Manila Developer Documentation

Release 13.2.1.dev21

Manila contributors

Feb 08, 2024

CONTENTS

1 What is Manila? 1

2 For end users 3
- 2.1 Tools for using Manila . . . 3
  - 2.1.1 User . . . 3
    - Create and manage shares . . . 3
    - Create and manage share networks . . . 34
    - Create and manage share network subnets . . . 47
    - Troubleshooting asynchronous failures . . . 51
- 2.2 Using the Manila API . . . 61

3 For operators 63
- 3.1 Installing Manila . . . 63
  - 3.1.1 Installation Tutorial . . . 63
    - Service Overview . . . 63
    - Install and configure controller node . . . 64
    - Install and configure a share node . . . 87
    - Verify operation . . . 123
    - Creating and using shared file systems . . . 124
    - Next steps . . . 134
- 3.2 Administrating Manila . . . 135
  - 3.2.1 Admin Guide . . . 135
    - Key concepts . . . 136
    - Share management . . . 138
    - Share types . . . 167
    - Share group types . . . 173
    - Share groups . . . 176
    - Share snapshots . . . 183
    - Share servers . . . 186
    - Share server management . . . 186
    - Share server limits (Since Wallaby release) . . . 189
    - Security services . . . 189
    - Share migration . . . 193
    - Share replication . . . 198
    - Multi-storage configuration . . . 209
    - Networking . . . 211
    - Troubleshoot Shared File Systems service . . . 220
    - Profiling the Shared File Systems service . . . 221
    - Upgrading the Shared File System service . . . 225

Share revert to snapshot . . . 227
Share server migration . . . 228
Manila share features support mapping . . . 234
Capabilities and Extra-Specs . . . 237
Group Capabilities and group-specs . . . 241
Export Location Metadata . . . 242
Supported share back ends . . . 243
3.3 Reference . . . 381
3.3.1 Configuration . . . 381
Introduction to the Shared File Systems service . . . 381
Shared File Systems API configuration . . . 383
Share drivers . . . 385
Log files used by Shared File Systems . . . 477
Additional options . . . 477
Shared File Systems service sample configuration files . . . 483
3.3.2 Command Line Interface . . . 487
Shared File Systems service (manila) command-line client . . . 487
manila-manage . . . 533
manila-status . . . 536
3.4 Additional resources . . . 537

4 For contributors **539**
4.1 Contributor/Developer Guide . . . 539
4.1.1 Basic Information . . . 539
So You Want to Contribute . . . 539
4.1.2 Programming HowTos and Tutorials . . . 543
Setting Up a Development Environment . . . 543
Setting up a development environment with devstack . . . 546
Running manila API with a web server . . . 550
Unit Tests . . . 552
Tempest Tests . . . 553
Adding a Method to the OpenStack Manila API . . . 556
Documenting your work . . . 557
Release Notes . . . 560
Using Commit Message Tags in Manila . . . 564
Guru Meditation Reports . . . 565
User Messages . . . 566
Ganesha Library . . . 572
4.1.3 Background Concepts for manila . . . 578
Manila System Architecture . . . 578
Threading model . . . 579
Internationalization . . . 580
AMQP and manila . . . 581
Manila minimum requirements and features . . . 586
Manila optional requirements and features since Mitaka . . . 590
Manila experimental features since Mitaka . . . 591
Pool-Aware Scheduler Support . . . 592
4.1.4 Other Resources . . . 596
Project hosting with Launchpad . . . 596
Code Reviews with Gerrit . . . 597
Manila team code review policy . . . 597

Manila Project Team Lead guide . . . 599
4.1.5 API Reference . . . 602
API Microversions . . . 602
REST API Version History . . . 608
Experimental APIs . . . 617
4.1.6 Module Reference . . . 618
Introduction to the Shared File Systems service . . . 618
Services, Managers and Drivers . . . 618
The Database Layer . . . 622
Shared Filesystems . . . 654
Manila share driver hooks . . . 703
Authentication and Authorization . . . 705
Scheduler . . . 717
Scheduler Filters . . . 723
Scheduler Weighers . . . 727
Fake Drivers . . . 729
Common and Misc Libraries . . . 735
Share Replication . . . 763
Configure and use driver filter and weighing for scheduler . . . 782
Share Migration . . . 787
Share Server Migration . . . 796
4.2 Additional reference . . . 806
4.2.1 Reference . . . 806
Glossary . . . 806

CHAPTER

ONE

WHAT IS MANILA?

Manila is the OpenStack Shared Filesystems service for providing Shared Filesystems as a service. Some of the goals of Manila are to be/have:

- **Component based architecture**: Quickly add new behaviors
- **Highly available**: Scale to very serious workloads
- **Fault-Tolerant**: Isolated processes avoid cascading failures
- **Recoverable**: Failures should be easy to diagnose, debug, and rectify
- **Open Standards**: Be a reference implementation for a community-driven api

CHAPTER

TWO

FOR END USERS

As an end user of Manila, youll use Manila to create a remote file system with either tools or the API directly: python-manilaclient, or by directly using the REST API.

2.1 Tools for using Manila

Contents:

2.1.1 User

Create and manage shares

- *General Concepts*
- *Usage and Limits*
- *Share types*
- *Share networks*
- *Create a share*
- *Allow read-write access*
- *Allow read-only access*
- *Update access rules metadata*
- *Deny access*
- *Create snapshot*
- *Create share from snapshot*
- *Delete share*
- *Delete snapshot*
- *Extend share*
- *Shrink share*
- *Share metadata*
- *Share revert to snapshot*

General Concepts

A `share` is filesystem storage that you can create with manila. You can pick a network protocol for the underlying storage, manage access and perform lifecycle operations on the share via the `manila` command line tool.

Before we review the operations possible, lets take a look at certain important terms:

- `share network`: This is a network that your shares can be exported to. Exporting shares to your own self-service isolated networks allows manila to provide `hard network path` data isolation guarantees in a multi-tenant cloud. To do so, under the hood, manila creates isolated `share servers`, and plugs them into your network. These share servers manage exports of your shares, and can connect to authentication domains that you determine. Manila performs all the lifecycle operations necessary on share servers, and you neednt worry about them. The important thing to note is that your cloud administrator must have made a share type with extra-spec `driver_handles_share_servers=True` for you to be able to use share networks and create shares on them. See *Create and manage share networks* and *Create and manage share network subnets* for more details.
- `share type`: A share type is a template made available by your administrator. You must always specify a share type when creating a share, unless you would like to use the default share type. Its possible that your cloud administrator has not made a default share type accessible to you. Share types specify some capabilities for your use:

Capability	Possible values	Consequence
driver_handles_share_servers	true or false	you can or cannot use share networks to create shares
snapshot_support	true or false	you can or cannot create snapshots of shares
create_share_from_snapshot_support	true or false	you can or cannot create clones of share snapshots
revert_to_snapshot_support	true or false	you can or cannot revert your shares in-place to the most recent snapshot
mount_snapshot_support	true or false	you can or cannot export your snapshots and mount them
replication_type	dr	you can create replicas for disaster recovery, only one active export allowed at a time
	readable	you can create read-only replicas, only one writable active export allowed at a time
	writable	you can create read/write replicas, any number of active exports per share
availability_zones	a list of one or more availability zones	shares are limited to these availability zones

Note:

- When `replication_type` extra specification is not present in the share type, you cannot create share replicas

- When the `availability_zones` extra specification is not present in the share type, the share type can be used in all availability zones of the cloud.

- `status` of resources: Resources that you create or modify with manila may not be available immediately. The API service is designed to respond immediately and the resource being created or modified is worked upon by the rest of the service stack. To indicate the readiness of resources, there are several attributes on the resources themselves and the user can watch these fields to know the state of the resource. For example, the `status` attribute in shares can convey some busy states such as creating, extending, shrinking, migrating. These -ing states end in a available state if everything goes well. They may end up in an error state in case there is an issue. See *Troubleshooting asynchronous failures* to determine if you can rectify these errors by yourself. If you cannot, consulting a more privileged user, usually a cloud administrator, might be useful.
- `snapshot`: This is a point-in-time copy of a share. In manila, snapshots are meant to be crash consistent, however, you may need to quiesce any applications using the share to ensure that the snapshots are application consistent. Cloud administrators can enable or disable snapshots via share type extra specifications.
- `security service`: This is an authentication domain that you define and associate with your share networks. It could be an Active Directory server, a Lightweight Directory Access Protocol server, or Kerberos. When used, access to shares can be controlled via these authentication domains. You may even combine multiple authentication domains.

Usage and Limits

- List the resource limits and usages that apply to your project

```
$ manila absolute-limits
+----------------------------+-------+
| Name                       | Value |
+----------------------------+-------+
| maxTotalReplicaGigabytes   | 1000  |
| maxTotalShareGigabytes     | 1000  |
| maxTotalShareNetworks      | 10    |
| maxTotalShareReplicas      | 100   |
| maxTotalShareSnapshots     | 50    |
| maxTotalShares             | 50    |
| maxTotalSnapshotGigabytes  | 1000  |
| totalReplicaGigabytesUsed  | 0     |
| totalShareGigabytesUsed    | 4     |
| totalShareNetworksUsed     | 1     |
| totalShareReplicasUsed     | 0     |
| totalShareSnapshotsUsed    | 1     |
| totalSharesUsed            | 4     |
| totalSnapshotGigabytesUsed | 1     |
+----------------------------+-------+
```

Share types

- List share types

```
$ manila type-list
+--------------------------------------+-------------------------------------
↪-+------------+------------+--------------------------------------+-----
↪----------------------------------------+-------------------------------
↪-------------------------+
| ID                                   | Name                                ␣
↪ | visibility | is_default | required_extra_specs                 |␣
↪optional_extra_specs                        | Description                 ␣
↪                                |
+--------------------------------------+-------------------------------------
↪-+------------+------------+--------------------------------------+-----
↪----------------------------------------+-------------------------------
↪-------------------------+
| af7b64ec-cdb3-4a5f-93c9-51672d72e172 | dhss_true                           ␣
↪ | public     | -          | driver_handles_share_servers : True  |␣
↪snapshot_support : True                     | None                        ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪create_share_from_snapshot_support : True  |                              ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪revert_to_snapshot_support : True          |                              ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪mount_snapshot_support : True              |                              ␣
↪                                |
| c39d3565-cee0-4a64-9e60-af06991ea4f7 | default                             ␣
↪ | public     | YES        | driver_handles_share_servers : False |␣
↪snapshot_support : True                     | None                        ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪create_share_from_snapshot_support : True  |                              ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪revert_to_snapshot_support : True          |                              ␣
↪                                |
|                                      |                                     ␣
↪ |            |            |                                      |␣
↪mount_snapshot_support : True              |                              ␣
↪                                |
| e88213ca-66e6-4ae1-ba1b-d9d2c65bae12 | dhss_false                          ␣
↪ | public     | -          | driver_handles_share_servers : False |␣
↪snapshot_support : True                     | None                        ␣
↪                                |
```

(continues on next page)

(continued from previous page)

```
|                                      |                                                                              |
↪ |             |             |                                                 |
↪create_share_from_snapshot_support : True   |                                                                              |
↪                                |
|                                      |                                                                              |
↪ |             |             |                                                 |
↪revert_to_snapshot_support : True           |                                                                              |
↪                                |
|                                      |                                                                              |
↪ |             |             |                                                 |
↪mount_snapshot_support : True               |                                                                              |
↪                                |
+--------------------------------------+----------------------------------
↪-+------------+------------+-----------------------------------------+-----
↪-----------------------------------------+----------------------------------
↪-------------------------+
```

Share networks

- Create a share network.

```
$ manila share-network-create \
    --name mysharenetwork \
    --description "My Manila network" \
    --neutron-net-id 23da40b4-0d5e-468c-8ac9-3766e9ceaacd \
    --neutron-subnet-id 4568bc9b-42fe-45ac-a49b-469e8276223c
+-----------------------+------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪----------------------------------+
| Property              | Value                                                                
```

(continued from previous page)

```
| name                  | mysharenetwork                                                 ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                      |
| project_id            | d9932a60d9ee4087b6cff9ce6e9b4e3b                               ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                      |
| created_at            | 2020-08-07T04:47:53.000000                                     ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                      |
| updated_at            | None                                                           ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                      |
| description           | My Manila network                                              ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                      |
| share_network_subnets | [{'id': '187dcd27-8478-45c1-bd5e-5423cafd15ae',
↪'availability_zone': None, 'created_at': '2020-08-07T04:47:53.000000',
↪'updated_at': None, 'segmentation_id': None, 'neutron_net_id':
↪'23da40b4-0d5e-468c-8ac9-3766e9ceaacd', 'neutron_subnet_id': '4568bc9b-
↪42fe-45ac-a49b-469e8276223c', 'ip_version': None, 'cidr': None,
↪'network_type': None, 'mtu': None, 'gateway': None}] |
+-----------------------+------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪----------------------------------+
```

Note: This Manila API does not validate the subnet information you supply right away. The validation is performed when creating a share with the share network. This is why, you do not see some subnet information populated on the share network resource until at least one share is created with it.

- List share networks.

```
$ manila share-network-list
+--------------------------------------+----------------+
| id                                   | name           |
+--------------------------------------+----------------+
| c4bfdd5e-7502-4a65-8876-0ce8b9914a64 | mysharenetwork |
+--------------------------------------+----------------+
```

Create a share

- Create a share

Note: If you use a share type that has the extra specification `driver_handles_share_servers=False`, you cannot use a share network to create your shares.

```
$ manila create NFS 1 \
    --name myshare \
    --description "My Manila share" \
    --share-network mysharenetwork \
    --share-type dhss_true
+--------------------------------------+--------------------------------------
↪-----+
| Property                             | Value                               ␣
↪     |
+--------------------------------------+--------------------------------------
↪-----+
| id                                   | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719 |
| size                                 | 1                                   ␣
↪     |
| availability_zone                    | None                                ␣
↪     |
| created_at                           | 2020-08-07T05:24:14.000000          ␣
↪     |
| status                               | creating                            ␣
↪     |
| name                                 | myshare                             ␣
↪     |
| description                          | My Manila share                     ␣
↪     |
| project_id                           |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b     |
| snapshot_id                          | None                                ␣
↪     |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-
↪0ce8b9914a64 |
```

(continues on next page)

(continued from previous page)

```
| share_proto                          | NFS                                  |
| metadata                             | {}                                   |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-51672d72e172 |
| is_public                            | False                                |
| snapshot_support                     | True                                 |
| task_state                           | None                                 |
| share_type_name                      | dhss_true                            |
| access_rules_status                  | active                               |
| replication_type                     | None                                 |
| has_replicas                         | False                                |
| user_id                              | 2cebd96a794f431caa06ce5215e0da21     |
| create_share_from_snapshot_support   | True                                 |
| revert_to_snapshot_support           | True                                 |
| share_group_id                       | None                                 |
| source_share_group_snapshot_member_id | None                                 |
| mount_snapshot_support               | True                                 |
| progress                             | None                                 |
+--------------------------------------+--------------------------------------+
```

- Show a share.

```
$ manila show myshare
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| id                                   | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719                                                          ␣
↪                    |
| size                                 | 1                              ␣
↪                                                                      ␣
↪             |
| availability_zone                    | nova                           ␣
↪                                                                      ␣
↪             |
| created_at                           | 2020-08-07T05:24:14.000000     ␣
↪                                                                      ␣
↪             |
| status                               | available                      ␣
↪                                                                      ␣
↪             |
| name                                 | myshare                        ␣
↪                                                                      ␣
↪             |
| description                          | My Manila share                ␣
↪                                                                      ␣
↪             |
| project_id                           |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b                                      ␣
↪                                                |
| snapshot_id                          | None                           ␣
↪                                                                      ␣
↪             |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-
↪0ce8b9914a64                                                          ␣
↪                    |
| share_proto                          | NFS                            ␣
↪                                                                      ␣
↪             |
| metadata                             | {}                             ␣
↪                                                                      ␣
↪             |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-
↪51672d72e172                                                          ␣
↪                    |
| is_public                            | False                          ␣
↪                                                                      ␣
↪             |
| snapshot_support                     | True                           ␣
↪                                                                      ␣
↪             |
| task_state                           | None                           ␣
↪                                                                      ␣
↪             |
| share_type_name                      | dhss_true                      ␣
↪                                                                      ␣
↪             |
```

(continues on next page)

(continued from previous page)

```
| access_rules_status                    | active                                                
```

(continued from previous page)

```
+--------------------------------------+-----------------------------------
↪-----------------------------------------------------------------------
↪-------------+
```

- List shares.

```
$ manila list
+--------------------------------------+--------------------+------+------
↪-------+-----------+-----------+-----------------+------+---------------
↪----+
| ID                                   | Name               | Size |␣
↪Share Proto | Status    | Is Public | Share Type Name | Host |␣
↪Availability Zone |
+--------------------------------------+--------------------+------+------
↪-------+-----------+-----------+-----------------+------+---------------
↪----+
| 83b0772b-00ad-4e45-8fad-106b9d4f1719 | myshare            | 1    | NFS ␣
↪       | available | False     | dhss_true       |      | nova          ␣
↪    |
+--------------------------------------+--------------------+------+------
↪-------+-----------+-----------+-----------------+------+---------------
↪----+
```

- List share export locations.

```
$ manila share-export-location-list myshare
+--------------------------------------+-----------------------------------
↪-----------------------------------------------------------------------
↪-----+-----------+
| ID                                   | Path                              ␣
↪                                                                        ␣
↪     | Preferred |
+--------------------------------------+-----------------------------------
↪-----------------------------------------------------------------------
↪-----+-----------+
| 395244a1-8aa9-44af-9fda-f7d6036ce2b9 | 10.0.0.10:/sharevolumes_10034/
↪share_83b0772b_00ad_4e45_8fad_106b9d4f1719_da404d59_4280_4b32_847f_
↪6cfa4f730bbd | False     |
| 908e5a28-c5ea-4627-b17c-1cfeb894ccd1 | 10.0.0.11:/sharevolumes_10034/
↪share_83b0772b_00ad_4e45_8fad_106b9d4f1719_da404d59_4280_4b32_847f_
↪6cfa4f730bbd | True      |
+--------------------------------------+-----------------------------------
↪-----------------------------------------------------------------------
↪-----+-----------+
```

Allow read-write access

- Allow access.

```
$ manila access-allow myshare ip 10.0.0.0/24 --metadata key1=value1
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| id           | e30bde96-9217-4f90-afdc-27c092af1c77 |
| share_id     | 83b0772b-00ad-4e45-8fad-106b9d4f1719 |
| access_level | rw                                   |
| access_to    | 10.0.0.0/24                          |
| access_type  | ip                                   |
| state        | queued_to_apply                      |
| access_key   | None                                 |
| created_at   | 2020-08-07T05:27:27.000000           |
| updated_at   | None                                 |
| metadata     | {'key1': 'value1'}                   |
+--------------+--------------------------------------+
```

Note: Since API version 2.38, access rules of type IP supports IPv6 addresses and subnets in CIDR notation.

Note: Since API version 2.45, metadata can be added, removed and updated for share access rules in a form of key=value pairs. Metadata can help you identify and filter access rules.

- List access.

```
$ manila access-list myshare
+--------------------------------------+-------------+-------------+------
↪--------+--------+-----------+----------------------------+------------
↪+
| id                                   | access_type | access_to   |␣
↪access_level | state  | access_key | created_at                 |␣
↪updated_at |
+--------------------------------------+-------------+-------------+------
↪--------+--------+-----------+----------------------------+------------
↪+
| e30bde96-9217-4f90-afdc-27c092af1c77 | ip          | 10.0.0.0/24 | rw  ␣
↪        | active | None       | 2020-08-07T05:27:27.000000 | None       ␣
↪|
+--------------------------------------+-------------+-------------+------
↪--------+--------+-----------+----------------------------+------------
↪+
```

An access rule is created.

Allow read-only access

- Allow access.

```
$ manila access-allow myshare ip fd31:7ee0:3de4:a41b::/64 --access-level␣
↪ro
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| id           | 45b0a030-306a-4305-9e2a-36aeffb2d5b7 |
| share_id     | 83b0772b-00ad-4e45-8fad-106b9d4f1719 |
| access_level | ro                                   |
| access_to    | fd31:7ee0:3de4:a41b::/64             |
| access_type  | ip                                   |
| state        | queued_to_apply                      |
| access_key   | None                                 |
| created_at   | 2020-08-07T05:28:35.000000           |
| updated_at   | None                                 |
| metadata     | {}                                   |
+--------------+--------------------------------------+
```

- List access.

```
$ manila access-list myshare
+--------------------------------------+-------------+--------------------
↪--------+--------------+--------+------------+--------------------------
↪--+------------+
| id                                   | access_type | access_to          ␣
↪        | access_level | state  | access_key | created_at                 ␣
↪  | updated_at |
+--------------------------------------+-------------+--------------------
↪--------+--------------+--------+------------+--------------------------
↪--+------------+
| 45b0a030-306a-4305-9e2a-36aeffb2d5b7 | ip          |␣
↪fd31:7ee0:3de4:a41b::/64   | ro           | active | None       | 2020-
↪08-07T05:28:35.000000 | None       |
| e30bde96-9217-4f90-afdc-27c092af1c77 | ip          | 10.0.0.0/24        ␣
↪        | rw           | active | None       | 2020-08-07T05:27:27.
↪000000 | None       |
+--------------------------------------+-------------+--------------------
↪--------+--------------+--------+------------+--------------------------
↪--+------------+
```

Another access rule is created.

Update access rules metadata

1. Add a new metadata.

```
$ manila access-metadata 0c8470ca-0d77-490c-9e71-29e1f453bf97 set␣
↪key2=value2
$ manila access-show 0c8470ca-0d77-490c-9e71-29e1f453bf97
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| id           | 0c8470ca-0d77-490c-9e71-29e1f453bf97 |
| share_id     | 8d8b854b-ec32-43f1-acc0-1b2efa7c3400 |
| access_level | rw                                   |
| access_to    | 10.0.0.0/24                          |
| access_type  | ip                                   |
| state        | active                               |
| access_key   | None                                 |
| created_at   | 2016-03-24T14:51:36.000000           |
| updated_at   | None                                 |
| metadata     | {'key1': 'value1', 'key2': 'value2'} |
+--------------+--------------------------------------+
```

2. Remove a metadata key value.

```
$ manila access-metadata 0c8470ca-0d77-490c-9e71-29e1f453bf97 unset key
$ manila access-show 0c8470ca-0d77-490c-9e71-29e1f453bf97
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| id           | 0c8470ca-0d77-490c-9e71-29e1f453bf97 |
| share_id     | 8d8b854b-ec32-43f1-acc0-1b2efa7c3400 |
| access_level | rw                                   |
| access_to    | 10.0.0.0/24                          |
| access_type  | ip                                   |
| state        | active                               |
| access_key   | None                                 |
| created_at   | 2016-03-24T14:51:36.000000           |
| updated_at   | None                                 |
| metadata     | {'key2': 'value2'}                   |
+--------------+--------------------------------------+
```

Deny access

- Deny access.

```
$ manila access-deny myshare 45b0a030-306a-4305-9e2a-36aeffb2d5b7
$ manila access-deny myshare e30bde96-9217-4f90-afdc-27c092af1c77
```

- List access.

```
$ manila access-list myshare
+----+-------------+-----------+--------------+-------+-----------+------
↪------+------------+
| id | access_type | access_to | access_level | state | access_key |␣
↪created_at | updated_at |
+----+-------------+-----------+--------------+-------+-----------+------
↪------+------------+
+----+-------------+-----------+--------------+-------+-----------+------
↪------+------------+
```

The access rules are removed.

Create snapshot

- Create a snapshot.

Note: To create a snapshot, the share type of the share must contain the capability extra-spec `snapshot_support=True`.

```
$ manila snapshot-create --name mysnapshot --description "My Manila␣
↪snapshot" myshare
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| id          | 8a18aa77-7500-4e56-be8f-6081146f47f1 |
| share_id    | 83b0772b-00ad-4e45-8fad-106b9d4f1719 |
| share_size  | 1                                    |
| created_at  | 2020-08-07T05:30:26.649430           |
| status      | creating                             |
| name        | mysnapshot                           |
| description | My Manila snapshot                   |
| size        | 1                                    |
| share_proto | NFS                                  |
| user_id     | 2cebd96a794f431caa06ce5215e0da21     |
| project_id  | d9932a60d9ee4087b6cff9ce6e9b4e3b     |
+-------------+--------------------------------------+
```

- List snapshots.

```
$ manila snapshot-list
+--------------------------------------+--------------------------------
↪----+-----------+------------+------------+
| ID                                   | Share ID                       ␣
↪    | Status    | Name       | Share Size |
+--------------------------------------+--------------------------------
↪----+-----------+------------+------------+
| 8a18aa77-7500-4e56-be8f-6081146f47f1 | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719 | available | mysnapshot | 1          |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+--------------------------------
↪----+-----------+-----------+-----------+
```

Create share from snapshot

- Create a share from a snapshot.

Note: To create a share from a snapshot, the share type of the parent share must contain the capability extra-spec `create_share_from_snapshot_support=True`.

```
$ manila create NFS 1 \
    --snapshot-id 8a18aa77-7500-4e56-be8f-6081146f47f1 \
    --share-network mysharenetwork \
    --name mysharefromsnap
+--------------------------------------+--------------------------------
↪-----+
| Property                             | Value                          ␣
↪     |
+--------------------------------------+--------------------------------
↪-----+
| id                                   | 2a9336ea-3afc-4443-80bb-
↪398f4bdb3a93 |
| size                                 | 1                              ␣
↪     |
| availability_zone                    | nova                           ␣
↪     |
| created_at                           | 2020-08-07T05:34:12.000000     ␣
↪     |
| status                               | creating                       ␣
↪     |
| name                                 | mysharefromsnap                ␣
↪     |
| description                          | None                           ␣
↪     |
| project_id                           |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b     |
| snapshot_id                          | 8a18aa77-7500-4e56-be8f-
↪6081146f47f1 |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-
↪0ce8b9914a64 |
| share_proto                          | NFS                            ␣
↪     |
| metadata                             | {}                             ␣
↪     |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-
↪51672d72e172 |
| is_public                            | False                          ␣
↪     |
```

(continues on next page)

(continued from previous page)

```
| snapshot_support                      | True                                 ␣
↪     |
| task_state                            | None                                 ␣
↪     |
| share_type_name                       | dhss_true                            ␣
↪     |
| access_rules_status                   | active                               ␣
↪     |
| replication_type                      | None                                 ␣
↪     |
| has_replicas                          | False                                ␣
↪     |
| user_id                               |␣
↪2cebd96a794f431caa06ce5215e0da21     |
| create_share_from_snapshot_support    | True                                 ␣
↪     |
| revert_to_snapshot_support            | True                                 ␣
↪     |
| share_group_id                        | None                                 ␣
↪     |
| source_share_group_snapshot_member_id | None                                 ␣
↪     |
| mount_snapshot_support                | True                                 ␣
↪     |
| progress                              | None                                 ␣
↪     |
+---------------------------------------+----------------------------------
↪-----+
```

- List shares.

```
$ manila list
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
| ID                                   | Name            | Size | Share␣
↪Proto | Status    | Is Public | Share Type Name | Host                 ␣
↪      | Availability Zone |
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
| 83b0772b-00ad-4e45-8fad-106b9d4f1719 | myshare         | 1    | NFS    ␣
↪    | available | False     | default         | nosb-devstack@london
↪#LONDON | nova              |
| 2a9336ea-3afc-4443-80bb-398f4bdb3a93 | mysharefromsnap | 1    | NFS    ␣
↪    | available | False     | default         | nosb-devstack@london
↪#LONDON | nova              |
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
```

- Show the share created from snapshot.

```
$ manila show mysharefromsnap
+-----------------------------------------+--------------------------------
↪--------------------------------------------------------------------------
↪-------------+
| Property                                | Value                          ␣
↪                                                                         ␣
↪             |
+-----------------------------------------+--------------------------------
↪--------------------------------------------------------------------------
↪-------------+
| id                                      | 2a9336ea-3afc-4443-80bb-
↪398f4bdb3a93                                                             ␣
↪                       |
| size                                    | 1                              ␣
↪                                                                         ␣
↪             |
| availability_zone                       | nova                           ␣
↪                                                                         ␣
↪             |
| created_at                              | 2020-08-07T05:34:12.000000     ␣
↪                                                                         ␣
↪             |
| status                                  | available                      ␣
↪                                                                         ␣
↪             |
| name                                    | mysharefromsnap                ␣
↪                                                                         ␣
↪             |
| description                             | None                           ␣
↪                                                                         ␣
↪             |
| project_id                              |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b                                          ␣
↪                                                    |
| snapshot_id                             | 8a18aa77-7500-4e56-be8f-
↪6081146f47f1                                                             ␣
↪                       |
| share_network_id                        | c4bfdd5e-7502-4a65-8876-
↪0ce8b9914a64                                                             ␣
↪                       |
| share_proto                             | NFS                            ␣
↪                                                                         ␣
↪             |
| metadata                                | {}                             ␣
↪                                                                         ␣
↪             |
| share_type                              | af7b64ec-cdb3-4a5f-93c9-
↪51672d72e172                                                             ␣
↪                       |
```

(continues on next page)

(continued from previous page)

```
| is_public                                | False                                ␣
↪                                                                                 ␣
↪                 |
| snapshot_support                         | True                                 ␣
↪                                                                                 ␣
↪                 |
| task_state                               | None                                 ␣
↪                                                                                 ␣
↪                 |
| share_type_name                          | dhss_true                            ␣
↪                                                                                 ␣
↪                 |
| access_rules_status                      | active                               ␣
↪                                                                                 ␣
↪                 |
| replication_type                         | None                                 ␣
↪                                                                                 ␣
↪                 |
| has_replicas                             | False                                ␣
↪                                                                                 ␣
↪                 |
| user_id                                  |␣
↪2cebd96a794f431caa06ce5215e0da21                                                  ␣
↪                                                    |
| create_share_from_snapshot_support       | True                                 ␣
↪                                                                                 ␣
↪                 |
| revert_to_snapshot_support               | True                                 ␣
↪                                                                                 ␣
↪                 |
| share_group_id                           | None                                 ␣
↪                                                                                 ␣
↪                 |
| source_share_group_snapshot_member_id    | None                                 ␣
↪                                                                                 ␣
↪                 |
| mount_snapshot_support                   | True                                 ␣
↪                                                                                 ␣
↪                 |
| progress                                 | 100%                                 ␣
↪                                                                                 ␣
↪                 |
| export_locations                         |                                      ␣
↪                                                                                 ␣
↪                 |
|                                          | id = 7928b361-cada-4505-a62e-
↪4cefb1cf6fc5                                                                     ␣
↪                   |
|                                          | path = 10.0.0.11:/path/to/fake/
↪share/share_2a9336ea_3afc_4443_80bb_398f4bdb3a93_97de2abe_d114_49a9_
↪9d01_ce5e71337e48 |
```

(continues on next page)

(continued from previous page)

```
|                                       | preferred = True                ␣
↪                                                                         ␣
↪              |
|                                       | id = e48d19ba-dee5-4492-b156-
↪5181530955be                                                             ␣
↪                 |
|                                       | path = 10.0.0.10:/path/to/fake/
↪share/share_2a9336ea_3afc_4443_80bb_398f4bdb3a93_97de2abe_d114_49a9_
↪9d01_ce5e71337e48 |
|                                       | preferred = False               ␣
↪                                                                         ␣
↪              |
+---------------------------------------+---------------------------------
↪-------------------------------------------------------------------------
↪-------------+
```

Delete share

- Delete a share.

```
$ manila delete mysharefromsnap
```

- List shares.

```
$ manila list
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
| ID                                   | Name            | Size | Share␣
↪Proto | Status    | Is Public | Share Type Name | Host                 ␣
↪      | Availability Zone |
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
| 83b0772b-00ad-4e45-8fad-106b9d4f1719 | myshare         | 1    | NFS    ␣
↪    | available | False     | default         | nosb-devstack@london
↪#LONDON | nova              |
| 2a9336ea-3afc-4443-80bb-398f4bdb3a93 | mysharefromsnap | 1    | NFS    ␣
↪    | deleting  | False     | default         | nosb-devstack@london
↪#LONDON | nova              |
+--------------------------------------+-----------------+------+---------
↪----+-----------+-----------+-----------------+-------------------------
↪----+-------------------+
```

The share is being deleted.

Delete snapshot

- Delete a snapshot.

```
$ manila snapshot-delete mysnapshot
```

- List snapshots after deleting.

```
$ manila snapshot-list

+----+----------+--------+------+------------+
| ID | Share ID | Status | Name | Share Size |
+----+----------+--------+------+------------+
+----+----------+--------+------+------------+
```

The snapshot is deleted.

Extend share

- Extend share.

```
$ manila extend myshare 2
```

- Show the share while it is being extended.

```
$ manila show myshare
+----------------------------------------+----------------------------------
↪--------------------------------------------------------------------------
↪-------------+
| Property                               | Value                           ␣
↪                                                                          ␣
↪             |
+----------------------------------------+----------------------------------
↪--------------------------------------------------------------------------
↪-------------+
| id                                     | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719                                                              ␣
↪                      |
| size                                   | 1                               ␣
↪                                                                          ␣
↪             |
| availability_zone                      | nova                            ␣
↪                                                                          ␣
↪             |
| created_at                             | 2020-08-07T05:24:14.000000      ␣
↪                                                                          ␣
↪             |
| status                                 | extending                       ␣
↪                                                                          ␣
↪             |
```

(continues on next page)

(continued from previous page)

```
| name                                 | myshare                                                
```

(continued from previous page)

```
| revert_to_snapshot_support             | True                                                
```

(continued from previous page)

```
+--------------------------------------+---------------------------------
↪-----------------------------------------------------------------------
↪-------------+
| id                                   | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719                                                           ␣
↪                     |
| size                                 | 2                               ␣
↪                                                                       ␣
↪             |
| availability_zone                    | nova                            ␣
↪                                                                       ␣
↪             |
| created_at                           | 2020-08-07T05:24:14.000000      ␣
↪                                                                       ␣
↪             |
| status                               | available                       ␣
↪                                                                       ␣
↪             |
| name                                 | myshare                         ␣
↪                                                                       ␣
↪             |
| description                          | My Manila share                 ␣
↪                                                                       ␣
↪             |
| project_id                           |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b                                       ␣
↪                                                |
| snapshot_id                          | None                            ␣
↪                                                                       ␣
↪             |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-
↪0ce8b9914a64                                                           ␣
↪                     |
| share_proto                          | NFS                             ␣
↪                                                                       ␣
↪             |
| metadata                             | {}                              ␣
↪                                                                       ␣
↪             |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-
↪51672d72e172                                                           ␣
↪                     |
| is_public                            | False                           ␣
↪                                                                       ␣
↪             |
| snapshot_support                     | True                            ␣
↪                                                                       ␣
↪             |
| task_state                           | None                            ␣
↪                                                                       ␣
↪             |
```

(continues on next page)

(continued from previous page)

```
| share_type_name                         | dhss_true                                                
```

(continued from previous page)

```
|                                       | preferred = False                ␣
↪                                                                          ␣
↪              |
+-------------------------------------+--------------------------------
↪-----------------------------------------------------------------------
↪-------------+
```

Shrink share

- Shrink a share.

```
$ manila shrink myshare 1
```

- Show the share while it is being shrunk.

```
$ manila show myshare
+-------------------------------------+--------------------------------
↪-----------------------------------------------------------------------
↪-------------+
| Property                            | Value                            ␣
↪                                                                          ␣
↪              |
+-------------------------------------+--------------------------------
↪-----------------------------------------------------------------------
↪-------------+
| id                                  | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719                                                              ␣
↪                      |
| size                                | 2                                ␣
↪                                                                          ␣
↪              |
| availability_zone                   | nova                             ␣
↪                                                                          ␣
↪              |
| created_at                          | 2020-08-07T05:24:14.000000       ␣
↪                                                                          ␣
↪              |
| status                              | shrinking                        ␣
↪                                                                          ␣
↪              |
| name                                | myshare                          ␣
↪                                                                          ␣
↪              |
| description                         | My Manila share                  ␣
↪                                                                          ␣
↪              |
| project_id                          |␣
↪d9932a60d9ee4087b6cff9ce6e9b4e3b                                          ␣
↪                                                |
```

(continues on next page)

(continued from previous page)

```
| snapshot_id                          | None                                 |
|                                      |                                      |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-0ce8b9914a64 |
| share_proto                          | NFS                                  |
| metadata                             | {}                                   |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-51672d72e172 |
| is_public                            | False                                |
| snapshot_support                     | True                                 |
| task_state                           | None                                 |
| share_type_name                      | dhss_true                            |
| access_rules_status                  | active                               |
| replication_type                     | None                                 |
| has_replicas                         | False                                |
| user_id                              | 2cebd96a794f431caa06ce5215e0da21     |
| create_share_from_snapshot_support   | True                                 |
| revert_to_snapshot_support           | True                                 |
| share_group_id                       | None                                 |
| source_share_group_snapshot_member_id | None                                 |
```

(continues on next page)

(continued from previous page)

```
| mount_snapshot_support                 | True                                ␣
↪                                                                              ␣
↪              |
| progress                               | 100%                                ␣
↪                                                                              ␣
↪              |
| export_locations                       |                                     ␣
↪                                                                              ␣
↪              |
|                                        | id = 908e5a28-c5ea-4627-b17c-
↪1cfeb894ccd1                                                                  ␣
↪                 |
|                                        | path = 10.0.0.11:/path/to/fake/
↪share/share_83b0772b_00ad_4e45_8fad_106b9d4f1719_da404d59_4280_4b32_
↪847f_6cfa4f730bbd |
|                                        | preferred = True                    ␣
↪                                                                              ␣
↪              |
|                                        | id = 395244a1-8aa9-44af-9fda-
↪f7d6036ce2b9                                                                  ␣
↪                 |
|                                        | path = 10.0.0.10:/path/to/fake/
↪share/share_83b0772b_00ad_4e45_8fad_106b9d4f1719_da404d59_4280_4b32_
↪847f_6cfa4f730bbd |
|                                        | preferred = False                   ␣
↪                                                                              ␣
↪              |
+----------------------------------------+---------------------------------
↪-------------------------------------------------------------------------
↪-------------+
```

- Show the share after it is being shrunk.

```
$ manila show myshare
+----------------------------------------+---------------------------------
↪-------------------------------------------------------------------------
↪-------------+
| Property                               | Value                               ␣
↪                                                                              ␣
↪              |
+----------------------------------------+---------------------------------
↪-------------------------------------------------------------------------
↪-------------+
| id                                     | 83b0772b-00ad-4e45-8fad-
↪106b9d4f1719                                                                  ␣
↪                      |
| size                                   | 1                                   ␣
↪                                                                              ␣
↪              |
```

(continues on next page)

(continued from previous page)

```
| availability_zone                    | nova                                 |
| created_at                           | 2020-08-07T05:24:14.000000           |
| status                               | available                            |
| name                                 | myshare                              |
| description                          | My Manila share                      |
| project_id                           | d9932a60d9ee4087b6cff9ce6e9b4e3b     |
| snapshot_id                          | None                                 |
| share_network_id                     | c4bfdd5e-7502-4a65-8876-0ce8b9914a64 |
| share_proto                          | NFS                                  |
| metadata                             | {}                                   |
| share_type                           | af7b64ec-cdb3-4a5f-93c9-51672d72e172 |
| is_public                            | False                                |
| snapshot_support                     | True                                 |
| task_state                           | None                                 |
| share_type_name                      | dhss_true                            |
| access_rules_status                  | active                               |
| replication_type                     | None                                 |
```

(continues on next page)

(continued from previous page)

```
| has_replicas                          | False                                                
```

Share metadata

- Set metadata items on your share

```
$ manila metadata myshare set purpose='storing financial data for analysis
↪' year_started=2020
```

- Show share metadata

```
$ manila metadata-show myshare
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| purpose      | storing financial data for analysis |
| year_started | 2020                                 |
+--------------+--------------------------------------+
```

- Query share list with metadata

```
$ manila list --metadata year_started=2020
+--------------------------------------+---------+------+-------------+---
↪--------+-----------+-----------------+------+-------------------+
| ID                                   | Name    | Size | Share Proto |␣
↪Status    | Is Public | Share Type Name | Host | Availability Zone |
+--------------------------------------+---------+------+-------------+---
↪--------+-----------+-----------------+------+-------------------+
| 83b0772b-00ad-4e45-8fad-106b9d4f1719 | myshare | 1    | NFS         |␣
↪available | False     | dhss_true       |      | nova              |
+--------------------------------------+---------+------+-------------+---
↪--------+-----------+-----------------+------+-------------------+
```

- Unset share metadata

```
$ manila metadata myshare unset year_started
```

Share revert to snapshot

- Share revert to snapshot

Note:

- To revert a share to its snapshot, the share type of the share must contain the capability extra-spec `revert_to_snapshot_support=True`.
- The revert operation can only be performed to the most recent available snapshot of the share known to manila. If revert to an earlier snapshot is desired, later snapshots must explicitly be deleted.

```
$ manila revert-to-snapshot mysnapshot
```

Create and manage share networks

- *Create share networks*
- *List share networks*
- *Update share networks*
- *Share network show*
- *Add security service/s*
- *List share network security services*
- *Remove a security service from a share network*
- *Delete share networks*
- *Update share network security service check (Since API version 2.63)*
- *Update share network security services (Since API version 2.63)*
- *Add share network security service check (Since API version 2.63)*

A share network stores network information to create and manage shares. A share network provides a way to designate a network to export shares upon. In the most common use case, you can create a share network with a private OpenStack (neutron) network that you own. If the share network is an isolated network, manila can provide hard guarantees of network and data isolation for your shared file systems in a multi-tenant cloud. In some clouds, however, shares cannot be exported directly upon private project networks; and the cloud may have provider networks that are designated for use with share networks.

In either case, as long as the underlying network is connected to the clients (virtual machines, containers or bare metals), there will exist a direct path to communicate with shares exported on the share networks.

Important: In order to use share networks, the share type you choose must have the extra specification `driver_handles_share_servers` set to True.

Create share networks

1. Create a share network.

```
$ manila share-network-create \
   --name sharenetwork1 \
   --description "Share Network created for demo purposes" \
   --neutron-net-id c297b020-025a-4f3e-8120-57ea90404afb \
   --neutron-subnet-id 29ecfbd5-a9be-467e-8b4a-3415d1f82888
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| name              | sharenetwork1                        |
| segmentation_id   | None                                 |
| created_at        | 2019-07-02T11:14:06.228816           |
```

(continues on next page)

(continued from previous page)

```
| neutron_subnet_id | 29ecfbd5-a9be-467e-8b4a-3415d1f82888     |
| updated_at        | None                                     |
| network_type      | None                                     |
| neutron_net_id    | c297b020-025a-4f3e-8120-57ea90404afb     |
| ip_version        | None                                     |
| cidr              | None                                     |
| project_id        | 907004508ef4447397ce6741a8f037c1         |
| id                | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1     |
| description       | Share Network created for demo purposes |
+-------------------+------------------------------------------+
```

2. Show the created share network.

```
$ manila share-network-show sharenetwork1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| id                | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1 |
| name              | sharenetwork1                        |
| project_id        | 5b23075b4b504261a5987b18588f86cf     |
| created_at        | 2019-10-09T04:19:31.000000           |
| updated_at        | None                                 |
| neutron_net_id    | c297b020-025a-4f3e-8120-57ea90404afb |
| neutron_subnet_id | 29ecfbd5-a9be-467e-8b4a-3415d1f82888 |
| network_type      | None                                 |
| segmentation_id   | None                                 |
| cidr              | None                                 |
| ip_version        | None                                 |
| description       | None                                 |
| gateway           | None                                 |
| mtu               | None                                 |
+-------------------+--------------------------------------+
```

Note: Since API version 2.51, a share network is able to span multiple subnets in different availability zones and the network information will be stored on each subnet. To accommodate adding multiple subnets, the share network create command was updated to accept an availability zone as parameter. This parameter will be used in the share network creation process which also creates a new subnet. If you do not specify an availability zone, the created subnet will be considered default by the Shared File Systems service. A default subnet is expected to be available in all availability zones of the cloud. So when you are creating a share network, the output will be similar to:

```
$ manila share-network-create \
   --name sharenetwork1 \
   --description "Share Network created for demo purposes" \
   --availability-zone manila-zone-0
+----------------------+--------------------------------------------------------------------
↪--------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------
↪-----------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| Property                | Value                                                
```

(continued from previous page)

List share networks

1. List share networks.

```
$ manila share-network-list
+--------------------------------------+---------------+
| id                                   | name          |
+--------------------------------------+---------------+
| feed6a6c-f9e0-45ba-9a2b-0db76bde63e1 | sharenetwork1 |
+--------------------------------------+---------------+
```

Update share networks

1. Update the share network data.

```
$ manila share-network-update sharenetwork1 \
  --neutron-net-id a27160ca-5595-4c62-bf54-a04fb7b14316 \
  --neutron-subnet-id f043f4b0-c05e-493f-bbe9-99689e2187d2
  +-------------------+--------------------------------------+
  | Property          | Value                                |
  +-------------------+--------------------------------------+
  | id                | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1 |
  | name              | sharenetwork1                        |
  | project_id        | 5b23075b4b504261a5987b18588f86cf     |
  | created_at        | 2019-10-09T04:19:31.000000           |
  | updated_at        | 2019-10-10T17:14:08.970945           |
  | neutron_net_id    | a27160ca-5595-4c62-bf54-a04fb7b14316 |
  | neutron_subnet_id | f043f4b0-c05e-493f-bbe9-99689e2187d2 |
  | network_type      | None                                 |
  | segmentation_id   | None                                 |
  | cidr              | None                                 |
  | ip_version        | None                                 |
  | description       | None                                 |
  | gateway           | None                                 |
  | mtu               | None                                 |
  +-------------------+--------------------------------------+
```

2. Show details of the updated share network.

```
$ manila share-network-show sharenetwork1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| id                | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1 |
| name              | sharenetwork1                        |
| project_id        | 5b23075b4b504261a5987b18588f86cf     |
```

(continues on next page)

(continued from previous page)

```
| created_at         | 2019-10-09T04:19:31.000000           |
| updated_at         | 2019-10-10T17:14:09.000000           |
| neutron_net_id     | a27160ca-5595-4c62-bf54-a04fb7b14316 |
| neutron_subnet_id  | f043f4b0-c05e-493f-bbe9-99689e2187d2 |
| network_type       | None                                 |
| segmentation_id    | None                                 |
| cidr               | None                                 |
| ip_version         | None                                 |
| description        | None                                 |
| gateway            | None                                 |
| mtu                | None                                 |
+--------------------+--------------------------------------+
```

Note: You cannot update the `neutron_net_id` and `neutron_subnet_id` of a share network that has shares exported onto it.

