shares from within OpenStack.

With this release, director can deploy the Shared File Systems with a native CephFS back end on the overcloud.

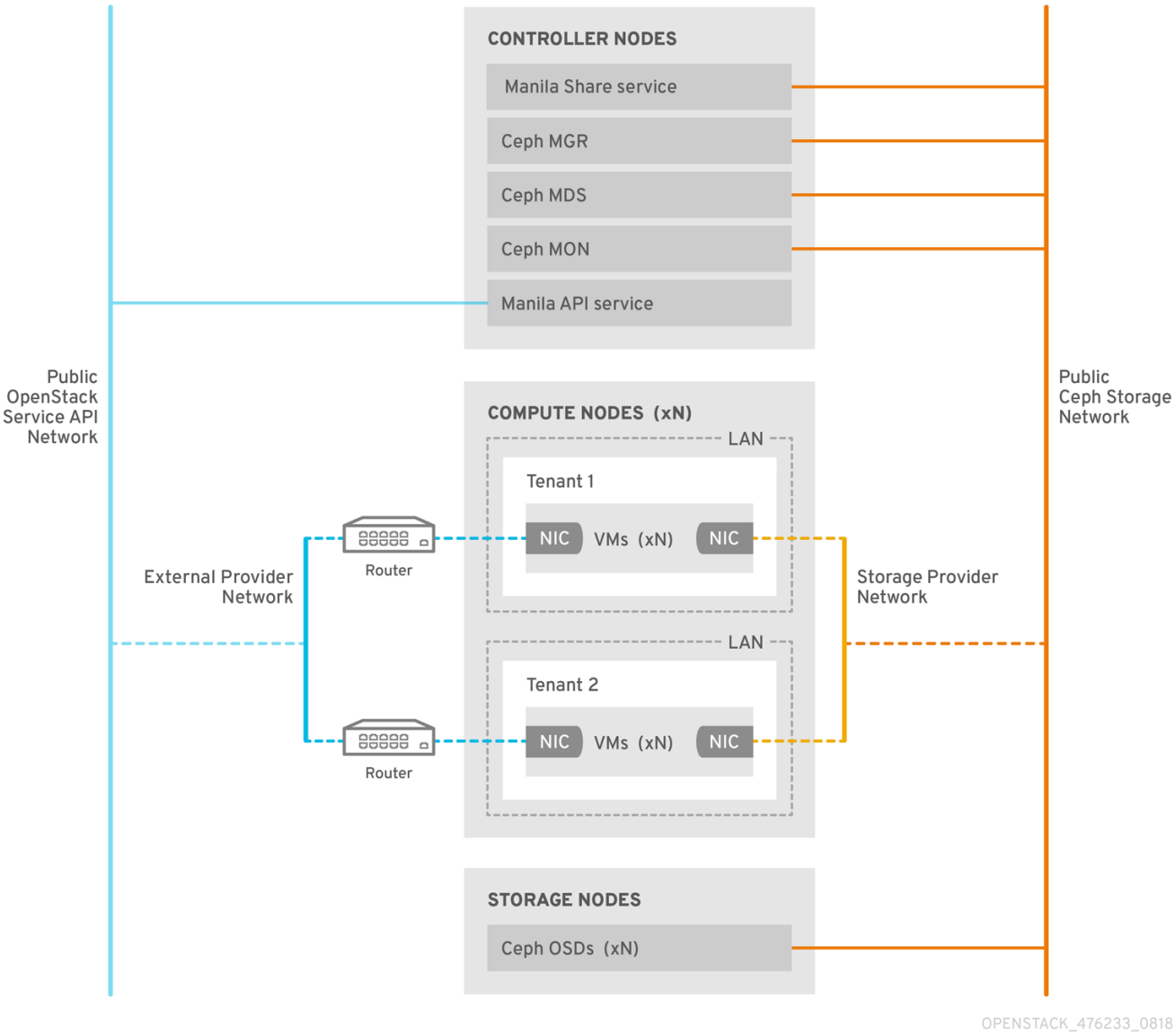
CHAPTER 2. CEPHFS WITH NATIVE DRIVER

The CephFS native driver combines the OpenStack Shared File Systems service (manila) and Red Hat Ceph Storage. When you use Red Hat OpenStack (RHOSP) director, the Controller nodes host the Ceph daemons, such as the manager, metadata servers (MDS), and monitors (MON) and the Shared File Systems services.

