Red Hat OpenStack Platform 9
Shared File System Service Deployment Guide

A Guide to Deploying the Shared File System Service in a Red Hat OpenStack Platform Overcloud

OpenStack Team

Red Hat OpenStack Platform 9 Shared File System Service Deployment Guide

A Guide to Deploying the Shared File System Service in a Red Hat OpenStack Platform Overcloud

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document describes how to deploy, configure, and test the OpenStack Shared File System Service, which offers file sharing as a service based on the upstream OpenStack Manila project. The test case presented in this document uses a CephFS back end, enabled through the manila.share.drivers.cephfs.cephfs_native driver. The OpenStack Shared File System Service is offered in this release as a Technology Preview, and should not be deployed in a production environment. For more information about Technology Previews, see:

Table of Contents

1. INTRODUCTION

Important

The OpenStack Shared File System service and Red Hat Ceph file system (CephFS) are available only as *Technology Previews*, and therefore not fully supported by Red Hat. The deployment scenario described in this document 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.

The OpenStack File Share Service (**openstack-manila**) provides the means to easily provision shared file systems that can be consumed by multiple instances. In the past, OpenStack users needed to manually deploy shared file systems before mounting them on instances. The OpenStack File Share Service, on the other hand, allows users to easily provision shares from a pre-configured storage pool, ready to be mounted securely. This pool, in turn, can be independently managed and scaled to meet demand.

The OpenStack File Share Service also allows administrators to define settings for different types of shares (namely, share types), in the same way that the OpenStack Block Storage service uses *volume types*. In addition, the OpenStack File Share Service service also provides the means to manage access, security, and snapshots for provisioned shares.

The test scenario presented in this document uses CephFS as a back end. This is implemented through the CephFS Native driver, namely **manila.share.drivers.cephfs.cephfs_native**. This driver allows instances to consume shared file systems through the Ceph network protocol.

Currently, the Shared File System Service can only be deployed manually or through Packstack. The service is not integrated yet into the Red Hat OpenStack Platform director. As such, when deploying the Shared File System Service on the overcloud, any subsequent overcloud updates may disable the service.

For this release, we recommend that you test the Shared File System service on a single-node Packstack deployment of Red Hat OpenStack Platform. Instructions for doing so are provided later in Section 3, “Deploy OpenStack with the File Share Service Enabled”.

2. REQUIREMENTS

The test scenario described in this document requires the following:

1. A Red Hat Ceph Storage cluster, with at least one Metadata Server (MDS) running. See the Red Hat Ceph Storage Administration Guide for information.
2. A Red Hat Ceph file system (CephFS). This file system is available as a Technology Preview on Red Hat Ceph 2.0. See Red Hat Ceph File System Guide (Technology Preview) for instructions.

Note

The Red Hat Ceph features and packages required for this test scenario are available in Red Hat Ceph 2.0. See the Red Hat Ceph Storage Release Notes for subscription details.

3. The Shared File System service and all instances that will be accessing CephFS shares must have network connectivity to the public network of the Red Hat Ceph Storage cluster.

In this test scenario, we will use Packstack to deploy Red Hat OpenStack Platform with the Shared File System service already installed. See Section 3, “Deploy OpenStack with the File Share Service Enabled” for instructions.

2.1. Limitations and Restrictions

Given the current state of the involved components, the test scenario in this document has the following limitations and restrictions:

1. Untrusted instance users pose a security risk to the Ceph Storage cluster, as they would have direct access to the public network of the Ceph Storage cluster. Ensure that the cluster you are using is quarantined from the production environment, and that only trusted users have access to the test environment.
2. This release only allows **read-write** access to shares.
3. With a CephFS provider, *share size quotas* are enforced on the *client side*; in this test scenario, through the **ceph-fuse** client (Section 7.4, “Mount the Share through the FUSE Client”). It is up to the client to respect share size quotas. For more information, see FUSE Client Configuration Reference (from the Red Hat Ceph Storage Administration Guide) and CephFS Quotas.