Note: From API version 2.51, updating the `neutron_net_id` and `neutron_subnet_id` is possible only for a default subnet. Non default subnets cannot be updated after they are created. You may delete the subnet in question, and re-create it. The output will look as shown below:

```
$ manila share-network-update sharenetwork1 \
   --neutron-net-id a27160ca-5595-4c62-bf54-a04fb7b14316 \
   --neutron-subnet-id f043f4b0-c05e-493f-bbe9-99689e2187d2
+----------------------+--------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪-----------------------------------------------------+
| Property             | Value                                            ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                          |
+----------------------+--------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪--------------------------------------------------------------------------
↪-----------------------------------------------------+
| id                   | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1             ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                                          ␣
↪                                                          |
```

(continues on next page)

(continued from previous page)

```
| name                  | sharenetwork1                                                  ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      |
| project_id            | 8c2962a4832743469a336f7c179f7d34                               ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      |
| created_at            | 2019-10-09T04:19:31.000000                                     ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      |
| updated_at            | 2019-10-10T17:14:09.000000                                     ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      |
| description           | Share Network created for demo purposes                        ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      ␣
↪                                                                                      |
| share_network_subnets | [{'id': '900d9ddc-7062-404e-8ef5-f63b84782d89',
↪'availability_zone': None, 'created_at': '2019-10-09T04:19:31.000000',
↪'updated_at': '2019-10-09T07:39:59.000000', 'segmentation_id': None,
↪'neutron_net_id': 'a27160ca-5595-4c62-bf54-a04fb7b14316', 'neutron_
↪subnet_id': 'f043f4b0-c05e-493f-bbe9-99689e2187d2', 'ip_version': None,
↪'cidr': None, 'network_type': None, 'mtu': None, 'gateway': None}] |
+-----------------------+-------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------+
```

Share network show

1. Show details of a share network.

```
$ manila share-network-show sharenetwork1
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| id                | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1 |
| name              | sharenetwork1                        |
| project_id        | 5b23075b4b504261a5987b18588f86cf     |
| created_at        | 2019-10-09T04:19:31.000000           |
| updated_at        | 2019-10-10T17:14:09.000000           |
| neutron_net_id    | fake_updated_net_id                  |
| neutron_subnet_id | fake_updated_subnet_id               |
| network_type      | None                                 |
| segmentation_id   | None                                 |
| cidr              | None                                 |
| ip_version        | None                                 |
| description       | None                                 |
| gateway           | None                                 |
| mtu               | None                                 |
+-------------------+--------------------------------------+
```

Note: Since API version 2.51, the `share-network-show` command also shows a list of subnets contained in the share network as show below.

```
+-----------------------+-------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-----------------------------------------------------------+
| Property              | Value                                           ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                 |
+-----------------------+-------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-------------------------------------------------------------------------
↪-----------------------------------------------------------+
| id                    | feed6a6c-f9e0-45ba-9a2b-0db76bde63e1            ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                         ␣
↪                                                                 |
```

(continues on next page)

(continued from previous page)

```
| name                  | sharenetwork1                                                 ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                    |
| project_id            | 8c2962a4832743469a336f7c179f7d34                              ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                    |
| created_at            | 2019-10-09T04:19:31.000000                                    ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                    |
| updated_at            | None                                                          ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                    |
| description           | Share Network created for demo purposes                       ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                    |
| share_network_subnets | [{'id': '900d9ddc-7062-404e-8ef5-f63b84782d89',
↪'availability_zone': None, 'created_at': '2019-10-09T04:19:31.000000',
↪'updated_at': '2019-10-09T07:39:59.000000', 'segmentation_id': None,
↪'neutron_net_id': 'fake_updated_net_id', 'neutron_subnet_id': 'fake_
↪updated_subnet_id', 'ip_version': None, 'cidr': None, 'network_type':␣
↪None, 'mtu': None, 'gateway': None}]                                    |
+-----------------------+-----------------------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------------------------------------------
↪-----------------------------------------------------------+
```

Note: Since API version 2.63, the `share-network-show` command also shows the `status` and `security_service_update_support` fields.

```
+---------------------------------+-----------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------------------------------------------
↪----------------------------------------
```

(continued from previous page)

```
| Property                    | Value                                                
```

(continued from previous page)

```
| status                          | active                                                
```

List share network security services

1. List all the security services existent in a share network.

```
$ manila share-network-security-service-list sharenetwork1
+--------------------------------------+----------------+--------+------+
| id                                   | name           | status | type |
+--------------------------------------+----------------+--------+------+
| 50303c35-2c53-4d37-a0d9-61dfe3789569 | my_sec_service | new    | ldap |
+--------------------------------------+----------------+--------+------+
```

Remove a security service from a share network

1. Remove a security service from a given share network.

```
$ manila share-network-security-service-remove \
   sharenetwork1 \
   my_sec_service
$ manila share-network-security-service-list sharenetwork1
+----+------+--------+------+
| id | name | status | type |
+----+------+--------+------+
+----+------+--------+------+
```

Delete share networks

1. Delete a share network.

```
$ manila share-network-delete sharenetwork1
```

2. List all share networks

```
$ manila share-network-list
+--------------------------------------+---------------+
| id                                   | name          |
+--------------------------------------+---------------+
+--------------------------------------+---------------+
```

Update share network security service check (Since API version 2.63)

1. Check if the update for security services of the same type can be performed:

```
$ manila share-network-security-service-update-check \
   sharenetwork1 \
   my_sec_service \
   my_sec_service_updated
+---------------------+-------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| Property             | Value                                                
```

Update share network security services (Since API version 2.63)

1. Replaces one security service for another of the same type.

```
$ manila share-network-security-service-update \
    sharenetwork1 \
    my_sec_service \
    my_sec_service_updated
$ manila share-network-security-service-list sharenetwork1
+--------------------------------------+------------------------+--------
↪+------+
| id                                   | name                   | status␣
↪| type |
+--------------------------------------+------------------------+--------
↪+------+
| 8971c5f6-52ec-4c53-bf6a-3fae38a9221e | my_sec_service_updated | new    ␣
↪| ldap |
+--------------------------------------+------------------------+--------
↪+------+
```

Note: The share network entity now contains a field called `security_service_update_support` which holds information whether all resources built within it can hold such operation. In order to update security services in share networks that currently contain shares, an operation to check if the operation can be completed must be performed. See *subsection*.

Add share network security service check (Since API version 2.63)

1. Check if it is possible to add a security service to a share network:

```
$ manila share-network-security-service-add-check \
   sharenetwork1 \
   my_sec_service
+---------------------+-----------------------------------------------------
↪-------------------------------------------------------------------------
↪--------------+
| Property            | Value                                               ␣
↪                                                                          ␣
↪              |
+---------------------+-----------------------------------------------------
↪-------------------------------------------------------------------------
↪--------------+
| compatible          | None                                                ␣
↪                                                                          ␣
↪              |
| requested_operation | {'operation': 'add_security_service', 'current_
↪security_service': None, 'new_security_service': '50303c35-2c53-4d37-
↪a0d9-61dfe3789569'} |
+---------------------+-----------------------------------------------------
↪-------------------------------------------------------------------------
↪--------------+
```

(continues on next page)

(continued from previous page)

2. Check if the result of the operation:

```
$ manila share-network-security-service-add-check \
   sharenetwork1 \
   my_sec_service
+---------------------+-------------------------------------------------------
↪------------------------------------------------------------------------------
↪--------------+
| Property            | Value                                                 
↪                                                                             
↪              |
+---------------------+-------------------------------------------------------
↪------------------------------------------------------------------------------
↪--------------+
| compatible          | True                                                  
↪                                                                             
↪              |
| requested_operation | {'operation': 'add_security_service', 'current_
↪security_service': None, 'new_security_service': '50303c35-2c53-4d37-
↪a0d9-61dfe3789569'} |
+---------------------+-------------------------------------------------------
↪------------------------------------------------------------------------------
↪--------------+
```

Create and manage share network subnets

- *Create a subnet in an existing share network*
- *Show a share network subnet*
- *Delete a share network subnet*

A share network subnet stores network information to create and manage shares. To create and manage your share network subnets, you can use `manila` client commands. You can create multiple subnets in a share network, and if you do not specify an availability zone, the subnet you are creating will be considered default by the Shared File Systems service. The default subnet spans all availability zones. You cannot have more than one default subnet per share network.

Important: In order to use share networks, the share type you choose must have the extra specification `driver_handles_share_servers` set to True.

Create a subnet in an existing share network

1. Create a subnet related to the given share network

```
$ manila share-network-subnet-create \
   sharenetwork1 \
   --availability-zone manila-zone-0 \
   --neutron-net-id a27160ca-5595-4c62-bf54-a04fb7b14316 \
   --neutron-subnet-id f043f4b0-c05e-493f-bbe9-99689e2187d2
+--------------------+--------------------------------------+
| Property           | Value                                |
+--------------------+--------------------------------------+
| id                 | be3ae5ad-a22c-494f-840e-5e3526e34e0f |
| availability_zone  | manila-zone-0                        |
| share_network_id   | 35f44d3c-8888-429e-b8c7-8a29dead6e5b |
| share_network_name | sharenetwork1                        |
| created_at         | 2019-10-09T04:54:48.000000           |
| segmentation_id    | None                                 |
| neutron_subnet_id  | f043f4b0-c05e-493f-bbe9-99689e2187d2 |
| updated_at         | None                                 |
| neutron_net_id     | a27160ca-5595-4c62-bf54-a04fb7b14316 |
| ip_version         | None                                 |
| cidr               | None                                 |
| network_type       | None                                 |
| mtu                | None                                 |
| gateway            | None                                 |
+--------------------+--------------------------------------+
```

2. Show the share network to verify if the created subnet is attached

```
$ manila share-network-show sharenetwork1
+-----------------------+-----------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------+
| Property              | Value                                               ␣
↪                                                                             ␣
↪                                                                             ␣
↪                                                                             ␣
↪                                                                             ␣
↪                                                    |
+-----------------------+-----------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------------------------------------
↪-----------------------------------------------+
| id                    | 35f44d3c-8888-429e-b8c7-8a29dead6e5b                ␣
↪                                                                             ␣
↪                                                                             ␣
↪                                                                             ␣
↪                                                    |
```

(continues on next page)

(continued from previous page)

```
| name                  | sharenetwork1                                                
```

Show a share network subnet

1. Show an existent subnet in a given share network

```
$ manila share-network-subnet-show \
   sharenetwork1 \
   be3ae5ad-a22c-494f-840e-5e3526e34e0f
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| id                | be3ae5ad-a22c-494f-840e-5e3526e34e0f |
| availability_zone | manila-zone-0                        |
| share_network_id  | 35f44d3c-8888-429e-b8c7-8a29dead6e5b |
| share_network_name| sharenetwork1                        |
| created_at        | 2019-10-09T04:54:48.000000           |
| segmentation_id   | None                                 |
| neutron_subnet_id | f043f4b0-c05e-493f-bbe9-99689e2187d2 |
| updated_at        | None                                 |
| neutron_net_id    | a27160ca-5595-4c62-bf54-a04fb7b14316 |
| ip_version        | None                                 |
| cidr              | None                                 |
| network_type      | None                                 |
| mtu               | None                                 |
| gateway           | None                                 |
+-------------------+--------------------------------------+
```

Delete a share network subnet

1. Delete a specific share network subnet

```
$ manila share-network-subnet-delete \
   sharenetwork1 \
   be3ae5ad-a22c-494f-840e-5e3526e34e0f
```

2. Verify that it has been deleted

```
$ manila share-network-show sharenetwork1
+----------------------+------------------------------------------+
| Property             | Value                                    |
+----------------------+------------------------------------------+
| id                   | 35f44d3c-8888-429e-b8c7-8a29dead6e5b     |
| name                 | sharenetwork1                            |
| project_id           | 8c2962a4832743469a336f7c179f7d34         |
| created_at           | 2019-10-09T04:19:31.000000               |
| updated_at           | None                                     |
| description          | Share Network created for demo purposes  |
| share_network_subnets| []                                       |
+----------------------+------------------------------------------+
```

Troubleshooting asynchronous failures

The Shared File Systems service performs many user actions asynchronously. For example, when a new share is created, the request is immediately acknowledged with a response containing the metadata of the share. Users can then query the resource and check the `status` attribute of the share. Usually an `...ing` status indicates that actions are performed asynchronously. For example, a new shares `status` attribute is set to `creating` by the service. If these asynchronous operations fail, the resources status will be set to `error`. More information about the error can be obtained with the help of the CLI client.

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. In order to create a share, you need to specify the share type that meets your requirements. Cloud administrators create share types; see these 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  ␣
↪                  |             |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-------------+-----------+-------
↪-----+--------------------------------------+--------------------------
↪-----------------+-------------+
```

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.

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

3. Create a share network from a private tenant network:

```
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  |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-------+
| 6c7ef9ef-3591-48b6-b18a-71a03059edd5 | mynet |
+--------------------------------------+-------+
```

4. 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                          ␣
↪     |
```

(continues on next page)

(continued from previous page)

```
| 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                             | {}                             
↪     |
+--------------------------------------+--------------------------------
↪-----+
```

5. 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.

6. To view the generated user message, use the `message-list` command. Use `--resource-id` to filter messages for a specific share resource.

```
clouduser1@client:~$ manila message-list
+--------------------------------------+---------------+------------------
↪-------------------+-----------+----------------------------------------
↪----------------------------------------------------------------------+----
↪-------+----------------------------+
| ID                                   | Resource Type | Resource ID
↪                   | Action ID | User Message
↪                                                                      |
↪Detail ID | Created At                 |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+--------------+------------------
↪--------------------+----------+--------------------------------------
↪-------------------------------------------------------------------+----
↪-------+----------------------------+
| 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 User Message column, you can see that the Shared File System service failed to create the share because of a capabilities mismatch.

7. To view more 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                                                       ␣
↪                                                |
+---------------+-----------------------------------------------------------
↪-----------------------------------------------+
```

As the cloud user, you know the related specs your share type has, 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                       ␣
↪     |
```

(continues on next page)

(continued from previous page)

```
| 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                               | {}                                  ␣
↪      |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-------------------------------
↪-----+
```

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. 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.

10. 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                           ␣
↪     |
```

(continues on next page)

(continued from previous page)

```
| 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 shares that failed to be created and corresponding support messages:

```
clouduser1@client:~$ manila delete 2d03d480-7cba-4122-ac9d-edc59c8df698
↪243f3a51-0624-4bdd-950e-7ed190b53b67
clouduser1@client:~$ manila message-list
+--------------------------------------+---------------+------------------
↪--------------------+-----------+-------------------------------------
↪-----------------------------------------------------------------+----
↪-------+----------------------------+
| ID                                   | Resource Type | Resource ID
↪                    | Action ID | User Message
↪                                                                  |
↪Detail ID | Created At                 |
```

(continues on next page)

(continued from previous page)

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

2.2 Using the Manila API

All features of Manila are exposed via a REST API that can be used to build more complicated logic or automation with Manila. This can be consumed directly or via various SDKs. The following resources can help you get started consuming the API directly:

- Manila API
- *Manila microversion history*

CHAPTER

THREE

FOR OPERATORS

This section has details for deploying and maintaining Manila services.

3.1 Installing Manila

Manila can be configured standalone using the configuration setting `auth_strategy = noauth`, but in most cases you will want to at least have the Keystone Identity service and other OpenStack services installed.

3.1.1 Installation Tutorial

Service Overview

The OpenStack Shared File Systems service (manila) provides file storage to a virtual machine. The Shared File Systems service provides an abstraction for managing and provisioning of file shares. The service also enables management of share types as well as share snapshots if a driver supports them.

The Shared File Systems service consists of the following components:

manila-api A WSGI app that authenticates and routes requests to the Shared File Systems service.

manila-data A standalone service whose purpose is to process data operations such as copying, share migration or backup.

manila-scheduler Schedules and routes requests to the appropriate share service. The scheduler uses configurable filters and weighers to route requests. The Filter Scheduler is the default and enables filters on various attributes of back ends, such as, Capacity, Availability Zone and other capabilities.

manila-share Manages back-end devices that provide shared file systems. A manila-share service talks to back-end devices by using share back-end drivers as interfaces. A share driver may operate in one of two modes, with or without handling of share servers. Share servers export file shares via share networks. When share servers are not managed by a driver within the shared file systems service, networking requirements should be handled out of band of the shared file systems service.

Messaging queue Routes information between the Shared File Systems processes.

For more information, see Configuration Reference Guide.

Install and configure controller node

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node. This service requires at least one additional share node that manages file storage back ends.

This section assumes that you already have a working OpenStack environment with at least the following components installed: Compute, Image Service, Identity.

Note that installation and configuration vary by distribution.

Install and configure controller node on openSUSE and SUSE Linux Enterprise

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node that runs openSUSE and SUSE Linux Enterprise. This service requires at least one additional share node that manages file storage back ends.

Prerequisites

Before you install and configure the Shared File Systems service, you must create a database, service credentials, and *API endpoints*.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the *manila* database:

     ```
     CREATE DATABASE manila;
     ```

   - Grant proper access to the `manila` database:

     ```
     GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'localhost' \
       IDENTIFIED BY 'MANILA_DBPASS';
     GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'%' \
       IDENTIFIED BY 'MANILA_DBPASS';
     ```

     Replace `MANILA_DBPASS` with a suitable password.

   - Exit the database access client.
2. Source the `admin` credentials to gain access to admin CLI commands:

   ```
   $ . admin-openrc.sh
   ```

3. To create the service credentials, complete these steps:
   - Create a `manila` user:

```
$ openstack user create --domain default --password-prompt manila
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | e0353a670a9e496da891347c589539e9 |
| enabled             | True                             |
| id                  | 83a3990fc2144100ba0e2e23886d8acc |
| name                | manila                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

- Add the `admin` role to the `manila` user:

```
$ openstack role add --project service --user manila admin
```

Note: This command provides no output.

- Create the `manila` and `manilav2` service entities:

```
$ openstack service create --name manila \
  --description "OpenStack Shared File Systems" share
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 82378b5a16b340aa9cc790cdd46a03ba |
  | name        | manila                           |
  | type        | share                            |
  +-------------+----------------------------------+
```

```
$ openstack service create --name manilav2 \
  --description "OpenStack Shared File Systems V2" sharev2
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems V2 |
  | enabled     | True                             |
  | id          | 30d92a97a81a4e5d8fd97a32bafd7b88 |
  | name        | manilav2                         |
  | type        | sharev2                          |
  +-------------+----------------------------------+
```

Note: The Shared File Systems services require two service entities.

4. Create the Shared File Systems service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  share public http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 0bd2bbf8d28b433aaea56a254c69f69d        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share internal http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | a2859b5732cc48b5b083dd36dafb6fd9        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share admin http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | f7f46df93a374cc49c0121bef41da03c        |
  | interface    | admin                                   |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  sharev2 public http://controller:8786/v2
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | d63cc0d358da4ea680178657291eddc1        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
  | service_name | manilav2                                |
  | service_type | sharev2                                 |
  | url          | http://controller:8786/v2               |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 internal http://controller:8786/v2
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | afc86e5f50804008add349dba605da54        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
  | service_name | manilav2                                |
  | service_type | sharev2                                 |
  | url          | http://controller:8786/v2               |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 admin http://controller:8786/v2
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | e814a0cec40546e98cf0c25a82498483        |
  | interface    | admin                                   |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88        |
  | service_name | manilav2                                |
  | service_type | sharev2                                 |
  | url          | http://controller:8786/v2               |
  +--------------+-----------------------------------------+
```

Note: The Shared File Systems services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# zypper install openstack-manila-api openstack-manila-scheduler python-
↪manilaclient
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

3. Complete the rest of the configuration in `manila.conf`:

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
share_name_template = share-%s
rootwrap_config = /etc/manila/rootwrap.conf
api_paste_config = /etc/manila/api-paste.ini
```

     Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference to the driver mode used. This is further explained in the section discussing the setup and configuration of the share node.

   - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
```

(continues on next page)

(continued from previous page)

```
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lock/manila
```

Finalize installation

1. Start the Shared File Systems services and configure them to start when the system boots:

```
# systemctl enable openstack-manila-api.service openstack-manila-
↪scheduler.service
# systemctl start openstack-manila-api.service openstack-manila-scheduler.
↪service
```

Install and configure controller node on Red Hat Enterprise Linux and CentOS

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node that runs Red Hat Enterprise Linux or CentOS. This service requires at least one additional share node that manages file storage back ends.

Prerequisites

Before you install and configure the Shared File Systems service, you must create a database, service credentials, and *API endpoints*.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

     ```
     $ mysql -u root -p
     ```

   - Create the *manila* database:

     ```
     CREATE DATABASE manila;
     ```

   - Grant proper access to the `manila` database:

     ```
     GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'localhost' \
       IDENTIFIED BY 'MANILA_DBPASS';
     GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'%' \
       IDENTIFIED BY 'MANILA_DBPASS';
     ```

     Replace `MANILA_DBPASS` with a suitable password.

   - Exit the database access client.
2. Source the `admin` credentials to gain access to admin CLI commands:

   ```
   $ . admin-openrc.sh
   ```

3. To create the service credentials, complete these steps:
   - Create a `manila` user:

     ```
     $ openstack user create --domain default --password-prompt manila
     User Password:
     Repeat User Password:
     +---------------------+----------------------------------+
     | Field               | Value                            |
     +---------------------+----------------------------------+
     | domain_id           | e0353a670a9e496da891347c589539e9 |
     | enabled             | True                             |
     | id                  | 83a3990fc2144100ba0e2e23886d8acc |
     | name                | manila                           |
     | options             | {}                               |
     | password_expires_at | None                             |
     +---------------------+----------------------------------+
     ```

- Add the `admin` role to the `manila` user:

```
$ openstack role add --project service --user manila admin
```

Note: This command provides no output.

- Create the `manila` and `manilav2` service entities:

```
$ openstack service create --name manila \
  --description "OpenStack Shared File Systems" share
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 82378b5a16b340aa9cc790cdd46a03ba |
  | name        | manila                           |
  | type        | share                            |
  +-------------+----------------------------------+
```

```
$ openstack service create --name manilav2 \
  --description "OpenStack Shared File Systems V2" sharev2
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems V2 |
  | enabled     | True                             |
  | id          | 30d92a97a81a4e5d8fd97a32bafd7b88 |
  | name        | manilav2                         |
  | type        | sharev2                          |
  +-------------+----------------------------------+
```

Note: The Shared File Systems services require two service entities.

4. Create the Shared File Systems service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  share public http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 0bd2bbf8d28b433aaea56a254c69f69d        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
```

(continues on next page)

(continued from previous page)

```
  | service_type | share                                        |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+----------------------------------------------+

$ openstack endpoint create --region RegionOne \
  share internal http://controller:8786/v1/%\(tenant_id\)s
  +--------------+----------------------------------------------+
  | Field        | Value                                        |
  +--------------+----------------------------------------------+
  | enabled      | True                                         |
  | id           | a2859b5732cc48b5b083dd36dafb6fd9             |
  | interface    | internal                                     |
  | region       | RegionOne                                    |
  | region_id    | RegionOne                                    |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba             |
  | service_name | manila                                       |
  | service_type | share                                        |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+----------------------------------------------+

$ openstack endpoint create --region RegionOne \
  share admin http://controller:8786/v1/%\(tenant_id\)s
  +--------------+----------------------------------------------+
  | Field        | Value                                        |
  +--------------+----------------------------------------------+
  | enabled      | True                                         |
  | id           | f7f46df93a374cc49c0121bef41da03c             |
  | interface    | admin                                        |
  | region       | RegionOne                                    |
  | region_id    | RegionOne                                    |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba             |
  | service_name | manila                                       |
  | service_type | share                                        |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+----------------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  sharev2 public http://controller:8786/v2
  +--------------+----------------------------------------------+
  | Field        | Value                                        |
  +--------------+----------------------------------------------+
  | enabled      | True                                         |
  | id           | d63cc0d358da4ea680178657291eddc1             |
  | interface    | public                                       |
  | region       | RegionOne                                    |
  | region_id    | RegionOne                                    |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88             |
  | service_name | manilav2                                     |
  | service_type | sharev2                                      |
```

(continues on next page)

(continued from previous page)

```
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 internal http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | afc86e5f50804008add349dba605da54         |
  | interface    | internal                                 |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88         |
  | service_name | manilav2                                 |
  | service_type | sharev2                                  |
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 admin http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | e814a0cec40546e98cf0c25a82498483         |
  | interface    | admin                                    |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88         |
  | service_name | manilav2                                 |
  | service_type | sharev2                                  |
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+
```

Note: The Shared File Systems services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# yum install openstack-manila python3-manilaclient
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

3. Complete the rest of the configuration in `manila.conf`:

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
share_name_template = share-%s
rootwrap_config = /etc/manila/rootwrap.conf
api_paste_config = /etc/manila/api-paste.ini
```

     Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference to the driver mode used. This is further explained in the section discussing the setup and configuration of the share node.

   - In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lock/manila
```

4. Populate the Shared File Systems database:

```
# su -s /bin/sh -c "manila-manage db sync" manila
```

Note: Ignore any deprecation messages in this output.

Finalize installation

1. Start the Shared File Systems services and configure them to start when the system boots:

```
# systemctl enable openstack-manila-api.service openstack-manila-
↪scheduler.service
# systemctl start openstack-manila-api.service openstack-manila-scheduler.
↪service
```

Install and configure controller node on Ubuntu

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node that runs Ubuntu. This service requires at least one additional share node that manages file storage back ends.

Prerequisites

Before you install and configure the Shared File Systems service, you must create a database, service credentials, and *API endpoints*.

1. To create the database, complete these steps:

- Use the database access client to connect to the database server as the `root` user:

```
$ mysql -u root -p
```

- Create the *manila* database:

```
CREATE DATABASE manila;
```

- Grant proper access to the `manila` database:

```
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'localhost' \
  IDENTIFIED BY 'MANILA_DBPASS';
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'%' \
  IDENTIFIED BY 'MANILA_DBPASS';
```

  Replace `MANILA_DBPASS` with a suitable password.

- Exit the database access client.

2. Source the `admin` credentials to gain access to admin CLI commands:

```
$ . admin-openrc.sh
```

3. To create the service credentials, complete these steps:

   - Create a `manila` user:

```
$ openstack user create --domain default --password-prompt manila
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | e0353a670a9e496da891347c589539e9 |
| enabled             | True                             |
| id                  | 83a3990fc2144100ba0e2e23886d8acc |
| name                | manila                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

   - Add the `admin` role to the `manila` user:

```
$ openstack role add --project service --user manila admin
```

   Note: This command provides no output.

   - Create the `manila` and `manilav2` service entities:

```
$ openstack service create --name manila \
  --description "OpenStack Shared File Systems" share
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 82378b5a16b340aa9cc790cdd46a03ba |
```

(continues on next page)

(continued from previous page)

```
  | name        | manila                           |
  | type        | share                            |
  +-------------+----------------------------------+
```

```
$ openstack service create --name manilav2 \
  --description "OpenStack Shared File Systems V2" sharev2
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems V2 |
  | enabled     | True                             |
  | id          | 30d92a97a81a4e5d8fd97a32bafd7b88 |
  | name        | manilav2                         |
  | type        | sharev2                          |
  +-------------+----------------------------------+
```

Note: The Shared File Systems services require two service entities.

4. Create the Shared File Systems service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  share public http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 0bd2bbf8d28b433aaea56a254c69f69d        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share internal http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | a2859b5732cc48b5b083dd36dafb6fd9        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
```

(continues on next page)

(continued from previous page)

```
  | service_type | share                                       |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+---------------------------------------------+

$ openstack endpoint create --region RegionOne \
  share admin http://controller:8786/v1/%\(tenant_id\)s
  +--------------+---------------------------------------------+
  | Field        | Value                                       |
  +--------------+---------------------------------------------+
  | enabled      | True                                        |
  | id           | f7f46df93a374cc49c0121bef41da03c            |
  | interface    | admin                                       |
  | region       | RegionOne                                   |
  | region_id    | RegionOne                                   |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba            |
  | service_name | manila                                      |
  | service_type | share                                       |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+---------------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  sharev2 public http://controller:8786/v2
  +--------------+---------------------------------------+
  | Field        | Value                                 |
  +--------------+---------------------------------------+
  | enabled      | True                                  |
  | id           | d63cc0d358da4ea680178657291eddc1      |
  | interface    | public                                |
  | region       | RegionOne                             |
  | region_id    | RegionOne                             |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88      |
  | service_name | manilav2                              |
  | service_type | sharev2                               |
  | url          | http://controller:8786/v2             |
  +--------------+---------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 internal http://controller:8786/v2
  +--------------+---------------------------------------+
  | Field        | Value                                 |
  +--------------+---------------------------------------+
  | enabled      | True                                  |
  | id           | afc86e5f50804008add349dba605da54      |
  | interface    | internal                              |
  | region       | RegionOne                             |
  | region_id    | RegionOne                             |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88      |
  | service_name | manilav2                              |
  | service_type | sharev2                               |
```

(continues on next page)

(continued from previous page)

```
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 admin http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | e814a0cec40546e98cf0c25a82498483         |
  | interface    | admin                                    |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88         |
  | service_name | manilav2                                 |
  | service_type | sharev2                                  |
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+
```

Note: The Shared File Systems services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# apt-get install manila-api manila-scheduler python3-manilaclient
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

3. Complete the rest of the configuration in `manila.conf`:

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
share_name_template = share-%s
rootwrap_config = /etc/manila/rootwrap.conf
api_paste_config = /etc/manila/api-paste.ini
```

Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference to the driver mode used. This is further explained in the section discussing the setup and configuration of the share node.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option to use the management interface IP address of the controller node:

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lock/manila
```

4. Populate the Shared File Systems database:

```
# su -s /bin/sh -c "manila-manage db sync" manila
```

Note: Ignore any deprecation messages in this output.

Finalize installation

1. Restart the Shared File Systems services:

```
# service manila-scheduler restart
# service manila-api restart
```

2. By default, the Ubuntu packages create an SQLite database. Because this configuration uses an SQL database server, you can remove the SQLite database file:

```
# rm -f /var/lib/manila/manila.sqlite
```

Install and configure controller node on Debian

This section describes how to install and configure the Shared File Systems service, code-named manila, on the controller node that runs a Debian distribution. This service requires at least one additional share node that manages file storage back ends.

Prerequisites

Before you install and configure the Shared File Systems service, you must create a database, service credentials, and *API endpoints*.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:

```
$ mysql -u root -p
```

   - Create the *manila* database:

```
CREATE DATABASE manila;
```

   - Grant proper access to the `manila` database:

```
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'localhost' \
  IDENTIFIED BY 'MANILA_DBPASS';
GRANT ALL PRIVILEGES ON manila.* TO 'manila'@'%' \
  IDENTIFIED BY 'MANILA_DBPASS';
```

     Replace `MANILA_DBPASS` with a suitable password.

   - Exit the database access client.

2. Source the `admin` credentials to gain access to admin CLI commands:

```
$ . admin-openrc.sh
```

3. To create the service credentials, complete these steps:

   - Create a `manila` user:

```
$ openstack user create --domain default --password-prompt manila
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | e0353a670a9e496da891347c589539e9 |
| enabled             | True                             |
| id                  | 83a3990fc2144100ba0e2e23886d8acc |
| name                | manila                           |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

   - Add the `admin` role to the `manila` user:

```
$ openstack role add --project service --user manila admin
```

   Note: This command provides no output.

   - Create the `manila` and `manilav2` service entities:

```
$ openstack service create --name manila \
  --description "OpenStack Shared File Systems" share
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems    |
  | enabled     | True                             |
  | id          | 82378b5a16b340aa9cc790cdd46a03ba |
  | name        | manila                           |
  | type        | share                            |
  +-------------+----------------------------------+
```

```
$ openstack service create --name manilav2 \
  --description "OpenStack Shared File Systems V2" sharev2
  +-------------+----------------------------------+
  | Field       | Value                            |
  +-------------+----------------------------------+
  | description | OpenStack Shared File Systems V2 |
  | enabled     | True                             |
  | id          | 30d92a97a81a4e5d8fd97a32bafd7b88 |
  | name        | manilav2                         |
```

(continues on next page)

(continued from previous page)

```
| type        | sharev2                          |
+-------------+----------------------------------+
```

Note: The Shared File Systems services require two service entities.

4. Create the Shared File Systems service API endpoints:

```
$ openstack endpoint create --region RegionOne \
  share public http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | 0bd2bbf8d28b433aaea56a254c69f69d        |
  | interface    | public                                  |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share internal http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | a2859b5732cc48b5b083dd36dafb6fd9        |
  | interface    | internal                                |
  | region       | RegionOne                               |
  | region_id    | RegionOne                               |
  | service_id   | 82378b5a16b340aa9cc790cdd46a03ba        |
  | service_name | manila                                  |
  | service_type | share                                   |
  | url          | http://controller:8786/v1/%(project_id)s |
  +--------------+-----------------------------------------+

$ openstack endpoint create --region RegionOne \
  share admin http://controller:8786/v1/%\(tenant_id\)s
  +--------------+-----------------------------------------+
  | Field        | Value                                   |
  +--------------+-----------------------------------------+
  | enabled      | True                                    |
  | id           | f7f46df93a374cc49c0121bef41da03c        |
  | interface    | admin                                   |
  | region       | RegionOne                               |
```

(continues on next page)

(continued from previous page)

```
| region_id    | RegionOne                                |
| service_id   | 82378b5a16b340aa9cc790cdd46a03ba         |
| service_name | manila                                   |
| service_type | share                                    |
| url          | http://controller:8786/v1/%(project_id)s |
+--------------+------------------------------------------+
```

```
$ openstack endpoint create --region RegionOne \
  sharev2 public http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | d63cc0d358da4ea680178657291eddc1         |
  | interface    | public                                   |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88         |
  | service_name | manilav2                                 |
  | service_type | sharev2                                  |
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 internal http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | afc86e5f50804008add349dba605da54         |
  | interface    | internal                                 |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
  | service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88         |
  | service_name | manilav2                                 |
  | service_type | sharev2                                  |
  | url          | http://controller:8786/v2                |
  +--------------+------------------------------------------+

$ openstack endpoint create --region RegionOne \
  sharev2 admin http://controller:8786/v2
  +--------------+------------------------------------------+
  | Field        | Value                                    |
  +--------------+------------------------------------------+
  | enabled      | True                                     |
  | id           | e814a0cec40546e98cf0c25a82498483         |
  | interface    | admin                                    |
  | region       | RegionOne                                |
  | region_id    | RegionOne                                |
```

(continues on next page)

(continued from previous page)

```
| service_id   | 30d92a97a81a4e5d8fd97a32bafd7b88          |
| service_name | manilav2                                  |
| service_type | sharev2                                   |
| url          | http://controller:8786/v2                 |
+--------------+-------------------------------------------+
```

Note: The Shared File Systems services require endpoints for each service entity.

Install and configure components

1. Install the packages:

```
# apt-get install manila-api manila-scheduler python3-manilaclient
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

3. Complete the rest of the configuration in `manila.conf`:

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
share_name_template = share-%s
rootwrap_config = /etc/manila/rootwrap.conf
api_paste_config = /etc/manila/api-paste.ini
```

     Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference

to the driver mode used. This is further explained in the section discussing the setup and configuration of the share node.

- In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace MANILA_PASS with the password you chose for the manila user in the Identity service.

- In the [DEFAULT] section, configure the my_ip option to use the management interface IP address of the controller node:

```
[DEFAULT]
...
my_ip = 10.0.0.11
```

- In the [oslo_concurrency] section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lock/manila
```

4. Populate the Shared File Systems database:

```
# su -s /bin/sh -c "manila-manage db sync" manila
```

Note: Ignore any deprecation messages in this output.

Finalize installation

1. Restart the Shared File Systems services:

```
# service manila-scheduler restart
# service manila-api restart
```

Install and configure a share node

This section describes how to install and configure a share node for the Shared File Systems service.

Note: The manila-share process can run in two modes, with and without handling of share servers. Some drivers may support either modes; while some may only support one of the two modes. See the Configuration Reference to determine if the driver you choose supports the driver mode desired. This tutorial describes setting up each driver mode using an example driver for the mode.

Note that installation and configuration vary by distribution.

Install and configure a share node running openSUSE and SUSE Linux Enterprise

This section describes how to install and configure a share node for the Shared File Systems service.

Note that installation and configuration vary by distribution. This section describes the instructions for a share node running openSUSE and SUSE Linux Enterprise.

Install and configure components

1. Install the packages:

```
# zypper install openstack-manila-share python-PyMySQL
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

4. Complete the rest of the configuration in `manila.conf`.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

Replace RABBIT_PASS with the password you chose for the openstack account in RabbitMQ.

- In the [DEFAULT] section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
```

Important: The default_share_type option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as driver_handles_share_servers) set appropriately with reference to the driver mode used. This is explained in further steps.

- In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace MANILA_PASS with the password you chose for the manila user in the Identity service.

- In the [DEFAULT] section, configure the my_ip option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace MANAGEMENT_INTERFACE_IP_ADDRESS with the IP address of the management network interface on your share node, typically 10.0.0.41 for the first node in the example architecture shown below:

- In the [oslo_concurrency] section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

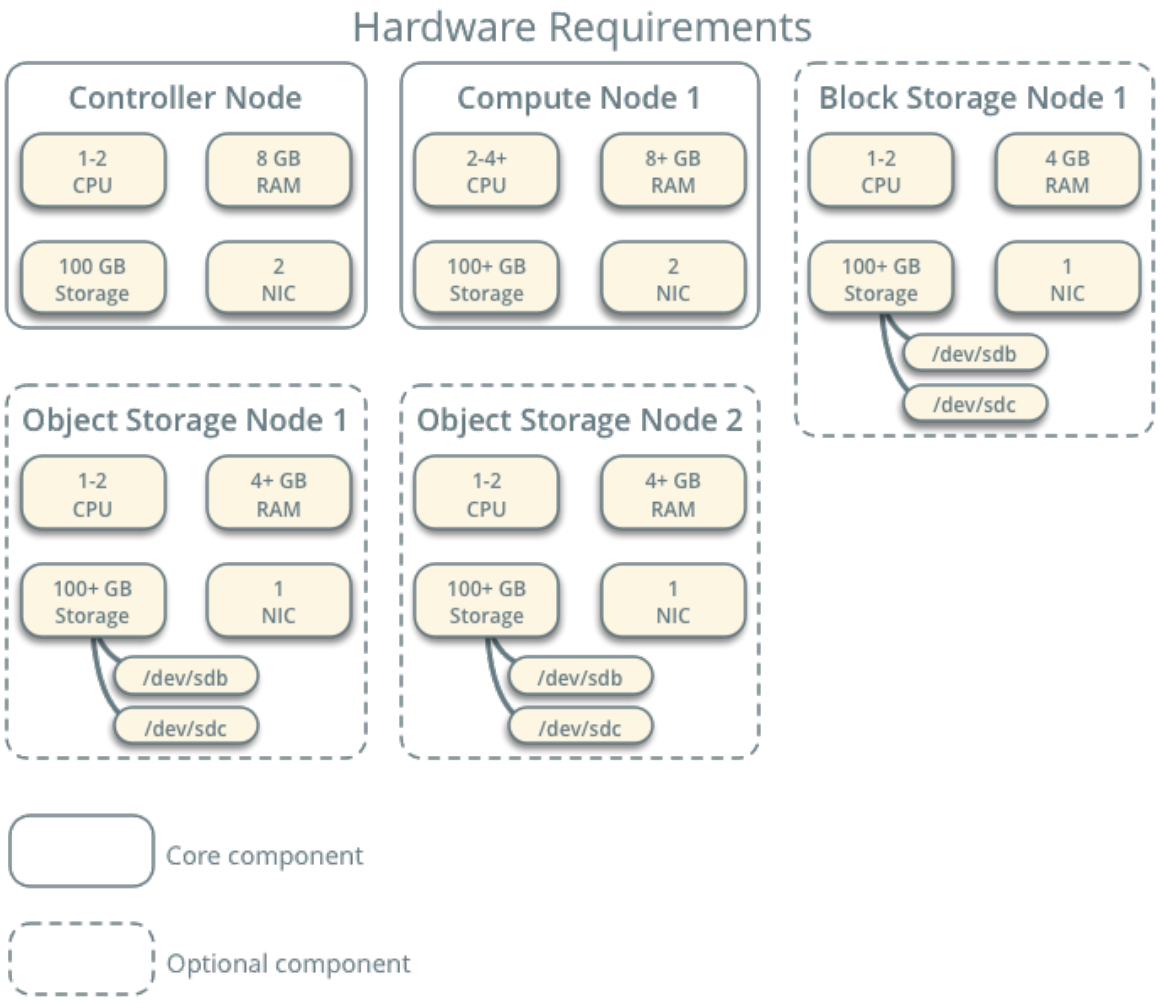


Fig. 1: **Hardware requirements**

Two driver modes

The share node can support two modes, with and without the handling of share servers. The mode depends on driver support.

Option 1

Deploying the service without driver support for share server management. In this mode, the service does not do anything related to networking. The operator must ensure network connectivity between instances and the NAS protocol based server.

This tutorial demonstrates setting up the LVM driver which creates LVM volumes on the share node and exports them with the help of an NFS server that is installed locally on the share node. It therefore requires LVM and NFS packages as well as an additional disk for the `manila-share` LVM volume group.

This driver mode may be referred to as `driver_handles_share_servers = False` mode, or simply `DHSS=False` mode.

Option 2

Deploying the service with driver support for share server management. In this mode, the service runs with a back end driver that creates and manages share servers. This tutorial demonstrates setting up the `Generic` driver. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares.

The information used for creating share servers is configured with the help of share networks.

This driver mode may be referred to as `driver_handles_share_servers = True` mode, or simply `DHSS=True` mode.

> **Warning:** When running the generic driver in `DHSS=True` driver mode, the share service should be run on the same node as the networking service. However, such a service may not be able to run the LVM driver that runs in `DHSS=False` driver mode effectively, due to a bug in some distributions of Linux. For more information, see LVM Driver section in the Configuration Reference Guide.

Choose one of the following options to configure the share driver:

Shared File Systems Option 1: No driver support for share servers management

For simplicity, this configuration references the same storage node configuration for the Block Storage service. However, the LVM driver requires a separate empty local block storage device to avoid conflict with the Block Storage service. The instructions use `/dev/sdc`, but you can substitute a different value for your particular node.

Prerequisites

Note: Perform these steps on the storage node.

1. Install the supporting utility packages:

   - Install LVM and NFS server packages:

   ```
   # zypper install lvm2 nfs-kernel-server
   ```

2. Create the LVM physical volume `/dev/sdc`:

   ```
   # pvcreate /dev/sdc
   Physical volume "/dev/sdc" successfully created
   ```

3. Create the LVM volume group `manila-volumes`:

   ```
   # vgcreate manila-volumes /dev/sdc
   Volume group "manila-volumes" successfully created
   ```

   The Shared File Systems service creates logical volumes in this volume group.