Compute nodes can host one or more projects. Projects(formerly known as tenants), which are represented in the following graphic by the white boxes, contain user-managed VMs, which are represented by gray boxes with two NICs. To access the ceph and manila daemons projects connect to the daemons over the public Ceph storage network. On this network, you can access data on the storage nodes provided by the Ceph Object Storage Daemons (OSDs). Instances (VMs) that are hosted on the project boot with two NICs: one dedicated to the storage provider network and the second to project-owned routers to the external provider network.

The storage provider network connects the VMs that run on the projects to the public Ceph storage network. The Ceph public network provides back-end access to the Ceph object storage nodes, metadata servers (MDS), and Controller nodes.

Using the native driver, CephFS relies on cooperation with the clients and servers to enforce quotas, guarantee project isolation, and for security. CephFS with the native driver works well in an environment with trusted end users on a private cloud. This configuration requires software that is running under user control to cooperate and work correctly.



2.1. SECURITY CONSIDERATIONS

The native CephFS back end requires a permissive trust model for OpenStack Platform tenants. This trust model is not appropriate for general purpose OpenStack Platform clouds that deliberately block users from directly accessing the infrastructure behind the services that the OpenStack Platform provides.

With native CephFS, user compute instances (VMs) must connect directly to the Ceph public network to access shares. Ceph infrastructure service daemons are deployed on the Ceph public network where they are exposed to user VMs. CephFS clients that run on user VMs interact cooperatively with the Ceph service daemons, and they interact directly with RADOS to read and write file data blocks.

CephFS quotas, which enforce Shared File Systems (manila) share sizes, are enforced on the client side, such as on VMs that are owned by Red Hat OpenStack Platform (RHOSP) users. Client-side software versions on user VMs might not be current, which can leave critical cloud infrastructure vulnerable to malicious or inadvertently harmful software that targets the Ceph service ports.

For these reasons, Red Hat recommends that you deploy native CephFS as a back end only in environments in which trusted users maintain the latest versions of client-side software. Users must also ensure that no software runs on their VMs that can impact the Ceph Storage infrastructure.

For a general purpose RHOSP deployment that serves many untrusted users, Red Hat recommends that you deploy CephFS through NFS. For more information about using CephFS through NFS, see Deploying the Shared File Systems service with CephFS through NFS .

Users might not keep client-side software current, and they might fail to exclude harmful software from their VMs, but using CephFS through NFS, users only have access to the public side of an NFS server, not to the Ceph infrastructure itself. NFS does not require the same kind of cooperative client and, in the worst case, an attack from a user VM could damage the NFS gateway without damaging the Ceph Storage infrastructure behind it.

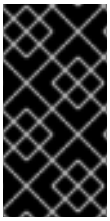
Red Hat recommends the following security measures:

- Configure the Storage network as a provider network.
- Impose role-based access control (RBAC) policies to secure the Storage provider network.
- Create a private share type for the native CephFS back end and expose it only to trusted tenants.

CHAPTER 3. NATIVE CEPHFS DEPLOYMENT

A typical native Ceph file system (CephFS) installation in a Red Hat OpenStack Platform (RHOSP) environment includes the following components:

- RHOSP Controller nodes that run containerized Ceph metadata server (MDS), Ceph monitor (MON) and Shared File Systems (manila) services. Some of these services can coexist on the same node or they can have one or more dedicated nodes.
- Ceph Storage cluster with containerized object storage daemons (OSDs) that run on Ceph Storage nodes.
- An isolated storage network that serves as the Ceph public network on which the clients can communicate with Ceph service daemons. To facilitate this, the storage network is made available as a provider network for users to connect their VMs and mount CephFS shares.



IMPORTANT

You cannot use the Shared File Systems service (manila) with the CephFS native driver to serve shares to OpenShift Container Platform through Manila CSI, because Red Hat does not support this type of deployment. For more information, contact Red Hat Support.

The Shared File Systems (manila) service provides APIs that allow the tenants to request file system shares, which are fulfilled by driver modules. The driver for Red Hat CephFS, **manila.share.drivers.cephfs.driver.CephFSDriver**, allows the Shared File Systems service to use native CephFS as a back end.

When director deploys the Shared File Systems service with a CephFS back end on the overcloud, it automatically creates the required data center storage network. However, you must create the corresponding storage provider network on the overcloud. For more information, see Chapter 4, *Completing post-deployment configuration*. For more information about network planning, see Overcloud networks in the *Director Installation and Usage* guide.