3. DEPLOY OPENSTACK WITH THE FILE SHARE SERVICE ENABLED

You can deploy OpenStack with the File Share Service enabled using the **Packstack** utility. Doing so will allow **Packstack** to configure the endpoints, identities, databases, and other settings/components required by the File Share Service:

1. Log in as *root* to the node that should host OpenStack and the File Share service.
2. Register the node and enable the required repositories. This procedure requires a Red Hat Subscription:

```
# subscription-manager register
# subscription-manager subscribe --auto
# subscription-manager repos --disable=*
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-rh-common-rpms
# subscription-manager repos --enable=rhel-7-server-openstack-9-rpms
```

3. Install the **yum** utility packages and update all the packages:

```
# yum install -y yum-utils
# yum update -y
```

4. Disable **NetworkManager** and reboot the system:

```
# systemctl disable NetworkManager
# reboot
```

5. After rebooting the system, install the **Packstack** utility:

```
# yum install -y openstack-packstack
```

6. Create an *answer file* named answer.txt for **Packstack**. The answer file allows you to configure the OpenStack deployment:

```
# packstack --gen-answer-file=answer.txt
```

7. Enable the File Share Service in the answer file. To do so, open answer.txt and set the CONFIG_MANILA_INSTALL option to y:

```
# Specify 'y' to install OpenStack Shared File System (manila). ['y', #
'n']
CONFIG_MANILA_INSTALL=y
```

8. Run **Packstack** with answer.txt:

```
# packstack --answer-file=answer.txt
```

Note

For detailed instructions on deploying a single-node, all-in-one instance of OpenStack (**without** the File Sharing Service enabled), see Evaluating OpenStack: Single-Node Deployment.

4. DEFINE CEPHFS BACK END

To define the CephFS back end for the Shared File System Service, add a new section for it in */etc/manila/manila.conf*. Use the following snippet as a reference for defining your back end:

```
[mycephbackend] # 1
share_backend_name = mycephbackend
driver_handles_share_servers = False # 2
share_driver = manila.share.drivers.cephfs.cephfs_native.CephFSNativeDriver
cephfs_conf_path = /etc/ceph/ceph.conf
cephfs_auth_id = manila
cephfs_cluster_name = ceph
```

1

The section title (**[mycephbackend]**) and **share_backend_name** define the name of the back end. Use this value later on to enable the back end and create a share type for it (as in Section 7.1, “Create a Share Type for the CephFS Back End”).

2

The **driver_handles_share_servers** setting specifies whether or not the back end should use the driver to handle the life cycle of share servers. As the CephFS back end does not create or destroy **manila** share servers, set this to **False**.

After adding the settings for the back end, enable it by:

- Specifying it in the **enabled_share_backends** setting (in the **[DEFAULT]** section), and
- Setting **CEPHFS** as the only protocol defined in **enabled_share_protocols** (also in the **[DEFAULT]** section). This is the protocol required by the **manila.share.drivers.cephfs.cephfs_native.CephFSNativeDriver** driver.

```
[DEFAULT]
...
enabled_share_backends = mycephbackend
...
enabled_share_protocols = CEPHFS
```

5. AUTHORIZE THE DRIVER TO COMMUNICATE WITH CEPH

After defining the back end, configure the driver to communicate with the Ceph back end. To do so, log in to any of the nodes of your Red Hat Ceph cluster. From there, create a Ceph identity (**client.manila**) for the Shared File System Service:

```
# read -d '' MON_CAPS << EOF
allow r,
allow command "auth del",
allow command "auth caps",
allow command "auth get",
allow command "auth get-or-create"
EOF

# ceph auth get-or-create client.manila -o manila.keyring \
    mds 'allow \' \*
    osd 'allow rw' \
    mon "$MON_CAPS"
```

These commands will create a *manila.keyring* file. Copy this file along with the */etc/ceph/ceph.conf* to */etc/ceph/* in the OpenStack node (where the Shared File System Service is hosted).