4. Only instances can access Shared File Systems service volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If projects use LVM on their volumes, the scanning tool detects these volumes and attempts to cache them which can cause a variety of problems with both the underlying operating system and project volumes. You must reconfigure LVM to scan only the devices that contain the `cinder-volume` and `manila-volumes` volume groups. Edit the `/etc/lvm/lvm.conf` file and complete the following actions:

   - In the `devices` section, add a filter that accepts the `/dev/sdb` and `/dev/sdc` devices and rejects all other devices:

   ```
   devices {
   ...
   filter = [ "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Warning: If your storage nodes use LVM on the operating system disk, you must also add the associated device to the filter. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Similarly, if your compute nodes use LVM on the operating system disk, you must also modify the filter in the `/etc/lvm/lvm.conf` file on those nodes to include only the operating system disk. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "r/.*/"]
   ```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable the LVM driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = lvm
enabled_share_protocols = NFS
```

   Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[lvm]` section, configure the LVM driver:

```
[lvm]
share_backend_name = LVM
share_driver = manila.share.drivers.lvm.LVMShareDriver
driver_handles_share_servers = False
lvm_share_volume_group = manila-volumes
lvm_share_export_ips = MANAGEMENT_INTERFACE_IP_ADDRESS
```

     Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your storage node. The value of this option can be a comma separated string of one or more IP addresses. In the example architecture shown below, the address would be 10.0.0.41:

Shared File Systems Option 2: Driver support for share servers management

For simplicity, this configuration references the same storage node as the one used for the Block Storage service.

Note: This guide describes how to configure the Shared File Systems service to use the `generic` driver with the driver handles share server mode (DHSS) enabled. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares. The information used for creating share servers is configured as share networks. Generic driver with DHSS enabled also requires the tenants private network (where the compute instances are running) to be attached to a public router.

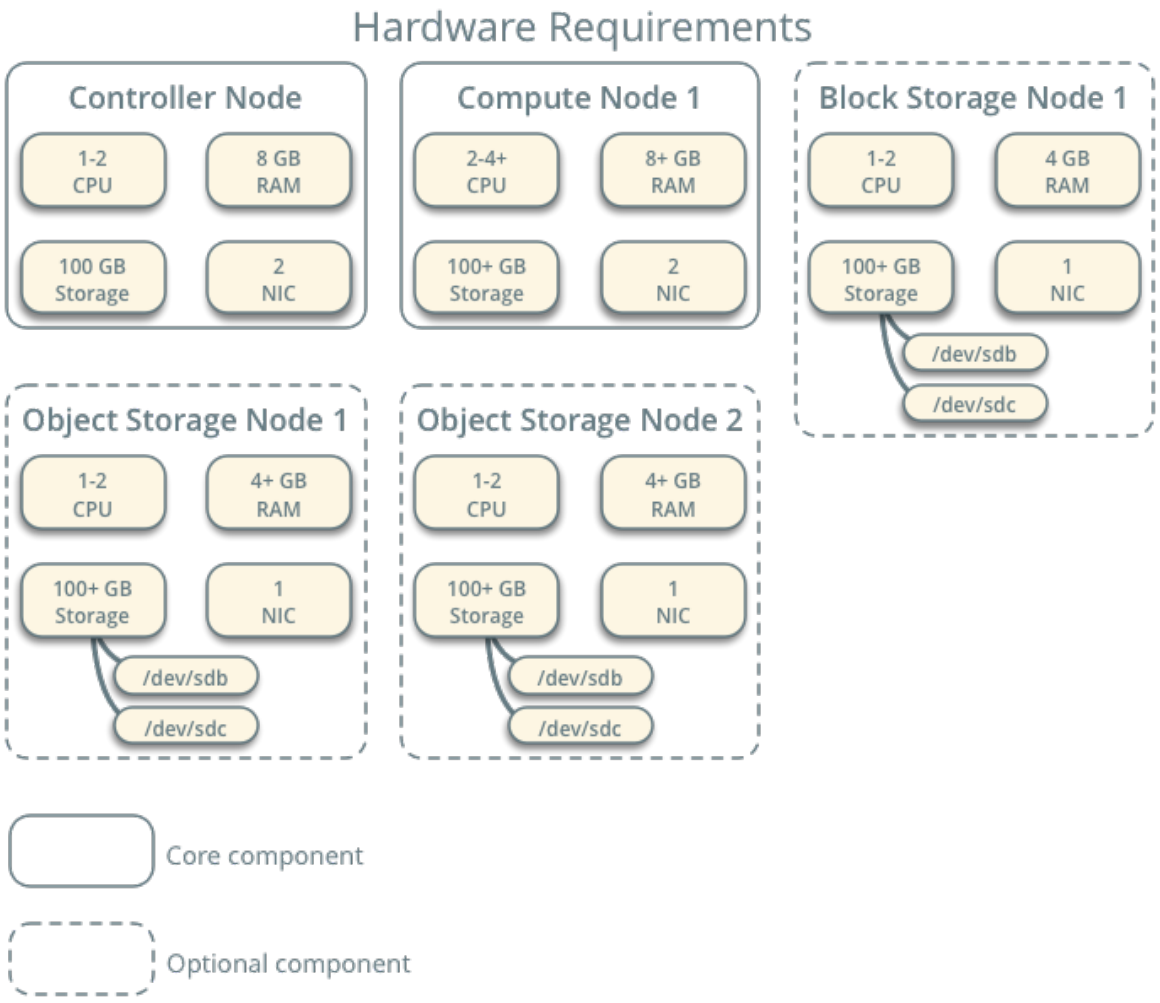


Fig. 2: **Hardware requirements**.

Prerequisites

Before you proceed, verify operation of the Compute, Networking, and Block Storage services. This options requires implementation of Networking option 2 and requires installation of some Networking service components on the storage node.

- Install the Networking service components:

```
# zypper install --no-recommends openstack-neutron-linuxbridge-agent
```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, enable the generic driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = generic
enabled_share_protocols = NFS
```

Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[neutron]`, `[nova]`, `[cinder]` and `[glance]` sections, enable authentication for those services:

```
[neutron]
...
url = http://controller:9696
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
```

(continues on next page)

(continued from previous page)

```
project_name = service
username = nova
password = NOVA_PASS

[cinder]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = cinder
password = CINDER_PASS

[glance]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = glance
password = GLANCE_PASS
```

- In the `[generic]` section, configure the generic driver:

```
[generic]
share_backend_name = GENERIC
share_driver = manila.share.drivers.generic.GenericShareDriver
driver_handles_share_servers = True
service_instance_flavor_id = 100
service_image_name = manila-service-image
service_instance_user = manila
service_instance_password = manila
interface_driver = manila.network.linux.interface.
↪BridgeInterfaceDriver
```

Note: You can also use SSH keys instead of password authentication for service instance credentials.

Important: The `service_image_name`, `service_instance_flavor_id`, `service_instance_user` and `service_instance_password` are with reference to

the service image that is used by the driver to create share servers. A sample service image for use with the `generic` driver is available in the `manila-image-elements` project. Its creation is explained in the post installation steps (See: *Creating and using shared file systems*).

Finalize installation

1. Prepare manila-share as start/stop service. Start the Shared File Systems service including its dependencies and configure them to start when the system boots:

```
# systemctl enable openstack-manila-share.service tgtd.service
# systemctl start openstack-manila-share.service tgtd.service
```

Install and configure a share node running Red Hat Enterprise Linux and CentOS

This section describes how to install and configure a share node for the Shared File Systems service. For simplicity, this configuration references one storage node with the generic driver managing the share servers. The generic backend manages share servers using compute, networking and block services for provisioning shares.

Note that installation and configuration vary by distribution. This section describes the instructions for a share node running Red Hat Enterprise Linux or CentOS.

Install and configure components

1. Install the packages:

```
# yum install openstack-manila-share python3-PyMySQL
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

4. Complete the rest of the configuration in `manila.conf`.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
```

Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference to the driver mode used. This is explained in further steps.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your share node, typically 10.0.0.41 for the first node in the example architecture shown below:

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

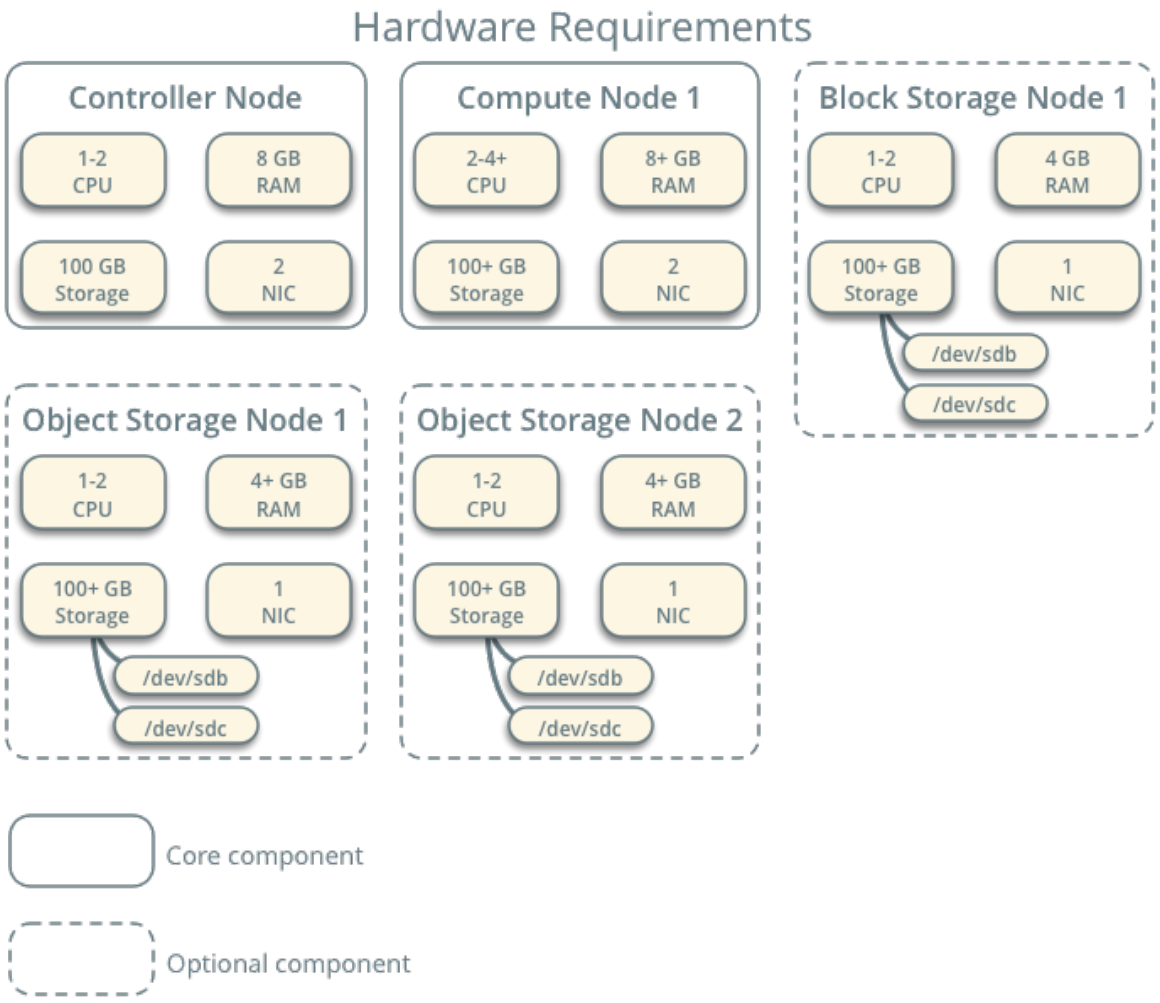


Fig. 3: **Hardware requirements**

Two driver modes

The share node can support two modes, with and without the handling of share servers. The mode depends on driver support.

Option 1

Deploying the service without driver support for share server management. In this mode, the service does not do anything related to networking. The operator must ensure network connectivity between instances and the NAS protocol based server.

This tutorial demonstrates setting up the LVM driver which creates LVM volumes on the share node and exports them with the help of an NFS server that is installed locally on the share node. It therefore requires LVM and NFS packages as well as an additional disk for the `manila-share` LVM volume group.

This driver mode may be referred to as `driver_handles_share_servers = False` mode, or simply `DHSS=False` mode.

Option 2

Deploying the service with driver support for share server management. In this mode, the service runs with a back end driver that creates and manages share servers. This tutorial demonstrates setting up the `Generic` driver. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares.

The information used for creating share servers is configured with the help of share networks.

This driver mode may be referred to as `driver_handles_share_servers = True` mode, or simply `DHSS=True` mode.

> **Warning:** When running the generic driver in `DHSS=True` driver mode, the share service should be run on the same node as the networking service. However, such a service may not be able to run the LVM driver that runs in `DHSS=False` driver mode effectively, due to a bug in some distributions of Linux. For more information, see LVM Driver section in the Configuration Reference Guide.

Choose one of the following options to configure the share driver:

Shared File Systems Option 1: No driver support for share servers management

For simplicity, this configuration references the same storage node configuration for the Block Storage service. However, the LVM driver requires a separate empty local block storage device to avoid conflict with the Block Storage service. The instructions use `/dev/sdc`, but you can substitute a different value for your particular node.

Prerequisites

Note: Perform these steps on the storage node.

1. Install the supporting utility packages:

   - Install LVM and NFS server packages:

   ```
   # yum install lvm2 nfs-utils nfs4-acl-tools portmap targetcli
   ```

   - Start the LVM metadata service and configure it to start when the system boots:

   ```
   # systemctl enable lvm2-lvmetad.service target.service
   # systemctl start lvm2-lvmetad.service target.service
   ```

2. Create the LVM physical volume `/dev/sdc`:

   ```
   # pvcreate /dev/sdc
   Physical volume "/dev/sdc" successfully created
   ```

3. Create the LVM volume group `manila-volumes`:

   ```
   # vgcreate manila-volumes /dev/sdc
   Volume group "manila-volumes" successfully created
   ```

   The Shared File Systems service creates logical volumes in this volume group.

4. Only instances can access Shared File Systems service volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If projects use LVM on their volumes, the scanning tool detects these volumes and attempts to cache them which can cause a variety of problems with both the underlying operating system and project volumes. You must reconfigure LVM to scan only the devices that contain the `cinder-volume` and `manila-volumes` volume groups. Edit the `/etc/lvm/lvm.conf` file and complete the following actions:

   - In the `devices` section, add a filter that accepts the `/dev/sdb` and `/dev/sdc` devices and rejects all other devices:

   ```
   devices {
   ...
   filter = [ "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Warning: If your storage nodes use LVM on the operating system disk, you must also add the associated device to the filter. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Similarly, if your compute nodes use LVM on the operating system disk, you must also modify the filter in the `/etc/lvm/lvm.conf` file on those nodes to include only the operating system disk. For example, if the `/dev/sda` device contains the operating system:

```
filter = [ "a/sda/", "r/.*/"]
```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable the LVM driver and the NFS protocol:

   ```
   [DEFAULT]
   ...
   enabled_share_backends = lvm
   enabled_share_protocols = NFS
   ```

   Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[lvm]` section, configure the LVM driver:

   ```
   [lvm]
   share_backend_name = LVM
   share_driver = manila.share.drivers.lvm.LVMShareDriver
   driver_handles_share_servers = False
   lvm_share_volume_group = manila-volumes
   lvm_share_export_ips = MANAGEMENT_INTERFACE_IP_ADDRESS
   ```

   Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your storage node. The value of this option can be a comma separated string of one or more IP addresses. In the example architecture shown below, the address would be 10.0.0.41:

Shared File Systems Option 2: Driver support for share servers management

For simplicity, this configuration references the same storage node as the one used for the Block Storage service.

Note: This guide describes how to configure the Shared File Systems service to use the `generic` driver with the driver handles share server mode (DHSS) enabled. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares. The information used for creating share servers is configured as share networks. Generic driver with DHSS enabled also requires the tenants private network (where the compute instances are running) to be attached to a public router.

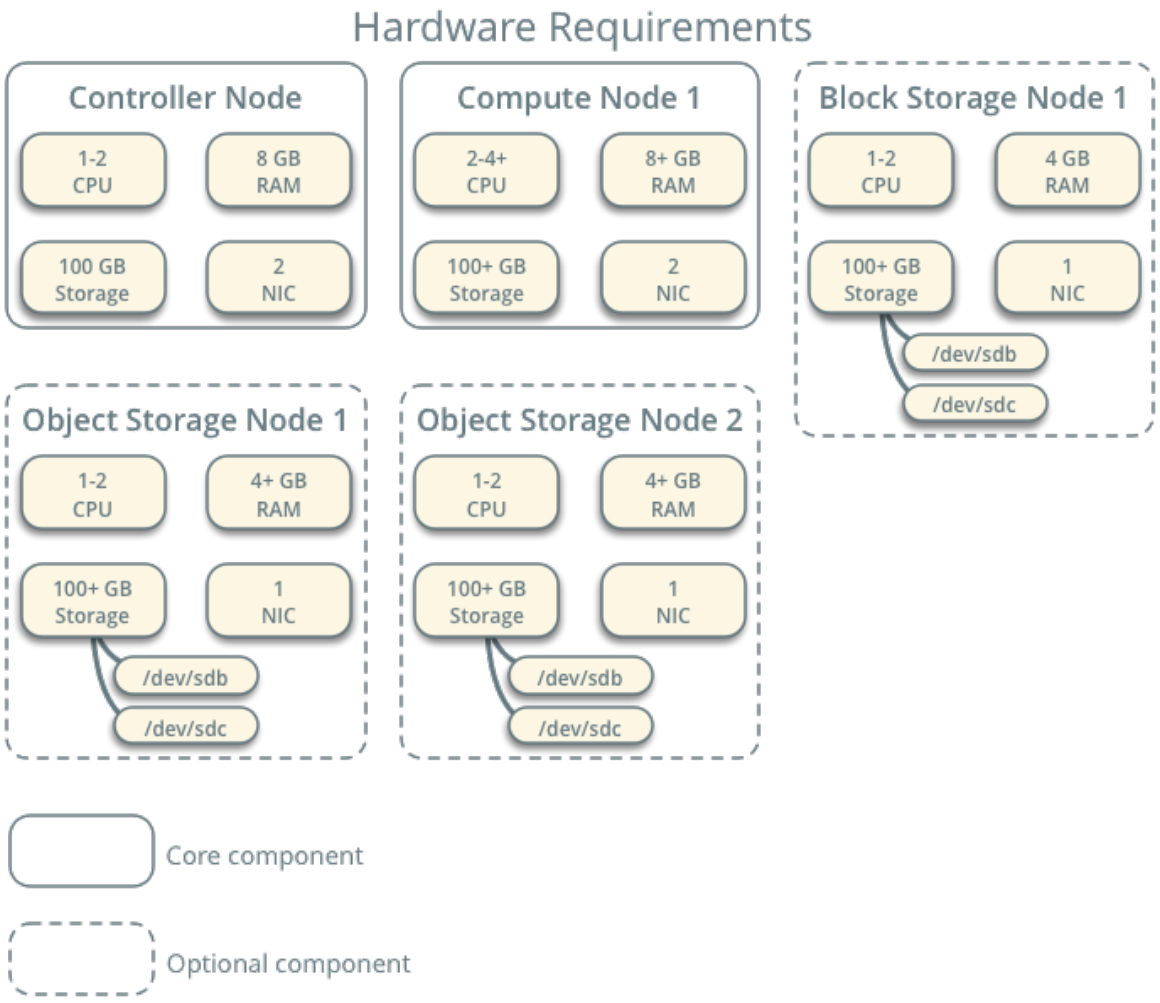


Fig. 4: **Hardware requirements**.

Prerequisites

Before you proceed, verify operation of the Compute, Networking, and Block Storage services. This options requires implementation of Networking option 2 and requires installation of some Networking service components on the storage node.

- Install the Networking service components:

```
# yum install openstack-neutron openstack-neutron-linuxbridge ebtables
```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, enable the generic driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = generic
enabled_share_protocols = NFS
```

Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[neutron]`, `[nova]`, `[cinder]` and `[glance]` sections, enable authentication for those services:

```
[neutron]
...
url = http://controller:9696
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
```

(continues on next page)

(continued from previous page)

```
project_name = service
username = nova
password = NOVA_PASS

[cinder]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = cinder
password = CINDER_PASS

[glance]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = glance
password = GLANCE_PASS
```

- In the `[generic]` section, configure the generic driver:

```
[generic]
share_backend_name = GENERIC
share_driver = manila.share.drivers.generic.GenericShareDriver
driver_handles_share_servers = True
service_instance_flavor_id = 100
service_image_name = manila-service-image
service_instance_user = manila
service_instance_password = manila
interface_driver = manila.network.linux.interface.
↪BridgeInterfaceDriver
```

Note: You can also use SSH keys instead of password authentication for service instance credentials.

Important: The `service_image_name`, `service_instance_flavor_id`, `service_instance_user` and `service_instance_password` are with reference to

the service image that is used by the driver to create share servers. A sample service image for use with the `generic` driver is available in the `manila-image-elements` project. Its creation is explained in the post installation steps (See: *Creating and using shared file systems*).

Finalize installation

1. Prepare manila-share as start/stop service. Start the Shared File Systems service including its dependencies and configure them to start when the system boots:

```
# systemctl enable openstack-manila-share.service
# systemctl start openstack-manila-share.service
```

Install and configure a share node running Ubuntu

This section describes how to install and configure a share node for the Shared File Systems service. For simplicity, this configuration references one storage node with the generic driver managing the share servers. The generic backend manages share servers using compute, networking and block services for provisioning shares.

Note that installation and configuration vary by distribution. This section describes the instructions for a share node running Ubuntu.

Install and configure components

1. Install the packages:

```
# apt-get install manila-share python3-pymysql
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

4. Complete the rest of the configuration in `manila.conf`.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in `RabbitMQ`.

   - In the `[DEFAULT]` section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
```

Important: The `default_share_type` option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as `driver_handles_share_servers`) set appropriately with reference to the driver mode used. This is explained in further steps.

- In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace `MANILA_PASS` with the password you chose for the `manila` user in the Identity service.

- In the `[DEFAULT]` section, configure the `my_ip` option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your share node, typically 10.0.0.41 for the first node in the example architecture shown below:

- In the `[oslo_concurrency]` section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

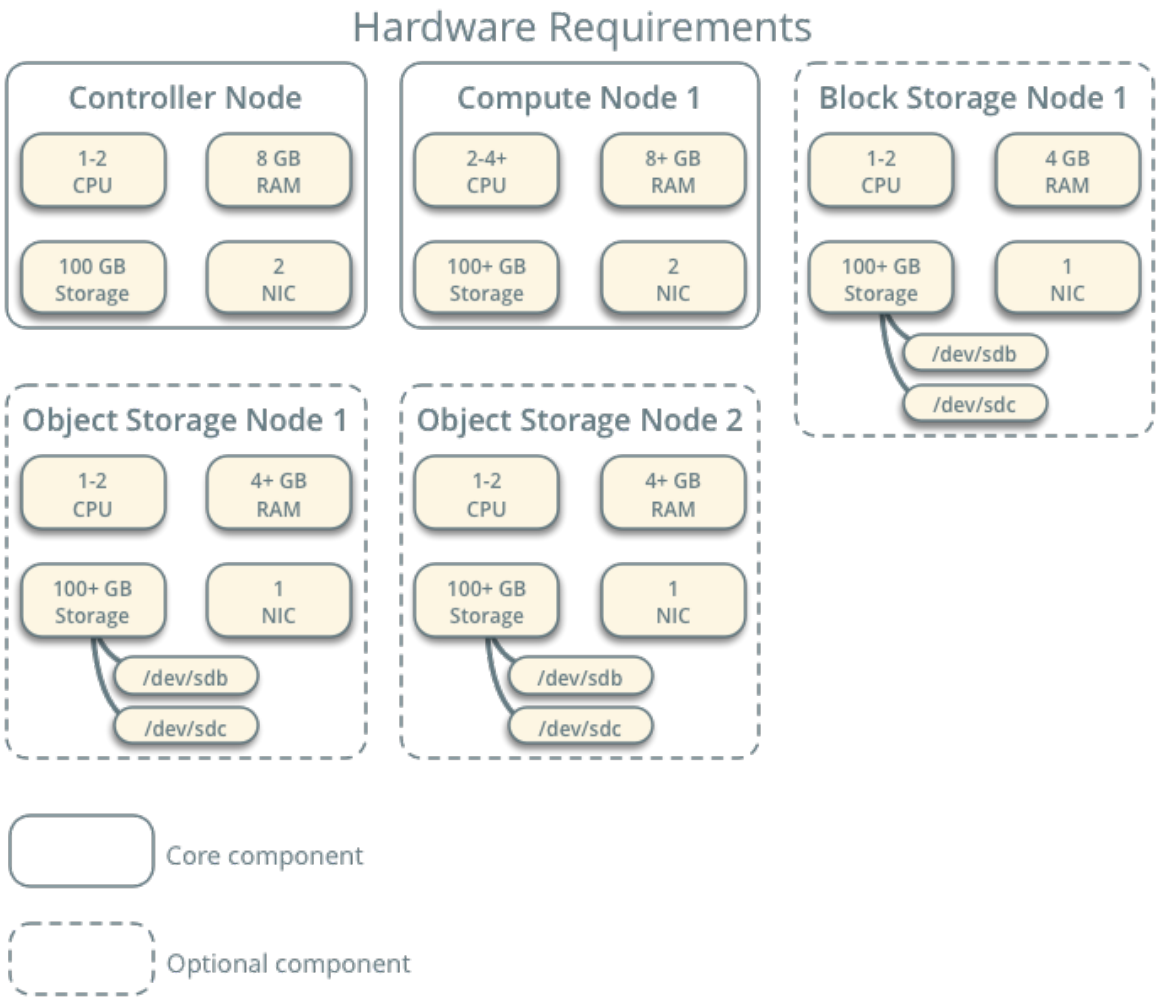


Fig. 5: **Hardware requirements**

Two driver modes

The share node can support two modes, with and without the handling of share servers. The mode depends on driver support.

Option 1

Deploying the service without driver support for share server management. In this mode, the service does not do anything related to networking. The operator must ensure network connectivity between instances and the NAS protocol based server.

This tutorial demonstrates setting up the LVM driver which creates LVM volumes on the share node and exports them with the help of an NFS server that is installed locally on the share node. It therefore requires LVM and NFS packages as well as an additional disk for the `manila-share` LVM volume group.

This driver mode may be referred to as `driver_handles_share_servers = False` mode, or simply `DHSS=False` mode.

Option 2

Deploying the service with driver support for share server management. In this mode, the service runs with a back end driver that creates and manages share servers. This tutorial demonstrates setting up the `Generic` driver. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares.

The information used for creating share servers is configured with the help of share networks.

This driver mode may be referred to as `driver_handles_share_servers = True` mode, or simply `DHSS=True` mode.

> **Warning:** When running the generic driver in `DHSS=True` driver mode, the share service should be run on the same node as the networking service. However, such a service may not be able to run the LVM driver that runs in `DHSS=False` driver mode effectively, due to a bug in some distributions of Linux. For more information, see LVM Driver section in the Configuration Reference Guide.

Choose one of the following options to configure the share driver:

Shared File Systems Option 1: No driver support for share servers management

For simplicity, this configuration references the same storage node configuration for the Block Storage service. However, the LVM driver requires a separate empty local block storage device to avoid conflict with the Block Storage service. The instructions use `/dev/sdc`, but you can substitute a different value for your particular node.

Prerequisites

Note: Perform these steps on the storage node.

1. Install the supporting utility packages:

   - Install LVM and NFS server packages:

   ```
   # apt-get install lvm2 nfs-kernel-server
   ```

2. Create the LVM physical volume `/dev/sdc`:

   ```
   # pvcreate /dev/sdc
   Physical volume "/dev/sdc" successfully created
   ```

3. Create the LVM volume group `manila-volumes`:

   ```
   # vgcreate manila-volumes /dev/sdc
   Volume group "manila-volumes" successfully created
   ```

   The Shared File Systems service creates logical volumes in this volume group.

4. Only instances can access Shared File Systems service volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If projects use LVM on their volumes, the scanning tool detects these volumes and attempts to cache them which can cause a variety of problems with both the underlying operating system and project volumes. You must reconfigure LVM to scan only the devices that contain the `cinder-volume` and `manila-volumes` volume groups. Edit the `/etc/lvm/lvm.conf` file and complete the following actions:

   - In the `devices` section, add a filter that accepts the `/dev/sdb` and `/dev/sdc` devices and rejects all other devices:

   ```
   devices {
   ...
   filter = [ "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Warning: If your storage nodes use LVM on the operating system disk, you must also add the associated device to the filter. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Similarly, if your compute nodes use LVM on the operating system disk, you must also modify the filter in the `/etc/lvm/lvm.conf` file on those nodes to include only the operating system disk. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "r/.*/"]
   ```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable the LVM driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = lvm
enabled_share_protocols = NFS
```

   Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[lvm]` section, configure the LVM driver:

```
[lvm]
share_backend_name = LVM
share_driver = manila.share.drivers.lvm.LVMShareDriver
driver_handles_share_servers = False
lvm_share_volume_group = manila-volumes
lvm_share_export_ips = MANAGEMENT_INTERFACE_IP_ADDRESS
```

     Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your storage node. The value of this option can be a comma separated string of one or more IP addresses. In the example architecture shown below, the address would be 10.0.0.41:

Shared File Systems Option 2: Driver support for share servers management

For simplicity, this configuration references the same storage node as the one used for the Block Storage service.

Note: This guide describes how to configure the Shared File Systems service to use the `generic` driver with the driver handles share server mode (DHSS) enabled. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares. The information used for creating share servers is configured as share networks. Generic driver with DHSS enabled also requires the tenants private network (where the compute instances are running) to be attached to a public router.

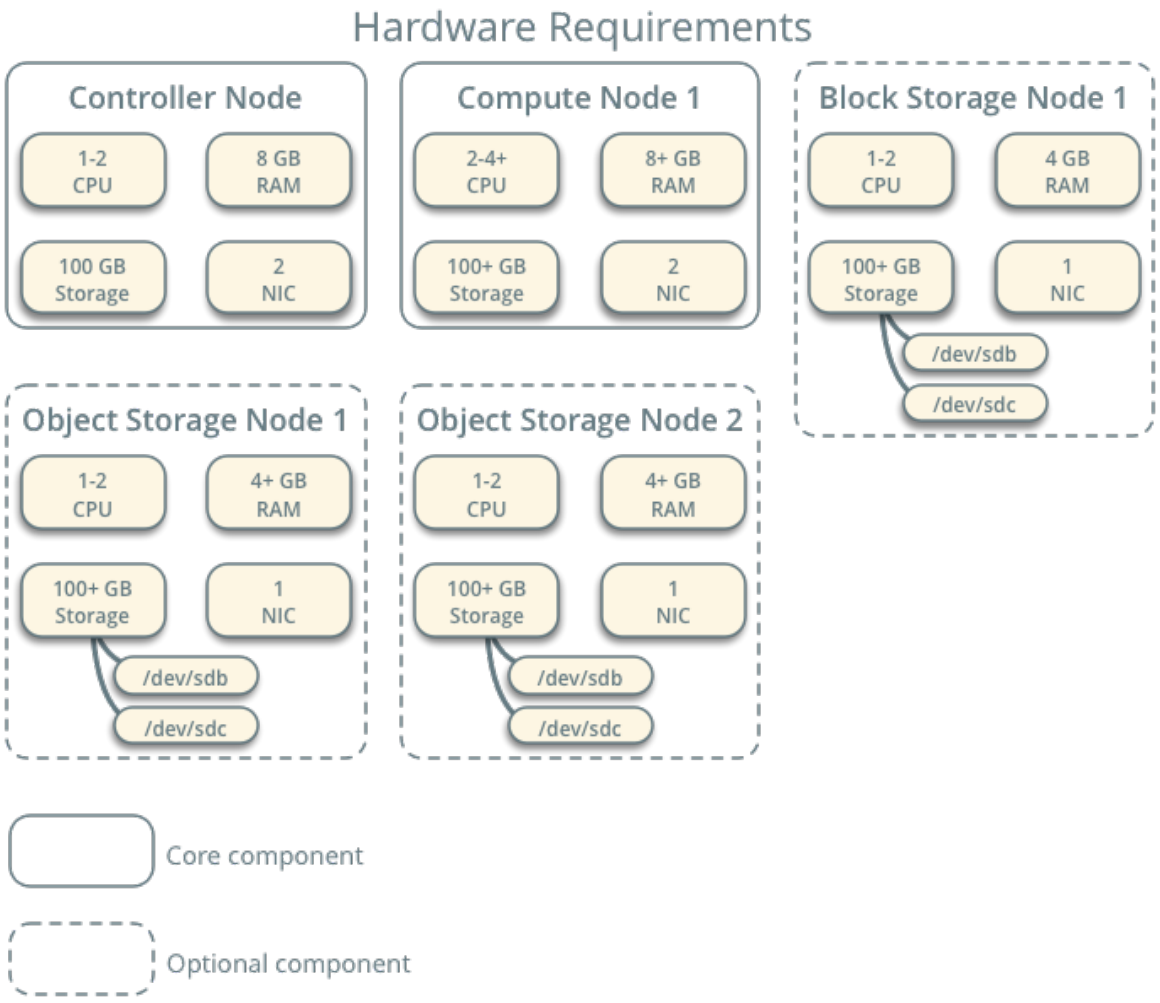


Fig. 6: **Hardware requirements**.

Prerequisites

Before you proceed, verify operation of the Compute, Networking, and Block Storage services. This options requires implementation of Networking option 2 and requires installation of some Networking service components on the storage node.

- Install the Networking service components:

```
# apt-get install neutron-plugin-linuxbridge-agent
```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, enable the generic driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = generic
enabled_share_protocols = NFS
```

Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[neutron]`, `[nova]`, `[cinder]` and `[glance]` sections, enable authentication for those services:

```
[neutron]
...
url = http://controller:9696
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
```

(continues on next page)

(continued from previous page)

```
project_name = service
username = nova
password = NOVA_PASS

[cinder]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = cinder
password = CINDER_PASS

[glance]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = glance
password = GLANCE_PASS
```

- In the `[generic]` section, configure the generic driver:

```
[generic]
share_backend_name = GENERIC
share_driver = manila.share.drivers.generic.GenericShareDriver
driver_handles_share_servers = True
service_instance_flavor_id = 100
service_image_name = manila-service-image
service_instance_user = manila
service_instance_password = manila
interface_driver = manila.network.linux.interface.
↪BridgeInterfaceDriver
```

Note: You can also use SSH keys instead of password authentication for service instance credentials.

Important: The `service_image_name`, `service_instance_flavor_id`, `service_instance_user` and `service_instance_password` are with reference to

the service image that is used by the driver to create share servers. A sample service image for use with the `generic` driver is available in the `manila-image-elements` project. Its creation is explained in the post installation steps (See: *Creating and using shared file systems*).

Finalize installation

1. Prepare manila-share as start/stop service. Start the Shared File Systems service including its dependencies:

```
# service manila-share restart
```

2. By default, the Ubuntu packages create an SQLite database. Because this configuration uses an SQL database server, remove the SQLite database file:

```
# rm -f /var/lib/manila/manila.sqlite
```

Install and configure a share node running Debian

This section describes how to install and configure a share node for the Shared File Systems service. For simplicity, this configuration references one storage node with the generic driver managing the share servers. The generic backend manages share servers using compute, networking and block services for provisioning shares.

Note that installation and configuration vary by distribution. This section describes the instructions for a share node running a Debian distribution.

Install and configure components

1. Install the packages:

```
# apt-get install manila-share python3-pymysql
```

2. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[database]` section, configure database access:

```
[database]
...
connection = mysql+pymysql://manila:MANILA_DBPASS@controller/manila
```

     Replace `MANILA_DBPASS` with the password you chose for the Shared File Systems database.

4. Complete the rest of the configuration in `manila.conf`.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

Replace RABBIT_PASS with the password you chose for the openstack account in RabbitMQ.

- In the [DEFAULT] section, set the following config values:

```
[DEFAULT]
...
default_share_type = default_share_type
rootwrap_config = /etc/manila/rootwrap.conf
```

Important: The default_share_type option specifies the default share type to be used when shares are created without specifying the share type in the request. The default share type that is specified in the configuration file has to be created with the necessary required extra-specs (such as driver_handles_share_servers) set appropriately with reference to the driver mode used. This is explained in further steps.

- In the [DEFAULT] and [keystone_authtoken] sections, configure Identity service access:

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = manila
password = MANILA_PASS
```

Replace MANILA_PASS with the password you chose for the manila user in the Identity service.

- In the [DEFAULT] section, configure the my_ip option:

```
[DEFAULT]
...
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace MANAGEMENT_INTERFACE_IP_ADDRESS with the IP address of the management network interface on your share node, typically 10.0.0.41 for the first node in the example architecture shown below:

- In the [oslo_concurrency] section, configure the lock path:

```
[oslo_concurrency]
...
lock_path = /var/lib/manila/tmp
```

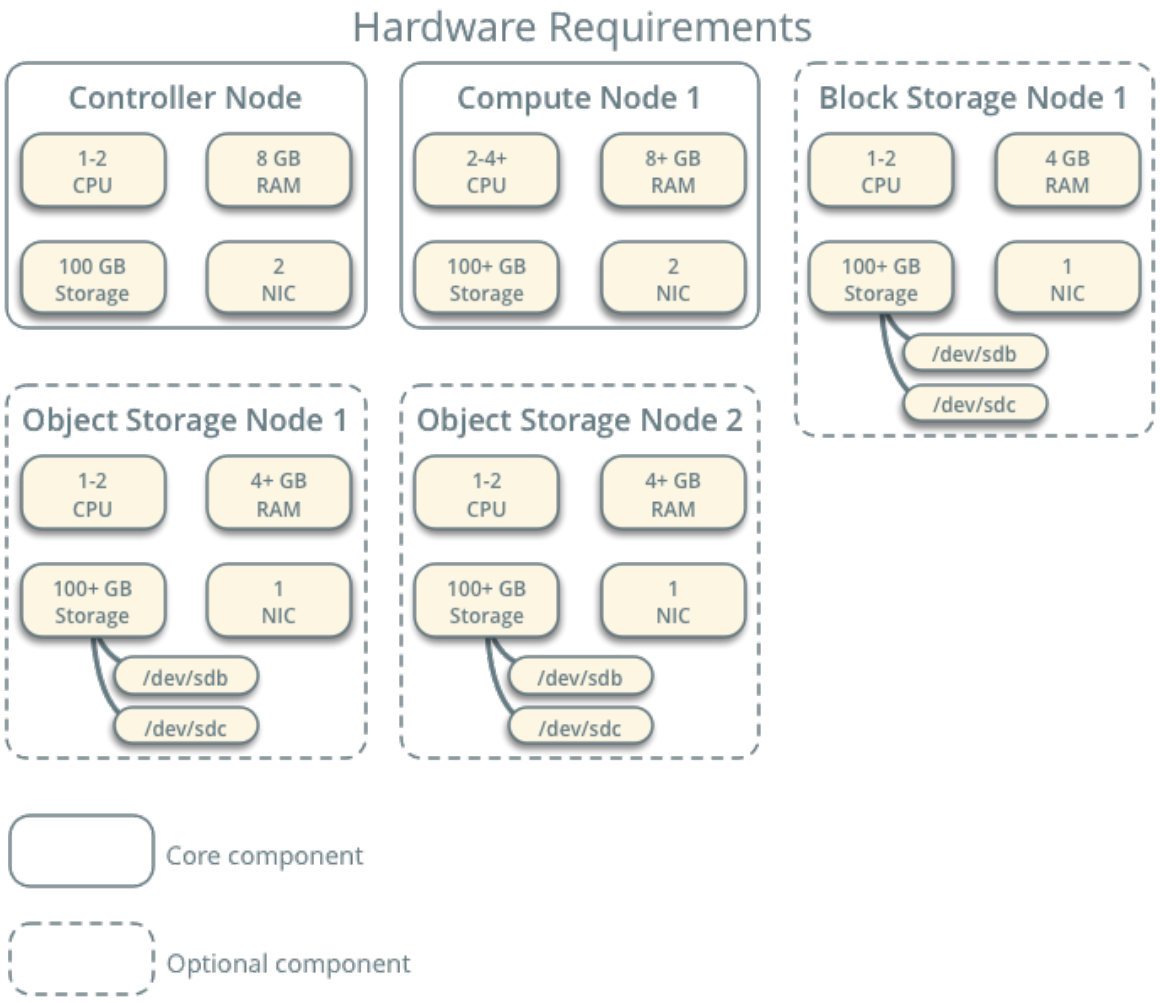


Fig. 7: **Hardware requirements**

Two driver modes

The share node can support two modes, with and without the handling of share servers. The mode depends on driver support.

Option 1

Deploying the service without driver support for share server management. In this mode, the service does not do anything related to networking. The operator must ensure network connectivity between instances and the NAS protocol based server.

This tutorial demonstrates setting up the LVM driver which creates LVM volumes on the share node and exports them with the help of an NFS server that is installed locally on the share node. It therefore requires LVM and NFS packages as well as an additional disk for the `manila-share` LVM volume group.

This driver mode may be referred to as `driver_handles_share_servers = False` mode, or simply `DHSS=False` mode.

Option 2

Deploying the service with driver support for share server management. In this mode, the service runs with a back end driver that creates and manages share servers. This tutorial demonstrates setting up the `Generic` driver. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares.

The information used for creating share servers is configured with the help of share networks.

This driver mode may be referred to as `driver_handles_share_servers = True` mode, or simply `DHSS=True` mode.

> **Warning:** When running the generic driver in `DHSS=True` driver mode, the share service should be run on the same node as the networking service. However, such a service may not be able to run the LVM driver that runs in `DHSS=False` driver mode effectively, due to a bug in some distributions of Linux. For more information, see LVM Driver section in the Configuration Reference Guide.

Choose one of the following options to configure the share driver:

Shared File Systems Option 1: No driver support for share servers management

For simplicity, this configuration references the same storage node configuration for the Block Storage service. However, the LVM driver requires a separate empty local block storage device to avoid conflict with the Block Storage service. The instructions use `/dev/sdc`, but you can substitute a different value for your particular node.

Prerequisites

Note: Perform these steps on the storage node.

1. Install the supporting utility packages:

   - Install LVM and NFS server packages:

   ```
   # apt-get install lvm2 nfs-kernel-server
   ```

2. Create the LVM physical volume `/dev/sdc`:

   ```
   # pvcreate /dev/sdc
   Physical volume "/dev/sdc" successfully created
   ```

3. Create the LVM volume group `manila-volumes`:

   ```
   # vgcreate manila-volumes /dev/sdc
   Volume group "manila-volumes" successfully created
   ```

   The Shared File Systems service creates logical volumes in this volume group.

4. Only instances can access Shared File Systems service volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If projects use LVM on their volumes, the scanning tool detects these volumes and attempts to cache them which can cause a variety of problems with both the underlying operating system and project volumes. You must reconfigure LVM to scan only the devices that contain the `cinder-volume` and `manila-volumes` volume groups. Edit the `/etc/lvm/lvm.conf` file and complete the following actions:

   - In the `devices` section, add a filter that accepts the `/dev/sdb` and `/dev/sdc` devices and rejects all other devices:

   ```
   devices {
   ...
   filter = [ "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Warning: If your storage nodes use LVM on the operating system disk, you must also add the associated device to the filter. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "a/sdb/", "a/sdc", "r/.*/"]
   ```

   Similarly, if your compute nodes use LVM on the operating system disk, you must also modify the filter in the `/etc/lvm/lvm.conf` file on those nodes to include only the operating system disk. For example, if the `/dev/sda` device contains the operating system:

   ```
   filter = [ "a/sda/", "r/.*/"]
   ```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:

   - In the `[DEFAULT]` section, enable the LVM driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = lvm
enabled_share_protocols = NFS
```

   Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[lvm]` section, configure the LVM driver:

```
[lvm]
share_backend_name = LVM
share_driver = manila.share.drivers.lvm.LVMShareDriver
driver_handles_share_servers = False
lvm_share_volume_group = manila-volumes
lvm_share_export_ips = MANAGEMENT_INTERFACE_IP_ADDRESS
```

     Replace `MANAGEMENT_INTERFACE_IP_ADDRESS` with the IP address of the management network interface on your storage node. The value of this option can be a comma separated string of one or more IP addresses. In the example architecture shown below, the address would be 10.0.0.41:

Shared File Systems Option 2: Driver support for share servers management

For simplicity, this configuration references the same storage node as the one used for the Block Storage service.

Note: This guide describes how to configure the Shared File Systems service to use the `generic` driver with the driver handles share server mode (DHSS) enabled. This driver requires Compute service (nova), Image service (glance) and Networking service (neutron) for creating and managing share servers; and Block storage service (cinder) for creating shares. The information used for creating share servers is configured as share networks. Generic driver with DHSS enabled also requires the tenants private network (where the compute instances are running) to be attached to a public router.

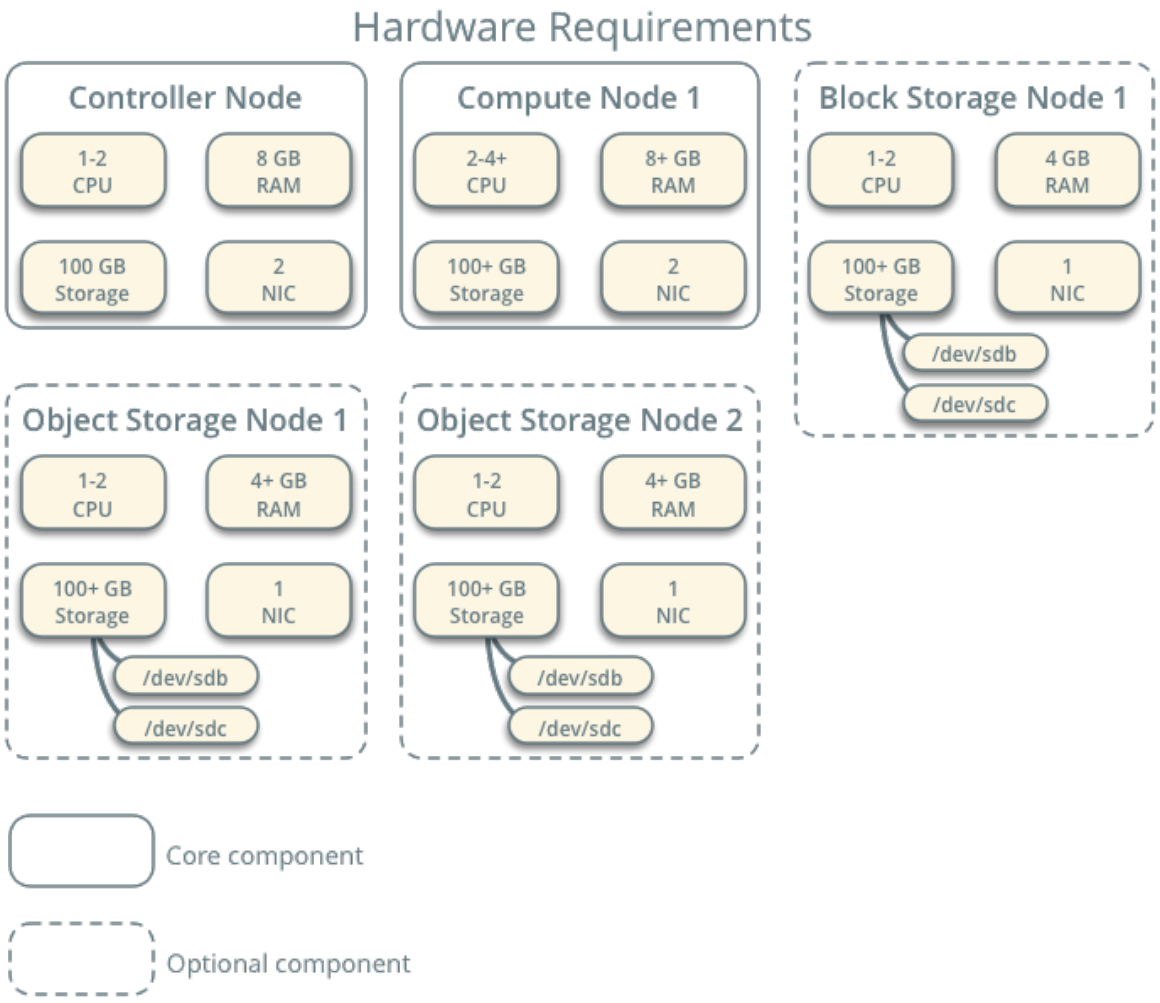


Fig. 8: **Hardware requirements**.

Prerequisites

Before you proceed, verify operation of the Compute, Networking, and Block Storage services. This options requires implementation of Networking option 2 and requires installation of some Networking service components on the storage node.

- Install the Networking service components:

```
# apt-get install neutron-plugin-linuxbridge-agent
```

Configure components

1. Edit the `/etc/manila/manila.conf` file and complete the following actions:
   - In the `[DEFAULT]` section, enable the generic driver and the NFS protocol:

```
[DEFAULT]
...
enabled_share_backends = generic
enabled_share_protocols = NFS
```

Note: Back end names are arbitrary. As an example, this guide uses the name of the driver.

   - In the `[neutron]`, `[nova]`, `[cinder]` and `[glance]` sections, enable authentication for those services:

```
[neutron]
...
url = http://controller:9696
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
```

(continues on next page)

(continued from previous page)

```
project_name = service
username = nova
password = NOVA_PASS

[cinder]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = cinder
password = CINDER_PASS

[glance]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = glance
password = GLANCE_PASS
```

- In the `[generic]` section, configure the generic driver:

```
[generic]
share_backend_name = GENERIC
share_driver = manila.share.drivers.generic.GenericShareDriver
driver_handles_share_servers = True
service_instance_flavor_id = 100
service_image_name = manila-service-image
service_instance_user = manila
service_instance_password = manila
interface_driver = manila.network.linux.interface.
↪BridgeInterfaceDriver
```

Note: You can also use SSH keys instead of password authentication for service instance credentials.

Important: The `service_image_name`, `service_instance_flavor_id`, `service_instance_user` and `service_instance_password` are with reference to

the service image that is used by the driver to create share servers. A sample service image for use with the `generic` driver is available in the `manila-image-elements` project. Its creation is explained in the post installation steps (See: *Creating and using shared file systems*).

Finalize installation

1. Prepare manila-share as start/stop service. Start the Shared File Systems service including its dependencies:

```
# service manila-share restart
```

Verify operation

Verify operation of the Shared File Systems service.

Note: Perform these commands on the controller node.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc.sh
```

2. List service components to verify successful launch of each process:

```
$ manila service-list
+------------------+----------------+------+---------+-------+------------
↪----------------+-----------------+
|      Binary      |    Host        | Zone |  Status | State |        ␣
↪Updated_at         | Disabled Reason |
+------------------+----------------+------+---------+-------+------------
↪----------------+-----------------+
| manila-scheduler | controller     | nova | enabled |   up  | 2014-10-
↪18T01:30:54.000000 |       None      |
| manila-share     | share1@generic | nova | enabled |   up  | 2014-10-
↪18T01:30:57.000000 |       None      |
+------------------+----------------+------+---------+-------+------------
↪----------------+-----------------+
```

Creating and using shared file systems

Depending on the option chosen while installing the share node (Option with share server management and one without); the steps to create and use your shared file systems will vary. When the Shared File Systems service handles the creation and management of share servers, you would need to specify the `share network` with the request to create a share. Either modes will vary in their respective share type definition. When using the driver mode with automatic handling of share servers, a service image is needed as specified in your configuration. The instructions below enumerate the steps for both driver modes. Follow what is appropriate for your installation.