Although you can manually configure the Shared File Systems service by editing the **/var/lib/config-data/puppet-generated/manila/etc/manila/manila.conf** file for the node, any settings can be overwritten by the Red Hat OpenStack Platform director in future overcloud updates. Red Hat only supports deployments of the Shared File Systems service that are managed by director.

This section describes how to install native CephFS in an integrated deployment managed by director.

3.1. REQUIREMENTS

You can deploy a native CephFS back end with new or existing Red Hat OpenStack Platform (RHOSP) environments if you observe the following requirements:

- Red Hat OpenStack Platform version 16.1 or later.
- Configure a new Red Hat Ceph Storage cluster at the same time as the native CephFS back end. For information about how to deploy Ceph Storage, see https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html/deploying_an_overcloud_with_containerized_red_hat_c

IMPORTANT

The RHOSP Shared File Systems service with the native CephFS back end is supported for use with Red Hat Ceph Storage version 4.1 or later. For more information about how to determine the version of Ceph Storage installed on your system, see Red Hat Ceph Storage releases and corresponding Ceph package versions.

- Install the Shared File Systems service on a Controller node. This is the default behavior.
- Use only a single instance of a CephFS back end for the Shared File Systems service.

3.2. FILE SHARES

File shares are handled differently between the Shared File Systems service (manila), Ceph File System (CephFS), and CephFS through NFS.

The Shared File Systems service provides shares, where a share is an individual file system namespace and a unit of storage with a defined size. Shared file system storage inherently allows multiple clients to connect, read, and write data to any given share, but you must give each client access to the share through the Shared File Systems service access control APIs before they can connect.

With CephFS, a share is considered a directory with a defined quota and a layout that points to a particular storage pool or namespace. CephFS quotas limit the size of a directory to the size share that the Shared File Systems service creates. Access to Ceph shares is determined by MDS authentication capabilities.

With native CephFS, file shares are provisioned and accessed through the CephFS protocol. Access control is performed with a CephX authentication scheme that uses CephFS usernames.

3.3. ISOLATED NETWORK USED BY NATIVE CEPHFS

Native CephFS deployments use the isolated storage network deployed by director as the Ceph public network. Clients use this network to communicate with various Ceph infrastructure service daemons. For more information about isolating networks, see Basic network isolation in the *Advanced Overcloud Customization* guide.

3.4. INSTALLING THE CEPH-ANSIBLE PACKAGE

Install the **ceph-ansible** package to be installed on an undercloud node to deploy containerized Ceph.

Procedure

1. Log in to an undercloud node as the **stack** user.
2. Install the ceph-ansible package:

```
[stack@undercloud-0 ~]$ sudo dnf install -y ceph-ansible
[stack@undercloud-0 ~]$ sudo dnf list ceph-ansible
...
Installed Packages
ceph-ansible.noarch    4.0.23-1.el8cp      @rhelosp-ceph-4-tools
```

3.5. DEPLOYING THE ENVIRONMENT

When you are ready to deploy the environment, use the **openstack overcloud deploy** command with the custom environments and roles required to configure the native CephFS back end.

The **openstack overcloud deploy** command has the following options in addition to other required options.

Action	Option	Additional Information
Specify the network configuration with **network_data.yaml**	[filename] **-n /usr/share/openstack-tripleo-heat-templates/network_data.yaml**	You can use a custom environment file to override values for the default networks specified in this network data environment file. This is the default network data file that is available when you use isolated networks. You can omit this file from the **openstack overcloud deploy** command for brevity.
Deploy the Ceph daemons with **ceph-ansible.yaml**	**-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**	Initiating Overcloud Deployment in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide
Deploy the Ceph metadata server with **ceph-mds.yaml**	**-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml**	Initiating Overcloud Deployment in the *Deploying an Overcloud with Containerized Red Hat Ceph* guide
Deploy the manila service with the native CephFS back end.	**-e /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsnative-config.yaml**	Section 3.5.1, “Environment file”

The following example shows an **openstack overcloud deploy command** that includes options to deploy a Ceph cluster, Ceph MDS, the native CephFS back end, and the networks required for the Ceph cluster:

```
[stack@undercloud ~]$ openstack overcloud deploy \
...
-n /usr/share/openstack-tripleo-heat-templates/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml   \
-e /home/stack/network-environment.yaml  \
-e/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml  \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml  \
-e /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsnative-config.yaml
```

For more information about the **openstack overcloud deploy** command, see Deployment command in the *Director Installation and Usage* guide.