Once the */etc/ceph/ceph.conf* and *manila.keyring* file are copied to the OpenStack node, log back in to that node. From there, set the **manila** user as the owner of both files:

```
# chown manila /etc/ceph/ceph.conf
# chown manila /etc/ceph/manila.keyring
```

Finally, enable the Ceph identity you created earlier (**client.manila**). To do so, add the following section to the */etc/ceph/ceph.conf* file:

```
[client.manila]
client mount uid = 0
client mount gid = 0
log file = /var/log/manila/ceph-client.manila.log
admin socket = /var/run/ceph/ceph-$name.$pid.asok
keyring = /etc/ceph/manila.keyring
```

6. RESTART THE SHARED FILE SYSTEM SERVICE

After configuring the Shared File System service, restart it to apply your settings:

```
# systemctl restart openstack-manila-api
# systemctl restart openstack-manila-share
# systemctl restart openstack-manila-scheduler
```

Then, check whether each service launched successfully:

```
# systemctl status openstack-manila-api
openstack-manila-api.service - OpenStack Manila API Server
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-api.service; enabled;
vendor preset: disabled)
Active: active (running) since Fri 2016-06-03 11:57:46 AEST; 32s ago
[...]

# systemctl status openstack-manila-share
openstack-manila-share.service - OpenStack Manila Share Service
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-share.service;
enabled; vendor preset: disabled)
Active: active (running) since Fri 2016-06-03 11:57:47 AEST; 31s ago
[...]

# systemctl status openstack-manila-scheduler
openstack-manila-scheduler.service - OpenStack Manila Scheduler
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-scheduler.service;
enabled; vendor preset: disabled)
Active: active (running) since Fri 2016-06-03 11:57:47 AEST; 31s ago
[...]
```

7. TEST THE SHARED FILE SYSTEM SERVICE

At this point, the Shared File System Service should already be deployed with a CephFS back end. In addition, you should also have a Compute instance ready to mount shares. For instructions on how to create an instance, see Manage Instances from the Instances and Images Guide.

Perform the following steps to finalize configuration and test the service.

7.1. Create a Share Type for the CephFS Back End

The Shared File System service allows you 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 (namely, extra specifications), and invoking the type during share creation applies those settings.

When creating a share on a non-default back end, you need to explicitly specify which back end to use. To make the process seamless for users, create a share type and associate it with the **share_backend_name** value of your back end. In Section 4, “Define CephFS Back End”, this value is **mycephbackend**.

To create a share type named *TYPENAME*, run the following as an OpenStack admin:

```
# manila type-create TYPENAME DHSS
```

DHSS specifies whether the share type will use drivers that that handle share servers. This should match the value set in **driver_handles_share_servers** of your back end definition (from Section 4, “Define CephFS Back End”). With **driver_handles_share_servers=False**, *DHSS* should also be false. So, to create a share type called **cephsharetype1**:

```
# manila type-create cephsharetype1 false
```

Next, associate the **cephsharetype1** type to a specific back end. You can specify the back end through its **share_backend_name** value. For example, to associate the share type **cephsharetype1** to the back end from Section 4, “Define CephFS Back End”, run:

```
# manila type-key cephsharetype1 set share_backend_name='mycephbackend'
```

Users should now be able to invoke the **cephsharetype1** type to create a share from the CephFS back end (namely, **mycephbackend**).

7.2. Create a Share

To create a share, run:

```
# manila create --share-type SHARETYPE --name SHARENAME SHAREPROTO SIZE
```

SHARETYPE

The share type whose settings you want to invoke (see Section 7.1, “Create a Share Type for the CephFS Back End”).

SHARENAME

The name of the share.

SHAREPROTO

The protocol the share should use. In this scenario, this should be **cephfs**.

SIZE

The size of the share.