Creating shares with Shared File Systems Option 1 (DHSS = False)

Create a share type

Disable DHSS (`driver_handles_share_servers`) before creating a share using the LVM driver.

1. Source the admin credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc
```

2. Create a default share type with DHSS disabled. A default share type will allow you to create shares with this driver, without having to specify the share type explicitly during share creation.

```
$ manila type-create default_share_type False
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| required_extra_specs | driver_handles_share_servers : False |
| Name                 | default_share_type                   |
| Visibility           | public                               |
| is_default           | -                                    |
| ID                   | 3df065c8-6ca4-4b80-a5cb-e633c0439097 |
| optional_extra_specs | snapshot_support : True              |
+----------------------+--------------------------------------+
```

Set this default share type in `manila.conf` under the `[DEFAULT]` section and restart the `manila-api` service before proceeding. Unless you do so, the default share type will not be effective.

Note: Creating and configuring a default share type is optional. If you wish to use the shared file system service with a variety of share types, where each share creation request could specify a type, please refer to the Share types usage documentation here.

Create a share

1. Source the demo credentials to perform the following steps as a non-administrative project:

```
$ . demo-openrc
```

2. Create an NFS share. Since a default share type has been created and configured, it need not be specified in the request.

```
$ manila create NFS 1 --name share1
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | default_share_type                   |
| description                 | None                                 |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| share_group_id              | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 55c401b3-3112-4294-aa9f-3cc355a4e361 |
| size                        | 1                                    |
| name                        | share1                               |
| share_type                  | 3df065c8-6ca4-4b80-a5cb-e633c0439097 |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-30T19:10:33.000000           |
| share_proto                 | NFS                                  |
| project_id                  | 3a46a53a377642a284e1d12efabb3b5a     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

3. After some time, the share status should change from `creating` to `available`:

```
$ manila list
+--------------------------------------+--------+------+-------------+----
↪-------+-----------+-------------------+------------------------------+-
↪------------------+
| ID                                   | Name   | Size | Share Proto |␣
↪Status    | Is Public | Share Type Name    | Host                         ␣
↪ | Availability Zone |
+--------------------------------------+--------+------+-------------+----
↪-------+-----------+-------------------+------------------------------+-
↪------------------+
| 55c401b3-3112-4294-aa9f-3cc355a4e361 | share1 | 1    | NFS         |␣
↪available | False     | default_share_type | storage@lvm#lvm-single-
↪pool | nova              |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+--------+------+-----------+----
↪-------+-----------+-------------------+---------------------------+-
↪------------------+
```

4. Determine export IP address of the share:

```
$ manila show share1
+----------------------------+-----------------------------------------
↪-----------------------------------------+
| Property                   | Value                                   ␣
↪                                         |
+----------------------------+-----------------------------------------
↪-----------------------------------------+
| status                     | available                               ␣
↪                                         |
| share_type_name            | default_share_type                      ␣
↪                                         |
| description                | None                                    ␣
↪                                         |
| availability_zone          | nova                                    ␣
↪                                         |
| share_network_id           | None                                    ␣
↪                                         |
| share_group_id             | None                                    ␣
↪                                         |
| export_locations           |                                         ␣
↪                                         |
|                            | path = 10.0.0.41:/var/lib/manila/mnt/
↪share-8e13a98f-c310-41df-ac90-fc8bce4910b8    |
|                            | id = 3c8d0ada-cadf-48dd-85b8-d4e8c3b1e204␣
↪                                         |
|                            | preferred = False                       ␣
↪                                         |
| host                       | storage@lvm#lvm-single-pool             ␣
↪                                         |
| access_rules_status        | active                                  ␣
↪                                         |
| snapshot_id                | None                                    ␣
↪                                         |
| is_public                  | False                                   ␣
↪                                         |
| task_state                 | None                                    ␣
↪                                         |
| snapshot_support           | True                                    ␣
↪                                         |
| id                         | 55c401b3-3112-4294-aa9f-3cc355a4e361    ␣
↪                                         |
| size                       | 1                                       ␣
↪                                         |
```

(continues on next page)

(continued from previous page)

```
| name                          | share1                                                
```

Creating shares with Shared File Systems Option 2 (DHSS = True)

Before being able to create a share, manila with the generic driver and the DHSS (`driver_handles_share_servers`) mode enabled requires the definition of at least an image, a network and a share-network for being used to create a share server. For that *back end* configuration, the share server is an instance where NFS shares are served.

Note: This configuration automatically creates a cinder volume for every share. The cinder volumes are attached to share servers according to the definition of a share network.

---

1. Source the admin credentials to gain access to admin-only CLI commands:

```
$ . admin-openrc.sh
```

2. Create a default share type with DHSS enabled. A default share type will allow you to create shares with this driver, without having to specify the share type explicitly during share creation.

```
$ manila type-create default_share_type True
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| required_extra_specs | driver_handles_share_servers : True  |
| Name                 | default_share_type                   |
| Visibility           | public                               |
| is_default           | -                                    |
| ID                   | 8a35da28-0f74-490d-afff-23664ecd4f01 |
| optional_extra_specs | snapshot_support : True              |
+----------------------+--------------------------------------+
```

   Set this default share type in `manila.conf` under the `[DEFAULT]` section and restart the `manila-api` service before proceeding. Unless you do so, the default share type will not be effective.

   Note: Creating and configuring a default share type is optional. If you wish to use the shared file system service with a variety of share types, where each share creation request could specify a type, please refer to the Share types usage documentation here.

   ---

3. Create a manila share server image in the Image service. You may skip this step and use any existing image. However, for mounting a share, the service image must contain the NFS packages as appropriate for the operating system. Whatever image you choose to be the service image, be sure to set the configuration values `service_image_name`, `service_instance_flavor_id`, `service_instance_user` and `service_instance_password` in `manila.conf`.

   Note: Any changes made to `manila.conf` while the `manila-share` service is running will require a restart of the service to be effective.

   Note: As an alternative to specifying a plain-text `service_instance_password` in

your configuration, a key-pair may be specified with options `path_to_public_key` and `path_to_private_key` to configure and allow password-less SSH access between the *share node* and the share server/s created.

```
$ curl -L \
https://tarballs.opendev.org/openstack/manila-image-elements/images/
↪manila-service-image-master.qcow2 | \
glance image-create \
--name "manila-service-image" \
--disk-format qcow2 \
--container-format bare \
--visibility public --progress
%   Total    %   Received %   Xferd  Average  Speed   Time     Time     ␣
↪Time      Current
                                    Dload    Upload  Total    Spent    ␣
↪Left      Speed
100 3008k    100 3008k    0   0      1042k    0       0:00:02  0:00:02  --
↪:--:--  1041k
+------------------+------------------------------------------------------
↪----------------------------+
| Property         | Value                                                ␣
↪                            |
+------------------+------------------------------------------------------
↪----------------------------+
| checksum         | 48a08e746cf0986e2bc32040a9183445                     ␣
↪                            |
| container_format | bare                                                 ␣
↪                            |
| created_at       | 2016-01-26T19:52:24Z                                 ␣
↪                            |
| direct_url       | rbd://3c3a4cbc-7331-4fc1-8cbb-79213b9cebff/images/
↪ff97deff-b184-47f8-827c-       |
|                  | 16c349c82720/snap                                    ␣
↪                            |
| disk_format      | qcow2                                                ␣
↪                            |
| id               | 1fc7f29e-8fe6-44ef-9c3c-15217e83997c                 ␣
↪                            |
| locations        | [{"url": "rbd://3c3a4cbc-7331-4fc1-8cbb-79213b9cebff/
↪images/ff97deff-b184-47f8   |
|                  | -827c-16c349c82720/snap", "metadata": {}}]           ␣
↪                            |
| min_disk         | 0                                                    ␣
↪                            |
| min_ram          | 0                                                    ␣
↪                            |
| name             | manila-service-image                                 ␣
↪                            |
| owner            | e2c965830ecc4162a002bf16ddc91ab7                     ␣
↪                            |
```

(continues on next page)

(continued from previous page)

```
| protected        | False                                                    ␣
↪                              |
| size             | 306577408                                                ␣
↪                              |
| status           | active                                                   ␣
↪                              |
| tags             | []                                                       ␣
↪                              |
| updated_at       | 2016-01-26T19:52:28Z                                     ␣
↪                              |
| virtual_size     | None                                                     ␣
↪                              |
| visibility       | public                                                   ␣
↪                              |
+------------------+-----------------------------------------------------------
↪----------------------------+
```

4. List available networks in order to get id and subnets of the private network:

```
$ neutron net-list
+--------------------------------------+---------+------------------------
↪----------------------------+
| id                                   | name    | subnets                ␣
↪                              |
+--------------------------------------+---------+------------------------
↪----------------------------+
| 0e62efcd-8cee-46c7-b163-d8df05c3c5ad | public  | 5cc70da8-4ee7-4565-
↪be53-b9c011fca011 10.3.31.0/24  |
| 7c6f9b37-76b4-463e-98d8-27e5686ed083 | private | 3482f524-8bff-4871-
↪80d4-5774c2730728 172.16.1.0/24 |
+--------------------------------------+---------+------------------------
↪----------------------------+
```

5. Source the demo credentials to perform the following steps as a non-administrative project:

```
$ . demo-openrc.sh
```

```
$ manila share-network-create --name demo-share-network1 \
--neutron-net-id PRIVATE_NETWORK_ID \
--neutron-subnet-id PRIVATE_NETWORK_SUBNET_ID
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| name              | demo-share-network1                  |
| segmentation_id   | None                                 |
| created_at        | 2016-01-26T20:03:41.877838           |
| neutron_subnet_id | 3482f524-8bff-4871-80d4-5774c2730728 |
| updated_at        | None                                 |
| network_type      | None                                 |
| neutron_net_id    | 7c6f9b37-76b4-463e-98d8-27e5686ed083 |
```

(continues on next page)

(continued from previous page)

```
| ip_version         | None                                 |
| cidr               | None                                 |
| project_id         | e2c965830ecc4162a002bf16ddc91ab7     |
| id                 | 58b2f0e6-5509-4830-af9c-97f525a31b14 |
| description        | None                                 |
+--------------------+--------------------------------------+
```

Create a share

1. Create an NFS share using the share network. Since a default share type has been created and configured, it need not be specified in the request.

```
$ manila create NFS 1 --name demo-share1 --share-network demo-share-
↪network1
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | None                                 |
| share_type_name             | default_share_type                   |
| description                 | None                                 |
| availability_zone           | None                                 |
| share_network_id            | 58b2f0e6-5509-4830-af9c-97f525a31b14 |
| share_group_id              | None                                 |
| host                        | None                                 |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 016ca18f-bdd5-48e1-88c0-782e4c1aa28c |
| size                        | 1                                    |
| name                        | demo-share1                          |
| share_type                  | 8a35da28-0f74-490d-afff-23664ecd4f01 |
| created_at                  | 2016-01-26T20:08:50.502877           |
| export_location             | None                                 |
| share_proto                 | NFS                                  |
| project_id                  | 48e8c35b2ac6495d86d4be61658975e7     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

2. After some time, the share status should change from `creating` to `available`:

```
$ manila list
+--------------------------------------+-------------+------+-------------
↪+-----------+-----------+-----------------------+----------------------
↪-------+-------------------+
| ID                                   | Name        | Size | Share Proto␣
↪| Status    | Is Public | Share Type Name       | Host                 ␣
↪       | Availability Zone |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+-------------+------+-------------
↪+-----------+-----------+----------------------+---------------------
↪-------+-------------------+
| 5f8a0574-a95e-40ff-b898-09fd8d6a1fac | demo-share1 | 1    | NFS         ␣
↪| available | False     |   default_share_type   | storagenode@generic
↪#GENERIC | nova              |
+--------------------------------------+-------------+------+-------------
↪+-----------+-----------+----------------------+---------------------
↪-------+-------------------+
```

3. Determine export IP address of the share:

```
$ manila show demo-share1
+------------------------------+-----------------------------------------
↪-----------------------------------------+
| Property                     | Value                                   ␣
↪                                         |
+------------------------------+-----------------------------------------
↪-----------------------------------------+
| status                       | available                               ␣
↪                                         |
| share_type_name              | default_share_type                      ␣
↪                                         |
| description                  | None                                    ␣
↪                                         |
| availability_zone            | nova                                    ␣
↪                                         |
| share_network_id             | 58b2f0e6-5509-4830-af9c-97f525a31b14    ␣
↪                                         |
| share_group_id               | None                                    ␣
↪                                         |
| export_locations             |                                         ␣
↪                                         |
|                              | path = 10.254.0.6:/shares/share-0bfd69a1-
↪27f0-4ef5-af17-7cd50bce6550               |
|                              | id = e525cbca-b3cc-4adf-a1cb-b1bf48fa2422␣
↪                                         |
|                              | preferred = False                       ␣
↪                                         |
| host                         | storagenode@generic#GENERIC             ␣
↪                                         |
| access_rules_status          | active                                  ␣
↪                                         |
| snapshot_id                  | None                                    ␣
↪                                         |
| is_public                    | False                                   ␣
↪                                         |
| task_state                   | None                                    ␣
↪                                         |
```

(continues on next page)

(continued from previous page)

```
| snapshot_support            | True                                 |
| id                          | 5f8a0574-a95e-40ff-b898-09fd8d6a1fac |
| size                        | 1                                    |
| name                        | demo-share1                          |
| share_type                  | 8a35da28-0f74-490d-afff-23664ecd4f01 |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-30T19:10:33.000000           |
| share_proto                 | NFS                                  |
| project_id                  | 48e8c35b2ac6495d86d4be61658975e7     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

Allow access to the share

1. Configure access to the new share before attempting to mount it via the network. The compute instance (whose IP address is referenced by the INSTANCE_IP below) must have network connectivity to the network specified in the share network.

```
$ manila access-allow demo-share1 ip INSTANCE_IP
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 5f8a0574-a95e-40ff-b898-09fd8d6a1fac |
| access_type  | ip                                   |
| access_to    | 10.0.0.46                            |
| access_level | rw                                   |
| state        | new                                  |
| id           | aefeab01-7197-44bf-ad0f-d6ca6f99fc96 |
+--------------+--------------------------------------+
```

Mount the share on a compute instance

1. Log into your compute instance and create a folder where the mount will be placed:

```
$ mkdir ~/test_folder
```

2. Mount the NFS share in the compute instance using the export location of the share:

```
$ mount -vt nfs 10.254.0.6:/shares/share-0bfd69a1-27f0-4ef5-af17-
↪7cd50bce6550 ~/test_folder
```

For more information about how to manage shares, see the OpenStack End User Guide

Next steps

Your OpenStack environment now includes the Shared File Systems service.

To add more services, see the additional documentation on installing OpenStack services

Continue to evaluate the Shared File Systems service by creating the service image and running the service with the correct driver mode that you chose while configuring the share node.

The OpenStack Shared File Systems service (manila) provides coordinated access to shared or distributed file systems. The method in which the share is provisioned and consumed is determined by the Shared File Systems driver, or drivers in the case of a multi-backend configuration. There are a variety of drivers that support NFS, CIFS, HDFS, GlusterFS, CEPHFS, MAPRFS and other protocols as well.

The Shared File Systems API and scheduler services typically run on the controller nodes. Depending upon the drivers used, the share service can run on controllers, compute nodes, or storage nodes.

Important: For simplicity, this guide describes configuring the Shared File Systems service to use one of either:

- the `generic` back end with the `driver_handles_share_servers` mode (DHSS) enabled that uses the *Compute service* (*nova*), *Image service* (*glance*), *Networking service* (*neutron*) and *Block storage service* (*cinder*); or,
- the `LVM` back end with `driver_handles_share_servers` mode (DHSS) disabled.

The storage protocol used and referenced in this guide is `NFS`. As stated above, the Shared File System service supports different storage protocols depending on the back end chosen.

For the `generic` back end, networking service configuration requires the capability of networks being attached to a public router in order to create share networks. If using this back end, ensure that Compute, Networking and Block storage services are properly working before you proceed. For networking service, ensure that option 2 (deploying the networking service with support for self-service networks) is properly configured.

This installation tutorial also assumes that installation and configuration of OpenStack packages, Network Time Protocol, database engine and message queue has been completed as per the instructions in the OpenStack Installation Guide.. The *Identity Service* (*keystone*) has to be pre-configured with suggested client environment scripts.

For more information on various Shared File Systems storage back ends, see the Shared File Systems Configuration Reference..

To learn more about installation dependencies noted above, see the OpenStack Installation Guide.

3.2 Administrating Manila

Contents:

3.2.1 Admin Guide

Shared File Systems service provides a set of services for management of shared file systems in a multi-project cloud environment. The service resembles OpenStack block-based storage management from the OpenStack Block Storage service project. With the Shared File Systems service, you can create a remote file system, mount the file system on your instances, and then read and write data from your instances to and from your file system.

The Shared File Systems service serves same purpose as the Amazon Elastic File System (EFS) does.

The Shared File Systems service can run in a single-node or multiple node configuration. The Shared File Systems service can be configured to provision shares from one or more back ends, so it is required to declare at least one back end. Shared File System service contains several configurable components.

It is important to understand these components:

- Share networks
- Shares
- Multi-tenancy
- Back ends

The Shared File Systems service consists of four types of services, most of which are similar to those of the Block Storage service:

- `manila-api`
- `manila-data`
- `manila-scheduler`
- `manila-share`

Installation of first three - `manila-api`, `manila-data`, and `manila-scheduler` is common for almost all deployments. But configuration of `manila-share` is backend-specific and can differ from deployment to deployment.

Key concepts

Share

In the Shared File Systems service `share` is the fundamental resource unit allocated by the Shared File System service. It represents an allocation of a persistent, readable, and writable filesystems. Compute instances access these filesystems. Depending on the deployment configuration, clients outside of OpenStack can also access the filesystem.

Note: A `share` is an abstract storage object that may or may not directly map to a share concept from the underlying storage provider. See the description of `share instance` for more details.

Share instance

This concept is tied with `share` and represents created resource on specific back end, when `share` represents abstraction between end user and back-end storages. In common cases, it is one-to-one relation. One single `share` has more than one `share instance` in two cases:

- When `share migration` is being applied
- When `share replication` is enabled

Therefore, each `share instance` stores information specific to real allocated resource on storage. And `share` represents the information that is common for `share instances`. A user with `member` role will not be able to work with it directly. Only a user with `admin` role has rights to perform actions against specific share instances.

Snapshot

A `snapshot` is a point-in-time, read-only copy of a `share`. You can create `Snapshots` from an existing, operational `share` regardless of whether a client has mounted the file system. A `snapshot` can serve as the content source for a new `share`. Specify the **Create from snapshot** option when creating a new `share` on the dashboard.

Storage Pools

With the Kilo release of OpenStack, Shared File Systems can use `storage pools`. The storage may present one or more logical storage resource pools that the Shared File Systems service will select as a storage location when provisioning `shares`.

Share Type

`Share type` is an abstract collection of criteria used to characterize `shares`. They are most commonly used to create a hierarchy of functional capabilities. This hierarchy represents tiered storage services levels. For example, an administrator might define a premium `share type` that indicates a greater level of performance than a basic `share type`. Premium represents the best performance level.

Share Access Rules

`Share access rules` define which users can access a particular `share`. For example, administrators can declare rules for NFS shares by listing the valid IP networks which will access the `share`. List the IP networks in CIDR notation.

Security Services

`Security services` allow granular client access rules for administrators. They can declare rules for authentication or authorization to access `share` content. External services including LDAP, Active Directory, and Kerberos can be declared as resources. Examine and consult these resources when making an access decision for a particular `share`. You can associate `Shares` with multiple security services, but only one service per one type.

Share Networks

A `share network` is an object that defines a relationship between a project network and subnet, as defined in an OpenStack Networking service or Compute service. The `share network` is also defined in `shares` created by the same project. A project may find it desirable to provision `shares` such that only instances connected to a particular OpenStack-defined network have access to the `share`. Also, `security services` can be attached to `share networks`, because most of auth protocols require some interaction with network services.

The Shared File Systems service has the ability to work outside of OpenStack. That is due to the `StandaloneNetworkPlugin`. The plugin is compatible with any network platform, and does not require specific network services in OpenStack like Compute or Networking service. You can set the network parameters in the `manila.conf` file.

Share Servers

A `share server` is a logical entity that hosts the shares created on a specific `share network`. A `share server` may be a configuration object within the storage controller, or it may represent logical resources provisioned within an OpenStack deployment used to support the data path used to access `shares`.

`Share servers` interact with network services to determine the appropriate IP addresses on which to export `shares` according to the related `share network`. The Shared File Systems service has a pluggable network model that allows `share servers` to work with different implementations of the Networking service.

Share management

A share is a remote, mountable file system. You can mount a share to and access a share from several hosts by several users at a time.

You can create a share and associate it with a network, list shares, and show information for, update, and delete a specified share. You can also create snapshots of shares. To create a snapshot, you specify the ID of the share that you want to snapshot.

The shares are based on of the supported Shared File Systems protocols:

- *NFS*. Network File System (NFS).
- *CIFS*. Common Internet File System (CIFS).
- *GLUSTERFS*. Gluster file system (GlusterFS).
- *HDFS*. Hadoop Distributed File System (HDFS).
- *CEPHFS*. Ceph File System (CephFS).
- *MAPRFS*. MapR File System (MAPRFS).

The Shared File Systems service provides set of drivers that enable you to use various network file storage devices, instead of the base implementation. That is the real purpose of the Shared File Systems service in production.

Share basic operations

General concepts

To create a file share, and access it, the following general concepts are prerequisite knowledge:

1. To create a share, use **`manila create`** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, `CephFS` or `MAPRFS` share file system protocols are supported.
2. You can also optionally specify the share network and the share type.
3. After the share becomes available, use the **`manila show`** command to get the share export locations.
4. After getting the share export locations, you can create an *access rule* for the share, mount it and work with files on the remote file system.

There are big number of the share drivers created by different vendors in the Shared File Systems service. As a Python class, each share driver can be set for the *back end* and run in the back end to manage the share operations.

Initially there are two driver modes for the back ends:

- no share servers mode
- share servers mode

Each share driver supports one or two of possible back end modes that can be configured in the `manila.conf` file. The configuration option `driver_handles_share_servers` in the `manila.conf` file sets the share servers mode or no share servers mode, and defines the driver mode for share storage lifecycle management:

Mode	Config option	Description
no share servers	driver_handles_share_servers = False	An administrator rather than a share driver manages the bare metal storage with some net interface instead of the presence of the share servers.
share servers	driver_handles_share_servers = True	The share driver creates the share server and manages, or handles, the share server life cycle.

It is *the share types* which have the extra specifications that help scheduler to filter back ends and choose the appropriate back end for the user that requested to create a share. The required extra boolean specification for each share type is `driver_handles_share_servers`. As an administrator, you can create the share types with the specifications you need. For details of managing the share types and configuration the back ends, see *Share types* and *Multi-storage configuration* documentation.

You can create a share in two described above modes:

- in a no share servers mode without specifying the share network and specifying the share type with `driver_handles_share_servers = False` parameter. See subsection *Create a share in no share servers mode*.
- in a share servers mode with specifying the share network and the share type with `driver_handles_share_servers = True` parameter. See subsection *Create a share in share servers mode*.

Create a share in no share servers mode

To create a file share in no share servers mode, you need to:

1. To create a share, use **`manila create`** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, `CephFS` or `MAPRFS` share file system protocols are supported.
2. You should specify the *share type* with `driver_handles_share_servers = False` extra specification.
3. You must not specify the `share network` because no share servers are created. In this mode the Shared File Systems service expects that administrator has some bare metal storage with some net interface.
4. The **`manila create`** command creates a share. This command does the following things:
   - The *manila-scheduler* service will find the back end with `driver_handles_share_servers = False` mode due to filtering the extra specifications of the share type.
   - The share is created using the storage that is specified in the found back end.
5. After the share becomes available, use the **`manila show`** command to get the share export locations.

In the example to create a share, the created already share type named `my_type` with `driver_handles_share_servers = False` extra specification is used.

Check share types that exist, run:

```
$ manila type-list
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
| ID   | Name    | visibility | is_default | required_extra_specs           ␣
↪    | optional_extra_specs    |
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
| %ID% | my_type | public     | -          | driver_handles_share_servers :␣
↪False | snapshot_support : True |
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
```

Create a private share with `my_type` share type, NFS shared file system protocol, and size 1 GB:

```
$ manila create nfs 1 --name Share1 --description "My share" --share-type my_
↪type
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | my_type                              |
| description                 | My share                             |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| share_server_id             | None                                 |
| share_group_id              | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| size                        | 1                                    |
| name                        | Share1                               |
| share_type                  | 14ee8575-aac2-44af-8392-d9c9d344f392 |
| has_replicas                | False                                |
| replication_type            | None                                 |
| created_at                  | 2016-03-25T12:02:46.000000           |
| share_proto                 | NFS                                  |
| project_id                  | 907004508ef4447397ce6741a8f037c1     |
| metadata                    | {}                                   |
+-----------------------------+--------------------------------------+
```

New share `Share2` should have a status `available`:

```
$ manila show Share2
+-----------------------------+------------------------------------------------
↪-----------+
| Property                    | Value                                          ␣
↪           |
```

(continues on next page)

(continued from previous page)

```
+------------------------------+------------------------------------------------
↪-----------+
| status                       | available                                      ␣
↪           |
| share_type_name              | my_type                                        ␣
↪           |
| description                  | My share                                       ␣
↪           |
| availability_zone            | nova                                           ␣
↪           |
| share_network_id             | None                                           ␣
↪           |
| export_locations             |                                                ␣
↪           |
|                              | path = 10.0.0.4:/shares/manila_share_a5fb1ab7_
↪...        |
|                              | preferred = False                              ␣
↪           |
|                              | is_admin_only = False                          ␣
↪           |
|                              | id = 9e078eee-bcad-40b8-b4fe-1c916cf98ed1      ␣
↪           |
|                              | share_instance_id = a5fb1ab7-0bbd-465b-ac14-
↪05706294b6e9 |
|                              | path = 172.18.198.52:/shares/manila_share_
↪a5fb1ab7_...   |
|                              | preferred = False                              ␣
↪           |
|                              | is_admin_only = True                           ␣
↪           |
|                              | id = 44933f59-e0e3-4483-bb88-72ba7c486f41      ␣
↪           |
|                              | share_instance_id = a5fb1ab7-0bbd-465b-ac14-
↪05706294b6e9 |
| share_server_id              | None                                           ␣
↪           |
| share_group_id               | None                                           ␣
↪           |
| host                         | manila@paris#epsilon                           ␣
↪           |
| access_rules_status          | active                                         ␣
↪           |
| snapshot_id                  | None                                           ␣
↪           |
| is_public                    | False                                          ␣
↪           |
| task_state                   | None                                           ␣
↪           |
| snapshot_support             | True                                           ␣
↪           |
```

(continues on next page)

(continued from previous page)

```
| id                           | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88
↪            |
| size                         | 1
↪            |
| name                         | Share1
↪            |
| share_type                   | 14ee8575-aac2-44af-8392-d9c9d344f392
↪            |
| has_replicas                 | False
↪            |
| replication_type             | None
↪            |
| created_at                   | 2016-03-25T12:02:46.000000
↪            |
| share_proto                  | NFS
↪            |
| project_id                   | 907004508ef4447397ce6741a8f037c1
↪            |
| metadata                     | {}
↪            |
+------------------------------+-----------------------------------------------
↪-----------+
```

Create a share in share servers mode

To create a file share in share servers mode, you need to:

1. To create a share, use **`manila create`** command and specify the required arguments: the size of the share and the shared file system protocol. `NFS`, `CIFS`, `GlusterFS`, `HDFS`, `CephFS` or `MAPRFS` share file system protocols are supported.
2. You should specify the *share type* with `driver_handles_share_servers = True` extra specification.
3. You should specify the *share network*.
4. The **`manila create`** command creates a share. This command does the following things:
   - The *manila-scheduler* service will find the back end with `driver_handles_share_servers = True` mode due to filtering the extra specifications of the share type.
   - The share driver will create a share server with the share network. For details of creating the resources, see the documentation of the specific share driver.
5. After the share becomes available, use the **`manila show`** command to get the share export location.

In the example to create a share, the default share type and the already existing share network are used.

Note: There is no default share type just after you started manila as the administrator. See *Share types* to create the default share type. To create a share network, use *Share networks*.

Check share types that exist, run:

```
$ manila type-list
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
| ID   | Name    | visibility | is_default | required_extra_specs           ␣
↪    | optional_extra_specs    |
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
| %id% | default | public     | YES        | driver_handles_share_servers :␣
↪True  | snapshot_support : True |
+------+---------+------------+------------+--------------------------------
↪----+-------------------------+
```

Check share networks that exist, run:

```
$ manila share-network-list
+--------------------------------------+--------------+
| id                                   | name         |
+--------------------------------------+--------------+
| c895fe26-92be-4152-9e6c-f2ad230efb13 | my_share_net |
+--------------------------------------+--------------+
```

Create a public share with `my_share_net` network, `default` share type, NFS shared file system protocol, and size 1 GB:

```
$ manila create nfs 1 \
    --name "Share2" \
    --description "My second share" \
    --share-type default \
    --share-network my_share_net \
    --metadata aim=testing \
    --public
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | default                              |
| description                 | My second share                      |
| availability_zone           | None                                 |
| share_network_id            | c895fe26-92be-4152-9e6c-f2ad230efb13 |
| share_server_id             | None                                 |
| share_group_id              | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | True                                 |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | 195e3ba2-9342-446a-bc93-a584551de0ac |
| size                        | 1                                    |
```

(continues on next page)

(continued from previous page)

```
| name                         | Share2                               |
| share_type                   | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf |
| has_replicas                 | False                                |
| replication_type             | None                                 |
| created_at                   | 2016-03-25T12:13:40.000000           |
| share_proto                  | NFS                                  |
| project_id                   | 907004508ef4447397ce6741a8f037c1     |
| metadata                     | {u'aim': u'testing'}                 |
+------------------------------+--------------------------------------+
```

The share also can be created from a share snapshot. For details, see *Share snapshots*.

See the share in a share list:

```
$ manila list
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+------------------
↪-+
| ID                                   | Name    | Size | Share Proto |␣
↪Status    | Is Public | Share Type Name | Host                 |␣
↪Availability Zone |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+------------------
↪-+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1  | 1    | NFS         |␣
↪available | False     | my_type         | manila@paris#epsilon | nova      ␣
↪       |
| 195e3ba2-9342-446a-bc93-a584551de0ac | Share2  | 1    | NFS         |␣
↪available | True      | default         | manila@london#LONDON | nova      ␣
↪       |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+------------------
↪-+
```

Check the share status and see the share export locations. After `creating` status share should have status `available`:

```
$ manila show Share2
+----------------------+-----------------------------------------------------------
↪----------------+
| Property             | Value                                                     ␣
↪                |
+----------------------+-----------------------------------------------------------
↪----------------+
| status               | available                                                 ␣
↪                |
| share_type_name      | default                                                   ␣
↪                |
| description          | My second share                                           ␣
↪                |
```

(continues on next page)

(continued from previous page)

```
| availability_zone     | nova                                                
```

(continued from previous page)

```
| has_replicas          | False                                                ␣
↪                  |
| replication_type      | None                                                 ␣
↪                  |
| created_at            | 2016-03-25T12:13:40.000000                           ␣
↪                  |
| share_proto           | NFS                                                  ␣
↪                  |
| project_id            | 907004508ef4447397ce6741a8f037c1                     ␣
↪                  |
| metadata              | {u'aim': u'testing'}                                 ␣
↪                  |
+----------------------+------------------------------------------------------
↪-----------------+
```

`is_public` defines the level of visibility for the share: whether other projects can or cannot see the share. By default, the share is private.

Update share

Update the name, or description, or level of visibility for all projects for the share if you need:

```
$ manila update Share2 --description "My second share. Updated" --is-public␣
↪False

$ manila show Share2
+----------------------+------------------------------------------------------
↪-----------------+
| Property             | Value                                                ␣
↪                  |
+----------------------+------------------------------------------------------
↪-----------------+
| status               | available                                            ␣
↪                  |
| share_type_name      | default                                              ␣
↪                  |
| description          | My second share. Updated                             ␣
↪                  |
| availability_zone    | nova                                                 ␣
↪                  |
| share_network_id     | c895fe26-92be-4152-9e6c-f2ad230efb13                 ␣
↪                  |
| export_locations     |                                                      ␣
↪                  |
|                      | path = 10.254.0.3:/shares/share-fe874928-39a2-441b-
↪8d24-29e6f0fde965 |
|                      | preferred = False                                    ␣
↪                  |
|                      | is_admin_only = False                                ␣
↪                  |
```

(continues on next page)

(continued from previous page)

```
|                      | id = de6d4012-6158-46f0-8b28-4167baca51a7                  ␣
↪                  |
|                      | share_instance_id = fe874928-39a2-441b-8d24-
↪29e6f0fde965              |
|                      | path = 10.0.0.3:/shares/share-fe874928-39a2-441b-
↪8d24-29e6f0fde965   |
|                      | preferred = False                                         ␣
↪                  |
|                      | is_admin_only = True                                      ␣
↪                  |
|                      | id = 602d0f5c-921b-4e45-bfdb-5eec8a89165a                 ␣
↪                  |
|                      | share_instance_id = fe874928-39a2-441b-8d24-
↪29e6f0fde965              |
| share_server_id      | 2e9d2d02-883f-47b5-bb98-e053b8d1e683                      ␣
↪                  |
| share_group_id       | None                                                      ␣
↪                 |
| host                 | manila@london#LONDON                                      ␣
↪                  |
| access_rules_status  | active                                                    ␣
↪                  |
| snapshot_id          | None                                                      ␣
↪                  |
| is_public            | False                                                     ␣
↪                  |
| task_state           | None                                                      ␣
↪                  |
| snapshot_support     | True                                                      ␣
↪                  |
| id                   | 195e3ba2-9342-446a-bc93-a584551de0ac                      ␣
↪                  |
| size                 | 1                                                         ␣
↪                  |
| name                 | Share2                                                    ␣
↪                  |
| share_type           | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf                      ␣
↪                  |
| has_replicas         | False                                                     ␣
↪                  |
| replication_type     | None                                                      ␣
↪                  |
| created_at           | 2016-03-25T12:13:40.000000                                ␣
↪                  |
| share_proto          | NFS                                                       ␣
↪                  |
| project_id           | 907004508ef4447397ce6741a8f037c1                          ␣
↪                  |
| metadata             | {u'aim': u'testing'}                                      ␣
↪                  |
```

(continues on next page)

(continued from previous page)

```
+----------------------+-------------------------------------------------------
↪---------------+
```

A share can have one of these status values:

Status	Description
creating	The share is being created.
deleting	The share is being deleted.
error	An error occurred during share creation.
error_deleting	An error occurred during share deletion.
available	The share is ready to use.
manage_starting	Share manage started.
manage_error	Share manage failed.
unmanage_starting	Share unmanage started.
unmanage_error	Share cannot be unmanaged.
unmanaged	Share was unmanaged.
extending	The extend, or increase, share size request was issued successfully.
extending_error	Extend share failed.
shrinking	Share is being shrunk.
shrinking_error	Failed to update quota on share shrinking.
shrinking_possible_data_loss_error	Shrink share failed due to possible data loss.
migrating	Share migration is in progress.

Share metadata

If you want to set the metadata key-value pairs on the share, run:

```
$ manila metadata Share2 set project=my_abc deadline=01/20/16
```

Get all metadata key-value pairs of the share:

```
$ manila metadata-show Share2
+----------+----------+
| Property | Value    |
+----------+----------+
| aim      | testing  |
| project  | my_abc   |
| deadline | 01/20/16 |
+----------+----------+
```

You can update the metadata:

```
$ manila metadata-update-all Share2 deadline=01/30/16
+----------+----------+
| Property | Value    |
+----------+----------+
```

(continues on next page)

(continued from previous page)

```
| deadline | 01/30/16 |
+----------+----------+
```

You also can unset the metadata using **manila metadata <share_name> unset <metadata_key(s)>**.

Reset share state

As administrator, you can reset the state of a share.

Use **manila reset-state [state <state>] <share>** command to reset share state, where `state` indicates which state to assign the share. Options include `available`, `error`, `creating`, `deleting`, `error_deleting` states.

```
$ manila reset-state Share2 --state deleting

$ manila show Share2
+----------------------+--------------------------------------------------------
↪----------------+
| Property             | Value                                                  ␣
↪                |
+----------------------+--------------------------------------------------------
↪----------------+
| status               | deleting                                               ␣
↪                |
| share_type_name      | default                                                ␣
↪                |
| description          | My second share. Updated                               ␣
↪                |
| availability_zone    | nova                                                   ␣
↪                |
| share_network_id     | c895fe26-92be-4152-9e6c-f2ad230efb13                   ␣
↪                |
| export_locations     |                                                        ␣
↪                |
|                      | path = 10.254.0.3:/shares/share-fe874928-39a2-441b-
↪8d24-29e6f0fde965 |
|                      | preferred = False                                      ␣
↪                |
|                      | is_admin_only = False                                  ␣
↪                |
|                      | id = de6d4012-6158-46f0-8b28-4167baca51a7              ␣
↪                |
|                      | share_instance_id = fe874928-39a2-441b-8d24-
↪29e6f0fde965              |
|                      | path = 10.0.0.3:/shares/share-fe874928-39a2-441b-
↪8d24-29e6f0fde965   |
|                      | preferred = False                                      ␣
↪                |
|                      | is_admin_only = True                                   ␣
↪                |
```

(continues on next page)

(continued from previous page)

```
|                      | id = 602d0f5c-921b-4e45-bfdb-5eec8a89165a
↪                 |
|                      | share_instance_id = fe874928-39a2-441b-8d24-
↪29e6f0fde965             |
| share_server_id      | 2e9d2d02-883f-47b5-bb98-e053b8d1e683
↪                 |
| share_group_id       | None
↪                    |
| host                 | manila@london#LONDON
↪                 |
| access_rules_status  | active
↪                 |
| snapshot_id          | None
↪                 |
| is_public            | False
↪                 |
| task_state           | None
↪                 |
| snapshot_support     | True
↪                 |
| id                   | 195e3ba2-9342-446a-bc93-a584551de0ac
↪                 |
| size                 | 1
↪                 |
| name                 | Share2
↪                 |
| share_type           | bf6ada49-990a-47c3-88bc-c0cb31d5c9bf
↪                 |
| has_replicas         | False
↪                 |
| replication_type     | None
↪                 |
| created_at           | 2016-03-25T12:13:40.000000
↪                 |
| share_proto          | NFS
↪                 |
| project_id           | 907004508ef4447397ce6741a8f037c1
↪                 |
| metadata             | {u'deadline': u'01/30/16'}
↪                 |
+----------------------+-------------------------------------------------------
↪----------------+
```

Delete and force-delete share

You also can force-delete a share. The shares cannot be deleted in transitional states. The transitional states are `creating`, `deleting`, `managing`, `unmanaging`, `migrating`, `extending`, and `shrinking` statuses for the shares. Force-deletion deletes an object in any state. Use the `policy.yaml` file to grant permissions for this action to other roles.

Tip: The configuration file `policy.yaml` may be used from different places. The path `/etc/manila/policy.yaml` is one of expected paths by default.

Use **manila delete <share_name_or_ID>** command to delete a specified share:

```
$ manila delete %share_name_or_id%
```

```
$ manila delete %share_name_or_id% --consistency-group %consistency-group-id%
```

If you try to delete the share in one of the transitional state using soft-deletion youll get an error:

```
$ manila delete Share2
Delete for share 195e3ba2-9342-446a-bc93-a584551de0ac failed: Invalid share:␣
↪Share status must be one of ('available', 'error', 'inactive'). (HTTP 403)␣
↪(Request-ID: req-9a77b9a0-17d2-4d97-8a7a-b7e23c27f1fe)
ERROR: Unable to delete any of the specified shares.
```

A share cannot be deleted in a transitional status, that it why an error from `python-manilaclient` appeared.

Print the list of all shares for all projects:

```
$ manila list --all-tenants
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
| ID                                   | Name    | Size | Share Proto |␣
↪Status    | Is Public | Share Type Name | Host                 |␣
↪Availability Zone |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1  | 1    | NFS         |␣
↪available | False     | my_type         | manila@paris#epsilon | nova      ␣
↪       |
| 195e3ba2-9342-446a-bc93-a584551de0ac | Share2  | 1    | NFS         |␣
↪available | False     | default         | manila@london#LONDON | nova      ␣
↪       |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
```

Force-delete Share2 and check that it is absent in the list of shares, run:

```
$ manila force-delete Share2

$ manila list
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
| ID                                   | Name    | Size | Share Proto |␣
↪Status    | Is Public | Share Type Name | Host                 |␣
↪Availability Zone |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
| 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 | Share1  | 1    | NFS         |␣
↪available | False     | my_type         | manila@paris#epsilon | nova      ␣
↪       |
+--------------------------------------+---------+------+-------------+-------
↪----+-----------+-----------------+----------------------+-----------------
↪-+
```

Manage access to share

The Shared File Systems service allows to grant or deny access to a specified share, and list the permissions for a specified share.

To grant or deny access to a share, specify one of these supported share access levels:

- **rw**. Read and write (RW) access. This is the default value.
- **ro**. Read-only (RO) access.

You must also specify one of these supported authentication methods:

- **ip**. Authenticates an instance through its IP address. A valid format is `XX.XX.XX.XX` or `XX.XX.XX.XX/XX`. For example `0.0.0.0/0`.
- **user**. Authenticates by a specified user or group name. A valid value is an alphanumeric string that can contain some special characters and is from 4 to 32 characters long.
- **cert**. Authenticates an instance through a TLS certificate. Specify the TLS identity as the IDENTKEY. A valid value is any string up to 64 characters long in the common name (CN) of the certificate. The meaning of a string depends on its interpretation.
- **cephx**. Ceph authentication system. Specify the Ceph auth ID that needs to be authenticated and authorized for share access by the Ceph back end. A valid value must be non-empty, consist of ASCII printable characters, and not contain periods.