3.5.1. Environment file

The integrated environment file that defines a native CephFS back end is located in the following path of an undercloud node: **/usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsnative-config.yaml**.

The **manila-cephfsnative-config.yaml** environment file contains settings relevant to the deployment of the Shared File Systems service. The back end default settings should work for most environments.

The example shows the default values that director uses during deployment of the Shared File Systems service:

```
[stack@undercloud ~]$ cat /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsnative-config.yaml

# A Heat environment file which can be used to enable a
# a Manila CephFS Native driver backend.
resource_registry:
  OS::TripleO::Services::ManilaApi: ../deployment/manila/manila-api-container-puppet.yaml
  OS::TripleO::Services::ManilaScheduler: ../deployment/manila/manila-scheduler-container-puppet.yaml
  # Only manila-share is pacemaker managed:
  OS::TripleO::Services::ManilaShare: ../deployment/manila/manila-share-pacemaker-puppet.yaml
  OS::TripleO::Services::ManilaBackendCephFs: ../deployment/manila/manila-backend-cephfs.yaml

parameter_defaults:
  ManilaCephFSBackendName: cephfs (1)
  ManilaCephFSDriverHandlesShareServers: false (2)
  ManilaCephFSCephFSAuthId: 'manila' (3)
  ManilaCephFSCephFSEnableSnapshots: true (4)
  ManilaCephFSCephVolumeMode: '0755' (5)
  # manila cephfs driver supports either native cephfs backend - 'CEPHFS'
  # (users mount shares directly from ceph cluster), or nfs-ganesha backend -
  # 'NFS' (users mount shares through nfs-ganesha server)
  ManilaCephFSCephFSProtocolHelperType: 'CEPHFS' (6)
```

The **parameter_defaults** header signifies the start of the configuration. Specifically, settings under this header let you override default values set in **resource_registry**. This includes values set by **OS::Tripleo::Services::ManilaBackendCephFs**, which sets defaults for a CephFS back end.

1. **ManilaCephFSBackendName** sets the name of the manila configuration of your CephFS backend. In this case, the default back end name is **cephfs**.

2. **ManilaCephFSDriverHandlesShareServers** controls the lifecycle of the share server. When set to **false**, the driver does not handle the lifecycle. This is the only supported option for CephFS back ends.

3. **ManilaCephFSCephFSAuthId** defines the Ceph auth ID that the director creates for the manila service to access the Ceph cluster.

4. **ManilaCephFSCephFSEnableSnapshots** controls snapshot activation. Snapshots are supported With Ceph Storage 4.1 and later, but the value of this parameter defaults to **false**. You can set the value to **true** to ensure that the driver reports the **snapshot_support** capability to the manila scheduler.

5. **ManilaCephFSCephVolumeMode** controls the UNIX permissions to set against the manila share created on the native CephFS back end. The value defaults to **755**.

6 **ManilaCephFSCephFSProtocolHelperType** must be set to **CEPHFS** to use the native CephFS driver.

For more information about environment files, see Environment Files in the *Director Installation and Usage* guide.

CHAPTER 4. COMPLETING POST-DEPLOYMENT CONFIGURATION

You must complete two post-deployment configuration tasks before you create NFS shares, grant user access, and mount NFS shares.

- Map the Networking service (neutron) storage network to the isolated data center storage network.
- Make the storage provider network available to trusted tenants only through custom role based access control (RBAC). Do not share the storage provider network globally.
- Create a private share type.
- Grant access to specific trusted tenants.

After you complete these steps, the tenant compute instances can create, allow access to, and mount native CephFS shares.

4.1. CREATING THE STORAGE PROVIDER NETWORK

You must map the new isolated storage network to a Networking (neutron) provider network. The Compute VMs attach to the network to access native CephFS share export locations.

For information about network security with the Shared File Systems service, see Hardening the Shared File Systems Service in the *Security and Hardening Guide*.

Procedure

The **openstack network create** command defines the configuration for the storage neutron network.

1. From an undercloud node, enter the following command:

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

2. On an undercloud node, create the storage network:

```
(overcloud) [stack@undercloud-0 ~]$ openstack network create Storage --provider-network-type vlan --provider-physical-network datacentre --provider-segment 30
```

   You can enter this command with the following options:

   - For the **--provider-physical-network** option, use the default value **datacentre**, unless you set another tag for the br-isolated bridge through NeutronBridgeMappings in your tripleo-heat-templates.
   - For the **--provider-segment** option, use the value set for the Storage isolated network in your network environment file. If this was not customized, the default environment file is **/usr/share/openstack-tripleo-heat-templates/network_data.yaml**. The VLAN associated with the Storage network value is **30** unless you modified the isolated network definitions.
   - For the **--provider-network-type** option, use the value **vlan**.