For example, to create a 1GB share named **myshare** using the share type from Section 7.1, “Create a Share Type for the CephFS Back End”:

```
# manila create --share-type cephsharetype1 --name myshare cephfs 1
```

Afterward, list the export location of the share:

```
# manila share-export-location-list myshare
```

The output of this command contains the *IP address and port* of the Ceph cluster’s monitoring node (or nodes), along with the path of the exported share. This information will be used later on to mount the share (in Section 7.4, “Mount the Share through the FUSE Client”) on an instance.

```
# manila share-export-location-list myshare
+--------------------------------------+--------------------------------------
--------------------------------------+---------+
|                  ID                  |                                  Path
```

```
|Preferred|
+-----------------------------------+---------------------------------
-----------------------------------+---------+
|3aaec994-a7ba-4350-995b-
9d61097c7f37|10.35.160.43:6789:/volumes/_nogroup/9096ddf0-405e-49dd-b7c8-
1c8815fb1f6e|  False  |
+-----------------------------------+---------------------------------
-----------------------------------+---------+
```

7.3. Configure Share Access

Before you can mount the shares on your Compute instance, you need to authorize the instance first. Doing so involves two steps:

1. Creating an authorized *client identity* that the Compute instance can use to access Ceph shares.
2. Retrieve the client identity's credentials and provide it to the Compute instance, ie. the client. This, in turn, will allow the client to mount shares from the Ceph cluster.

7.3.1. Create an Authorized Client Identity

To do so, first create an authorized *client identity* for the Compute instance. Run the following from the OpenStack node:

```
# manila access-allow SHARENAME IDENTTYPE CLIENTIDENT
```

SHARENAME

The name or ID of the share to which you want to grant access (for example, **myshare** from Section 7.2, "Create a Share").

IDENTTYPE

The identifying method that the Shared File System Service should use to authenticate a share user or Compute instance.

CLIENTIDENT

The client identity used by the chosen *IDENTTYPE*.

The value of the *CLIENTIDENT* varies depending on what identifying method you choose as *IDENTTYPE*:

- **cert**: this method is used for authenticating an instance through TLS certificates.
- **user**: use this to authenticate by user or group name.
- **ip**: use this to authenticate an instance through its IP address.
- **cephx**: use this to authenticate an instance through the Ceph authentication system.

In this test scenario, use **cephx** as your *IDENTTYPE*. So, to create a *client identity* named **cephclient** that the Compute instance can use to access **myshare**:

```
# manila access-allow myshare cephx cephclient
+------------+--------------------------------+
|  Property  |              Value             |
```

```
+------------+--------------------------------------+
|  share_id  |44ad4045-892b-4782-b550-11c2fb680d8e|
| access_type|              cephx                   |
| access_to  |            cephclient                |
|access_level|               rw                     |
|    state   |              new                     |
|     id     |e144e1d9-cce6-4012-bf02-d84909a42292|
+------------+--------------------------------------+
```

7.3.2. Retrieve and Use Client Identity Credentials

Once a client identity is created (**cephclient**), retrieve its credentials from Ceph. Run the following command on the OpenStack node:

```
# ceph --name=CEPHIDENT --keyring=KEYRING auth \
  get client.CLIENTIDENT -o CLIENTIDENT.keyring
```

CEPHIDENT

The Ceph identity you created earlier in Section 5, “Authorize the Driver to Communicate with Ceph”, namely **client.manila**.

KEYRING

The corresponding keyring of *CEPHIDENT*, namely */etc/ceph/manila.keyring*. In Section 5, “Authorize the Driver to Communicate with Ceph”, you copied this file locally from the Ceph cluster.

CLIENTIDENT

The identity you created earlier in Section 7.3.1, “Create an Authorized Client Identity”, namely **cephclient**.

For example:

```
# ceph --name=client.manila --keyring=/etc/ceph/manila.keyring auth get
client.cephclient -o cephclient.keyring
exported keyring for client.cephclient
```

This command will output a keyring file named *cephclient.keyring*, which you will use to authenticate and authorize a Compute instance to mount a CephFS share.

To view the contents of the keyring:

```
# cat cephclient.keyring
[client.cephclient]
key = AQAx+4RXvzYoHRAAMABEsmb6guFEs6MAi0ltuQ==
caps mds = "allow rw path=/volumes/_nogroup/9096ddf0-405e-49dd-b7c8-
1c8815fb1f6e"
caps mon = "allow r"
caps osd = "allow rw pool=cephfs_data namespace=fsvolumens_9096ddf0-405e-49dd-
b7c8-1c8815fb1f6e"
```

Copy this keyring file to the instance you will use and see Section 7.4, “Mount the Share through the FUSE Client” for instructions on mounting the share.