Try to mount NFS share with export path `10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9` on the node with IP address `10.0.0.13`:

```
$ sudo mount -v -t nfs 10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_
↪05706294b6e9 /mnt/
mount.nfs: timeout set for Tue Oct  6 10:37:23 2015
```

(continues on next page)

(continued from previous page)

```
mount.nfs: trying text-based options 'vers=4,addr=10.0.0.4,clientaddr=10.0.0.
↪13'
mount.nfs: mount(2): Permission denied
mount.nfs: access denied by server while mounting 10.0.0.4:/shares/manila_
↪share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
```

An error message Permission denied appeared, so you are not allowed to mount a share without an access rule. Allow access to the share with `ip` access type and `10.0.0.13` IP address:

```
$ manila access-allow Share1 ip 10.0.0.13 --access-level rw
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| share_id     | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| access_type  | ip                                   |
| access_to    | 10.0.0.13                            |
| access_level | rw                                   |
| state        | new                                  |
| id           | de715226-da00-4cfc-b1ab-c11f3393745e |
+--------------+--------------------------------------+
```

Try to mount a share again. This time it is mounted successfully:

```
$ sudo mount -v -t nfs 10.0.0.4:/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_
↪05706294b6e9 /mnt/
```

Since it is allowed node on 10.0.0.13 read and write access, try to create a file on a mounted share:

```
$ cd /mnt
$ ls
lost+found
$ touch my_file.txt
```

Connect via SSH to the `10.0.0.4` node and check new file *my_file.txt* in the `/shares/manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9` directory:

```
$ ssh 10.0.0.4
$ cd /shares
$ ls
manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
$ cd manila_share_a5fb1ab7_0bbd_465b_ac14_05706294b6e9
$ ls
lost+found  my_file.txt
```

You have successfully created a file from instance that was given access by its IP address.

Allow access to the share with `user` access type:

```
$ manila access-allow Share1 user demo --access-level rw
+--------------+--------------------------------------+
| Property     | Value                                |
```

(continues on next page)

(continued from previous page)

```
+--------------+--------------------------------------+
| share_id     | 10f5a2a1-36f5-45aa-a8e6-00e94e592e88 |
| access_type  | user                                 |
| access_to    | demo                                 |
| access_level | rw                                   |
| state        | new                                  |
| id           | 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a |
+--------------+--------------------------------------+
```

Note: Different share features are supported by different share drivers. For the example, the Generic driver with the Block Storage service as a back-end doesnt support `user` and `cert` authentications methods. For details of supporting of features by different drivers, see Manila share features support mapping.

To verify that the access rules (ACL) were configured correctly for a share, you list permissions for a share:

```
$ manila access-list Share1
+--------------------------------------+-------------+------------+-----------
↪---+--------+
| id                                   | access type | access to  | access␣
↪level | state  |
+--------------------------------------+-------------+------------+-----------
↪---+--------+
| 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a | user        | demo       | rw       ␣
↪   | error  |
| de715226-da00-4cfc-b1ab-c11f3393745e | ip          | 10.0.0.13  | rw       ␣
↪   | active |
+--------------------------------------+-------------+------------+-----------
↪---+--------+
```

Deny access to the share and check that deleted access rule is absent in the access rule list:

```
$ manila access-deny Share1 de715226-da00-4cfc-b1ab-c11f3393745e

$ manila access-list Share1
+--------------------------------------+-------------+-----------+------------
↪--+-------+
| id                                   | access type | access to | access␣
↪level | state |
+--------------------------------------+-------------+-----------+------------
↪--+-------+
| 4f391c6b-fb4f-47f5-8b4b-88c5ec9d568a | user        | demo      | rw       ␣
↪  | error |
+--------------------------------------+-------------+-----------+------------
↪--+-------+
```

Manage and unmanage share

To `manage` a share means that an administrator, rather than a share driver, manages the storage lifecycle. This approach is appropriate when an administrator already has the custom non-manila share with its size, shared file system protocol, and export path, and an administrator wants to register it in the Shared File System service.

To `unmanage` a share means to unregister a specified share from the Shared File Systems service. Administrators can revert an unmanaged share to managed status if needed.

Unmanage a share

Note: The `unmanage` operation is not supported for shares that were created on top of share servers and created with share networks until Shared File Systems API version `2.49` (Stein/Manila 8.0.0 release).

Important: Shares that have dependent snapshots or share replicas cannot be removed from the Shared File Systems service unless the snapshots have been removed or unmanaged and the share replicas have been removed.

Unmanaging a share removes it from the management of the Shared File Systems service without deleting the share. It is a non-disruptive operation and existing clients are not disconnected, and the functionality is aimed at aiding infrastructure operations and maintenance workflows. To unmanage a share, run the **`manila unmanage <share>`** command. Then try to print the information about the share. The returned result should indicate that Shared File Systems service wont find the share:

```
$ manila unmanage share_for_docs
$ manila show share_for_docs
ERROR: No share with a name or ID of 'share_for_docs' exists.
```

Manage a share

Note: The `manage` operation is not supported for shares that are exported on share servers via share networks until Shared File Systems API version `2.49` (Stein/Manila 8.0.0 release).

Note: From API version 2.53, if the requester specifies a share type containing a `replication_type` extra spec while managing a share, manila quota system will reserve and consume resources for two additional quotas: `share_replicas` and `replica_gigabytes`. From API version 2.62, manila quota system will validate size of the share against `per_share_gigabytes` quota.

To register the non-managed share in the File System service, run the **`manila manage`** command:

```
manila manage [--name <name>] [--description <description>]
              [--share_type <share-type>]
```

(continues on next page)

(continued from previous page)

```
              [--share-server-id <share_server_id>]
              [--driver_options [<key=value> [<key=value> ...]]]
              <service_host> <protocol> <export_path>
```

The positional arguments are:

- service_host. The manage-share service host in `host@backend#POOL` format, which consists of the host name for the back end, the name of the back end, and the pool name for the back end.
- protocol. The Shared File Systems protocol of the share to manage. Valid values are NFS, CIFS, GlusterFS, HDFS or MAPRFS.
- export_path. The share export path in the format appropriate for the protocol:
  - NFS protocol. 10.0.0.1:/foo_path.
  - CIFS protocol. \\10.0.0.1\foo_name_of_cifs_share.
  - HDFS protocol. hdfs://10.0.0.1:foo_port/foo_share_name.
  - GlusterFS. 10.0.0.1:/foo_volume.
  - MAPRFS. maprfs:///share-0 -C -Z -N foo.

The optional arguments are:

- name. The name of the share that is being managed.
- share_type. The share type of the share that is being managed. If not specified, the service will try to manage the share with the configured default share type.
- share_server_id. must be provided to manage shares within share networks. This argument can only be used with File Systems API version `2.49` (Stein/Manila 8.0.0 release) and beyond.
- driver_options. An optional set of one or more key and value pairs that describe driver options. As a result, a special share type named `for_managing` was used in example.

To manage share, run:

```
$ manila manage \
    manila@paris#shares \
    nfs \
    1.0.0.4:/shares/manila_share_6d2142d8_2b9b_4405_867f_8a48094c893f \
    --name share_for_docs \
    --description "We manage share." \
    --share_type for_managing
+-----------------------------+----------------------------------------+
| Property                    | Value                                  |
+-----------------------------+----------------------------------------+
| status                      | manage_starting                        |
| share_type_name             | for_managing                           |
| description                 | We manage share.                       |
| availability_zone           | None                                   |
| share_network_id            | None                                   |
| share_server_id             | None                                   |
| share_group_id              | None                                   |
```

(continues on next page)

(continued from previous page)

```
| host                         | manila@paris#shares                  |
| access_rules_status          | active                               |
| snapshot_id                  | None                                 |
| is_public                    | False                                |
| task_state                   | None                                 |
| snapshot_support             | True                                 |
| id                           | ddfb1240-ed5e-4071-a031-b842035a834a |
| size                         | None                                 |
| name                         | share_for_docs                       |
| share_type                   | 14ee8575-aac2-44af-8392-d9c9d344f392 |
| has_replicas                 | False                                |
| replication_type             | None                                 |
| created_at                   | 2016-03-25T15:22:43.000000           |
| share_proto                  | NFS                                  |
| project_id                   | 907004508ef4447397ce6741a8f037c1     |
| metadata                     | {}                                   |
+------------------------------+--------------------------------------+
```

Check that the share is available:

```
$ manila show share_for_docs
+----------------------+------------------------------------------------------------
↪--------------------+
| Property             | Value                                                      ␣
↪                    |
+----------------------+------------------------------------------------------------
↪--------------------+
| status               | available                                                  ␣
↪                    |
| share_type_name      | for_managing                                               ␣
↪                    |
| description          | We manage share.                                           ␣
↪                    |
| availability_zone    | None                                                       ␣
↪                    |
| share_network_id     | None                                                       ␣
↪                    |
| export_locations     |                                                            ␣
↪                    |
|                      | path = 1.0.0.4:/shares/manila_share_6d2142d8_2b9b_
↪4405_867f_8a48094c893f |
|                      | preferred = False                                          ␣
↪                    |
|                      | is_admin_only = False                                      ␣
↪                    |
|                      | id = d4d048bf-4159-4a94-8027-e567192b8d30                  ␣
↪                    |
|                      | share_instance_id = 4c8e3887-4f9a-4775-bab4-
↪e5840a09c34e                 |
```

(continues on next page)

(continued from previous page)

```
|                      | path = 2.0.0.3:/shares/manila_share_6d2142d8_2b9b_
↪4405_867f_8a48094c893f |
|                      | preferred = False                                 ␣
↪                    |
|                      | is_admin_only = True                              ␣
↪                    |
|                      | id = 1dd4f0a3-778d-486a-a851-b522f6e7cf5f         ␣
↪                    |
|                      | share_instance_id = 4c8e3887-4f9a-4775-bab4-
↪e5840a09c34e                 |
| share_server_id      | None                                              ␣
↪                    |
| share_group_id       | None                                              ␣
↪                    |
| host                 | manila@paris#shares                               ␣
↪                    |
| access_rules_status  | active                                            ␣
↪                    |
| snapshot_id          | None                                              ␣
↪                    |
| is_public            | False                                             ␣
↪                    |
| task_state           | None                                              ␣
↪                    |
| snapshot_support     | True                                              ␣
↪                    |
| id                   | ddfb1240-ed5e-4071-a031-b842035a834a              ␣
↪                    |
| size                 | 1                                                 ␣
↪                    |
| name                 | share_for_docs                                    ␣
↪                    |
| share_type           | 14ee8575-aac2-44af-8392-d9c9d344f392              ␣
↪                    |
| has_replicas         | False                                             ␣
↪                    |
| replication_type     | None                                              ␣
↪                    |
| created_at           | 2016-03-25T15:22:43.000000                        ␣
↪                    |
| share_proto          | NFS                                               ␣
↪                    |
| project_id           | 907004508ef4447397ce6741a8f037c1                  ␣
↪                    |
| metadata             | {}                                                ␣
↪                    |
+----------------------+---------------------------------------------------
↪--------------------+
```

Manage and unmanage share snapshot

To `manage` a share snapshot means that an administrator, rather than a share driver, manages the storage lifecycle. This approach is appropriate when an administrator manages share snapshots outside of the Shared File Systems service and wants to register it with the service.

To `unmanage` a share snapshot means to unregister a specified share snapshot from the Shared File Systems service. Administrators can revert an unmanaged share snapshot to managed status if needed.

Unmanage a share snapshot

The `unmanage` operation is not supported for shares that were created on top of share servers and created with share networks. The Share service should have the option `driver_handles_share_servers = False` set in the `manila.conf` file.

To unmanage managed share snapshot, run the **`manila snapshot-unmanage <share_snapshot>`** command. Then try to print the information about the share snapshot. The returned result should indicate that Shared File Systems service wont find the share snapshot:

```
$ manila snapshot-unmanage my_test_share_snapshot
$ manila snapshot-show my_test_share_snapshot
ERROR: No sharesnapshot with a name or ID of 'my_test_share_snapshot'
exists.
```

Manage a share snapshot

To register the non-managed share snapshot in the File System service, run the **`manila snapshot-manage`** command:

```
manila snapshot-manage [--name <name>] [--description <description>]
              [--driver_options [<key=value> [<key=value> ...]]]
              <share> <provider_location>
```

The positional arguments are:

- share. Name or ID of the share.
- provider_location. Provider location of the share snapshot on the backend.

The `driver_options` is an optional set of one or more key and value pairs that describe driver options.

To manage share snapshot, run:

```
$ manila snapshot-manage \
    9ba52cc6-c97e-4b40-8653-4bcbaaf9628d \
    4d1e2863-33dd-4243-bf39-f7354752097d \
    --name my_test_share_snapshot \
    --description "My test share snapshot" \
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| status           | manage_starting                      |
```

(continues on next page)

(continued from previous page)

```
| share_id          | 9ba52cc6-c97e-4b40-8653-4bcbaaf9628d |
| user_id           | d9f4003655c94db5b16c591920be1f91     |
| description       | My test share snapshot               |
| created_at        | 2016-07-25T04:49:42.600980           |
| size              | None                                 |
| share_proto       | NFS                                  |
| provider_location | 4d1e2863-33dd-4243-bf39-f7354752097d |
| id                | 89c663b5-026d-45c7-a43b-56ef0ba0faab |
| project_id        | aaa33a0ca4324965a3e65ae47e864e94     |
| share_size        | 1                                    |
| name              | my_test_share_snapshot               |
+-------------------+--------------------------------------+
```

Check that the share snapshot is available:

```
$ manila snapshot-show my_test_share_snapshot
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | available                            |
| share_id          | 9ba52cc6-c97e-4b40-8653-4bcbaaf9628d |
| user_id           | d9f4003655c94db5b16c591920be1f91     |
| description       | My test share snapshot               |
| created_at        | 2016-07-25T04:49:42.000000           |
| size              | 1                                    |
| share_proto       | NFS                                  |
| provider_location | 4d1e2863-33dd-4243-bf39-f7354752097d |
| id                | 89c663b5-026d-45c7-a43b-56ef0ba0faab |
| project_id        | aaa33a0ca4324965a3e65ae47e864e94     |
| share_size        | 1                                    |
| name              | my_test_share_snapshot               |
+-------------------+--------------------------------------+
```

Resize share

To change file share size, use the **`manila extend`** command and the **`manila shrink`** command. For most drivers it is safe operation. If you want to be sure that your data is safe, you can make a share back up by creating a snapshot of it.

You can extend and shrink the share with the **`manila extend`** and **`manila shrink`** commands respectively, and specify the share with the new size that does not exceed the quota. For details, see *Quotas and Limits*. You also cannot shrink share size to 0 or to a greater value than the current share size.

Note: From API version 2.53, extending a replicated share, manila quota system will reserve and consume resources for two additional quotas: `share_replicas` and `replica_gigabytes`. This request will fail if there is no available quotas to extend the share and all of its share replicas.

While extending, the share has an `extending` status. This means that the increase share size request was

issued successfully.

To extend the share and check the result, run:

```
$ manila extend docs_resize 2
$ manila show docs_resize
+----------------------+--------------------------------------------------------
↪---------------------+
| Property             | Value                                                  ␣
↪                     |
+----------------------+--------------------------------------------------------
↪---------------------+
| status               | available                                              ␣
↪                     |
| share_type_name      | my_type                                                ␣
↪                     |
| description          | None                                                   ␣
↪                     |
| availability_zone    | nova                                                   ␣
↪                     |
| share_network_id     | None                                                   ␣
↪                     |
| export_locations     |                                                        ␣
↪                     |
|                      | path = 1.0.0.4:/shares/manila_share_b8afc508_8487_
↪442b_b170_ea65b07074a8 |
|                      | preferred = False                                      ␣
↪                     |
|                      | is_admin_only = False                                  ␣
↪                     |
|                      | id = 3ffb76f4-92b9-4639-83fd-025bc3e302ff              ␣
↪                     |
|                      | share_instance_id = b8afc508-8487-442b-b170-
↪ea65b07074a8                 |
|                      | path = 2.0.0.3:/shares/manila_share_b8afc508_8487_
↪442b_b170_ea65b07074a8 |
|                      | preferred = False                                      ␣
↪                     |
|                      | is_admin_only = True                                   ␣
↪                     |
|                      | id = 1f0e263f-370d-47d3-95f6-1be64088b9da              ␣
↪                     |
|                      | share_instance_id = b8afc508-8487-442b-b170-
↪ea65b07074a8                 |
| share_server_id      | None                                                   ␣
↪                     |
| share_group_id       | None                                                   ␣
↪                     |
| host                 | manila@paris#shares                                    ␣
↪                     |
| access_rules_status  | active                                                 ␣
↪                     |
```

(continues on next page)

(continued from previous page)

```
| snapshot_id          | None                                                 ␣
↪                      |
| is_public            | False                                                ␣
↪                      |
| task_state           | None                                                 ␣
↪                      |
| snapshot_support     | True                                                 ␣
↪                      |
| id                   | b07dbebe-a328-403c-b402-c8871c89e3d1                 ␣
↪                      |
| size                 | 2                                                    ␣
↪                      |
| name                 | docs_resize                                          ␣
↪                      |
| share_type           | 14ee8575-aac2-44af-8392-d9c9d344f392                 ␣
↪                      |
| has_replicas         | False                                                ␣
↪                      |
| replication_type     | None                                                 ␣
↪                      |
| created_at           | 2016-03-25T15:33:18.000000                           ␣
↪                      |
| share_proto          | NFS                                                  ␣
↪                      |
| project_id           | 907004508ef4447397ce6741a8f037c1                     ␣
↪                      |
| metadata             | {}                                                   ␣
↪                      |
+----------------------+------------------------------------------------------
↪---------------------+
```

While shrinking, the share has a `shrinking` status. This means that the decrease share size request was issued successfully. To shrink the share and check the result, run:

```
$ manila shrink docs_resize 1
$ manila show docs_resize
+----------------------+------------------------------------------------------
↪---------------------+
| Property             | Value                                                ␣
↪                      |
+----------------------+------------------------------------------------------
↪---------------------+
| status               | available                                            ␣
↪                      |
| share_type_name      | my_type                                              ␣
↪                      |
| description          | None                                                 ␣
↪                      |
| availability_zone    | nova                                                 ␣
↪                      |
```

(continues on next page)

(continued from previous page)

```
| share_network_id     | None                                                 ␣
↪                      |
| export_locations     |                                                      ␣
↪                      |
|                      | path = 1.0.0.4:/shares/manila_share_b8afc508_8487_
↪442b_b170_ea65b07074a8 |
|                      | preferred = False                                    ␣
↪                      |
|                      | is_admin_only = False                                ␣
↪                      |
|                      | id = 3ffb76f4-92b9-4639-83fd-025bc3e302ff            ␣
↪                      |
|                      | share_instance_id = b8afc508-8487-442b-b170-
↪ea65b07074a8                  |
|                      | path = 2.0.0.3:/shares/manila_share_b8afc508_8487_
↪442b_b170_ea65b07074a8 |
|                      | preferred = False                                    ␣
↪                      |
|                      | is_admin_only = True                                 ␣
↪                      |
|                      | id = 1f0e263f-370d-47d3-95f6-1be64088b9da            ␣
↪                      |
|                      | share_instance_id = b8afc508-8487-442b-b170-
↪ea65b07074a8                  |
| share_server_id      | None                                                 ␣
↪                      |
| share_group_id       | None                                                 ␣
↪                      |
| host                 | manila@paris#shares                                  ␣
↪                      |
| access_rules_status  | active                                               ␣
↪                      |
| snapshot_id          | None                                                 ␣
↪                      |
| is_public            | False                                                ␣
↪                      |
| task_state           | None                                                 ␣
↪                      |
| snapshot_support     | True                                                 ␣
↪                      |
| id                   | b07dbebe-a328-403c-b402-c8871c89e3d1                 ␣
↪                      |
| size                 | 1                                                    ␣
↪                      |
| name                 | docs_resize                                          ␣
↪                      |
| share_type           | 14ee8575-aac2-44af-8392-d9c9d344f392                 ␣
↪                      |
| has_replicas         | False                                                ␣
↪                      |
```

(continues on next page)

(continued from previous page)

```
| replication_type      | None                                                 ␣
↪                       |
| created_at            | 2016-03-25T15:33:18.000000                           ␣
↪                       |
| share_proto           | NFS                                                  ␣
↪                       |
| project_id            | 907004508ef4447397ce6741a8f037c1                     ␣
↪                       |
| metadata              | {}                                                   ␣
↪                       |
+-----------------------+------------------------------------------------------
↪--------------------+
```

Quotas and limits

Limits

Limits are the resource limitations that are allowed for each project. An administrator can configure limits in the `manila.conf` file.

Users can query their rate and absolute limits.

To see the absolute limits, run:

```
$ manila absolute-limits
+----------------------------+-------+
| Name                       | Value |
+----------------------------+-------+
| maxTotalShareGigabytes     | 1000  |
| maxTotalShareNetworks      | 10    |
| maxTotalShareSnapshots     | 50    |
| maxTotalShares             | 50    |
| maxTotalSnapshotGigabytes  | 1000  |
| maxTotalShareReplicas      | 100   |
| maxTotalReplicaGigabytes   | 1000  |
| totalShareGigabytesUsed    | 1     |
| totalShareNetworksUsed     | 2     |
| totalShareSnapshotsUsed    | 1     |
| totalSharesUsed            | 1     |
| totalSnapshotGigabytesUsed | 1     |
| totalShareReplicasUsed     | 1     |
| totalReplicaGigabytesUsed  | 1     |
+----------------------------+-------+
```

Rate limits control the frequency at which users can issue specific API requests. Administrators use rate limiting to configure limits on the type and number of API calls that can be made in a specific time interval. For example, a rate limit can control the number of `GET` requests processed during a one-minute period.

To set the API rate limits, modify the `etc/manila/api-paste.ini` file, which is a part of the WSGI

pipeline and defines the actual limits. You need to restart `manila-api` service after you edit the `etc/manila/api-paste.ini` file.

```
[filter:ratelimit]
paste.filter_factory = manila.api.v1.limits:RateLimitingMiddleware.factory
limits = (POST, "*/shares", ^/shares, 120, MINUTE);(PUT, "*/shares", .*, 120,␣
↪MINUTE);(DELETE, "*", .*, 120, MINUTE)
```

Also, add the `ratelimit` to `noauth`, `keystone`, `keystone_nolimit` parameters in the `[composite:openstack_share_api]` and `[composite:openstack_share_api_v2]` groups.

```
[composite:openstack_share_api]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap ssl ratelimit sizelimit noauth api
keystone = cors faultwrap ssl ratelimit sizelimit authtoken keystonecontext␣
↪api
keystone_nolimit = cors faultwrap ssl ratelimit sizelimit authtoken␣
↪keystonecontext api

[composite:openstack_share_api_v2]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap ssl ratelimit sizelimit noauth apiv2
keystone = cors faultwrap ssl ratelimit sizelimit authtoken keystonecontext␣
↪apiv2
keystone_nolimit = cors faultwrap ssl ratelimit sizelimit authtoken␣
↪keystonecontext apiv2
```

To see the rate limits, run:

```
$ manila rate-limits
+--------+------------+-------+--------+--------+----------------------+
| Verb   | URI        | Value | Remain | Unit   | Next_Available       |
+--------+------------+-------+--------+--------+----------------------+
| DELETE | "*"        | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
| POST   | "*/shares" | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
| PUT    | "*/shares" | 120   | 120    | MINUTE | 2015-10-20T15:17:20Z |
+--------+------------+-------+--------+--------+----------------------+
```

Quotas

Quota sets provide quota management support.

To list the quotas for a project or user, use the **`manila quota-show`** command. If you specify the optional `--user` parameter, you get the quotas for this user in the specified project. If you omit this parameter, you get the quotas for the specified project.

Note: The Shared File Systems service does not perform mapping of usernames and project names to IDs. Provide only ID values to get correct setup of quotas. Setting it by names you set quota for nonexistent project/user. In case quota is not set explicitly by project/user ID, The Shared File Systems service just applies default quotas.

```
$ manila quota-show --tenant %project_id% --user %user_id%
+----------------------+----------------------------------+
| Property             | Value                            |
+----------------------+----------------------------------+
| id                   | d99c76b43b1743fd822d26ccc915989c |
| gigabytes            | 1000                             |
| per_share_gigabytes  | -1                               |
| snapshot_gigabytes   | 1000                             |
| snapshots            | 50                               |
| shares               | 50                               |
| share_networks       | 10                               |
| share_groups         | 50                               |
| share_group_snapshots| 50                               |
| share_replicas       | 100                              |
| replica_gigabytes    | 1000                             |
+----------------------+----------------------------------+
```

There are default quotas for a project that are set from the `manila.conf` file. To list the default quotas for a project, use the **`manila quota-defaults`** command:

```
$ manila quota-defaults --tenant %project_id%
+----------------------+----------------------------------+
| Property             | Value                            |
+----------------------+----------------------------------+
| id                   | 1cc2154937bd40f4815d5f168d372263 |
| gigabytes            | 1000                             |
| per_share_gigabytes  | -1                               |
| snapshot_gigabytes   | 1000                             |
| snapshots            | 50                               |
| shares               | 50                               |
| share_networks       | 10                               |
| share_groups         | 50                               |
| share_group_snapshots| 50                               |
| share_replicas       | 100                              |
| replica_gigabytes    | 1000                             |
+----------------------+----------------------------------+
```

The administrator can update the quotas for a specific project, or for a specific user by providing both the `--tenant` and `--user` optional arguments. It is possible to update the `shares`, `snapshots`, `gigabytes`, `snapshot-gigabytes`, `share-networks`, `share_groups`, `share_group_snapshots` and `share-type` quotas.

Note: Since API version 2.53, the administrator is also able to update quotas for share replicas and replica gigabytes by specifying `share_replicas` and/or `replica_gigabytes`. Since API version 2.62, the administrator is also able to update quotas for per share gigabytes by specifying `per_share_gigabytes`

```
$ manila quota-update %project_id% --user %user_id% --shares 49 --snapshots 49
```

As administrator, you can also permit or deny the force-update of a quota that is already used, or if the requested value exceeds the configured quota limit. To force-update a quota, use `force` optional key.

```
$ manila quota-update %project_id% --shares 51 --snapshots 51 --force
```

The administrator can also update the quotas for a specific share type. Share Type quotas cannot be set for individual users within a project. They can only be applied across all users of a particular project.

```
$ manila quota-update %project_id% --share-type %share_type_id%
```

To revert quotas to default for a project or for a user, delete quotas:

```
$ manila quota-delete --tenant %project_id% --user-id %user_id%
```

To revert quotas to default, use the specific project or share type. Share Type quotas can not be reverted for individual users within a project. They can only be reverted across all users of a particular project.

```
$ manila quota-delete --tenant %project_id% --share-type %share_type_id%
```

Share types

The Shared File System service back-end storage drivers offer a wide range of capabilities. The variation in these capabilities allows cloud administrators to provide a storage service catalog to their end users. Share types can be used to create this storage service catalog. Cloud administrators can influence provisioning of users shares with the help of Share types. All shares are associated with a share type. Share types are akin to `flavors` in the OpenStack Compute service (nova), or `volume types` in the OpenStack Block Storage service (cinder), or `storage classes` in Kubernetes. You can allow a share type to be accessible to all users in your cloud if you wish. You can also create private share types that allow only users belonging to certain OpenStack projects to access them. You can have an unlimited number of share types in your cloud, but for practical purposes, you may want to create only a handful of publicly accessible share types.

Each share type is an object that encompasses `extra-specs` (extra specifications). These extra-specs can map to storage back-end capabilities, or can be directives to the service.

Consider for example, offering three share types in your cloud to map to service levels:

Type	Capabilities/Instructions
Gold	Allow creating snapshots, reverting to snapshots and share replication, thick provision shares
Silver	Allow creating snapshots, thin provision shares
Bronze	Dont allow creating snapshots, thin provision shares

Capabilities or instructions such as the ones above are coded as extra-specs that your users and the Shared File System service understand. Users in OpenStack projects can see all public share types along with private share types that are made accessible to them. Not all extra-specs that you configure in a share type are visible to your users. This design helps preserve the cloud abstraction. Along with the share type names, they can see the share type descriptions and tenant-visible extra-specs.

For more details on extra-specs, see *Capabilities and Extra-Specs*.

The Shared File Systems service also allows using quota controls with share types. Quotas can help you maintain your SLAs by limiting the number of consumable resources or aid in billing. See *Quotas and limits* for more details.

Driver Handles Share Servers (DHSS)

To provide secure and hard multi-tenancy on the network data path, the Shared File Systems service allows users to use their own share networks. When shares are created on a share network, users can be sure they have their own isolated share servers that export their shares on the share network that have the ability plug into user-determined authentication domains (security services). Not all Shared File System service storage drivers support share networks. Those that do assert the capability `driver_handles_share_servers=True`.

When creating a share type, you are *required* to set an extra-spec that matches this capability. It is visible to end users.

Default Share Type

When you are operating a cloud where all your tenants are trusted, you may want to create a default share type that applies to all of them. It simplifies share creation for your end users since they dont need to worry about share types.

Use of a default share type is not recommended in a multi-tenant cloud where you may want to separate your user workloads, or offer different service capabilities. In such instances, you must always encourage your users to specify a share type at share creation time, and not rely on the default share type.

Important: If you do not create and configure a default share type, users *must* specify a valid share type during share creation, or share creation requests will fail.

To configure the default share type, edit the `manila.conf` file, and set the configuration option [DEFAULT]/default_share_type.

You must then create a share type, using **`manila type-create`**:

```
manila type-create [--is_public <is_public>]
                   [--description <description>]
                   [--extra-specs <other-extra-specs>]
                   <name> <spec_driver_handles_share_servers>
```

where:

- `name` is the share type name
- `is_public` defines the visibility for the share type (true/false)
- `description` is a free form text field to describe the characteristics of the share type for your users benefit
- `extra-specs` defines a comma separated set of key=value pairs of optional extra specifications
- `spec_driver_handles_share_servers` is the mandatory extra-spec (true/false)

Share type operations

To create a new share type you need to specify the name of the new share type. You also require an extra spec `driver_handles_share_servers`. The new share type can be public or private.

```
$ manila manila type-create default-shares False \
  --description "Default share type for the cloud, no fancy capabilities"

$ manila type-list
 +--------------------------------------+--------------------------------------+-
↪----------+------------+--------------------------------------+------------
↪------------------------------+-----------------------------------------------
↪-------------+
 | ID                                   | Name                                 |␣
↪visibility | is_default | required_extra_specs                 | optional_
↪extra_specs                         | Description                                  ␣
↪               |
 +--------------------------------------+--------------------------------------+-
↪----------+------------+--------------------------------------+------------
↪------------------------------+-----------------------------------------------
↪-------------+
 | cf1f92ec-4d0a-4b79-8f18-6bb82c22840a | default-shares                       |␣
↪public     | -          | driver_handles_share_servers : False |              ␣
↪                                    | Default share type for the cloud, no fancy␣
↪capabilities |
 +--------------------------------------+--------------------------------------+-
↪----------+------------+--------------------------------------+------------
↪------------------------------+-----------------------------------------------
↪-------------+

 $ manila type-show default-shares
 +----------------------+-----------------------------------------------------
↪----+
 | Property             | Value                                                ␣
↪    |
 +----------------------+-----------------------------------------------------
↪----+
 | id                   | cf1f92ec-4d0a-4b79-8f18-6bb82c22840a                 ␣
↪    |
 | name                 | default-shares                                       ␣
↪    |
 | visibility           | public                                               ␣
↪    |
 | is_default           | NO                                                   ␣
↪    |
 | description          | Default share type for the cloud, no fancy␣
↪capabilities |
 | required_extra_specs | driver_handles_share_servers : False                 ␣
↪    |
 | optional_extra_specs |                                                      ␣
↪    |
```

(continues on next page)

(continued from previous page)

```
 +----------------------+--------------------------------------------------------
↪----+
```

You did not provide optional capabilities, so they are all *assumed to be off by default*. So, Non-privileged users see some tenant-visible capabilities explicitly.

```
$ source demorc
$ manila type-list
+--------------------------------------+-------------------------------------+--
↪---------+------------+---------------------------------------+-------------
↪--------------------------------+-----------------------------------------------
↪-------------+
| ID                                   | Name                                |␣
↪visibility | is_default | required_extra_specs                  | optional_
↪extra_specs                          | Description                           ␣
↪                |
+--------------------------------------+-------------------------------------+--
↪---------+------------+---------------------------------------+-------------
↪--------------------------------+-----------------------------------------------
↪-------------+
| cf1f92ec-4d0a-4b79-8f18-6bb82c22840a | default-shares                      |␣
↪public     | -          | driver_handles_share_servers : False | snapshot_
↪support : False                      | Default share type for the cloud, no␣
↪fancy capabilities |
+--------------------------------------+-------------------------------------+--
↪---------+------------+---------------------------------------+-------------
↪--------------------------------+-----------------------------------------------
↪-------------+

$ manila type-show default-shares
+----------------------+-------------------------------------------------------
↪---+
| Property             | Value                                                 ␣
↪   |
+----------------------+-------------------------------------------------------
↪---+
| id                   | cf1f92ec-4d0a-4b79-8f18-6bb82c22840a                  ␣
↪   |
| name                 | default-shares                                        ␣
↪   |
| visibility           | public                                                ␣
↪   |
| is_default           | NO                                                    ␣
↪   |
| description          | Default share type for the cloud, no fancy␣
↪capabilities |
| required_extra_specs | driver_handles_share_servers : False                  ␣
↪   |
| optional_extra_specs | snapshot_support : False                              ␣
↪   |
```

(continues on next page)

(continued from previous page)

```
|                      | create_share_from_snapshot_support : False          |
↪    |
|                      | revert_to_snapshot_support : False                  |
↪    |
|                      | mount_snapshot_support : False                      |
↪    |
+----------------------+-----------------------------------------------------
↪---+
```

You can set or unset extra specifications for a share type using **manila type-key <share_type> set <key=value>** command.

```
$ manila type-key default-shares set snapshot_support=True

$ manila type-show default-shares
 +----------------------+-----------------------------------------------------
↪----+
 | Property             | Value                                               |
↪    |
 +----------------------+-----------------------------------------------------
↪----+
 | id                   | cf1f92ec-4d0a-4b79-8f18-6bb82c22840a                |
↪    |
 | name                 | default-shares                                      |
↪    |
 | visibility           | public                                              |
↪    |
 | is_default           | NO                                                  |
↪    |
 | description          | Default share type for the cloud, no fancy
↪capabilities |
 | required_extra_specs | driver_handles_share_servers : False                |
↪    |
 | optional_extra_specs | snapshot_support : True                             |
↪    |
 +----------------------+-----------------------------------------------------
↪----+
```

Use **manila type-key <share_type> unset <key>** to unset an extra specification.

A share type can be deleted with the **manila type-delete <share_type>** command. However, a share type can only be deleted if there are no shares, share groups or share group types associated with the share type.

Share type access control

You can provide access, revoke access, and retrieve list of allowed projects for a specified private share.

Create a private type:

```
$ manila type-create my_type1 True \
        --is_public False \
        --extra-specs snapshot_support=True
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| required_extra_specs | driver_handles_share_servers : True  |
| Name                 | my_type1                             |
| Visibility           | private                              |
| is_default           | -                                    |
| ID                   | 06793be5-9a79-4516-89fe-61188cad4d6c |
| optional_extra_specs | snapshot_support : True              |
+----------------------+--------------------------------------+
```

Note: If you run **manila type-list** only public share types appear. To see private share types, run **manila type-list --all`**.

Grant access to created private type for a demo and alt_demo projects by providing their IDs:

```
$ manila type-access-add my_type1 d8f9af6915404114ae4f30668a4f5ba7
$ manila type-access-add my_type1 e4970f57f1824faab2701db61ee7efdf
```

To view information about access for a private share, type `my_type1`:

```
$ manila type-access-list my_type1
+----------------------------------+
| Project_ID                       |
+----------------------------------+
| d8f9af6915404114ae4f30668a4f5ba7 |
| e4970f57f1824faab2701db61ee7efdf |
+----------------------------------+
```

After granting access to the share, the users in the allowed projects can see the share type and use it to create shares.

To deny access for a specified project, use **manila type-access-remove <share_type> <project_id>** command.

Share group types

Share group types are types for share groups just like *share types for shares*. A group type is associated with group specs similar to the way extra specs are associated with a share type.

A share group type aids the scheduler to filter or choose back ends when you create a share group and to set any backend specific parameters on the share group. Any driver that can perform a group operation in an advantaged way may report that as a group capability, such as:

- Ordered writes
- Consistent snapshots
- Group replication
- Group backup

Share group types may contain group specs corresponding to the group capabilities reported by the backends. A group capability applies across all the shares inside the share group, for example, a backend may support *consistent_snapshot_support*, and using this group type extra spec in the group type will allow scheduling share groups onto that backend. Any time a snapshot of the group is initiated, a crash consistent simultaneous snapshot of all the constituent shares is taken. Shares in a share group may each have different share types because they can each be on separate pools, have different capabilities and perhaps end users can even be billed differently for using each of them. To allow for this possibility, one or more share types can be associated with a group type. The admin also specifies which share type(s) a given group type may contain. At least one share type must be provided to create a share group type. When an user creates a share group, the scheduler creates the group on one of the backends that match the specified share type(s) and share group type.

In the Shared File Systems configuration file `manila.conf`, the administrator can set the share group type used by default for the share group creation.

To create a share group type, use **`manila share-group-type-create`** command as:

```
manila share-group-type-create [--is_public <is_public>]
                               [--group-specs [<key=value> [<key=value> ...]]]
                               <name> <share_types>
```

Where the `name` is the share group type name and `--is_public` defines the level of the visibility for the share group type. One share group can include multiple `share_types`. `--group-specs` are the extra specifications used to filter back ends.

Note: The extra specifications set in the share group types are explained further in *Scheduling*.

Administrators can create share group types with these extra specifications for the back ends filtering. An administrator can use the `policy.json` file to grant permissions for share group type creation with extra specifications to other roles.

You set a share group type to private or public and *manage the access* to the private share group types. By default a share group type is created as publicly accessible. Set `--is_public` to `False` to make the share group type private.

Share group type operations

To create a new share group type you need to specify the name of the new share group type and existing share types. The new share group type can also be public. One share group can include multiple share types.

```
$ manila share-group-type-create group_type_for_cg default_share_type --is_
↪public True
+-----------+--------------------------------------+
| Property  | Value                                |
+-----------+--------------------------------------+
| is_default | -                                   |
| ID        | cfe42f20-d13e-4348-9370-f0763e426db3 |
| Visibility | public                              |
| Name      | group_type_for_cg                    |
+-----------+--------------------------------------+

$ manila share-group-type-list
+--------------------------------------+------------------+------------+-----
↪-------+
| ID                                   | Name             | visibility | is_
↪default |
+--------------------------------------+------------------+------------+-----
↪-------+
| cfe42f20-d13e-4348-9370-f0763e426db3 | group_type_for_cg | public     | -  ␣
↪       |
+--------------------------------------+------------------+------------+-----
↪-------+
```

You can set or unset extra specifications for a share group type using **manila share-group-type-key <share_group_type> set <key=value>** command.

```
$ manila share-group-type-key group_type_for_cg set consistent_snapshot_
↪support=host
```

It is also possible to view a list of current share group types and extra specifications:

```
$ manila share-group-type-specs-list
+--------------------------------------+------------------+------------------
↪-----------------+
| ID                                   | Name             | all_extra_specs ␣
↪                 |
+--------------------------------------+------------------+------------------
↪-----------------+
| cfe42f20-d13e-4348-9370-f0763e426db3 | group_type_for_cg | consistent_
↪snapshot_support : host |
+--------------------------------------+------------------+------------------
↪-----------------+
```

Use **manila share-group-type-key <share_group_type> unset <key>** to unset an extra specification.

A public or private share group type can be deleted with the **manila share-group-type-delete <share_group_type>** command.

Share group type access

You can manage access to a private share group type for different projects. Administrators can provide access, revoke access, and retrieve information about access for a specified private share group.

Create a private group type:

```
$ manila share-group-type-create my_type1 default_share_type --is_public False
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| is_default | -                                    |
| ID         | f57cf3db-2503-4c0f-915c-4f1335d95465 |
| Visibility | private                              |
| Name       | my_type1                             |
+------------+--------------------------------------+
```

Note: If you run **manila share-group-type-list** only public share group types appear. To see private share group types, run **manila share-group-type-list** with `--all` optional argument.

Grant access to created private type for a demo and alt_demo projects by providing their IDs:

```
$ manila share-group-type-access-add my_type1 d8f9af6915404114ae4f30668a4f5ba7
$ manila share-group-type-access-add my_type1 e4970f57f1824faab2701db61ee7efdf
```

To view information about access for a private share, **manila type-access-list my_type1**:

```
$ manila type-access-list my_type1
+----------------------------------+
| Project_ID                       |
+----------------------------------+
| d8f9af6915404114ae4f30668a4f5ba7 |
| e4970f57f1824faab2701db61ee7efdf |
+----------------------------------+
```

After granting access to the share group type, the target project can see the share group type in the list, and create private share groups.

To deny access for a specified project, use **manila share-group-type-access-remove <share_group_type> <project_id>** command.

```
$ manila share-group-type-access-remove my_type1␣
↪e4970f57f1824faab2701db61ee7efdf
```

Share groups

Share group support is available in Manila since the Ocata release. A share group is a group of shares upon which users can perform group based operations, such as taking a snapshot together. This framework is meant to allow migrating or replicating a group of shares in unison in future releases of manila. Support currently exists for creating group types and group specs, creating groups of shares, and creating snapshots of groups. These group operations can be performed using the command line client.

To create a share group, and access it, the following general concepts are prerequisite knowledge:

1. To create a share group, use **`manila share-group-create`** command.
2. You can specify the `share-network`, *share group type*, `source-share-group-snapshot`, `availability-zone`, *share type*.
3. After the share group becomes available, use the **`manila create`** command to create a share within the share group.

Note: A share group is limited to a single backend, i.e. all shares created within a particular share group end up on the same backend. If the backend supports pools, the shares may be created within separate pools. So this feature is apt for those that would like co-locality of different shares.

Actions on a share group

A few actions, such as extend & shrink, are inherently applicable only to individual shares. One could theoretically apply extend to a group, increasing the size of each member, but this would not be a use-case covered initially. Any actions in this category must remain available to group members, and other actions such as taking snapshots of group members can be allowed, but actions such as migration or replication would be available only at the group level and not on its members.

Share Action	Share Group Action
Create (share type)	Create (share types, group type)
Delete	Delete (group)
Snapshot	Snapshot (may or may not be a consistent group snapshot)
Create from snapshot	Create from group snapshot
Clone	Clone group (and all members) (planned)
Replicate	Replicate (planned)
Migrate	Migrate (planned)
Extend/shrink	N/A

Creating a share with share group

Creating a share group type

In this example, we will create a new share group type and specify the *consistent_snapshot_support* as an group-spec within the share-group-type-create being used.

Use the **`manila type-list`** command to get a share type. Then use the share type to create a share group type.

```
$ manila type-list
+--------------------------------------+--------------------+------------+---
↪---------+--------------------------------------+--------------------------
↪--+
| ID                                   | Name               | visibility |␣
↪is_default | required_extra_specs                 | optional_extra_specs   ␣
↪    |
+--------------------------------------+--------------------+------------+---
↪---------+--------------------------------------+--------------------------
↪--+
| ee6287aa-448b-432b-a928-41ce9d8e149f | default_share_type | public     | -␣
↪         | driver_handles_share_servers : False |                          ␣
↪  |
+--------------------------------------+--------------------+------------+---
↪---------+--------------------------------------+--------------------------
↪--+
```

Use the **manila share-group-type-create** command to create a new share group type. Specify the name and share types.

```
$ manila share-group-type-create group_type_for_cg default_share_type
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| is_default | -                                    |
| ID         | cfe42f20-d13e-4348-9370-f0763e426db3 |
| Visibility | public                               |
| Name       | group_type_for_cg                    |
+------------+--------------------------------------+
```

Use the **manila share-group-type-key** command to set a group-spec to the share group type.

```
$ manila share-group-type-key group_type_for_cg set consistent_snapshot_
↪support=host
```

Note: This command has no output. To verify the group-spec, use the **manila share-group-type-specs-list** command and specify the share group types name or ID as a parameter.

Creating a share group

Use the **manila share-group-create** command to create a share group. Specify the share group type that we created.

```
$ manila share-group-create --share-group-type group_type_for_cg
+-------------------------------+--------------------------------------+
| Property                      | Value                                |
+-------------------------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| status                          | creating                             |
| description                     | None                                 |
| created_at                      | 2017-09-11T02:08:52.319921           |
| source_share_group_snapshot_id | None                                 |
| share_network_id                | None                                 |
| share_server_id                 | None                                 |
| host                            | None                                 |
| share_group_type_id             | cfe42f20-d13e-4348-9370-f0763e426db3 |
| project_id                      | 87ba30b5315c40ec8ec5e3346112eae4     |
| share_types                     | ee6287aa-448b-432b-a928-41ce9d8e149f |
| id                              | ecf78d45-546a-48df-a969-c153e68f0376 |
| name                            | None                                 |
+---------------------------------+--------------------------------------+
```

Note: One share group can include multiple share types. The share types are going to be inherited directly from the share group type.