4.2. CONFIGURING THE STORAGE PROVIDER NETWORK

Create a corresponding **StorageSubnet** on the neutron provider network. Ensure that the subnet is the same for the **storage_subnet** in the undercloud, and that the allocation range for the storage subnet and the corresponding undercloud subnet do not overlap.

Requirements

- The start and ending IP range for the allocation pool
- The subnet IP range

Procedure

1. From an undercloud node, enter the following command:

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

2. Use the sample command to provision the network. Update the values to suit your environment.

```
(overcloud) [stack@undercloud-0 ~]$ openstack subnet create \
--allocation-pool start=172.17.3.10,end=172.17.3.149 \
--dhcp \
--network Storage \
--subnet-range 172.17.3.0/24 \
--gateway none StorageSubnet
```

   - For the **--allocation-pool** option, replace the **start=172.17.3.10,end=172.17.3.149** IP values with the IP values for your network.
   - For the **--subnet-range** option, replace the **172.17.3.0/24** subnet range with the subnet range for your network.

4.3. CONFIGURING ROLE-BASED ACCESS CONTROL FOR THE STORAGE PROVIDER NETWORK

After you identify the trusted tenants or projects that can use the storage network, configure role-based access control (RBAC) rules for them through the Networking service (neutron).

Requirements

Names of the projects that need access to the storage network

Procedure

1. From an undercloud node, enter the following command:

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

2. Identify the projects that require access:

```
(overcloud) [stack@undercloud-0 ~]$ openstack project list
+----------------------------------+---------+
| ID                               | Name    |
+----------------------------------+---------+
| 06f1068f79d2400b88d1c2c33eacea87 | demo    |
```

```
| 5038dde12dfb44fdaa0b3ee4bfe487ce | service |
| 820e2d9c956644c2b1530b514127fd0d | admin   |
+----------------------------------+---------+
```

3. Create network RBAC rules with the desired projects:

```
(overcloud) [stack@undercloud-0 ~]$ openstack network rbac create \
--action access_as_shared Storage \
--type network \
--target-project demo
```

Repeat this step for all of the projects that require access to the storage network.

4.4. CONFIGURING A DEFAULT SHARE TYPE

You can use the Shared File Systems service to define share types that you can use to create shares with specific settings. Share types work like Block Storage volume types. Each type has associated settings, for example, extra specifications. When you invoke the type during share creation the settings apply to the shared file system.

To secure the native CephFS back end against untrusted users, Red Hat recommends that you do not create a default share type. When a default share type does not exist, users are forced to specify a share type, and trusted users can use a custom private share type to which they have exclusive access rights.

If you must create a default share type for untrusted tenants, you can steer provisioning away from the native CephFS back end.

Procedure

1. From an undercloud node, enter the following command:

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

2. Set an extra specification on the share type:

```
(overcloud) [stack@undercloud-0 ~] manila type-create default False
(overcloud) [stack@undercloud-0 ~] manila type-key default set share_backend_name='s!=
cephfs'
```

3. Create a private share type and provide trusted tenants with access to this share type:

```
(overcloud) [stack@undercloud-0 ~]$ manila type-create --is-public false nativecephfstype
false
(overcloud) [stack@undercloud-0 ~]$ manila type-key nativecephfstype set
share_backend_name='cephfs'
(overcloud) [stack@undercloud-0 ~]$ manila type-access-add nativecephfstype
<trusted_tenant_project_id>
```

For more information about share types, see Creating a share type in the *Storage Guide*.

CHAPTER 5. VERIFYING SUCCESSFUL NATIVE CEPHFS BACK END DEPLOYMENT

Deploying native CephFS as a back end of the Shared File Systems service (manila) adds the following new elements to the overcloud environment:

- Storage provider network
- Ceph MDS service on the Controller nodes

For more information about using the Shared File Systems service with native CephFS, see Shared File Systems service in the *Storage Guide*.

The cloud administrator must verify the stability of the native CephFS environment before making it available to service users.