7.4. Mount the Share through the FUSE Client

At this point, the Compute instance that will be mounting the CephFS share should contain the *cephclient.keyring* file from Section 7.3.2, “Retrieve and Use Client Identity Credentials”.

Instances must use a Ceph client to mount shared file systems exported through this protocol. Preferably, use the **ceph-fuse** client to ensure that share size limits set through the Shared File System Service are obeyed.

Log in to the Compute instance that will mount the CephFS share. Then, install the **ceph-fuse** client there:

```
$ sudo yum install -y ceph-fuse
```

Next, configure the **ceph-fuse** client to connect to the Ceph cluster hosting the CephFS share. To do so, create a *ceph.conf* file containing the following settings:

```
[client]
    client quota = true
    mon host = MONIP:MONPORT
```

MONIP

The IP of the Ceph cluster’s monitoring nodes.

MONPORT

The corresponding ports of *MONIP*.

You retrieved the *MONIP* and *MONPORT* values earlier in Section 7.2, “Create a Share”. Using those values:

```
[client]
    client quota = true
    mon host = 10.35.160.43:6789
```

Note

If your Ceph cluster has multiple monitoring nodes, list all their IPs and ports as well, separated by commas.

You can now mount the share from Section 7.2, “Create a Share”. To mount the share on */mnt* in the Compute instance, run:

```
$ sudo ceph-fuse ~/mnt \
    --id=cephclient \
    --conf=./ceph.conf \
    --keyring=./cephclient.keyring \
    --client-mountpoint=SHAREPATH
```

Replace *SHAREPATH* with the path of the exported share. You retrieved this information earlier in Section 7.2, “Create a Share” (namely, **/volumes/_nogroup/9096ddf0-405e-49dd-b7c8-1c8815fb1f6e**).

Upon mounting the share, check if you can write to it at its mount point.

7.5. Test Other Functions

After confirming that you can mount to the share, exit the instance and log back into the OpenStack host. From there, you can test:

- Increasing the share size
- Shrinking the share size
- Denying access to the share
- Deleting the share

To perform any of these operations, you will need the share's name or ID (*SHARENAME*). For example, we created a share named **myshare** earlier in Section 7.2, "Create a Share".

Changing a Share's Size

To increase the share size:

```
# manila extend SHARENAME SIZE
```

Conversely, to shrink the size of a share:

```
# manila shrink SHARENAME SIZE
```

In both cases, replace *SIZE* with the target size of the share, in GB. When shrinking a share, you cannot shrink beyond the amount of storage already consumed by the share.

Denying Share Access

In Section 7.3, "Configure Share Access", you authorized the instance (through its *CLIENTIDENT*, ie. client identity) to access and mount the share (identified by its *SHARENAME*, ie. share iD or name). This authorization is provided by an *access rule*; to revoke the authorization (and, by extension, the client identity's ability to access the share), you have to revoke the access rule. Before doing so, you need to retrieve access rule's ID:

```
# manila access-list SHARENAME
```

For example, given the share we created earlier in Section 7.2, "Create a Share" (**myshare**):

```
# manila access-list myshare
+--------------------------------------+-----------+----------+
|                  id                  |access_type|access_to |...
+--------------------------------------+-----------+----------+
|e144e1d9-cce6-4012-bf02-d84909a42292|   cephx   |cephclient|...
+--------------------------------------+-----------+----------+
```

Upon retrieving the access rule ID (*ACCESSID*), you can deny it, thereby revoking the authorization:

```
# manila access-deny SHARENAME ACCESSID
```

For example:

```
# manila access-deny myshare e144e1d9-cce6-4012-bf02-d84909a42292
```

Deleting a Share

Upon revoking access, the instance should no longer be able to write to the share. You can then delete it with:

```
# manila delete SHARENAME
```