Use the **manila share-group-show** command to retrieve details of the share. Specify the share ID or name as a parameter.

```
$ manila share-group-show ecf78d45-546a-48df-a969-c153e68f0376
+---------------------------------+------------------------------------------+
| Property                        | Value                                    |
+---------------------------------+------------------------------------------+
| status                          | available                                |
| description                     | None                                     |
| created_at                      | 2017-09-11T02:08:53.000000               |
| source_share_group_snapshot_id | None                                     |
| share_network_id                | None                                     |
| share_server_id                 | None                                     |
| host                            | ubuntu@generic2#test_pool                |
| share_group_type_id             | cfe42f20-d13e-4348-9370-f0763e426db3     |
| project_id                      | 87ba30b5315c40ec8ec5e3346112eae4         |
| share_types                     | ee6287aa-448b-432b-a928-41ce9d8e149f     |
| id                              | ecf78d45-546a-48df-a969-c153e68f0376     |
| name                            | None                                     |
+---------------------------------+------------------------------------------+
```

Create a share with the share group

Use the **manila create** command to create a share. Specify the share protocol, size, share group type and the share name.

```
$ manila create NFS 1 --share-group ecf78d45-546a-48df-a969-c153e68f0376 --
↪name test_group_share_1
+-------------------------------------+-----------------------------------
↪------+
```

(continues on next page)

(continued from previous page)

```
| Property                              | Value                                ␣
↪      |
+---------------------------------------+--------------------------------------
↪------+
| status                                | creating                             ␣
↪      |
| share_type_name                       | default_share_type                   ␣
↪      |
| description                           | None                                 ␣
↪      |
| availability_zone                     | None                                 ␣
↪      |
| share_network_id                      | None                                 ␣
↪      |
| share_server_id                       | None                                 ␣
↪      |
| share_group_id                        | ecf78d45-546a-48df-a969-
↪c153e68f0376      |
| host                                  | ubuntu@generic2#test_pool            ␣
↪      |
| 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                                    | 21997eaf-712e-433e-8872-
↪4ff085683657      |
| size                                  | 1                                    ␣
↪      |
| source_share_group_snapshot_member_id | None                                 ␣
↪      |
| user_id                               | b7f2c522a5644a83b78b3f61f50c6d71     ␣
↪      |
| name                                  | test_group_share_1                   ␣
↪      |
| share_type                            | ee6287aa-448b-432b-a928-
↪41ce9d8e149f      |
| has_replicas                          | False                                ␣
↪      |
| replication_type                      | None                                 ␣
↪      |
```

(continues on next page)

(continued from previous page)

```
| created_at                           | 2017-09-11T02:28:16.000000
↪       |
| share_proto                          | NFS
↪       |
| mount_snapshot_support               | False
↪       |
| project_id                           | 87ba30b5315c40ec8ec5e3346112eae4
↪       |
| metadata                             | {}
↪       |
+--------------------------------------+--------------------------------------
↪------+
```

Create another share with a same share group, and named test_group_share_2.

```
$ manila create NFS 1 --share-group ecf78d45-546a-48df-a969-c153e68f0376 --
↪name test_group_share_2
+--------------------------------------+--------------------------------------
↪------+
| Property                             | Value
↪       |
+--------------------------------------+--------------------------------------
↪------+
| status                               | creating
↪       |
| share_type_name                      | default_share_type
↪       |
| description                          | None
↪       |
| availability_zone                    | None
↪       |
| share_network_id                     | None
↪       |
| share_server_id                      | None
↪       |
| share_group_id                       | ecf78d45-546a-48df-a969-
↪c153e68f0376      |
| host                                 | ubuntu@generic2#test_pool
↪       |
| revert_to_snapshot_support           | False
↪       |
| access_rules_status                  | active
↪       |
| snapshot_id                          | None
↪       |
| create_share_from_snapshot_support   | False
↪       |
| is_public                            | False
↪       |
```

(continues on next page)

(continued from previous page)

```
| task_state                          | None                                 ␣
↪       |
| snapshot_support                    | False                                ␣
↪       |
| id                                  | 8d34a9a3-3b8c-4771-af2c-
↪66c78fe1e0b1      |
| size                                | 1                                    ␣
↪       |
| source_share_group_snapshot_member_id | None                                 ␣
↪       |
| user_id                             | b7f2c522a5644a83b78b3f61f50c6d71     ␣
↪       |
| name                                | test_group_share_2                   ␣
↪       |
| share_type                          | ee6287aa-448b-432b-a928-
↪41ce9d8e149f      |
| has_replicas                        | False                                ␣
↪       |
| replication_type                    | None                                 ␣
↪       |
| created_at                          | 2017-09-11T21:01:36.000000           ␣
↪       |
| share_proto                         | NFS                                  ␣
↪       |
| mount_snapshot_support              | False                                ␣
↪       |
| project_id                          | 87ba30b5315c40ec8ec5e3346112eae4     ␣
↪       |
| metadata                            | {}                                   ␣
↪       |
+-------------------------------------+--------------------------------------
↪------+
```

Creating a share group snapshot

Create a share group sanpshot of the share group

Use the **`manila share-group-snapshot-create`** command to create a share group snapshot. Specify the share group ID or name.

```
$ manila share-group-snapshot-create  ecf78d45-546a-48df-a969-c153e68f0376
+---------------+--------------------------------------+
| Property      | Value                                |
+---------------+--------------------------------------+
| status        | creating                             |
| name          | None                                 |
| created_at    | 2017-09-11T21:04:54.612737           |
| share_group_id | ecf78d45-546a-48df-a969-c153e68f0376 |
| project_id    | 87ba30b5315c40ec8ec5e3346112eae4     |
```

(continues on next page)

(continued from previous page)

```
| id              | ac387240-08dc-4b23-80f6-ffc481e6c87a |
| description     | None                                 |
+-----------------+--------------------------------------+
```

Show the members of the share group snapshot

Use the **manila share-group-snapshot-create** command to see all share members of share group snapshot. Specify the share group snapshot ID or name.

```
$ manila share-group-snapshot-list-members  ac387240-08dc-4b23-80f6-
↪ffc481e6c87a
+--------------------------------------+------+
| Share ID                             | Size |
+--------------------------------------+------+
| 21997eaf-712e-433e-8872-4ff085683657 | 1    |
| 8d34a9a3-3b8c-4771-af2c-66c78fe1e0b1 | 1    |
+--------------------------------------+------+
```

Show the details of the share group snapshot

```
$ manila share-group-snapshot-show ac387240-08dc-4b23-80f6-ffc481e6c87a
+----------------+--------------------------------------+
| Property       | Value                                |
+----------------+--------------------------------------+
| status         | available                            |
| name           | None                                 |
| created_at     | 2017-09-11T21:04:55.000000           |
| share_group_id | ecf78d45-546a-48df-a969-c153e68f0376 |
| project_id     | 87ba30b5315c40ec8ec5e3346112eae4     |
| id             | ac387240-08dc-4b23-80f6-ffc481e6c87a |
| description    | None                                 |
+----------------+--------------------------------------+
```

Deleting share groups

Use the **manila share-group-delete <group_id>** to delete share groups.

Deleting share group snapshots

Use the **manila share-group-snapshot-delete <group_snapshot_id>** to delete share a share group snapshot.

Important: Before attempting to delete a share group or a share group snapshot, make sure that all its constituent shares and snapshots were deleted. Users will need to delete share group snapshots before attempting to delete shares within ashare group or the group itself.

Share snapshots

The Shared File Systems service provides a snapshot mechanism to help users restore data by running the **manila snapshot-create** command.

To export a snapshot, create a share from it, then mount the new share to an instance. Copy files from the attached share into the archive.

To import a snapshot, create a new share with appropriate size, attach it to instance, and then copy a file from the archive to the attached file system.

Note: You cannot delete a share while it has saved dependent snapshots.

Create a snapshot from the share:

```
$ manila snapshot-create Share1 --name Snapshot1 --description "Snapshot of␣
↪Share1"
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | creating                             |
| share_id    | aca648eb-8c03-4394-a5cc-755066b7eb66 |
| user_id     | 5c7bdb6eb0504d54a619acf8375c08ce     |
| description | Snapshot of Share1                   |
| created_at  | 2015-09-25T05:27:38.000000           |
| size        | 1                                    |
| share_proto | NFS                                  |
| id          | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
| project_id  | cadd7139bc3148b8973df097c0911016     |
| share_size  | 1                                    |
| name        | Snapshot1                            |
+-------------+--------------------------------------+
```

Update snapshot name or description if needed:

```
$ manila snapshot-rename Snapshot1 Snapshot_1 --description "Snapshot of␣
↪Share1. Updated."
```

Check that status of a snapshot is `available`:

```
$ manila snapshot-show Snapshot1
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | available                            |
| share_id    | aca648eb-8c03-4394-a5cc-755066b7eb66 |
| user_id     | 5c7bdb6eb0504d54a619acf8375c08ce     |
| name        | Snapshot1                            |
| created_at  | 2015-09-25T05:27:38.000000           |
| share_proto | NFS                                  |
| id          | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
```

(continues on next page)

(continued from previous page)

```
| project_id  | cadd7139bc3148b8973df097c0911016     |
| size        | 1                                    |
| share_size  | 1                                    |
| description | Snapshot of Share1                   |
+-------------+--------------------------------------+
```

To create a copy of your data from a snapshot, use **manila create** with key `--snapshot-id`. This creates a new share from an existing snapshot. Create a share from a snapshot and check whether it is available:

```
$ manila create nfs 1 --name Share2 --metadata source=snapshot --description
↪"Share from a snapshot." --snapshot-id 962e8126-35c3-47bb-8c00-f0ee37f42ddd
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | None                                 |
| share_type_name             | default                              |
| description                 | Share from a snapshot.               |
| availability_zone           | None                                 |
| share_network_id            | None                                 |
| export_locations            | []                                   |
| share_server_id             | None                                 |
| share_group_id              | None                                 |
| host                        | None                                 |
| snapshot_id                 | 962e8126-35c3-47bb-8c00-f0ee37f42ddd |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
| id                          | b6b0617c-ea51-4450-848e-e7cff69238c7 |
| size                        | 1                                    |
| name                        | Share2                               |
| share_type                  | c0086582-30a6-4060-b096-a42ec9d66b86 |
| created_at                  | 2015-09-25T06:25:50.240417           |
| export_location             | None                                 |
| share_proto                 | NFS                                  |
| project_id                  | 20787a7ba11946adad976463b57d8a2f     |
| metadata                    | {u'source': u'snapshot'}             |
+-----------------------------+--------------------------------------+

$ manila show Share2
+-----------------------------+-----------------------------------------+
| Property                    | Value                                   |
+-----------------------------+-----------------------------------------+
| status                      | available                               |
| share_type_name             | default                                 |
| description                 | Share from a snapshot.                  |
| availability_zone           | nova                                    |
| share_network_id            | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a    |
| export_locations            | 10.254.0.3:/shares/share-1dc2a471-3d47-...|
```

(continues on next page)

(continued from previous page)

```
| share_server_id              | 41b7829d-7f6b-4c96-aea5-d106c2959961      |
| share_group_id               | None                                      |
| host                         | manila@generic1#GENERIC1                  |
| snapshot_id                  | 962e8126-35c3-47bb-8c00-f0ee37f42ddd      |
| is_public                    | False                                     |
| task_state                   | None                                      |
| snapshot_support             | True                                      |
| id                           | b6b0617c-ea51-4450-848e-e7cff69238c7      |
| size                         | 1                                         |
| name                         | Share2                                    |
| share_type                   | c0086582-30a6-4060-b096-a42ec9d66b86      |
| created_at                   | 2015-09-25T06:25:50.000000                |
| share_proto                  | NFS                                       |
| project_id                   | 20787a7ba11946adad976463b57d8a2f          |
| metadata                     | {u'source': u'snapshot'}                  |
+------------------------------+-------------------------------------------+
```

By default, the Shared File Systems service will place the new share in the source shares pool, unless a different destination availability zone is provided by the user, using the key `--availability-zone`.

Starting from Ussuri release, a new filter and weigher were added to the scheduler to enhance the selection of a destination pool when creating shares from snapshot. Drivers that support creating shares from snapshots across back ends also need the back end configuration option `replication_domain` to be specified. This option can be an arbitrary string. As an administrator, you are expected to determine which back ends are compatible to copy data between each other. Once you have identified these back ends, configure `replication_domain` in their respective configuration sections to the same string. Refer to the *feature support mapping* for identifying which back ends support this feature. The use of scheduler when creating share from a snapshot must be enabled using the configuration flag `[DEFAULT]/use_scheduler_creating_share_from_snapshot`. This option is disabled by default.

Note: When combining both `--snapshot-id` and `--availability-zone` keys, youll need to make sure that the configuration flag `[DEFAULT]/use_scheduler_creating_share_from_snapshot` is enabled, or the operation will be denied when source and destination availability zones are different.

You can soft-delete a snapshot using **`manila snapshot-delete <snapshot_name_or_ID>`**. If a snapshot is in busy state, and during the delete an `error_deleting` status appeared, administrator can force-delete it or explicitly reset the state.

Use **`snapshot-reset-state [--state <state>] <snapshot>`** to update the state of a snapshot explicitly. A valid value of a status are `available`, `error`, `creating`, `deleting`, `error_deleting`. If no state is provided, the `available` state will be used.

Use **`manila snapshot-force-delete <snapshot>`** to force-delete a specified share snapshot in any state.

Share servers

A share server is a resource created by the Shared File Systems service when the driver is operating in the *driver_handles_share_servers = True* mode. A share server exports users shares, manages their exports and access rules.

Share servers are abstracted away from end users. Drivers operating in *driver_handles_share_servers = True* mode manage the lifecycle of these share servers automatically. Administrators can however remove the share servers from the management of the Shared File Systems service without destroying them. They can also bring in existing share servers under the Shared File Systems service. They can list all available share servers and update their status attribute. They can delete an specific share server if it has no dependent shares.

Share server management

To `manage` a share server means that when the driver is operating in the `driver_handles_share_servers = True` mode, the administrator can bring a pre-existing share server under the management of the Shared File Systems service.

To `unmanage` means that the administrator is able to unregister an existing share server from the Shared File Systems service without deleting it from the storage back end. To be unmanaged, the referred share server cannot have any shares known to the Shared File Systems service.

Manage a share server

To bring a share server under the Shared File System service, use the **`manila share-server-manage`** command:

```
manila share-server-manage
    [--driver_options [<key=value> [<key=value> ...]]]
    [--share_network_subnet <share-network-subnet>]]
    <host> <share_network> <identifier>
```

The positional arguments are:

- host. The manage-share service host in `host@backend` format, which consists of the host name for the back end and the name of the back end.
- share_network. The share network where the share server is contained.
- identifier. The identifier of the share server on the back end storage.

The `driver_options` is an optional set of one or more driver-specific metadata items as key and value pairs. The specific key-value pairs necessary vary from driver to driver. Consult the driver-specific documentation to determine if any specific parameters must be supplied. Ensure that the share type has the `driver_handles_share_servers = True` extra-spec.

The `share_network_subnet` is an optional parameter which was introduced in Train release. Due to a change in the share networks structure, a share network no longer contains the following attributes: `neutron_net_id`, `neutron_subnet_id`, `gateway`, `mtu`, `network_type`, `ip_version`, `segmentation_id`. These attributes now pertain to the share network subnet entity, and a share network can span multiple share network subnets in different availability zones. If you do not specify a

share network subnet, the Shared File Systems Service will choose the default one (which does not pertain to any availability zone).

If using an OpenStack Networking (Neutron) based plugin, ensure that:

- There are some ports created, which correspond to the share server interfaces.
- The correct IP addresses are allocated to these ports.
- `manila:share` is set as the owner of these ports.

To manage a share server, run:

```
$ manila share-server-manage \
    manila@paris \
    share_net_test \
    backend_server_1 \
+--------------------+------------------------------------------+
| Property           | Value                                    |
+--------------------+------------------------------------------+
| id                 | 441d806f-f0e0-4c90-b7e2-a553c6aa76b2     |
| project_id         | 907004508ef4447397ce6741a8f037c1         |
| updated_at         | None                                     |
| status             | manage_starting                          |
| host               | manila@paris                             |
| share_network_name | share_net_test                           |
| share_network_id   | c895fe26-92be-4152-9e6c-f2ad230efb13     |
| created_at         | 2019-04-25T18:25:23.000000               |
| backend_details    | {}                                       |
| is_auto_deletable  | False                                    |
| identifier         | backend_server_1                         |
+--------------------+------------------------------------------+
```

Note: The `is_auto_deletable` property is used by the Shared File Systems service to identify a share server that can be deleted by internal routines.

The service can automatically delete share servers if there are no shares associated with them. To delete a share server when the last share is deleted, set the option: `delete_share_server_with_last_share`. If a scheduled cleanup is desired instead, `automatic_share_server_cleanup` and `unused_share_server_cleanup_interval` options can be set. Only one of the cleanup methods can be used at one time.

Any share server that has a share unmanaged from it cannot be automatically deleted by the Shared File Systems service. The same is true for share servers that have been managed into the service. Cloud administrators can delete such share servers manually if desired.

Unmanage a share server

To unmanage a share server, run **manila share-server-unmanage <share-server>**.

```
$ manila share-server-unmanage 441d806f-f0e0-4c90-b7e2-a553c6aa76b2
$ manila share-server-show 441d806f-f0e0-4c90-b7e2-a553c6aa76b2
ERROR: Share server 441d806f-f0e0-4c90-b7e2-a553c6aa76b2 could not be
found.
```

Reset the share server state

As administrator you are able to reset a share server state. To reset the state of a share server, run **manila share-server-reset-state <share-server> --state <state>**.

The positional arguments are:

- share-server. The share server name or id.
- **state. The state to be assigned to the share server. The options are:**
  - active
  - error
  - deleting
  - creating
  - managing
  - unmanaging
  - unmanage_error
  - manage_error

List share servers

To list share servers, run **manila share-server-list** command:

```
manila share-server-list [--host <hostname>] [--status <status>]
                         [--share-network <share_network>]
                         [--project-id <project_id>]
                         [--columns <columns>]
```

All the arguments above are optional. They can ben used to filter share servers. The options to filter:

- host. Shows all the share servers pertaining to the specified host.
- status. Shows all the share servers that are in the specified status.
- share_network. Shows all the share servers that pertain in the same share network.
- project_id. Shows all the share servers pertaining to the same project.
- columns. The administrator specifies which columns to display in the result of the list operation.

```
$ manila share-server-list
+--------------------------------------+--------------+--------+--------------
↪--+--------------------------------+------------+
| Id                                   | Host         | Status | Share␣
↪Network  | Project Id                     | Updated_at |
+--------------------------------------+--------------+--------+--------------
↪--+--------------------------------+------------+
| 441d806f-f0e0-4c90-b7e2-a553c6aa76b2 | manila@paris | active | share_net_
↪test | fd6d30efa5ff4c99834dc0d13f96e8eb | None       |
+--------------------------------------+--------------+--------+--------------
↪--+--------------------------------+------------+
```

Share server limits (Since Wallaby release)

Since Wallaby release, it is possible to specify limits for share servers size and amount of instances. It helps administrators to provision their resources in the cloud system and balance the share servers size. If a value is not configured, there is no behavioral change and manila will consider it as unlimited. Then, will reuse share servers regardless their size and amount of built instances.

- `max_share_server_size`: Maximum sum of gigabytes a share server can have considering all its share instances and snapshots.
- `max_shares_per_share_server`: Maximum number of share instances created in a share server.

Note: If one of these limits is reached during a request that requires a share server to be provided, manila will create a new share server to place such request.

Security services

A security service stores client configuration information used for authentication and authorization (AuthN/AuthZ). For example, a share server will be the client for an existing service such as LDAP, Kerberos, or Microsoft Active Directory.

You can associate a share with one to three security service types:

- `ldap`: LDAP.
- `kerberos`: Kerberos.
- `active_directory`: Microsoft Active Directory.

You can configure a security service with these options:

- A DNS IP address.
- An IP address or host name.
- A domain.
- A user or group name.
- The password for the user, if you specify a user name.

You can add the security service to the *share network*.

To create a security service, specify the security service type, a description of a security service, DNS IP address used inside projects network, security service IP address or host name, domain, security service user or group used by project, and a password for the user. The share name is optional.

Create a `ldap` security service:

```
$ manila security-service-create ldap --dns-ip 8.8.8.8 --server 10.254.0.3 --
↪name my_ldap_security_service
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | new                                  |
| domain      | None                                 |
| password    | None                                 |
| name        | my_ldap_security_service             |
| dns_ip      | 8.8.8.8                              |
| created_at  | 2015-09-25T10:19:06.019527           |
| updated_at  | None                                 |
| server      | 10.254.0.3                           |
| user        | None                                 |
| project_id  | 20787a7ba11946adad976463b57d8a2f     |
| type        | ldap                                 |
| id          | 413479b2-0d20-4c58-a9d3-b129fa592d8e |
| description | None                                 |
+-------------+--------------------------------------+
```

To create `kerberos` security service, run:

```
$ manila security-service-create kerberos --server 10.254.0.3 --user demo --
↪password secret --name my_kerberos_security_service --description "Kerberos␣
↪security service"
+-------------+--------------------------------------+
| Property    | Value                                |
+-------------+--------------------------------------+
| status      | new                                  |
| domain      | None                                 |
| password    | secret                               |
| name        | my_kerberos_security_service         |
| dns_ip      | None                                 |
| created_at  | 2015-09-25T10:26:03.211849           |
| updated_at  | None                                 |
| server      | 10.254.0.3                           |
| user        | demo                                 |
| project_id  | 20787a7ba11946adad976463b57d8a2f     |
| type        | kerberos                             |
| id          | 7f46a447-2534-453d-924d-bd7c8e63bbec |
| description | Kerberos security service            |
+-------------+--------------------------------------+
```

To see the list of created security service use **`manila security-service-list`**:

```
$ manila security-service-list
+--------------------------------------+-----------------------------+-------
↪-+----------+
| id                                   | name                        |␣
↪status | type     |
+--------------------------------------+-----------------------------+-------
↪-+----------+
| 413479b2-0d20-4c58-a9d3-b129fa592d8e | my_ldap_security_service    | new  ␣
↪ | ldap     |
| 7f46a447-2534-453d-924d-bd7c8e63bbec | my_kerberos_security_service | new  ␣
↪ | kerberos |
+--------------------------------------+-----------------------------+-------
↪-+----------+
```

You can add a security service to the existing *share network*, which is not yet used (a `share network` not associated with a share).

Add a security service to the share network with `share-network-security-service-add` specifying share network and security service. The command returns information about the security service. You can see view new attributes and `share_networks` using the associated share network ID.

```
$ manila share-network-security-service-add share_net2 my_ldap_security_
↪service

$ manila security-service-show my_ldap_security_service
+----------------+-----------------------------------------+
| Property       | Value                                   |
+----------------+-----------------------------------------+
| status         | new                                     |
| domain         | None                                    |
| password       | None                                    |
| name           | my_ldap_security_service                |
| dns_ip         | 8.8.8.8                                 |
| created_at     | 2015-09-25T10:19:06.000000              |
| updated_at     | None                                    |
| server         | 10.254.0.3                              |
| share_networks | [u'6d36c41f-d310-4aff-a0c2-ffd870e91cab'] |
| user           | None                                    |
| project_id     | 20787a7ba11946adad976463b57d8a2f        |
| type           | ldap                                    |
| id             | 413479b2-0d20-4c58-a9d3-b129fa592d8e    |
| description    | None                                    |
+----------------+-----------------------------------------+
```

It is possible to see the list of security services associated with a given share network. List security services for `share_net2` share network with:

```
$ manila share-network-security-service-list share_net2
+--------------------------------------+--------------------------+--------+--
↪----+
| id                                   | name                     | status |␣
↪type |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+--------------------------+--------+--
↪----+
| 413479b2-0d20-4c58-a9d3-b129fa592d8e | my_ldap_security_service | new    |␣
↪ldap |
+--------------------------------------+--------------------------+--------+--
↪----+
```

You also can dissociate a security service from the share network and confirm that the security service now has an empty list of share networks:

```
$ manila share-network-security-service-remove share_net2 my_ldap_security_
↪service

$ manila security-service-show my_ldap_security_service
+----------------+--------------------------------------+
| Property       | Value                                |
+----------------+--------------------------------------+
| status         | new                                  |
| domain         | None                                 |
| password       | None                                 |
| name           | my_ldap_security_service             |
| dns_ip         | 8.8.8.8                              |
| created_at     | 2015-09-25T10:19:06.000000           |
| updated_at     | None                                 |
| server         | 10.254.0.3                           |
| share_networks | []                                   |
| user           | None                                 |
| project_id     | 20787a7ba11946adad976463b57d8a2f     |
| type           | ldap                                 |
| id             | 413479b2-0d20-4c58-a9d3-b129fa592d8e |
| description    | None                                 |
+----------------+--------------------------------------+
```

The Shared File Systems service allows you to update a security service field using **manila security-service-update** command with optional arguments such as `--dns-ip`, `--server`, `--domain`, `--user`, `--password`, `--name`, or `--description`.

To remove a security service not associated with any share networks run:

```
$ manila security-service-delete my_ldap_security_service
```

Share migration

Share migration is the feature that migrates a share between different storage pools.

Use cases

As an administrator, you may want to migrate your share from one storage pool to another for several reasons. Examples include:

- Maintenance or evacuation
  - Evacuate a back end for hardware or software upgrades
  - Evacuate a back end experiencing failures
  - Evacuate a back end which is tagged end-of-life
- Optimization
  - Defragment back ends to empty and be taken offline to conserve power
  - Rebalance back ends to maximize available performance
  - Move data and compute closer together to reduce network utilization and decrease latency or increase bandwidth
- Moving shares
  - Migrate from old hardware generation to a newer generation
  - Migrate from one vendor to another

Migration workflows

Moving shares across different storage pools is generally expected to be a disruptive operation that disconnects existing clients when the source ceases to exist. For this reason, share migration is implemented in a 2-phase approach that allows the administrator to control the timing of the disruption. The first phase performs data copy while users retain access to the share. When copying is complete, the second phase may be triggered to perform a switchover that may include a last sync and deleting the source, generally requiring users to reconnect to continue accessing the share.

In order to migrate a share, one of two possible mechanisms may be employed, which provide different capabilities and affect how the disruption occurs with regards to user access during data copy phase and disconnection during switchover phase. Those two mechanisms are:

- Driver-assisted migration: This mechanism is intended to make use of driver optimizations to migrate shares between pools of the same storage vendor. This mechanism allows migrating shares nondisruptively while the source remains writable, preserving all filesystem metadata and snapshots. The migration workload is performed in the storage back end.
- Host-assisted migration: This mechanism is intended to migrate shares in an agnostic manner between two different pools, regardless of storage vendor. The implementation for this mechanism does not offer the same properties found in driver-assisted migration. In host-assisted migration, the source remains readable, snapshots must be deleted prior to starting the migration, filesystem metadata may be lost, and the clients will get disconnected by the end of migration. The migration

workload is performed by the Data Service, which is a dedicated manila service for intensive data operations.

When starting a migration, driver-assisted migration is attempted first. If the shared file system service detects it is not possible to perform the driver-assisted migration, it proceeds to attempt host-assisted migration.

Using the migration APIs

The commands to interact with the share migration API are:

- `migration_start`: starts a migration while retaining access to the share. Migration is paused and waits for `migration_complete` invocation when it has copied all data and is ready to take down the source share.

```
$ manila migration-start share_1 ubuntu@generic2#GENERIC2 --writable␣
↪False --preserve-snapshots False --preserve-metadata False --
↪nondisruptive False
```

Note: This command has no output.

- `migration_complete`: completes a migration, removing the source share and setting the destination share instance to `available`.

```
$ manila migration-complete share_1
```

Note: This command has no output.

- `migration_get_progress`: obtains migration progress information of a share.

```
$ manila migration-get-progress share_1

+----------------+--------------------------+
| Property       | Value                    |
+----------------+--------------------------+
| task_state     | data_copying_in_progress |
| total_progress | 37                       |
+----------------+--------------------------+
```

- `migration_cancel`: cancels an in-progress migration of a share.

```
$ manila migration-cancel share_1
```

Note: This command has no output.

The parameters

To start a migration, an administrator should specify several parameters. Among those, two of them are key for the migration.

- `share`: The share that will be migrated.
- `destination_pool`: The destination pool to which the share should be migrated to, in format host@backend#pool.

Several other parameters, referred to here as `driver-assisted parameters`, *must* be specified in the `migration_start` API. They are:

- `preserve_metadata`: whether preservation of filesystem metadata should be enforced for this migration.
- `preserve_snapshots`: whether preservation of snapshots should be enforced for this migration.
- `writable`: whether the source share remaining writable should be enforced for this migration.
- `nondisruptive`: whether it should be enforced to keep clients connected throughout the migration.

Specifying any of the boolean parameters above as `True` will disallow a host-assisted migration.

In order to appropriately move a share to a different storage pool, it may be required to change one or more share properties, such as the share type, share network, or availability zone. To accomplish this, use the optional parameters:

- `new_share_type_id`: Specify the ID of the share type that should be set in the migrated share.
- `new_share_network_id`: Specify the ID of the share network that should be set in the migrated share.

If driver-assisted migration should not be attempted, you may provide the optional parameter:

- `force_host_assisted_migration`: whether driver-assisted migration attempt should be skipped. If this option is set to `True`, all driver-assisted options must be set to `False`.

Configuration

For share migration to work in the cloud, there are several configuration requirements that need to be met:

For driver-assisted migration: it is necessary that the configuration of all back end stanzas is present in the file manila.conf of all manila-share nodes. Also, network connectivity between the nodes running manila-share service and their respective storage back ends is required.

For host-assisted migration: it is necessary that the Data Service (manila-data) is installed and configured in a node connected to the clouds administrator network. The drivers pertaining to the source back end and destination back end involved in the migration should be able to provide shares that can be accessed from the administrator network. This can easily be accomplished if the driver supports `admin_only` export locations, else it is up to the administrator to set up means of connectivity.

In order for the Data Service to mount the source and destination instances, it must use manila share access APIs to grant access to mount the instances. The access rule type varies according to the share protocol, so there are a few config options to set the access value for each type:

- data_node_access_ips: For IP-based access type, provide one or more administrator network IP addresses of the host running the Data Service. For NFS shares, drivers should always add rules with the no_root_squash property.
- data_node_access_cert: For certificate-based access type, provide the value of the certificate name that grants access to the Data Service.
- data_node_access_admin_user: For user-based access type, provide the value of a username that grants access and administrator privileges to the files in the share.
- data_node_mount_options: Provide the value of a mapping of protocol name to respective mount options. The Data Service makes use of mount command templates that by default have a dedicated field to inserting mount options parameter. The default value for this config option already includes the username and password parameters for CIFS shares and NFS v3 enforcing parameter for NFS shares.
- mount_tmp_location: Provide the value of a string representing the path where the share instances used in migration should be temporarily mounted. The default value is /tmp/.
- check_hash: This boolean config option value determines whether the hash of all files copied in migration should be validated. Setting this option increases the time it takes to migrate files, and is recommended for ultra-dependable systems. It defaults to disabled.

The configuration options above are respective to the Data Service only and should be defined the DEFAULT group of the manila.conf configuration file. Also, the Data Service node must have all the protocol-related libraries pre-installed to be able to run the mount commands for each protocol.

You may need to change some driver-specific configuration options from their default value to work with specific drivers. If so, they must be set under the driver configuration stanza in manila.conf. See a detailed description for each one below:

- migration_ignore_files: Provide value as a list containing the names of files or folders to be ignored during migration for a specific driver. The default value is a list containing only lost+found folder.
- share_mount_template: Provide a string that defines the template for the mount command for a specific driver. The template should contain the following entries to be formatted by the code:
  - proto: The share protocol. Automatically formatted by the Data Service.
  - options: The mount options to be formatted by the Data Service according to the data_node_mount_options config option.
  - export: The export path of the share. Automatically formatted by the Data Service with the shares admin_only export location.
  - path: The path to mount the share. Automatically formatted by the Data Service according to the mount_tmp_location config option.

  The default value for this config option is:

```
mount -vt %(proto)s %(options)s %(export)s %(path)s.
```

- share_unmount_template: Provide the value of a string that defines the template for the unmount command for a specific driver. The template should contain the path of where the shares are mounted, according to the mount_tmp_location config option, to be formatted automatically by the Data Service. The default value for this config option is:

```
umount -v %(path)s
```

- `protocol_access_mapping`: Provide the value of a mapping of access rule type to protocols supported. The default value specifies IP and user based access types mapped to NFS and CIFS respectively, which are the combinations supported by manila. If a certain driver uses a different protocol for IP or user access types, or is not included in the default mapping, it should be specified in this configuration option.

Other remarks

- There is no need to manually add any of the previously existing access rules after a migration is complete, they will be persisted on the destination after the migration.
- Once migration of a share has started, the user will see the status `migrating` and it will block other share actions, such as adding or removing access rules, creating or deleting snapshots, resizing, among others.
- The destination share instance export locations, although it may exist from the beginning of a host-assisted migration, are not visible nor accessible as access rules cannot be added.
- During a host-assisted migration, an access rule granting access to the Data Service will be added and displayed by querying the `access-list` API. This access rule should not be tampered with, it will otherwise cause migration to fail.
- Resources allocated are cleaned up automatically when a migration fails, except if this failure occurs during the 2nd phase of a driver-assisted migration. Each step in migration is saved to the field `task_state` present in the Share model. If for any reason the state is not set to `migration_error` during a failure, it will need to be reset using the `reset-task-state` API.
- It is advised that the node running the Data Service is well secured, since it will be mounting shares with highest privileges, temporarily exposing user data to whoever has access to this node.
- The two mechanisms of migration are affected differently by service restarts:
  - If performing a host-assisted migration, all services may be restarted except for the manila-data service when performing the copy (the `task_state` field value starts with `data_copying_`). In other steps of the host-assisted migration, both the source and destination manila-share services should not be restarted.
  - If performing a driver-assisted migration, the migration is affected minimally by driver restarts if the `task_state` is `migration_driver_in_progress`, while the copy is being done in the back end. Otherwise, the source and destination manila-share services should not be restarted.

Share replication

Replication of data has a number of use cases in the cloud. One use case is High Availability of the data in a shared file system, used for example, to support a production database. Another use case is ensuring Data Protection; i.e being prepared for a disaster by having a replication location that will be ready to back up your primary data source.

The Shared File System service supports user facing APIs that allow users to create shares that support replication, add and remove share replicas and manage their snapshots and access rules. Three replication types are currently supported and they vary in the semantics associated with the primary share and the secondary copies.

Important: **Share replication** is an **experimental** Shared File Systems API in the Mitaka release. Contributors can change or remove the experimental part of the Shared File Systems API in further releases without maintaining backward compatibility. Experimental APIs have an `X-OpenStack-Manila-API-Experimental: true` header in their HTTP requests.

Replication types supported

Before using share replication, make sure the Shared File System driver that you are running supports this feature. You can check it in the `manila-scheduler` service reports. The `replication_type` capability reported can have one of the following values:

writable The driver supports creating `writable` share replicas. All share replicas can be accorded read/write access and would be synchronously mirrored.

readable The driver supports creating `read-only` share replicas. All secondary share replicas can be accorded read access. Only the primary (or `active` share replica) can be written into.

dr The driver supports creating `dr` (abbreviated from Disaster Recovery) share replicas. A secondary share replica is inaccessible until after a `promotion`.

None The driver does not support Share Replication.

Note: The term `active` share replica refers to the `primary` share. In `writable` style of replication, all share replicas are `active`, and there could be no distinction of a `primary` share. In `readable` and `dr` styles of replication, a `secondary` share replica may be referred to as `passive`, `non-active` or simply, `replica`.

Configuration

Two new configuration options have been introduced to support Share Replication.

replica_state_update_interval Specify this option in the `DEFAULT` section of your `manila.conf`. The Shared File Systems service requests periodic update of the *replica_state* of all `non-active` share replicas. The update occurs with respect to an interval corresponding to this option. If it is not specified, it defaults to 300 seconds.

replication_domain Specify this option in the backend stanza when using a multi-backend style configuration. The value can be any ASCII string. Two backends that can replicate between each other

would have the same `replication_domain`. This comes from the premise that the Shared File Systems service expects Share Replication to be performed between symmetric backends. This option is *required* for using the Share Replication feature.

Health of a share replica

Apart from the `status` attribute, share replicas have the `replica_state` attribute to denote the state of data replication on the storage backend. The `primary` share replica will have its *replica_state* attribute set to *active*. The `secondary` share replicas may have one of the following as their `replica_state`:

in_sync The share replica is up to date with the `active` share replica (possibly within a backend-specific `recovery point objective`).

out_of_sync The share replica is out of date (all new share replicas start out in this `replica_state`).

error When the scheduler fails to schedule this share replica or some potentially irrecoverable error occurred with regard to updating data for this replica.

Promotion or failover

For `readable` and `dr` types of replication, we refer to the task of switching a *non-active* share replica with the `active` replica as *promotion*. For the `writable` style of replication, promotion does not make sense since all share replicas are `active` (or writable) at all times.

The *status* attribute of the non-active replica being promoted will be set to `replication_change` during its promotion. This has been classified as a `busy` state and thus API interactions with the share are restricted while one of its share replicas is in this state.

Share replication workflows

The following examples have been implemented with the ZFSonLinux driver that is a reference implementation in the Shared File Systems service. It operates in `driver_handles_share_servers=False` mode and supports the `readable` type of replication. In the example, we assume a configuration of two Availability Zones[1], called *availability_zone_1* and *availability_zone_2*.

Since the Train release, some drivers operating in `driver_handles_share_server=True` mode support share replication.

Multiple availability zones are not necessary to use the replication feature. However, the use of an availability zone as a `failure domain` is encouraged.

Pay attention to the network configuration for the ZFS driver. Here, we assume a configuration of `zfs_service_ip` and `zfs_share_export_ip` from two separate networks. The service network is reachable from the host where the `manila-share` service is running. The share export IP is from a network that allows user access.

See Configuring the ZFSonLinux driver for information on how to set up the ZFSonLinux driver.

[1] When running in a multi-backend configuration, until the Stein release, deployers could only configure one Availability Zone per manila configuration file. This is achieved with the option `storage_availability_zone` defined under the `[DEFAULT]` section.

Beyond the Stein release, the option `backend_availability_zone` can be specified in each back end stanza. The value of this configuration option will override any configuration of the `storage_availability_zone` from the `[DEFAULT]` section.

Creating a share that supports replication

Create a new share type and specify the *replication_type* as an extra-spec within the share-type being used.

Use the **manila type-create** command to create a new share type. Specify the name and the value for the extra-spec `driver_handles_share_servers`.

```
$ manila type-create readable_type_replication False
+----------------------+--------------------------------------+
| Property             | Value                                |
+----------------------+--------------------------------------+
| required_extra_specs | driver_handles_share_servers : False |
| Name                 | readable_type_replication            |
| Visibility           | public                               |
| is_default           | -                                    |
| ID                   | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074 |
| optional_extra_specs | snapshot_support : True              |
+----------------------+--------------------------------------+
```

Use the **manila type-key** command to set an extra-spec to the share type.

```
$ manila type-key readable_type_replication set replication_type=readable
```

Note: This command has no output. To verify the extra-spec, use the **manila extra-specs-list** command and specify the share types name or ID as a parameter.

Create a share with the share type

Use the **manila create** command to create a share. Specify the share protocol, size and the availability zone.

```
$ manila create NFS 1 --share_type readable_type_replication --name my_share -
↪-description "This share will have replicas" --az availability_zone_1
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | creating                             |
| share_type_name             | readable_type_replication            |
| description                 | This share will have replicas        |
| availability_zone           | availability_zone_1                  |
| share_network_id            | None                                 |
| share_server_id             | None                                 |
| share_group_id              | None                                 |
| host                        |                                      |
| access_rules_status         | active                               |
| snapshot_id                 | None                                 |
| is_public                   | False                                |
| task_state                  | None                                 |
| snapshot_support            | True                                 |
```

(continues on next page)

(continued from previous page)

```
| id                           | e496ed61-8f2e-436b-b299-32c3e90991cc |
| size                         | 1                                    |
| name                         | my_share                             |
| share_type                   | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074 |
| has_replicas                 | False                                |
| replication_type             | readable                             |
| created_at                   | 2016-03-29T20:22:18.000000           |
| share_proto                  | NFS                                  |
| project_id                   | 48a5ca76ac69405e99dc1c13c5195186     |
| metadata                     | {}                                   |
+------------------------------+--------------------------------------+
```

Note: If you are creating a share with the share type specification `driver_handles_share_servers=True`, the share network parameter is required for the operation to be performed.

Use the **`manila show`** command to retrieve details of the share. Specify the share ID or name as a parameter.

```
$ manila show my_share
+------------------------------+-----------------------------------------------
↪---------------------+
| Property                     | Value                                        ␣
↪                     |
+------------------------------+-----------------------------------------------
↪---------------------+
| status                       | available                                    ␣
↪                     |
| share_type_name              | readable_type_replication                    ␣
↪                     |
| description                  | This share will have replicas                ␣
↪                     |
| availability_zone            | availability_zone_1                          ␣
↪                     |
| share_network_id             | None                                         ␣
↪                     |
| export_locations             |                                              ␣
↪                     |
|                              | path =                                       ␣
↪                     |
|                              |10.32.62.26:/alpha/manila_share_38efc042_50c2_
↪4825_a6d8_cba2a8277b28|
|                              | preferred = False                            ␣
↪                     |
|                              | is_admin_only = False                        ␣
↪                     |
|                              | id = e1d754b5-ec06-42d2-afff-3e98c0013faf    ␣
↪                     |
```

(continues on next page)

(continued from previous page)

```
|                              | share_instance_id = 38efc042-50c2-4825-a6d8-
↪cba2a8277b28            |
|                              | path =                                        ␣
↪                        |
|                              |172.21.0.23:/alpha/manila_share_38efc042_50c2_
↪4825_a6d8_cba2a8277b28|
|                              | preferred = False                             ␣
↪                        |
|                              | is_admin_only = True                          ␣
↪                        |
|                              | id = 6f843ecd-a7ea-4939-86de-e1e01d9e8672     ␣
↪                        |
|                              | share_instance_id = 38efc042-50c2-4825-a6d8-
↪cba2a8277b28            |
| share_server_id              | None                                          ␣
↪                        |
| share_group_id               | None                                          ␣
↪                        |
| host                         | openstack4@zfsonlinux_1#alpha                 ␣
↪                        |
| access_rules_status          | active                                        ␣
↪                        |
| snapshot_id                  | None                                          ␣
↪                        |
| is_public                    | False                                         ␣
↪                        |
| task_state                   | None                                          ␣
↪                        |
| snapshot_support             | True                                          ␣
↪                        |
| id                           | e496ed61-8f2e-436b-b299-32c3e90991cc          ␣
↪                        |
| size                         | 1                                             ␣
↪                        |
| name                         | my_share                                      ␣
↪                        |
| share_type                   | 3b3ee3f7-6e43-4aa1-859d-0b0511c43074          ␣
↪                        |
| has_replicas                 | False                                         ␣
↪                        |
| replication_type             | readable                                      ␣
↪                        |
| created_at                   | 2016-03-29T20:22:18.000000                    ␣
↪                        |
| share_proto                  | NFS                                           ␣
↪                        |
| project_id                   | 48a5ca76ac69405e99dc1c13c5195186              ␣
↪                        |
| metadata                     | {}                                            ␣
↪                        |
```

(continues on next page)

(continued from previous page)

```
+----------------------------+-------------------------------------------------
↪--------------------+
```

Note: When you create a share that supports replication, an `active` replica is created for you. You can verify this with the **`manila share-replica-list`** command.

From API version 2.53, when creating a replicated share, the manila quota system will reserve and consume resources for two additional quotas: `share_replicas` and `replica_gigabytes`.

Creating and promoting share replicas

Create a share replica

Use the **`manila share-replica-create`** command to create a share replica. Specify the share ID or name as a parameter. You may optionally provide the *availability_zone*.

```
$ manila share-replica-create my_share --az availability_zone_2
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_2                  |
| created_at        | 2016-03-29T20:24:53.148992           |
| updated_at        | None                                 |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
| host              |                                      |
| replica_state     | None                                 |
| id                | 78a5ef96-6c36-42e0-b50b-44efe7c1807e |
+-------------------+--------------------------------------+
```

See details of the newly created share replica

Note: Since API version 2.51 (Train release), a share network is able to span multiple subnets in different availability zones. So, when using a share type with specification `driver_handles_share_servers=True`, users must ensure that the share network has a subnet in the availability zone that they desire the share replica to be created in.

Use the **`manila share-replica-show`** command to see details of the newly created share replica. Specify the share replicas ID as a parameter.

```
$ manila share-replica-show 78a5ef96-6c36-42e0-b50b-44efe7c1807e
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | available                            |
```

(continues on next page)

(continued from previous page)

```
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_2                  |
| created_at        | 2016-03-29T20:24:53.000000           |
| updated_at        | 2016-03-29T20:24:58.000000           |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
| host              | openstack4@zfsonlinux_2#beta         |
| replica_state     | in_sync                              |
| id                | 78a5ef96-6c36-42e0-b50b-44efe7c1807e |
+-------------------+--------------------------------------+
```

See all replicas of the share

Use the **`manila share-replica-list`** command to see all the replicas of the share. Specify the share ID or name as an optional parameter.

```
$ manila share-replica-list --share-id my_share
+--------------------------------------+-----------+---------------+----------
↪----------------------------+--------------------------------+--------------
↪------+----------------------------+
| ID                                   | Status    | Replica State | Share ID␣
↪                            | Host                           | Availability␣
↪Zone    | Updated At                 |
+--------------------------------------+-----------+---------------+----------
↪----------------------------+--------------------------------+--------------
↪------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available | active        | e496ed61-
↪8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha | availability_
↪zone_1 | 2016-03-29T20:22:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | in_sync       | e496ed61-
↪8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  | availability_
↪zone_2 | 2016-03-29T20:24:58.000000 |
+--------------------------------------+-----------+---------------+----------
↪----------------------------+--------------------------------+--------------
↪------+----------------------------+
```

Promote the secondary share replica to be the new active replica

Use the **`manila share-replica-promote`** command to promote a non-active share replica to become the `active` replica. Specify the non-active replicas ID as a parameter.

```
$ manila share-replica-promote 78a5ef96-6c36-42e0-b50b-44efe7c1807e
```

Note: This command has no output.