Prerequisites

- Complete the steps in Chapter 3, *Native CephFS deployment*

5.1. VERIFYING CREATION OF ISOLATED STORAGE NETWORK

The **network_data.yaml** file used to deploy native CephFS as a Shared File Systems service back end creates the storage VLAN. Use this procedure to confirm you successfully created the storage VLAN.

Procedure

1. Log in to one of the Controller nodes in the overcloud.
2. Check the connected networks and verify the existence of the VLAN as set in the **network_data.yaml** file:

```
$ ip a
8: vlan30: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
    link/ether 52:9c:82:7a:d4:75 brd ff:ff:ff:ff:ff:ff
    inet 172.17.3.144/24 brd 172.17.3.255 scope global vlan30
       valid_lft forever preferred_lft forever
    inet6 fe80::509c:82ff:fe7a:d475/64 scope link
       valid_lft forever preferred_lft forever
```

5.2. VERIFYING CEPH MDS SERVICE

Use the **systemctl status** command to verify the Ceph MDS service status.

Procedure

- Enter the following command on all Controller nodes to check the status of the MDS container:

```
$ systemctl status ceph-mds<@CONTROLLER-HOST>
```

Example:

```
$ systemctl status ceph-mds@controller-0.service

ceph-mds@controller-0.service - Ceph MDS
   Loaded: loaded (/etc/systemd/system/ceph-mds@.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2018-09-18 20:11:53 UTC; 6 days ago
 Main PID: 65066 (conmon)
   Tasks: 16 (limit: 204320)
   Memory: 38.2M
   CGroup: /system.slice/system-ceph\x2dmds.slice/ceph-mds@controller-0.service
           └─60921 /usr/bin/podman run --rm --net=host --memory=32000m --cpus=4 -v
/var/lib/ceph:/var/lib/ceph:z -v /etc/ceph:/etc/ceph:z -v
/var/run/ceph:/var/run/ceph:z -v /etc/localtime:/etc/localtime:ro>
```

5.3. VERIFYING CEPH CLUSTER STATUS

Complete the following steps to verify the Ceph cluster status.

Procedure

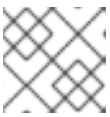
1. Log in to any Controller node.
2. From the Ceph monitor daemon, enter the following command:

```
$ sudo podman exec ceph-mon-controller-0 ceph -s
  cluster:
    id:     670dc288-cd36-4772-a4fc-47287f8e2ebf
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum controller-1,controller-2,controller-0 (age 14h)
    mgr: controller-1(active, since 8w), standbys: controller-0, controller-2
    mds: cephfs:1 {0=controller-2=up:active} 2 up:standby
    osd: 15 osds: 15 up (since 8w), 15 in (since 8w)

  task status:
    scrub status:
        mds.controller-2: idle

  data:
    pools: 6 pools, 192 pgs
    objects: 309 objects, 1.6 GiB
    usage: 21 GiB used, 144 GiB / 165 GiB avail
    pgs: 192 active+clean
```



NOTE

There is one active MDS and two MDSs on standby.

3. To see a detailed status of the Ceph File System, enter the following command:

```
$ sudo ceph fs ls

name: cephfs metadata pool: manila_metadata, data pools: [manila_data]
```

NOTE

In this example output, *cephfs* is the name of Ceph File System that director creates to host CephFS shares that users create through the Shared File Systems service.

5.4. VERIFYING MANILA-SHARE SERVICE STATUS

Complete the following step to verify the status of the manila-share service.

Procedure

1. Enter the following command from one of the Controller nodes to confirm that **openstack-manila-share** started:

```
$ sudo pcs status resources | grep manila

* Container bundle: openstack-manila-share [cluster.common.tag/rhosp16-openstack-manila-share:pcmklatest]:
* openstack-manila-share-podman-0 (ocf::heartbeat:podman): Started controller-0
```

5.5. VERIFYING MANILA-API SERVICES ACKNOWLEDGES SCHEDULER AND SHARE SERVICES

Complete the following steps to confirm the **manila-api** service acknowledges the scheduler and share services.

Procedure

1. Log in to the undercloud.
2. Enter the following command:

```
$ source /home/stack/overcloudrc
```

3. Enter the following command to confirm **manila-scheduler** and **manila-share** are enabled:

```
$ manila service-list

| Id | Binary          | Host              | Zone | Status | State | Updated_at |

| 2 | manila-scheduler | hostgroup         | nova | enabled | up | 2018-08-08T04:15:03.000000 |
| 5 | manila-share | hostgroup@cephfs | nova | enabled | up | 2018-08-08T04:15:03.000000 |
```