The promotion may take time. During the promotion, the `replica_state` attribute of the share replica being promoted will be set to `replication_change`.

```
$ manila share-replica-list --share-id my_share
+--------------------------------------+-----------+--------------------+-----
```

(continues on next page)

```
↪----------------------------------+----------------------------+
↪-----------+----------------------------+
```

(continued from previous page)

```
| ID                                   | Status    |    Replica State    |␣
↪Share ID                              | Host                              |␣
↪Availability Zone    | Updated At                 |
+--------------------------------------+-----------+---------------------+-----
↪----------------------------------+-----------------------------------+----------
↪----------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available |       active        |␣
↪e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha |␣
↪availability_zone_1 | 2016-03-29T20:32:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | replication_change |␣
↪e496ed61-8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  |␣
↪availability_zone_2 | 2016-03-29T20:32:19.000000 |
+--------------------------------------+-----------+---------------------+-----
↪----------------------------------+-----------------------------------+----------
↪----------+----------------------------+
```

Once the promotion is complete, the `replica_state` will be set to `active`.

```
$ manila share-replica-list --share-id my_share
+--------------------------------------+-----------+---------------+----------
↪----------------------------+-----------------------------------+---------------
↪------+----------------------------+
| ID                                   | Status    | Replica State | Share ID␣
↪                          | Host                              | Availability␣
↪Zone    | Updated At                 |
+--------------------------------------+-----------+---------------+----------
↪----------------------------+-----------------------------------+---------------
↪------+----------------------------+
| 38efc042-50c2-4825-a6d8-cba2a8277b28 | available | in_sync       | e496ed61-
↪8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_1#alpha | availability_
↪zone_1 | 2016-03-29T20:32:19.000000 |
| 78a5ef96-6c36-42e0-b50b-44efe7c1807e | available | active        | e496ed61-
↪8f2e-436b-b299-32c3e90991cc | openstack4@zfsonlinux_2#beta  | availability_
↪zone_2 | 2016-03-29T20:32:19.000000 |
+--------------------------------------+-----------+---------------+----------
↪----------------------------+-----------------------------------+---------------
↪------+----------------------------+
```

Access rules

Create an IP access rule for the share

Use the **`manila access-allow`** command to add an access rule. Specify the share ID or name, protocol and the target as parameters.

```
$ manila access-allow my_share ip 0.0.0.0/0 --access-level rw
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| share_id     | e496ed61-8f2e-436b-b299-32c3e90991cc |
| access_type  | ip                                   |
| access_to    | 0.0.0.0/0                            |
| access_level | rw                                   |
| state        | new                                  |
| id           | 8b339cdc-c1e0-448f-bf6d-f068ee6e8f45 |
+--------------+--------------------------------------+
```

Note: Access rules are not meant to be different across the replicas of the share. However, as per the type of replication, drivers may choose to modify the access level prescribed. In the above example, even though read/write access was requested for the share, the driver will provide read-only access to the non-active replica to the same target, because of the semantics of the replication type: `readable`. However, the target will have read/write access to the (currently) non-active replica when it is promoted to become the `active` replica.

The **`manila access-deny`** command can be used to remove a previously applied access rule.

List the export locations of the share

Use the **`manila share-export-locations-list`** command to list the export locations of a share.

```
$ manila share-export-location-list my_share
+--------------------------------------+--------------------------------------
↪--------------------------------------+-----------+
| ID                                   | Path                                 
↪                                      | Preferred |
+--------------------------------------+--------------------------------------
↪--------------------------------------+-----------+
| 3ed3fbf5-2fa1-4dc0-8440-a0af72398cb6 | 10.32.62.21:/beta/subdir/manila_
↪share_78a5ef96_6c36_42e0_b50b_44efe7c1807e| False     |
| 6f843ecd-a7ea-4939-86de-e1e01d9e8672 | 172.21.0.23:/alpha/manila_share_
↪38efc042_50c2_4825_a6d8_cba2a8277b28      | False     |
| e1d754b5-ec06-42d2-afff-3e98c0013faf | 10.32.62.26:/alpha/manila_share_
↪38efc042_50c2_4825_a6d8_cba2a8277b28      | False     |
| f3c5585f-c2f7-4264-91a7-a4a1e754e686 | 172.21.0.29:/beta/subdir/manila_
↪share_78a5ef96_6c36_42e0_b50b_44efe7c1807e| False     |
+--------------------------------------+--------------------------------------
↪--------------------------------------+-----------+
```

Identify the export location corresponding to the share replica on the user accessible network and you may mount it on the target node.

Note: As an administrator, you can list the export locations for a particular share replica by using the **`manila share-instance-export-location-list`** command and specifying the share replicas ID as a parameter.

Snapshots

Create a snapshot of the share

Use the **`manila snapshot-create`** command to create a snapshot of the share. Specify the share ID or name as a parameter.

```
$ manila snapshot-create my_share --name "my_snapshot"
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | creating                             |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| user_id           | 5c7bdb6eb0504d54a619acf8375c08ce     |
| description       | None                                 |
| created_at        | 2016-03-29T21:14:03.000000           |
| share_proto       | NFS                                  |
| provider_location | None                                 |
| id                | 06cdccaf-93a0-4e57-9a39-79fb1929c649 |
| project_id        | cadd7139bc3148b8973df097c0911016     |
| size              | 1                                    |
| share_size        | 1                                    |
| name              | my_snapshot                          |
+-------------------+--------------------------------------+
```

Show the details of the snapshot

Use the **`manila snapshot-show`** to view details of a snapshot. Specify the snapshot ID or name as a parameter.

```
$ manila snapshot-show my_snapshot
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | available                            |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| user_id           | 5c7bdb6eb0504d54a619acf8375c08ce     |
| description       | None                                 |
| created_at        | 2016-03-29T21:14:03.000000           |
| share_proto       | NFS                                  |
| provider_location | None                                 |
| id                | 06cdccaf-93a0-4e57-9a39-79fb1929c649 |
| project_id        | cadd7139bc3148b8973df097c0911016     |
| size              | 1                                    |
| share_size        | 1                                    |
| name              | my_snapshot                          |
+-------------------+--------------------------------------+
```

Note: The `status` attribute of a snapshot will transition from `creating` to `available` only when it is present on all the share replicas that have their `replica_state` attribute set to `active` or `in_sync`.

Likewise, the `replica_state` attribute of a share replica will transition from `out_of_sync` to `in_sync`

only when all `available` snapshots are present on it.

Planned failovers

As an administrator, you can use the **`manila share-replica-resync`** command to attempt to sync data between `active` and `non-active` share replicas of a share before promotion. This will ensure that share replicas have the most up-to-date data and their relationships can be safely switched.

```
$ manila share-replica-resync 38efc042-50c2-4825-a6d8-cba2a8277b28
```

Note: This command has no output.

Updating attributes

If an error occurs while updating data or replication relationships (during a `promotion`), the Shared File Systems service may not be able to determine the consistency or health of a share replica. It may require administrator intervention to make any fixes on the storage backend as necessary. In such a situation, state correction within the Shared File Systems service is possible.

As an administrator, you can:

Reset the `status` attribute of a share replica

Use the **`manila share-replica-reset-state`** command to reset the `status` attribute. Specify the share replicas ID as a parameter and use the `--state` option to specify the state intended.

```
$ manila share-replica-reset-state 38efc042-50c2-4825-a6d8-cba2a8277b28 --
↪state available
```

Note: This command has no output.

Reset the `replica_state` attribute

Use the **`manila share-replica-reset-replica-state`** command to reset the `replica_state` attribute. Specify the share replicas ID and use the `--state` option to specify the state intended.

```
$ manila share-replica-reset-replica-state 38efc042-50c2-4825-a6d8-
↪cba2a8277b28 --state out_of_sync
```

Note: This command has no output.

Force delete a specified share replica in any state

Use the **`manila share-replica-delete`** command with the force key to remove the share replica, regardless of the state it is in.

```
$ manila share-replica-show 9513de5d-0384-4528-89fb-957dd9b57680
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| status            | error                                |
| share_id          | e496ed61-8f2e-436b-b299-32c3e90991cc |
| availability_zone | availability_zone_1                  |
| created_at        | 2016-03-30T01:32:47.000000           |
| updated_at        | 2016-03-30T01:34:25.000000           |
| share_network_id  | None                                 |
| share_server_id   | None                                 |
| host              | openstack4@zfsonlinux_1#alpha        |
| replica_state     | out_of_sync                          |
| id                | 38efc042-50c2-4825-a6d8-cba2a8277b28 |
+-------------------+--------------------------------------+

$ manila share-replica-delete --force 38efc042-50c2-4825-a6d8-cba2a8277b28
```

Note: This command has no output.

Use the `policy.yaml` file to grant permissions for these actions to other roles.

Deleting share replicas

Use the **`manila share-replica-delete`** command with the share replicas ID to delete a share replica.

```
$ manila share-replica-delete 38efc042-50c2-4825-a6d8-cba2a8277b28
```

Note: This command has no output.

Note: You cannot delete the last `active` replica with this command. You should use the **`manila delete`** command to remove the share.

Multi-storage configuration

The Shared File Systems service can provide access to one or more file storage back ends. In general, the workflow with multiple back ends looks similar to the Block Storage service one.

Using `manila.conf`, you can spawn multiple share services. To do it, you should set the *enabled_share_backends* flag in the `manila.conf` file. This flag defines the comma-separated names of the configuration stanzas for the different back ends. One name is associated to one configuration group for a back end.

The following example runs three configured share services:

```
[DEFAULT]
enabled_share_backends=backendEMC1,backendGeneric1,backendNetApp

[backendGeneric1]
share_driver=manila.share.drivers.generic.GenericShareDriver
share_backend_name=one_name_for_two_backends
service_instance_user=ubuntu_user
service_instance_password=ubuntu_user_password
service_image_name=ubuntu_image_name
path_to_private_key=/home/foouser/.ssh/id_rsa
path_to_public_key=/home/foouser/.ssh/id_rsa.pub

[backendEMC1]
share_driver=manila.share.drivers.emc.driver.EMCShareDriver
share_backend_name=backendEMC2
emc_share_backend=vnx
emc_nas_server=1.1.1.1
emc_nas_password=password
emc_nas_login=user
emc_nas_server_container=server_3
emc_nas_pool_name="Pool 2"

[backendNetApp]
share_driver = manila.share.drivers.netapp.common.NetAppDriver
driver_handles_share_servers = True
share_backend_name=backendNetApp
netapp_login=user
netapp_password=password
netapp_server_hostname=1.1.1.1
netapp_root_volume_aggregate=aggr01
```

To spawn separate groups of share services, you can use separate configuration files. If it is necessary to control each back end in a separate way, you should provide a single configuration file per each back end.

Scheduling

The Shared File Systems service uses a scheduler to provide unified access for a variety of different types of shared file systems. The scheduler collects information from the active shared services, and makes decisions such as what shared services will be used to create a new share. To manage this process, the Shared File Systems service provides Share types API.

A share type is a list from key-value pairs called extra-specs. The scheduler uses required and un-scoped extra-specs to look up the shared service most suitable for a new share with the specified share type. For more information about extra-specs and their type, see Capabilities and Extra-Specs section in developer documentation.

The general scheduler workflow:

1. Share services report information about their existing pool number, their capacities, and their capabilities.

2. When a request on share creation arrives, the scheduler picks a service and pool that best serves the request, using share type filters and back end capabilities. If back end capabilities pass through, all filters request the selected back end where the target pool resides.
3. The share driver receives a reply on the request status, and lets the target pool serve the request as the scheduler instructs. The scoped and un-scoped share types are available for the driver implementation to use as needed.

Manage shares services

The Shared File Systems service provides API that allows to manage running share services (Share services API). Using the **`manila service-list`** command, it is possible to get a list of all kinds of running services. To select only share services, you can pick items that have field `binary` equal to `manila-share`. Also, you can enable or disable share services using raw API requests. Disabling means that share services are excluded from the scheduler cycle and new shares will not be placed on the disabled back end. However, shares from this service stay available.

Networking

Unlike the OpenStack Block Storage service, the Shared File Systems service must connect to the Networking service. The share service requires the option to self-manage share servers. For client authentication and authorization, you can configure the Shared File Systems service to work with different network authentication services, like LDAP, Kerberos protocols, or Microsoft Active Directory.

Share networks

Share networks are essential to allow end users a path to hard multi-tenancy. When backed by isolated networks, the Shared File Systems service can guarantee hard network path isolation for the users shares. Users can be allowed to designate their project networks as share networks. When a share network is provided during share creation, the share driver sets up a virtual share server (NAS server) on the share network and exports shares using this NAS server. The share server itself is abstracted away from the user. You must ensure that the storage system can connect the share servers it provisions to the networks users can use as their share networks.

Note: Not all shared file systems storage backends support share networks. Share networks can only be used when using a share type that has the specification `driver_handles_share_servers=True`. To see what storage back ends support this specification, refer to the *Manila share features support mapping*.

How to create share network

To list networks in a project, run:

```
$ openstack network list
+--------------+---------+--------------------+
| ID           | Name    | Subnets            |
+--------------+---------+--------------------+
```

(continues on next page)

(continued from previous page)

```
| bee7411d-... | public  | 884a6564-0f11-...  |
|              |         | e6da81fa-5d5f-...  |
| 5ed5a854-... | private | 74dcfb5a-b4d7-...  |
|              |         | cc297be2-5213-...  |
+--------------+---------+--------------------+
```

A share network stores network information that share servers can use where shares are hosted. You can associate a share with a single share network. You must always specify a share network when creating a share with a share type that requests hard multi-tenancy, i.e., has extra-spec driver_handles_share_servers=True.

For more information about supported plug-ins for share networks, see *Network plug-ins*.

A share network has these attributes:

- The IP block in Classless Inter-Domain Routing (CIDR) notation from which to allocate the network.
- The IP version of the network.
- The network type, which is *vlan*, *vxlan*, *gre*, or *flat*.

If the network uses segmentation, a segmentation identifier. For example, VLAN, VXLAN, and GRE networks use segmentation.

To create a share network with private network and subnetwork, run:

```
$ manila share-network-create --neutron-net-id 5ed5a854-21dc-4ed3-870a-
↪117b7064eb21 \
--neutron-subnet-id 74dcfb5a-b4d7-4855-86f5-a669729428dc --name my_share_net \
--description "My first share network" --availability-zone manila-zone-0
+-------------------+--------------------------------------+
| Property          | Value                                |
+-------------------+--------------------------------------+
| name              | my_share_net                         |
| segmentation_id   | None                                 |
| created_at        | 2015-09-24T12:06:32.602174           |
| neutron_subnet_id | 74dcfb5a-b4d7-4855-86f5-a669729428dc |
| updated_at        | None                                 |
| network_type      | None                                 |
| neutron_net_id    | 5ed5a854-21dc-4ed3-870a-117b7064eb21 |
| ip_version        | None                                 |
| cidr              | None                                 |
| project_id        | 20787a7ba11946adad976463b57d8a2f     |
| id                | 5c3cbabb-f4da-465f-bc7f-fadbe047b85a |
| description       | My first share network               |
+-------------------+--------------------------------------+
```

The `segmentation_id`, `cidr`, `ip_version`, and `network_type` share network attributes are automatically set to the values determined by the network provider.

Note: You are able to specify the parameter `availability_zone` only with API versions >= 2.51. From the version 2.51, a share network is able to span multiple subnets in different availability

zones. The network parameters `neutron_net_id`, `neutron_subnet_id`, `segmentation_id`, `cidr`, `ip_version`, `network_type`, `gateway` and `mtu` were moved to the share network subnet and no longer pertain to the share network. If you do not specify an availability zone during the share network creation, the created subnet will be considered default by the Shared File Systems Service. A default subnet is expected to be reachable from all availability zones in the cloud.

Note: Since API version 2.63, the share network will have two additional fields: `status` and `security_service_update_support`. The former indicates the current status of a share network, and the latter informs if all the share networks resources can hold updating or adding security services after they are already deployed.

To check the network list, run:

```
$ manila share-network-list
+--------------------------------------+--------------+
| id                                   | name         |
+--------------------------------------+--------------+
| 5c3cbabb-f4da-465f-bc7f-fadbe047b85a | my_share_net |
+--------------------------------------+--------------+
```

If you configured the generic driver with `driver_handles_share_servers = True` (with the share servers) and already had previous operations in the Shared File Systems service, you can see `manila_service_network` in the neutron list of networks. This network was created by the generic driver for internal use.

```
$ openstack network list
+--------------+------------------------+--------------------+
| ID           | Name                   | Subnets            |
+--------------+------------------------+--------------------+
| 3b5a629a-e...| manila_service_network | 4f366100-50...     |
| bee7411d-... | public                 | 884a6564-0f11-...  |
|              |                        | e6da81fa-5d5f-...  |
| 5ed5a854-... | private                | 74dcfb5a-b4d7-...  |
|              |                        | cc297be2-5213-...  |
+--------------+------------------------+--------------------+
```

You also can see detailed information about the share network including `network_type`, and `segmentation_id` fields:

```
$ openstack network show manila_service_network
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        | nova                                 |
| created_at                | 2016-12-13T09:31:30Z                 |
| description               |                                      |
| id                        | 3b5a629a-e7a1-46a3-afb2-ab666fb884bc |
```

(continues on next page)

(continued from previous page)

```
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| mtu                       | 1450                                 |
| name                      | manila_service_network               |
| port_security_enabled     | True                                 |
| project_id                | f6ac448a469b45e888050cf837b6e628     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 73                                   |
| revision_number           | 7                                    |
| router:external           | Internal                             |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 682e3329-60b0-440f-8749-83ef53dd8544 |
| tags                      | []                                   |
| updated_at                | 2016-12-13T09:31:36Z                 |
+---------------------------+--------------------------------------+
```

You also can add and remove the security services from the share network. For more detail, see *Security services*.

How to reset the state of a share network (Since API version 2.63)

To reset the state of a given share network, run:

```
$ manila share-network-reset-state manila_service_network --state active
```

Share network subnets (Since API version 2.51)

Share network subnet is an entity that stores network data from the OpenStack Networking service. A share network can span multiple share network subnets in different availability zones.

How to create share network subnet

When you create a share network, a primary share network subnet is automatically created. The share network subnet stores network information that share servers can use where shares are hosted. If a share network subnet is not assigned to a specific availability zone, it is considered to be available across all availability zones. Such a subnet is referred to as `default` subnet. A share network can have only one default subnet. However, having a default subnet is not necessary. A share can be associated with only one share network. To list share networks in a project, run:

```
$ manila share-network-list
+--------------------------------------+----------------------+
| id                                   | name                 |
+--------------------------------------+----------------------+
| 483a9787-5116-48b2-bd89-473022fad060 | sharenetwork1        |
```

(continues on next page)

(continued from previous page)

```
| bcb9c650-a501-410d-a418-97f28b8ab61a | sharenetwork2          |
+--------------------------------------+------------------------+
```

You can attach any number of share network subnets into a share network. However, only one share network subnet is allowed per availability zone in a given share network. If you try to create another subnet in a share network that already contains a subnet in a specific availability zone, the operation will be denied.

To create a share network subnet in a specific share network, run:

```
$ manila share-network-subnet-create sharenetwork1 \
    --availability-zone manila-zone-0 \
    --neutron-net-id 5ed5a854-21dc-4ed3-870a-117b7064eb21 \
    --neutron-subnet-id 74dcfb5a-b4d7-4855-86f5-a669729428dc
+--------------------+--------------------------------------+
| Property           | Value                                |
+--------------------+--------------------------------------+
| id                 | 20f3cd2c-0faa-4b4b-a00a-4f188eb1cf38 |
| availability_zone  | manila-zone-0                        |
| share_network_id   | 483a9787-5116-48b2-bd89-473022fad060 |
| share_network_name | sharenetwork1                        |
| created_at         | 2019-12-03T00:37:30.000000           |
| segmentation_id    | None                                 |
| neutron_subnet_id  | 74dcfb5a-b4d7-4855-86f5-a669729428dc |
| updated_at         | None                                 |
| neutron_net_id     | 5ed5a854-21dc-4ed3-870a-117b7064eb21 |
| ip_version         | None                                 |
| cidr               | None                                 |
| network_type       | None                                 |
| mtu                | None                                 |
| gateway            | None                                 |
+--------------------+--------------------------------------+
```

To list all the share network subnets of a given share network, you need to show the share network, and then all subnets will be displayed, as shown below:

```
$ manila share-network-show sharenetwork1
+----------------------+--------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪----------------------+
| Property             | Value
↪
↪
↪
↪
```

(continues on next page)

(continued from previous page)

```
+-----------------------+--------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪--------------------------------------------------------------------------------------------------------
↪-----------------------+
| id                    | 483a9787-5116-48b2-bd89-473022fad060                                          ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                     |
| name                | sharenetwork1                                                                 ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                     |
| project_id          | 58ff89e14f9245d7843b8cf290525b5b                                              ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                     |
| created_at          | 2019-12-03T00:16:39.000000                                                    ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                                                                                                      ␣
↪                     |
```

(continues on next page)

(continued from previous page)

```
| updated_at             | 2019-12-03T00:31:58.000000                                    ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                        |
| description            | None                                                  ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                                                                                ␣
↪                        |
| share_network_subnets | [{'id': '20f3cd2c-0faa-4b4b-a00a-4f188eb1cf38',
↪'availability_zone': 'manila-zone-0', 'created_at': '2019-12-03T00:37:30.
↪000000', 'updated_at': None, 'segmentation_id': None, 'neutron_net_id':
↪'5ed5a854-21dc-4ed3-870a-117b7064eb21', 'neutron_subnet_id': '74dcfb5a-b4d7-
↪4855-86f5-a669729428dc', 'ip_version': None, 'cidr': None, 'network_type':␣
↪None, 'mtu': None, 'gateway': None}, {'id': '8b532c15-3ac7-4ea1-b1bc-
↪732614a82313', 'availability_zone': None, 'created_at': '2019-12-
↪03T00:16:39.000000', 'updated_at': None, 'segmentation_id': None, 'neutron_
↪net_id': None, 'neutron_subnet_id': None, 'ip_version': None, 'cidr': None,
↪'network_type': None, 'mtu': None, 'gateway': None}] |
+------------------------+-------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------
↪----------------------+
```

To show a specific share network subnet, run:

```
$ manila share-network-subnet-show sharenetwork1 20f3cd2c-0faa-4b4b-a00a-
↪4f188eb1cf38
+--------------------+--------------------------------------+
| Property           | Value                                |
+--------------------+--------------------------------------+
| id                 | 20f3cd2c-0faa-4b4b-a00a-4f188eb1cf38 |
```

(continues on next page)

(continued from previous page)

```
| availability_zone  | manila-zone-0                        |
| share_network_id   | 483a9787-5116-48b2-bd89-473022fad060 |
| share_network_name | sharenetwork1                        |
| created_at         | 2019-12-03T00:37:30.000000           |
| segmentation_id    | None                                 |
| neutron_subnet_id  | 74dcfb5a-b4d7-4855-86f5-a669729428dc |
| updated_at         | None                                 |
| neutron_net_id     | 5ed5a854-21dc-4ed3-870a-117b7064eb21 |
| ip_version         | None                                 |
| cidr               | None                                 |
| network_type       | None                                 |
| mtu                | None                                 |
| gateway            | None                                 |
+--------------------+--------------------------------------+
```

To delete a share network subnet, run:

```
$ manila share-network-subnet-delete sharenetwork1 20f3cd2c-0faa-4b4b-a00a-
↪4f188eb1cf38
```

If you want to remove a share network subnet, make sure that no other resource is using the subnet, otherwise the Shared File Systems Service will deny the operation.

Network plug-ins

The Shared File Systems service architecture defines an abstraction layer for network resource provisioning and allowing administrators to choose from a different options for how network resources are assigned to their projects networked storage. There are a set of network plug-ins that provide a variety of integration approaches with the network services that are available with OpenStack.

What is a network plugin in Manila?

A network plugin is a python class that uses a specific facility (e.g. Neutron network) to provide network resources to the *manila-share* service.

When to use a network plugin?

A Manila *share driver* may be configured in one of two modes, where it is managing the lifecycle of *share servers* on its own or where it is merely providing storage resources on a pre-configured share server. This mode is defined using the boolean option *driver_handles_share_servers* in the Manila configuration file. A network plugin is only useful when a driver is handling its own share servers.

Note: Not all share drivers support both modes. Each driver must report which mode(s) it supports to the manila-share service.

When *driver_handles_share_servers* is set to *True*, a share driver will be called to create share servers for shares using information provided within a *share network*. This information will be provided to one

of the enabled network plugins that will handle reservation, creation and deletion of network resources including *IP addresses* and *network interfaces*.

The Shared File Systems service may need a network resource provisioning if share service with specified driver works in mode, when a share driver manages lifecycle of share servers on its own. This behavior is defined by a flag `driver_handles_share_servers` in share service configuration. When `driver_handles_share_servers` is set to `True`, a share driver will be called to create share servers for shares using information provided within a share network. This information will be provided to one of the enabled network plug-ins that will handle reservation, creation and deletion of network resources including IP addresses and network interfaces.

What network plug-ins are available?

There are two network plug-ins and three python classes in the Shared File Systems service:

1. Network plug-in for using the OpenStack Networking service. It allows to use any network segmentation that the Networking service supports. It is up to each share driver to support at least one network segmentation type.

   a) `manila.network.neutron.neutron_network_plugin.NeutronNetworkPlugin`. This is a default network plug-in. It requires the `neutron_net_id` and the `neutron_subnet_id` to be provided when defining the share network that will be used for the creation of share servers. The user may define any number of share networks corresponding to the various physical network segments in a project environment.

   b) `manila.network.neutron.neutron_network_plugin.NeutronSingleNetworkPlugin`. This is a simplification of the previous case. It accepts values for `neutron_net_id` and `neutron_subnet_id` from the `manila.conf` configuration file and uses one network for all shares.

   When only a single network is needed, the NeutronSingleNetworkPlugin (1.b) is a simple solution. Otherwise NeutronNetworkPlugin (1.a) should be chosen.

2. Network plug-in for specifying networks independently from OpenStack networking services.

   a) `manila.network.standalone_network_plugin.StandaloneNetworkPlugin`. This plug-in uses a pre-existing network that is available to the manila-share host. This network may be handled either by OpenStack or be created independently by any other means. The plug-in supports any type of network - flat and segmented. As above, it is completely up to the share driver to support the network type for which the network plug-in is configured.

Note: The ip version of the share network is defined by the flags of `network_plugin_ipv4_enabled` and `network_plugin_ipv6_enabled` in the `manila.conf` configuration since Pike. The `network_plugin_ipv4_enabled` default value is set to True. The `network_plugin_ipv6_enabled` default value is set to False. If `network_plugin_ipv6_enabled` option is True, the value of `network_plugin_ipv4_enabled` will be ignored, it means to support both IPv4 and IPv6 share network.

Troubleshoot Shared File Systems service

Failures in Share File Systems service during a share creation

Problem

New shares can enter `error` state during the creation process.

Solution

1. Make sure, that share services are running in debug mode. If the debug mode is not set, you will not get any tips from logs how to fix your issue.
2. Find what share service holds a specified share. To do that, run command **`manila show <share_id_or_name>`** and find a share host in the output. Host uniquely identifies what share service holds the broken share.
3. Look thought logs of this share service. Usually, it can be found at `/etc/var/log/manila-share.log`. This log should contain kind of traceback with extra information to help you to find the origin of issues.

No valid host was found

Problem

If a share type contains invalid extra specs, the scheduler will not be able to locate a valid host for the shares.

Solution

To diagnose this issue, make sure that scheduler service is running in debug mode. Try to create a new share and look for message `Failed to schedule create_share: No valid host was found.` in `/etc/var/log/manila-scheduler.log`.

To solve this issue look carefully through the list of extra specs in the share type, and the list of share services reported capabilities. Make sure that extra specs are pointed in the right way.

Created share is unreachable

Problem

By default, a new share does not have any active access rules.

Solution

To provide access to new share, you need to create appropriate access rule with the right value. The value must defines access.

Service becomes unavailable after upgrade

Problem

After upgrading the Shared File Systems service from version v1 to version v2.x, you must update the service endpoint in the OpenStack Identity service. Otherwise, the service may become unavailable.

Solution

1. To get the service type related to the Shared File Systems service, run:

```
# openstack endpoint list

# openstack endpoint show <share-service-type>
```

   You will get the endpoints expected from running the Shared File Systems service.

2. Make sure that these endpoints are updated. Otherwise, delete the outdated endpoints and create new ones.

Failures during management of internal resources

Problem

The Shared File System service manages internal resources effectively. Administrators may need to manually adjust internal resources to handle failures.

Solution

Some drivers in the Shared File Systems service can create service entities, like servers and networks. If it is necessary, you can log in to project `service` and take manual control over it.

Profiling the Shared File Systems service

Profiler

The detailed description of the profiler and its config options is available at Profiler docs.

Using Profiler

To start profiling Manila code, the following steps have to be taken:

1. Add the following lines to the `/etc/manila/manila.conf` file (the profiling is disabled by default).

```
[profiler]
connection_string = redis://localhost:6379
hmac_keys = SECRET_KEY
trace_sqlalchemy = True
enabled = True
```

   Examples of possible values for `connection_string` option:

   - `messaging://` - use oslo_messaging driver for sending spans.
   - `redis://127.0.0.1:6379` - use redis driver for sending spans.
   - `mongodb://127.0.0.1:27017` - use mongodb driver for sending spans.
   - `elasticsearch://127.0.0.1:9200` - use elasticsearch driver for sending spans.
   - `jaeger://127.0.0.1:6831` - use jaeger tracing as driver for sending spans.

2. Restart all manila services and keystone service.

3. To verify profiler with manilaclient, run any command with `--profile <key>`. The key (e.g. SECRET_KEY) should be one of the `hmac_keys` mentioned in manila.conf. To generate correct profiling information across all services at least one key needs to be consistent between OpenStack projects.

```
$ manila --profile SECRET_KEY create NFS 1 --name Share1 --share-network␣
↪testNetwork --share-type dhss_true
 +-------------------------------------+--------------------------------
↪------+
 | Property                            | Value                          ␣
↪      |
 +-------------------------------------+--------------------------------
↪------+
 | id                                  | 9703da88-25ba-41e6-827d-
↪a6932f708dd4 |
 | size                                | 1                              ␣
↪      |
 | availability_zone                   | None                           ␣
↪      |
 | created_at                          | 2021-02-23T11:21:38.000000     ␣
↪      |
 | status                              | creating                       ␣
↪      |
 | name                                | Share1                         ␣
↪      |
 | description                         | None                           ␣
↪      |
 | project_id                          |␣
↪c67b2fd35b054060971d28cf654ee92a     |
```

(continues on next page)

(continued from previous page)

```
| snapshot_id                          | None                                 |
| share_network_id                     | 03754c58-1456-497f-b7d6-8f36a4d644f0 |
| share_proto                          | NFS                                  |
| metadata                             | {}                                   |
| share_type                           | 5b1a4133-371c-4583-a801-f2b6e1ae102d |
| is_public                            | False                                |
| snapshot_support                     | False                                |
| task_state                           | None                                 |
| share_type_name                      | dhss_true                            |
| access_rules_status                  | active                               |
| replication_type                     | None                                 |
| has_replicas                         | False                                |
| user_id                              | 7ecd60ddae1448b79449dc6434460eaf     |
| create_share_from_snapshot_support   | False                                |
| revert_to_snapshot_support           | False                                |
| share_group_id                       | None                                 |
| source_share_group_snapshot_member_id | None                                 |
| mount_snapshot_support               | False                                |
| progress                             | None                                 |
| share_server_id                      | None                                 |
| host                                 |                                      |
+--------------------------------------+--------------------------------------+
Profiling trace ID: 1705dfd8-e45a-46cd-b0e2-2e40fd9e5f22
To display trace use next command:
osprofiler trace show --html 1705dfd8-e45a-46cd-b0e2-2e40fd9e5f22
```

4. To verify profiler with openstackclient, run any command with `--os-profile <key>`.

```
$ openstack --os-profile SECRET_KEY share create NFS 1 --name Share2 --
↪share-network testNetwork --share-type dhss_true
 +---------------------------------------+--------------------------------
↪------+
 | Field                                 | Value                          
↪      |
 +---------------------------------------+--------------------------------
↪------+
 | access_rules_status                   | active                         
↪      |
 | availability_zone                     | None                           
↪      |
 | create_share_from_snapshot_support    | False                          
↪      |
 | created_at                            | 2021-02-23T11:23:41.000000     
↪      |
 | description                           | None                           
↪      |
 | has_replicas                          | False                          
↪      |
 | host                                  |                                
↪      |
 | id                                    | 78a19734-394f-4967-9671-
↪c226df00a023 |
 | is_public                             | False                          
↪      |
 | metadata                              | {}                             
↪      |
 | mount_snapshot_support                | False                          
↪      |
 | name                                  | Share2                         
↪      |
 | progress                              | None                           
↪      |
 | project_id                            |
↪c67b2fd35b054060971d28cf654ee92a     |
 | replication_type                      | None                           
↪      |
 | revert_to_snapshot_support            | False                          
↪      |
 | share_group_id                        | None                           
↪      |
 | share_network_id                      | 03754c58-1456-497f-b7d6-
↪8f36a4d644f0 |
 | share_proto                           | NFS                            
↪      |
 | share_server_id                       | None                           
↪      |
 | share_type                            | 5b1a4133-371c-4583-a801-
↪f2b6e1ae102d |
```

(continues on next page)

(continued from previous page)

```
| share_type_name                      | dhss_true                        ␣
↪      |
| size                                 | 1                                ␣
↪      |
| snapshot_id                          | None                             ␣
↪      |
| snapshot_support                     | False                            ␣
↪      |
| source_share_group_snapshot_member_id | None                             ␣
↪      |
| status                               | creating                         ␣
↪      |
| task_state                           | None                             ␣
↪      |
| user_id                              |␣
↪7ecd60ddae1448b79449dc6434460eaf     |
| volume_type                          | dhss_true                        ␣
↪      |
+--------------------------------------+--------------------------------
↪------+
Trace ID: 0ca7ce01-36a9-481c-8b3d-263a3b5caa35
Short trace ID for OpenTracing-based drivers: 8b3d263a3b5caa35
Display trace data with command:
osprofiler trace show --html 0ca7ce01-36a9-481c-8b3d-263a3b5caa35
```

5. To display the trace date in HTML format, run below command.

```
$ osprofiler trace show --html 0ca7ce01-36a9-481c-8b3d-263a3b5caa35 --
↪connection-string redis://localhost:6379 --out /opt/stack/output.html
```

Upgrading the Shared File System service

This document outlines steps and notes for operators for reference when upgrading their Shared File System service (manila) from previous versions of OpenStack. The service aims to provide a minimal downtime upgrade experience. Since the service does not operate in the data plane, the accessibility of any provisioned resources such as shares, share snapshots, share groups, share replicas, share servers, security services and share networks will not be affected during an upgrade. Clients can continue to actively use these resources while the service control plane is being upgraded.

Plan the upgrade

It is highly recommended that you:

- update the Shared File System service to the latest code from the release you are currently using.
- read the Shared File System service release notes for the release that you intended to upgrade to. Pay special attention to the deprecations and upgrade notes.
- consider the impact of the service control plane upgrade to your clouds users. The upgrade process interrupts provisioning of new shared file systems and associated resources. It also prevents

management operations on existing shared file systems and associated resources. Data path access to shared file systems will remain uninterrupted.

- take a backup of the shared file system service database so you can rollback any failed upgrades to a previous version of the software. Although the `manila-manage` command offers a database downgrade command, it is not supported for production use. The only way to recover from a failed update is to restore the database from a backup.
- identify your Shared File System service back end storage systems/solutions and their drivers. Ensure that the version of each storage system is supported by the respective driver in the target release. If youre using a storage solution from a third party vendor, consult their product pages to determine if the solution is supported by the release of OpenStack that you are upgrading to. Many vendors publish a support matrix either within this service administration guide, or on their own websites. If you find an incompatibility, stop, and determine if you have to upgrade the storage solution first.
- develop an upgrade procedure and assess it thoroughly by using a test environment similar to your production environment.

Graceful service shutdown

Shared File System service components (scheduler, share-manager, data-manager) are python processes listening for messages on a AMQP queue. When the operator sends SIGTERM signal to the process, they stop getting new work from the queue, complete any outstanding work and then terminate.

Database Migration

The Shared File System service only supports cold upgrades, meaning that the service plane is expected to be down during the database upgrade. Database upgrades include schema changes as well as data migrations to accommodate newer versions of the schema. Once upgraded, downgrading the database is not supported. When the database has been upgraded, older services may misbehave when accessing database objects, so ensure all `manila-*` services are down before you upgrade the database.

Prune deleted database rows

Shared File System service resources are soft deleted in the database, so users are able to track instances in the DB that are created and destroyed in production. Soft-deletion also helps cloud operators adhere to data retention policies. Not purging soft-deleted entries affects DB performance as indices grow very large and data migrations take longer as there is more data to migrate. It is recommended that you prune the service database before upgrading to prevent unnecessary data migrations. Pruning permanently deletes soft deleted database records.

```
manila-manage db purge <age_in_days>
```

Upgrade procedure

1. Ensure youre running the latest Shared File System service packages for the OpenStack release that you currently use.
2. Run the `manila-status upgrade check` command to validate that the service is ready for upgrade.
3. Backup the manila database
4. Gracefully stop all Shared File System service processes. We recommend in this order: manila-api, manila-scheduler, manila-share and manila-data.

Note: The manila-data service may be processing time consuming data migrations. Shutting it down will interrupt any ongoing migrations, and these will not be automatically started when the service comes back up. You can check the status on ongoing migrations with `manila migration-get-progress` command; issue `manila migration-complete` for any ongoing migrations that have completed their data copy phase.

---

1. Upgrade all the service packages. If upgrading from distribution packages, your system package manager is expected to handle this automatically.
2. Fix any deprecated configuration options used.
3. Fix any deprecated api policies used.
4. Run `manila-manage db sync` from any node with the latest manila packages.
5. Start all the Shared File System service processes.
6. Inspect the `services` by running `manila service-list`. If there are any orphaned records, run `manila-manage service cleanup` to delete them.

Upgrade testing

The Shared File System service code is continually tested for upgrade from a previous release to the current release using Grenade. Grenade is an OpenStack test harness project that validates upgrade scenarios between releases. It uses DevStack to initially perform a base OpenStack install and then upgrade to a target version. Tests include the creation of a variety of Shared File System service resources on the prior release, and verification for their existence and functionality after the upgrade.

Share revert to snapshot

To revert a share to the latest available snapshot, use the **`manila revert-to-snapshot`**.

Note:

- In order to use this feature, the available backend in your deployment must have support for it. The list of backends that support this feature in the manila can be found in the *Manila share features support mapping*.

- This feature is only available in API version 2.27 and beyond. To create shares that are revertible, the share type used must contain the extra-spec `revert_to_snapshot_support` set to `True`. The default value for this is `False`.
- The revert operation can only be performed to the most recent available snapshot of the share known to manila. If revert to an earlier snapshot is desired, later snapshots must explicitly be deleted. In order to determine the most recent snapshot, the `created_at` field on the snapshot object is used.

While reverting, the share is in `reverting` status and the snapshot is in `restoring` status. After a successful restoration, the share and snapshot states will again be set to `available`. If the restoration fails the share will be set to `reverting_error` state and the snapshot will be set to `available`.

When a replicated share is reverted, the share becomes ready to be used only when all `active` replicas have been reverted. All secondary replicas will remain in `out-of-sync` state until they are consistent with the `active` replicas.

To revert a share to a snapshot, run:

```
$ manila revert-to-snapshot 14ee8575-aac2-44af-8392-d9c9d344f392
```

Share server migration

Share server migration is a functionality that lets administrators migrate a share server, and all its shares and snapshots, to a new destination.

As with share migration, a 2-phase approach was implemented for share server migration, which allows to control the right time to complete the operation, that usually ends on clients disruption.

The process of migrating a share server involves different operations over the share server, but can be achieved by invoking two main operations: start and complete. Youll need to begin with the start operation and wait until the service has completed the first phase of the migration to call the complete operation. When a share server is undergoing the first phase, its possible to choose to cancel it, or get a report of the progress.

A new operation called migration check is available to assist on a pre-migration phase, by validating within the destination host if the migration can or not be completed, providing an output with the compatible capabilities supported by the driver.

Share server migration is driven by share drivers, which means that both source and destination backends must support this functionality, and the driver must provide such operation in an efficient way.

Server migration workflows

Before actually starting the migration, you can use the operation *migration_check* to verify if the destination host and the requested capabilities are supported by the driver. If the answer is `compatible` equal to `True`, you can proceed with the migration process, otherwise youll need to identify the conflicting parameters or, in more complex scenarios, search for messages directly in the manila logs. The available capabilities are: `writable`, `nondisruptive`, `preserve_snapshots` and `new_share_network_id`, which are detailed in *Migration check and migration start parameters*.

The migration process starts by invoking the *migration_start* operation for a given share server. This operation will start the first phase of the migration that copies all data, from source to destination, including

all shares, their access rules and even snapshots if supported by the driver controlling the destination host.

For all ongoing migrations, you can optionally request the current status of a share server migration using *migration_get_progress* operation to retrieve the total progress of the data copy and its current task state. If supported by the driver, you can also cancel this operation by issuing *migration_cancel* and wait until all status become `active` and `available` again.

After completing the data copy, the first phase is completed and the next operation, *migration_complete*, can be initiated to finish the migration. The *migration_complete* operation usually disrupts clients access, since the export locations of the shares will change. The new export locations will be derived from the new share server that is provisioned at the destination, which is instantiated with distinct network allocations.

A new field `task_state` is available in the share server model to help track which operation is being executed during this process. The following tables show, for each phase, the expected `task_state`, along with their order of execution and a brief description of the actions that are being executed in the back end.

Table 1: **Share server migration states - 1st phase**

Sequence	*task_state*	Description
1	migration_starting	All initial validations passed, all shares and snapshots cant be modified until the end of the migration.
2	migration_in_progress	The destination host started the process of migration. If the driver doesnt support remain `writable`, all access rules are modified to read only.
3	migration_driver_starting	The driver was called to initiate the process of migrating the share server. Manila will wait for drivers answer.
4	migration_driver_in_progress	The driver accepted the request and started copying the data to the new share server. It will remain in this state until the end of the data copy.
5	migration_driver_phase1_done	Driver finished copying the data and its ready to complete the migration.

Along with the share server migration progress (in percentage) and the the current task state, the API also provides the destination share server ID. Alternatively, you may check the destination share server ID by querying the share server for a `source_share_server_id` set to the ID of the share server being migrated. During the entire migration process, the source source share server will remain with `server_migrating` status while the destination share server will remain with `server_migrating_to` status.

If an error occurs during the 1st phase of the migration, the source share server has its status reverted to `active` again, while the destination server has its status set to `error`. Both share servers will have their `task_state` updated to `migration_error`. All shares and snapshots are updated to `available` and any `read-only` rules are reset to allow writing into the shares.

Table 2: **Share server migration states - 2nd phase**

Sequence	*task_state*	Description
1	migration_completing	The destination host started processing the operation and the driver is called to complete the share server migration.
2	migration_success	The migration was completed with success. All shares and snapshots are `available` again.

After finishing the share server migration, all shares and snapshots have their status updated to `available`. The source share server status is set to `inactive` and the destination share server to `active`.

If an error occurs during the 2nd phase of the migration, both source and destination share servers will have their status updated to `error`, along with their shares and snapshots, since its not possible to infer if they are working properly and the current status of the migration. In this scenario, you will need to manually verify the health of all share servers resources and manually fix their statuses. Both share servers will have their `task_state` set to `migration_error`.

Table 3: **Share server migration states - migration cancel**

Sequence	*task_state*	Description
1	migration_cancel_in_progress	The destination host started the cancel process. It will remain in this state until the driver finishes all tasks that are in progress.
2	migration_cancelled	The migration was successfully cancelled.

If an error occurs during the migration cancel operation, the source share server has its status reverted to `active` again, while the destination server has its status updated to `error`. Both share servers will have their `task_state` set to `migration_error`. All shares and snapshots have their statuses updated to `available`.

Using share server migration CLI

The available commands to interact with the share server migration API are the following:

- `migration_check`: call a migration check operation to validate if the provided destination host is compatible with the requested operation and its parameters. The output shows if the destination host is compatible or not and the migration capabilities supported by the back end.

```
$ manila share-server-migration-check f3089d4f-89e8-4730-b6e6-
↪7cab553df071 stack@dummy2 --nondisruptive False --writable True --
↪preserve_snapshots True

+-----------------------+-----------------------------------------------
↪-----------------------------------------------------------------------
↪-----------------------------------------------------------------------+
| Property              | Value                                         ␣
↪                                                                       ␣
↪                                                                       |
+-----------------------+-----------------------------------------------
↪-----------------------------------------------------------------------
↪-----------------------------------------------------------------------+
| compatible            | True                                          ␣
↪                                                                       ␣
↪                                                                       |
| requested_capabilities | {'writable': 'True', 'nondisruptive': 'False',
↪'preserve_snapshots': 'True', 'share_network_id': None, 'host':
↪'stack@dummy2'}                                                        ␣
↪      |
```

(continues on next page)

(continued from previous page)

```
| supported_capabilities | {'writable': True, 'nondisruptive': False,
↪'preserve_snapshots': True, 'share_network_id': 'ac8e103f-c21a-4442-
↪bddc-fdadee093099', 'migration_cancel': True, 'migration_get_progress':␣
↪True} |
+------------------------+-----------------------------------------------
↪-----------------------------------------------------------------------
↪-----------------------------------------------------------------------+
```

The `share_network_id` attribute in the `supported_capabilities` will correspond to the value `--new_share_network` option if provided, otherwise it will be the same as the source share network. In the output it is possible to identify if the destination host supports the `migration_cancel` and `migration_get_progress` operations before starting the migration. The request parameters are the same for both `migration_check` and `migration_start` operations and are detailed in the following section.

Note: Back ends might use this operation to do many other validations with regards of storage compatibility, free space checks, share-type extra-specs validations, and so on. A `compatible` equal to `False` answer may not carry the actual conflict. You must check the `manila-share` logs for more details.

- `migration_start`: starts a share server migration to the provided destination host. This command starts the 1st phase of the migration that is an asynchronous operation and can take long to finish, depending on the size of the share server and the efficiency of the storage on copying all the data.

```
$ manila share-server-migration-start f3089d4f-89e8-4730-b6e6-
↪7cab553df071 stack@dummy2 --nondisruptive False --writable True --
↪preserve_snapshots True
```

The parameters description is detailed in the following section.

Note: This operation doesnt support migrating share servers with shares that have replicas or that belong to share groups.

Note: The current migration state and progress can be retrieve using the `migration-get-progress` command.

Note: This command has no output.

- `migration_complete`: completes a migration that already finished the 1st phase. This operation cant be cancelled and might end up on disrupting clients access after all shares migrate to the new share server.

```
 $ manila share-server-migration-complete f3089d4f-89e8-4730-b6e6-
↪7cab553df071
```

(continues on next page)

(continued from previous page)

```
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| destination_share_server_id | f3fb808f-c2a4-4caa-9805-7caaf55c0522 |
+-----------------------------+--------------------------------------+
```

- `migration_cancel`: cancels an in-progress share server migration. This operation can only be started while the migration is still on the 1st phase of the migration.

```
$ manila share-server-migration-cancel f3089d4f-89e8-4730-b6e6-
↪7cab553df071
```

Note: This command has no output.

- `migration_get_progress`: obtains the current progress information of a share server migration.

```
 $ manila share-server-migration-get-progress f3089d4f-89e8-4730-b6e6-
↪7cab553df071

+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| total_progress              | 50                                   |
| task_state                  | migration_driver_in_progress         |
| destination_share_server_id | f3fb808f-c2a4-4caa-9805-7caaf55c0522 |
+-----------------------------+--------------------------------------+
```

Migration check and migration start parameters

Share server *migration_check* and *migration_start* operations have specific parameters that have the semantic detailed below. From these, only `new_share_network` stands as an optional parameter.

- `share_server_id`: The ID of the share server that will be migrated.
- `destination_host`: The destination host to which the share server should be migrated to, in format `host@backend`.
- `preserve_snapshots`: enforces when the preservation of snapshots is mandatory for the requested migration. If the destination host doesnt support it, the operation will be denied. If this parameter is set to `False`, it will be the drivers supported capability that will define if the snapshots will be preserved or not.

Note: If the driver doesnt support preserving snapshots but at least one share has a snapshot, the operation will fail and the you will need to manually remove the remaining snapshots before proceeding.

- `writable`: enforces whether the source share server should remain writable for the requested migration. If the destination host doesnt support it, the operation will be denied. If this parameter is set to `False`, it will be the drivers supported capability that will define if all shares will remain writable or not.
- `nondisruptive`: enforces whether the migration should keep clients connected throughout the migration process. If the destination host doesnt support it, the operation will be denied. If this parameter is set to `False`, it will be the drivers supported capability that will define if all clients will remain connected or not.

In order to appropriately move a share server to a different host, it may be required to change the destination share network to be used by the new share server. In this case, a new share network can be provided using the following optional parameter:

- `new_share_network_id`: specifies the ID of the share network that should be used when setting up the new share server.

Note: It is not possible to choose the destination share network subnet since it will be automatically selected according to the destination hosts availability zone. If the new share network doesnt have a share network subnet in the destination hosts availability zone or doesnt have a default subnet, the operation will fail.

Configuration

For share server migration to work it is necessary to have compatible back end stanzas present in the manila configuration of all `manila-share` nodes.

Some drivers may provide some driver-specific configuration options that can be changed to adapt to specific workload. Check *Share drivers* documentation for more details.

Important notes

- Once the migration of a share server has started, the user will see that the status of all associated resources change to `server_migrating` and this will block any other share actions, such as adding or removing access rules, creating or deleting snapshots, resizing, among others.
- Since this is a driver-assisted migration, there is no guarantee that the destination share server will be cleaned up after a migration failure. For this reason, the destination share server will be always updated to `error` if any failure occurs. The same assumption is made for a source share server after a successful migration, where manila updates its status to `inactive` to avoid being reused for new shares.
- If a failure occurs during the 2nd phase of the migration, you will need to manually identify the current status of the source share server in order to revert it back to `active` again. If the share server and all its resources remain healthy, you will need to reset the status using `reset_status` API for each affected resource.
- Each step in the migration process is saved to the field `task_state` present in the share server model. If for any reason the state is not set to `migration_error` after a failure, it will need to be reset using the `reset_task_state` API, to unlock new share actions.

- After a failure occurs, the destination share server will have its status updated to `error` and will continue pointing to the original source share server. This can help you to identify the failed share servers when running multiple migrations in parallel.

Manila share features support mapping

Here we provide information on support of different share features by different share drivers.

Column values contain the OpenStack release letter when a feature was added to the driver. Column value ? means that this field requires an update with current information. Column value - means that this feature is not currently supported.

Mapping of share drivers and share features support

Driver name	create delete share	manage unmanage share	extend share	shrini
ZFSonLinux	M	N	M	M
Container	N	-	N	-
Generic (Cinder as back-end)	J	K	L	L
NetApp Clustered Data ONTAP	J	L	L	L
EMC VMAX	O	-	O	-
EMC VNX	J	-	-	-
EMC Unity	N	U	N	S
EMC Isilon	K	-	M	-
GlusterFS	J	-	directory layout (T)	direct
GlusterFS-Native	J	-	-	-
HDFS	K	-	M	-
Hitachi HNAS	L	L	L	M
Hitachi HSP	N	N	N	N
HPE 3PAR	K	-	-	-
Huawei	K	L	L	L
IBM GPFS	K	O	L	-
INFINIDAT	Q	-	Q	-
INSPUR AS13000	R	-	R	-
INSPUR InStorage	T	-	T	-
Infortrend	T	T	T	T
LVM	M	-	M	-
Quobyte	K	-	M	M
Windows SMB	L	L	L	L
Oracle ZFSSA	K	N	M	M
CephFS	M	-	M	M
Tegile	M	-	M	M
NexentaStor4	N	-	N	-
NexentaStor5	N	T	N	N
MapRFS	O	O	O	O
QNAP	O	O	O	-
Pure Storage FlashBlade	X	-	X	X

Mapping of share drivers and share access rules support

Driver name	Read & Write			
	IPv4	IPv6	USER	Cert
ZFSonLinux	NFS (M)	-	-	-
Container	-	-	CIFS (N)	-
Generic (Cinder as back-end)	NFS,CIFS (J)	-	-	-
NetApp Clustered Data ONTAP	NFS (J)	NFS (Q)	CIFS (J)	-
EMC VMAX	NFS (O)	NFS (R)	CIFS (O)	-
EMC VNX	NFS (J)	NFS (Q)	CIFS (J)	-
EMC Unity	NFS (N)	NFS (Q)	CIFS (N)	-
EMC Isilon	NFS,CIFS (K)	-	CIFS (M)	-
GlusterFS	NFS (J)	-	-	-
GlusterFS-Native	-	-	-	J
HDFS	-	-	HDFS(K)	-
Hitachi HNAS	NFS (L)	-	CIFS (N)	-
Hitachi HSP	NFS (N)	-	-	-
HPE 3PAR	NFS,CIFS (K)	-	CIFS (K)	-
Huawei	NFS (K)	-	NFS (M),CIFS (K)	-
LVM	NFS (M)	NFS (P)	CIFS (M)	-
Quobyte	NFS (K)	-	-	-
Windows SMB	-	-	CIFS (L)	-
IBM GPFS	NFS (K)	-	-	-
INFINIDAT	NFS (Q)	-	-	-
INSPUR AS13000	NFS (R)	-	CIFS (R)	-
INSPUR InStorage	NFS (T)	-	CIFS (T)	-
Infortrend	NFS (T)	-	CIFS (T)	-
Oracle ZFSSA	NFS,CIFS(K)	-	-	-
CephFS	NFS (P)	NFS (T)	-	-
Tegile	NFS (M)	-	NFS (M),CIFS (M)	-
NexentaStor4	NFS (N)	-	-	-
NexentaStor5	NFS (N)	T	-	-
MapRFS	-	-	MapRFS(O)	-
QNAP	NFS (O)	-	-	-
Pure Storage FlashBlade	NFS (X)	-	-	-

Mapping of share drivers and security services support

Driver name	Active Directory	LDAP	Kerberos
ZFSonLinux	-	-	-
Container	-	-	-
Generic (Cinder as back-end)	-	-	-
NetApp Clustered Data ONTAP	J	J	J
EMC VMAX	O	-	-
EMC VNX	J	-	-
EMC Unity	N	-	-

continues on next page

Table 6 – continued from previous page

Driver name	Active Directory	LDAP	Kerberos
EMC Isilon	-	-	-
GlusterFS	-	-	-
GlusterFS-Native	-	-	-
HDFS	-	-	-
Hitachi HNAS	-	-	-
Hitachi HSP	-	-	-
HPE 3PAR	-	-	-
Huawei	M	M	-
LVM	-	-	-
Quobyte	-	-	-
Windows SMB	L	-	-
IBM GPFS	-	-	-
INFINIDAT	-	-	-
INSPUR AS13000	-	-	-
INSPUR InStorage	-	-	-
Infortrend	-	-	-
Oracle ZFSSA	-	-	-
CephFS	-	-	-
Tegile	-	-	-
NexentaStor4	-	-	-
NexentaStor5	-	-	-
MapRFS	-	-	-
QNAP	-	-	-
Pure Storage FlashBlade	-	-	-

Mapping of share drivers and common capabilities

More information: *Capabilities and Extra-Specs*

Driver name	DHSS=True	DHSS=False	dedupe	compression	thin_provisioning	t
ZFSonLinux	-	M	M	M	M	-
Container	N	-	-	-	-	
Generic (Cinder as back-end)	J	K	-	-	-	
NetApp Clustered Data ONTAP	J	K	M	M	M	
EMC VMAX	O	-	-	-	-	-
EMC VNX	J	-	-	-	-	
EMC Unity	N	T	-	-	N	-
EMC Isilon	-	K	-	-	-	
GlusterFS	-	J	-	-	-	
GlusterFS-Native	-	J	-	-	-	
HDFS	-	K	-	-	-	
Hitachi HNAS	-	L	N	-	L	-
Hitachi HSP	-	N	-	-	N	-
HPE 3PAR	L	K	L	-	L	
Huawei	M	K	L	L	L	
INFINIDAT	-	Q	-	-	Q	C

Table 7 –

Driver name	DHSS=True	DHSS=False	dedupe	compression	thin_provisioning	t
Infortrend	-	T	-	-	-	-
LVM	-	M	-	-	-	I
Quobyte	-	K	-	-	-	I
Windows SMB	L	L	-	-	-	I
IBM GPFS	-	K	-	-	-	I
Oracle ZFSSA	-	K	-	-	-	I
CephFS	-	M	-	-	-	I
Tegile	-	M	M	M	M	-
NexentaStor4	-	N	N	N	N	I
NexentaStor5	-	N	-	N	N	I
MapRFS	-	N	-	-	-	I
QNAP	-	O	Q	Q	O	C
INSPUR AS13000	-	R	-	-	R	-
INSPUR InStorage	-	T	-	-	-	T
Pure Storage FlashBlade	-	X	-	-	X	-

Note: The common capability reported by back ends differs from some names seen in the above table:

- *DHSS* is reported as `driver_handles_share_servers` (See details for *DHSS*)
- *create share from snapshot* is reported as `create_share_from_snapshot_support`

Capabilities and Extra-Specs

Cloud Administrators create *Share types* with extra-specs to:

- influence the schedulers decision to place new shares, and
- instruct the Shared File System service or its storage driver/s to perform certain special actions with respect to the users shares.

As an administrator, you can choose a descriptive name or provide good descriptions for your share types to convey the share type capabilities to end users. End users can view standard `tenant-visible` extra-specs that can let them seek required behavior and automate their applications accordingly. By design, however, all other extra-specs of a share type are not exposed to non-privileged users.

Types of Extra-Specs

The Shared File Systems service back-end storage drivers offer a wide range of capabilities. The variation in these capabilities allows cloud administrators to provide a storage service catalog to their end users. Share type extra-specs tie-in with these capabilities.

Some back-end capabilities are very specific to a storage system, and are opaque to the Shared File System service or the end users. These capabilities are invoked with the help of scoped extra-specs. Using scoped extra-specs is a way to provide programmatic directives to the concerned storage driver to do something during share creation or share manipulation. You can learn about the opaque capabilities through driver documentation and configure these capabilities within share types as scoped extra-specs

(e.g.: hpe3par:nfs_options). The Shared File System service scheduler ignores scoped extra-specs during its quest to find the right back end to provision shares.

There are some back-end capabilities in manila that do matter to the scheduler. For our understanding, lets call these non-scoped or non-opaque capabilities. All non-scoped capabilities can be directly used as share types extra-specs. They are considered by the schedulers capabilities filter (and any custom filter defined by deployers).

You can get a list of non-scoped capabilities from the scheduler by using:

```
$ manila pool-list --detail
```

The non-scoped capabilities can be of three types:

- **Capabilities pertaining to a specific back end storage system driver**: For example, *huawei_smartcache*. No Shared File System service API relies on non-opaque back end specific capabilities.
- **Common capabilities that are not visible to end users**: The manila community has standardized some cross-platform capabilities like *thin_provisioning*, *dedupe*, *compression*, *qos*, *ipv6_support* and *ipv4_support*. Values of these options do not matter to any Shared File System service APIs; however, they can signify something to the manila services themselves. For example when a back end supports thin_provisioning, the scheduler service performs over-provisioning, and if a back end does not report *ipv6_support* as True, the share-manager service drops IPv6 access rules before invoking the storage driver to update access rules.
- **Common capabilities that are visible to end users**: Some capabilities affect functionality exposed via the Shared File System service API. For example, not all back ends support snapshots, and even if they do, they may not support all of the snapshot operations. For example, cloning snapshots into new shares, reverting shares in-place to snapshots, etc.

  The support for these capabilities determines whether users would be able to perform certain control-plane operations with manila. For example, a back end driver may report *snapshot_support=True* allowing end users to create share snapshots, however, the driver can report *create_share_from_snapshot_support=False*. This reporting allows cloud administrators to create share types that support snapshots but not creating shares from snapshots. When a user uses such a share type, they will not be able to clone snapshots into new shares. Tenant-visible capabilities aid manila in validating requests and failing fast on requests it cannot accommodate. They also help level set the user expectations on some failures. For example, if snapshot_support is set to False on the share type, since users can see this, they will not invoke the create snapshot API, and even if they do, they will understand the HTTP 400 (and error message) in better context.

Important: All extra-specs are optional, except one: *driver_handles_share_servers*.

Schedulers treatment of non-scoped extra specs

The CapabilitiesFilter in the Shared File System scheduler uses the following for matching operators:

- No operator This defaults to doing a python ==. Additionally it will match boolean values.
- **<=, >=, ==, !=**

  This does a float conversion and then uses the python operators as expected.

- **<in>**

  This either chooses a host that has partially matching string in the capability or chooses a host if it matches any value in a list. For example, if <in> sse4 is used, it will match a host that reports capability of sse4_1 or sse4_2.

- **<or>**

  This chooses a host that has one of the items specified. If the first word in the string is <or>, another <or> and value pair can be concatenated. Examples are <or> 3, <or> 3 <or> 5, and <or> 1 <or> 3 <or> 7. This is for string values only.

- **<is>**

  This chooses a host that matches a boolean capability. An example extra-spec value would be <is> True.

- **=**

  This does a float conversion and chooses a host that has equal to or greater than the resource specified. This operator behaves this way for historical reasons.

- **s==, s!=, s>=, s>, s<=, s<**

  The s indicates it is a string comparison. These choose a host that satisfies the comparison of strings in capability and specification. For example, if capabilities:replication_type s== dr, a host that reports replication_type of dr will be chosen. If share_backend_name s!= cephfs is used, any host not named cephfs can be chosen.

For vendor-specific non-scoped capabilities (which need to be visible to the scheduler), drivers are recommended to use the vendor prefix followed by an underscore. This is not a strict requirement, but can provide a consistent look along-side the scoped extra-specs and will be a clear indicator of vendor capabilities vs. common capabilities.

Common Capabilities

Common capabilities apply to multiple backends. Like all other backend reported capabilities, these capabilities can be used verbatim as extra_specs in share types used to create shares.

Share type common capability extra-specs that are visible to end users:

- **driver_handles_share_servers** is a special, required common capability. When set to True, the scheduler matches requests with back ends that can isolate user workloads with dedicated share servers exporting shares on user provided share networks.
- **snapshot_support** indicates whether snapshots are supported for shares created on the pool/backend. When administrators do not set this capability as an extra-spec in a share type, the scheduler can place new shares of that type in pools without regard for whether snapshots are supported, and those shares will not support snapshots.
- **create_share_from_snapshot_support** indicates whether a backend can create a new share from a snapshot. When administrators do not set this capability as an extra-spec in a share type, the scheduler can place new shares of that type in pools without regard for whether creating shares from snapshots is supported, and those shares will not support creating shares from snapshots.
- **revert_to_snapshot_support** indicates that a driver is capable of reverting a share in place to its most recent snapshot. When administrators do not set this capability as an extra-spec in a share type, the scheduler can place new shares of that type in pools without regard for whether reverting shares to snapshots is supported, and those shares will not support reverting shares to snapshots.
- **mount_snapshot_support** indicates that a driver is capable of exporting share snapshots for mounting. Users can provide and revoke access to mountable snapshots just like they can with their shares.
- **replication_type** indicates the style of replication supported for the backend/pool. This extra_spec will have a string value and could be one of *writable*, *readable* or *dr*. *writable* replication type involves synchronously replicated shares where all replicas are writable. Promotion is not supported and not needed. *readable* and *dr* replication types involve a single *active* or *primary* replica and one or more *non-active* or secondary replicas per share. In *readable* type of replication, *non-active* replicas have one or more export_locations and can thus be mounted and read while the *active* replica is the only one that can be written into. In *dr* style of replication, only the *active* replica can be mounted, read from and written into.
- **availability_zones** indicates a comma separated list of availability zones that can be used for provisioning. Users can always provide a specific availability zone during share creation, and they will receive a synchronous failure message if they attempt to create a share in an availability zone that the share type does not permit. If you do not set this extra-spec, the share type is assumed to be serviceable in all availability zones known to the Shared File Systems service.

Share type common capability extra-specs that are not visible to end users:

- **dedupe** indicates that a backend/pool can provide shares using some deduplication technology. The default value of the dedupe capability (if a driver doesnt report it) is False. Drivers can support both dedupe and non-deduped shares in a single storage pool by reporting `dedupe=[True, False]`. You can make a share type use deduplication by setting this extra-spec to <is> True, or prevent it by setting this extra-spec to <is> False.
- **compression** indicates that a backend/pool can provide shares using some compression technology. The default value of the compression capability (if a driver doesnt report it) is False. Drivers can support compressed and non-compressed shares in a single storage pool by reporting `compression=[True, False]`. You can make a share type use compression by setting this extra-spec to <is> True, or prevent it by setting this extra-spec to <is> False.

- **thin_provisioning** can be enabled where shares will not be guaranteed space allocations and over-provisioning will be enabled. This capability defaults to False. Back ends/pools that support thin provisioning report True for this capability. Administrators can make a share type use thin provisioned shares by setting this extra-spec to <is> True. If a driver reports thin_provisioning=False (the default) then its assumed that the driver is doing thick provisioning and overprovisioning is turned off. A driver can support thin provisioned and thick provisioned shares in the same pool by reporting `thin_provisioning=[True, False]`.

  To provision a thick share on a back end that supports both thin and thick provisioning, set one of the following in extra specs:

```
{'thin_provisioning': 'False'}
{'thin_provisioning': '<is> False'}
{'capabilities:thin_provisioning': 'False'}
{'capabilities:thin_provisioning': '<is> False'}
```

- **qos** indicates that a backend/pool can provide shares using some QoS (Quality of Service) specification. The default value of the qos capability (if a driver doesnt report it) is False. You can make a share type use QoS by setting this extra-spec to <is> True and also setting the relevant QoS-related extra specs for the drivers being used. Administrators can prevent a share type from using QoS by setting this extra-spec to <is> False. Different drivers have different ways of specifying QoS limits (or guarantees) and this extra spec merely allows the scheduler to filter by pools that either have or dont have QoS support enabled.
- **ipv4_support** indicates whether a back end can create a share that can be accessed via IPv4 protocol. If administrators do not set this capability as an extra-spec in a share type, the scheduler can place new shares of that type in pools without regard for whether IPv4 is supported.
- **ipv6_support** - indicates whether a back end can create a share that can be accessed via IPv6 protocol. If administrators do not set this capability as an extra-spec in a share type, the scheduler can place new shares of that type in pools without regard for whether IPv6 is supported.
- **provisioning:max_share_size** can set the max size of share, the value must be an integer and greater than 0. If administrators set this capability as an extra-spec in a share type, the size of share created with the share type can not be greater than the specified value.
- **provisioning:min_share_size** can set the min size of share, the value must be an integer and greater than 0. If administrators set this capability as an extra-spec in a share type, the size of share created with the share type can not be less than the specified value.

Group Capabilities and group-specs

Manila Administrators create share group types with *Share types* and group-specs to allow users to request a group type of share group to create. The Administrator chooses a name for the share group type and decides how to communicate the significance of the different share group types in terms that the users should understand or need to know. By design, most of the details of a share group type (the extra- specs) are not exposed to users only Administrators.

Share group Types

Refer to the manila client command-line help for information on how to create a share group type and set share types, group-spec key/value pairs for a share group type.

Group-Specs

The group specs contains the group capabilities, similar to snapshot_support in share types. Users know what a group can do from group specs.

The group specs is an exact match requirement in share group filter (such as ConsistentSnapshotFilter). When the ConsistentSnapshotFilter is enabled (it is enabled by default), the scheduler will only create a share group on a backend that reports capabilities that match the share group types group-spec keys.

Common Group Capabilities

For group capabilities that apply to multiple backends a common capability can be created. Like all other backend reported group capabilities, these group capabilities can be used verbatim as group_specs in share group types used to create share groups.

- *consistent_snapshot_support* - indicates that a backend can enable you to create snapshots at the exact same point in time from multiple shares. The default value of the consistent_snapshot_support capability (if a driver doesnt report it) is None. Administrators can make a share group type use consistent snapshot support by setting this group-spec to host.

Export Location Metadata

Manila shares can have one or more export locations. The exact number depends on the driver and the storage controller, and there is no preference for more or fewer export locations. Usually drivers create an export location for each physical network interface through which the share can be accessed.

Because not all export locations have the same qualities, Manila allows drivers to add additional keys to the dict returned for each export location when a share is created. The share manager stores these extra keys and values in the database and they are available to the API service, which may expose them through the REST API or use them for filtering.

Metadata Keys

Only keys defined in this document are valid. Arbitrary driver-defined keys are not allowed. The following keys are defined:

- *is_admin_only* - May be True or False. Defaults to False. Indicates that the export location exists for administrative purposes. If is_admin_only=True, then the export location is hidden from non-admin users calling the REST API. Also, these export locations are assumed to be reachable directly from the admin network, which is important for drivers that support share servers and which have some export locations only accessible to tenants.
- *preferred* - May be True or False. Defaults to False. Indicates that clients should prefer to mount this export location over other export locations that are not preferred. This may be used by drivers which have fast/slow paths to indicate to clients which paths are faster. It could be used to indicate

a path is preferred for another reason, as long as the reason isnt one that changes over the life of the manila-share service. This key is always visible through the REST API.

Supported share back ends

The manila share service must be configured to use drivers for one or more storage back ends, as described in general terms below. See the drivers section in the Configuration Reference for detailed configuration options for each back end.

Container Driver

The Container driver provides a lightweight solution for share servers management. It allows to use Docker containers for hosting userspace shared file systems services.

Supported operations

- Create CIFS share;
- Delete CIFS share;
- Allow user access to CIFS share;
- Deny user access to CIFS share;
- Extend CIFS share.

Restrictions

- Current implementation has been tested only on Ubuntu. Devstack plugin wont work on other distributions however it should be possible to install prerequisites and set the driver up manually;
- The only supported protocol is CIFS;
- The following features are not implemented: * Manage/unmanage share; * Shrink share; * Create/delete snapshots; * Create a share from a snapshot; * Manage/unmanage snapshots.

Known problems

- May demonstrate unstable behaviour when running concurrently. It is strongly suggested that the driver should be used with extreme care in cases other than building lightweight development and testing environments.

Setting up container driver with devstack

The driver could be set up via devstack. This requires the following update to local.conf:

```
enable_plugin manila https://opendev.org/openstack/manila <ref>
MANILA_ENABLED_BACKENDS=london
MANILA_OPTGROUP_london_driver_handles_share_servers=True
MANILA_OPTGROUP_london_neutron_host_id=<hostname>
SHARE_DRIVER=manila.share.drivers.container.driver.ContainerShareDriver
SHARE_BACKING_FILE_SIZE=<backing file size>
MANILA_DEFAULT_SHARE_TYPE_EXTRA_SPECS='snapshot_support=false'
```

where <ref> is change reference, which could be copied from gerrit web-interface, <hostname> is the name of the host with running neutron

Setting Container Driver Up Manually

This section describes steps needed to be performed to set the driver up manually. The driver has been tested on Ubuntu 14.04, thus in case of any other distribution package names might differ. The following packages must be installed:

- docker.io

One can verify if the package is installed by issuing `sudo docker info` command. In case of normal operation it should return docker usage statistics. In case it fails complaining on inaccessible socket try installing `apparmor`. Please note that docker usage requires superuser privileges.

After docker is successfully installed a docker image containing necessary packages must be provided. Currently such image could be downloaded from https://github.com/a-ovchinnikov/manila-image-elements-lxd-images/releases/download/0.1.0/manila-docker-container.tar.gz The image has to be unpacked but not untarred. This could be achieved by running gzip -d <imagename> command. Resulting tar-archive of the image could be uploaded to docker via

```
sudo docker load --input <imagename.tar>
```

If the previous command finished successfully you will be able to see the image in the image list:

```
sudo docker images
```

The driver expects to find a folder /tmp/shares on the host where it is running as well as a logical volume group manila_docker_volumes.

When installing the driver manually one must make sure that brctl and docker commands are present in the /etc/manila/rootwrap.d/share.filters and could be executed as root.

Finally to use the driver one must add a backend to the config file containing the following settings:

```
driver_handles_share_servers = True
share_driver = manila.share.drivers.container.driver.ContainerShareDriver
neutron_host_id = <hostname>
```

where <hostname> is the name of the host running neutron. (In case of single VM devstack it is VMs name).

After restarting manila services you should be able to use the driver.

ZFS (on Linux) Driver

Manila ZFSonLinux share driver uses ZFS filesystem for exporting NFS shares. Written and tested using Linux version of ZFS.

Requirements

- NFS daemon that can be handled via exportfs app.
- ZFS filesystem packages, either Kernel or FUSE versions.
- ZFS zpools that are going to be used by Manila should exist and be configured as desired. Manila will not change zpool configuration.
- For remote ZFS hosts according to manila-share service host SSH should be installed.
- **For ZFS hosts that support replication:**
  - SSH access for each other should be passwordless.
  - Service IP addresses should be available by ZFS hosts for each other.

Supported Operations

The following operations are supported:

- Create NFS Share
- Delete NFS Share
- Manage NFS Share
- Unmanage NFS Share
- **Allow NFS Share access**
  - Only IP access type is supported for NFS
  - Both access levels are supported - RW and RO
- Deny NFS Share access
- Create snapshot
- Delete snapshot
- Manage snapshot
- Unmanage snapshot
- Create share from snapshot
- Extend share
- Shrink share
- **Replication (experimental):**

  - Create/update/delete/promote replica operations are supported

- Share migration (experimental)

Possibilities

- Any amount of ZFS zpools can be used by share driver.
- Allowed to configure default options for ZFS datasets that are used for share creation.
- Any amount of nested datasets is allowed to be used.
- All share replicas are read-only, only active one is RW.
- All share replicas are synchronized periodically, not continuously. So, status in_sync means latest sync was successful. Time range between syncs equals to value of config global opt replica_state_update_interval.
- Driver is able to use qualified extra spec zfsonlinux:compression. It can contain any value that is supported by used ZFS app. But if it is disabled via config option with value compression=off, then it will not be used.

Restrictions

The ZFSonLinux share driver has the following restrictions:

- Only IP access type is supported for NFS.
- Only FLAT network is supported.
- Promote share replica operation will switch roles of current secondary replica and active. It does not make more than one active replica available.
- SaMBa based sharing is not yet implemented.
- Thick provisioning is not yet implemented.

Known problems

- Promote share replica operation will make ZFS filesystem that became secondary as RO only on NFS level. On ZFS level system will stay mounted as was - RW.

Backend Configuration

The following parameters need to be configured in the manila configuration file for the ZFSonLinux driver:

- share_driver = manila.share.drivers.zfsonlinux.driver.ZFSonLinuxShareDriver
- driver_handles_share_servers = False
- **replication_domain = custom_str_value_as_domain_name**
  - if empty, then replication will be disabled
  - if set then will be able to be used as replication peer for other backend with same value.

- zfs_share_export_ip = <user_facing IP address of ZFS host>
- zfs_service_ip = <IP address of service network interface of ZFS host>
- **zfs_zpool_list = zpoolname1,zpoolname2/nested_dataset_for_zpool2**
  - can be one or more zpools
  - can contain nested datasets
- **zfs_dataset_creation_options = <list of ZFS dataset options>**
  - readonly,quota,sharenfs and sharesmb options will be ignored
- **zfs_dataset_name_prefix = <prefix>**
  - Prefix to be used in each dataset name.
- **zfs_dataset_snapshot_name_prefix = <prefix>**
  - Prefix to be used in each dataset snapshot name.
- **zfs_use_ssh = <boolean_value>**
  - set False if ZFS located on the same host as manila-share service
  - set True if manila-share service should use SSH for ZFS configuration
- **zfs_ssh_username = <ssh_username>**
  - required for replication operations
  - required for SSHing to ZFS host if zfs_use_ssh is set to True
- **zfs_ssh_user_password = <ssh_user_password>**
  - password for zfs_ssh_username of ZFS host.
  - used only if zfs_use_ssh is set to True
- **zfs_ssh_private_key_path = <path_to_private_ssh_key>**
  - used only if zfs_use_ssh is set to True
- **zfs_share_helpers = NFS=manila.share.drivers.zfsonlinux.utils.NFSviaZFSHelper**
  - Approach for setting up helpers is similar to various other share driver
  - At least one helper should be used.
- **zfs_replica_snapshot_prefix = <prefix>**
  - Prefix to be used in dataset snapshot names that are created by update replica operation.
- **zfs_migration_snapshot_prefix = <prefix>**
  - Prefix to be used in dataset snapshot names that are created for migration operation.

Restart of *manila-share* service is needed for the configuration changes to take effect.

The manila.share.drivers.zfsonlinux.driver Module

Module with ZFSonLinux share driver that utilizes ZFS filesystem resources and exports them as shares.

class ZFSonLinuxShareDriver(**args*, ***kwargs*)

Bases: manila.share.drivers.zfsonlinux.utils.ExecuteMixin, *manila.share.driver.ShareDriver*

create_replica(*context*, **args*, ***kwargs*)

Replicate the active replica to a new replica on this backend.

Note: This call is made on the host that the new replica is being created upon.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share. This list also contains the replica to be created. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **new_replica** The share replica dictionary.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'creating',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'out_of_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': 'e6155221-ea00-49ef-abf9-9f89b7dd900a',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules. These are rules that other instances of the share already obey. Drivers are expected to apply access rules to the new replica or disregard access rules that dont apply.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica that will need to exist on the new share replica that is being created. The driver needs to ensure that this snapshot instance is truly available before transitioning the replica from out_of_sync to in_sync. Snapshots

instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the `update_replicated_snapshot` method.

Example:

```
[
 {
 'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
    },
 'share_replica_snapshot': {
    'id': '',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'status': 'available',
    'provider_location': None,
        ...
    },
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica being created.

Returns None or a dictionary. The dictionary can contain export_locations replica_state and access_rules_status. export_locations is a list of paths and replica_state is one of active, in_sync, out_of_sync or error.

Important: A backend supporting writable type replication should return active as the replica_state.

Export locations should be in the same format as returned during the `create_share` call.

Example:

```
{
    'export_locations': [
        {
            'path': '172.16.20.22/sample/export/path',
             'is_admin_only': False,
             'metadata': {'some_key': 'some_value'},
        },
    ],
     'replica_state': 'in_sync',
     'access_rules_status': 'in_sync',
}
```

create_replicated_snapshot(*context*, **args*, ***kwargs*)

Create a snapshot on active instance and update across the replicas.

Note: This call is made on the active replicas host. Drivers are expected to transfer the snapshot created to the respective replicas.

The driver is expected to return model updates to the share manager. If it was able to confirm the creation of any number of the snapshot instances passed in this interface, it can set their status to available as a cue for the share manager to set the progress attr to 100%.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to creating.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
```

(continues on next page)

(continued from previous page)

```
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being created.

Raises Exception. Any exception in this method will set all instances to error.

create_share(*context*, **args*, ***kwargs*)

Is called to create share.

create_share_from_snapshot(*context*, **args*, ***kwargs*)

Is called to create share from snapshot.

Creating a share from snapshot can take longer than a simple clone operation if data copy is required from one host to another. For this reason driver will be able complete this creation asynchronously, by providing a creating_from_snapshot status in the model update.

When answering asynchronously, drivers must implement the call get_share_status in order to provide updates for shares with creating_from_snapshot status.

It is expected that the driver returns a model update to the share manager that contains: share status and a list of export_locations. A list of export_locations is mandatory only for share in available status. The current supported status are available and creating_from_snapshot.

Parameters

- **context** Current context
- **share** Share instance model with share data.
- **snapshot** Snapshot instance model .
- **share_server** Share server model or None.
- **parent_share** Share model from parent snapshot with share data and share server model.

Returns

a dictionary of updates containing current share status and its export_location (if available).

Example:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance to error and the operation will end.

create_snapshot(*context*, **args*, ***kwargs*)

Is called to create snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

Returns None or a dictionary with key export_locations containing a list of export locations, if snapshots can be mounted.

delete_replica(*context*, **args*, ***kwargs*)

Delete a replica.

Note: This call is made on the host that hosts the replica being deleted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be deleted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
```

(continues on next page)

(continued from previous page)

```
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the share replica being deleted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '53099868-65f1-11e5-9d70-feff819cdc9f',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. The dict contains snapshot instances that are associated with the share replica being deleted. No model updates to snapshot instances are possible in this method. The driver should return when the cleanup is completed on the backend for both, the snapshots and the replica itself. Drivers must handle situations where the snapshot may not yet have finished creating on this replica.

Example:

```
[
    {
    'id': '89dafd00-0999-4d23-8614-13eaa6b02a3b',
```

(continues on next page)

(continued from previous page)

```
    'snapshot_id': '3ce1caf7-0945-45fd-a320-714973e949d3',
    'status: 'available',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be deleted.

Returns None.

Raises Exception. Any exception raised will set the share replicas status and replica_state attributes to error_deleting. It will not affect snapshots belonging to this replica.

delete_replicated_snapshot(*context*, **args*, ***kwargs*)

Delete a snapshot by deleting its instances across the replicas.

Note: This call is made on the active replicas host, since drivers may not be able to delete the snapshot from an individual replica.

The driver is expected to return model updates to the share manager. If it was able to confirm the removal of any number of the snapshot instances passed in this interface, it can set their status to deleted as a cue for the share manager to clean up that instance from the database.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
```

(continues on next page)

(continued from previous page)

```
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to deleting.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status': 'deleting',
    'progress': '100%',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'deleting',
    'progress': '100%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being deleted. To confirm the deletion of the snapshot instance, set the status attribute of the instance to deleted (constants.STATUS_DELETED)

Raises Exception. Any exception in this method will set the status attribute of all snapshot instances to error_deleting.

delete_share(*context*, **args*, ***kwargs*)
Is called to remove share.

delete_snapshot(*context*, **args*, ***kwargs*)
Is called to remove snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

do_setup(*context*)
Perform basic setup and checks.

ensure_share(*context*, **args*, ***kwargs*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

extend_share(*context*, **args*, ***kwargs*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

get_network_allocations_number()
ZFS does not handle networking. Return 0.

get_pool(*share*)
Return pool name where the share resides on.

Parameters **share** The share hosted by the driver.

manage_existing(*share*, *driver_options*)
Manage existing ZFS dataset as manila share.

ZFSonLinux driver accepts only one driver_option size. If an administrator provides this option, then such quota will be set to dataset and used as share size. Otherwise, driver will set quota equal to nearest bigger rounded integer of usage size. Driver does not expect mountpoint to be changed (should be equal to default that is /%(dataset_name)s).

Parameters

- **share** share data
- **driver_options** Empty dict or dict with size option.

Returns dict with share size and its export locations.

manage_existing_snapshot(*snapshot_instance*, *driver_options*)
Manage existing share snapshot with manila.

Parameters

- **snapshot_instance** SnapshotInstance data
- **driver_options** expects only one optional key size.

Returns

dict with share snapshot instance fields for update, example:

```
{
    size: 1, provider_location: path/to/some/dataset@some_snapshot_tag,
}
```

migration_cancel(*context*, **args*, ***kwargs*)

Cancels migration of a given share to another host.

Note: Is called in source shares backend to cancel migration.

If possible, driver can implement a way to cancel an in-progress migration.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

migration_check_compatibility(*context*, **args*, ***kwargs*)

Checks destination compatibility for migration of a given share.

Note: Is called to test compatibility with destination backend.

Driver should check if it is compatible with destination backend so driver-assisted migration can proceed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the share to be migrated.
- **destination_share** Reference to the share model to be used by migrated share.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

A dictionary containing values indicating if destination backend is compatible, if share can remain writable during migration, if it can preserve all file metadata and if it can perform migration of given share non-disruptively.

Example:

```
{
    'compatible': True,
    'writable': True,
    'preserve_metadata': True,
    'nondisruptive': True,
    'preserve_snapshots': True,
}
```

migration_complete(*context*, **args*, ***kwargs*)

Completes migration of a given share to another host.

Note: Is called in source shares backend to complete migration.

If driver is implementing 2-phase migration, this method should perform the disruptive tasks related to the 2nd phase of migration, thus completing it. Driver should also delete all original share data from source backend.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

If the migration changes the share export locations, snapshot provider locations or snapshot export locations, this method should return a dictionary with the relevant info. In such case, a dictionary containing a list of export locations and a list of model updates for each snapshot indexed by their IDs.

Example:

```
{
    'export_locations':
    [
        {
```

(continues on next page)

(continued from previous page)

```
        'path': '1.2.3.4:/foo',
        'metadata': {},
        'is_admin_only': False
        },
        {
        'path': '5.6.7.8:/foo',
        'metadata': {},
        'is_admin_only': True
        },
    ],
    'snapshot_updates':
    {
        'bc4e3b28-0832-4168-b688-67fdc3e9d408':
        {
        'provider_location': '/snapshots/foo/bar_1',
        'export_locations':
        [
            {
            'path': '1.2.3.4:/snapshots/foo/bar_1',
            'is_admin_only': False,
            },
            {
            'path': '5.6.7.8:/snapshots/foo/bar_1',
            'is_admin_only': True,
            },
        ],
        },
        '2e62b7ea-4e30-445f-bc05-fd523ca62941':
        {
        'provider_location': '/snapshots/foo/bar_2',
        'export_locations':
        [
            {
            'path': '1.2.3.4:/snapshots/foo/bar_2',
            'is_admin_only': False,
            },
            {
            'path': '5.6.7.8:/snapshots/foo/bar_2',
            'is_admin_only': True,
            },
        ],
        },
    },
}
```

migration_continue(*context*, **args*, ***kwargs*)

Continues migration of a given share to another host.

Note: Is called in source shares backend to continue migration.

Driver should implement this method to continue monitor the migration progress in storage and perform following steps until 1st phase is completed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns Boolean value to indicate if 1st phase is finished.

migration_start(*context*, **args*, ***kwargs*)

Starts migration of a given share to another host.

Note: Is called in source shares backend to start migration.

Driver should implement this method if willing to perform migration in a driver-assisted way, useful for when source shares backend driver is compatible with destination backend driver. This method should start the migration procedure in the backend and end. Following steps should be done in migration_continue.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

promote_replica(*context*, **args*, ***kwargs*)

Promote a replica to active replica state.

Note: This call is made on the host that hosts the replica being promoted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be promoted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica to be promoted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
```

(continues on next page)

(continued from previous page)

```
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be promoted.

Returns updated_replica_list or None. The driver can return the updated list as in the request parameter. Changes that will be updated to the Database are: export_locations, access_rules_status and replica_state.

Raises Exception. This can be any exception derived from BaseException. This is re-raised by the manager after some necessary cleanup. If the driver raises an exception during promotion, it is assumed that all of the replicas of the share are in an inconsistent state. Recovery is only possible through the periodic update call and/or administrator intervention to correct the status of the affected replicas if they become healthy again.

shrink_share(*context*, **args*, ***kwargs*)

Shrinks size of existing share.

If consumed space on share larger than new_size driver should raise ShareShrinkingPossibleDataLoss exception: raise ShareShrinkingPossibleDataLoss(share_id=share[id])

Parameters

- **share** Share model
- **new_size** New size of share (new_size < share[size])

- **share_server** Optional Share server model

:raises ShareShrinkingPossibleDataLoss, NotImplementedError

unmanage(*share*)
Removes the specified share from Manila management.

unmanage_snapshot(*snapshot_instance*)
Unmanage dataset snapshot.

update_access(*context*, **args*, ***kwargs*)
Update access rules for given share.

`access_rules` contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the `add_rules` and `delete_rules` parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in `add_rules` and rules that need to be removed to be present in `delete_rules`.

When a rule in `delete_rules` was never applied, drivers must not raise an exception, or attempt to set the rule to `error` state.

`add_rules` and `delete_rules` can be empty lists, in this situation, drivers should ensure that the rules present in `access_rules` are the same as those on the back end. One scenario where this situation is forced is when the access_level is changed for all existing rules (share migration and for readable replicas).

Drivers must be mindful of this call for share replicas. When update_access is called on one of the replicas, the call is likely propagated to all replicas belonging to the share, especially when individual rules are added or removed. If a particular access rule does not make sense to the driver in the context of a given replica, the driver should be careful to report a correct behavior, and take meaningful action. For example, if R/W access is requested on a replica that is part of a readable type replication; R/O access may be added by the driver instead of R/W. Note that raising an exception *will* result in the access_rules_status on the replica, and the share itself being out_of_sync. Drivers can sync on the valid access rules that are provided on the `create_replica` and `promote_replica` calls.

Parameters

- **context** Current context
- **share** Share model with share data.
- **access_rules** A list of access rules for given share
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

Returns

None, or a dictionary of updates in the format:

```
{
```

09960614-8574-4e03-89cf-7cf267b0bd08: {

access_key: alice31493e5441b8171d2310d80e37e, state: error,

},

28f6eabb-4342-486a-a7f4-45688f0c0295: {

access_key: bob0078aa042d5a7325480fd13228b, state: active,

},

}

The top level keys are access_id fields of the access rules that need to be updated. `access_key``s are credentials (str) of the entities granted access. Any rule in the ``access_rules` parameter can be updated.

Important: Raising an exception in this method will force *all* rules in applying and denying states to error.

An access rule can be set to error state, either explicitly via this return parameter or because of an exception raised in this method. Such an access rule will no longer be sent to the driver on subsequent access rule updates. When users deny that rule however, the driver will be asked to deny access to the client/s represented by the rule. We expect that a rule that was error-ed at the driver should never exist on the back end. So, do not fail the deletion request.

Also, it is possible that the driver may receive a request to add a rule that is already present on the back end. This can happen if the share manager service goes down while the driver is committing access rule changes. Since we cannot determine if the rule was applied successfully by the driver before the disruption, we will treat all applying transitional rules as new rules and repeat the request.

---

update_replica_state(*context*, **args*, ***kwargs*)

Update the replica_state of a replica.

Note: This call is made on the host which hosts the replica being updated.

Drivers should fix replication relationships that were broken if possible inside this method.

This method is called periodically by the share manager; and whenever requested by the administrator through the resync API.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be updated. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
```

(continues on next page)

(continued from previous page)

```
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica being updated Replica state will always be in_sync, out_of_sync, or error. Replicas in active state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share. The driver could attempt to sync on any unapplied access_rules.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica being updated. The driver needs to ensure that this snapshot instance is truly available before transitioning from out_of_sync to in_sync. Snapshots instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the update_replicated_snapshot method.

Example:

```
 [
  {
'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
   },
 'share_replica_snapshot': {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'status': 'creating',
    'provider_location': None,
      ...
   },
  }
 ]
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_state: a str value denoting the replica_state. Valid values are in_sync and out_of_sync or None (to leave the current replica_state unchanged).

update_replicated_snapshot(*context*, **args*, ***kwargs*)

Update the status of a snapshot instance that lives on a replica.

Note: For DR and Readable styles of replication, this call is made on the replicas host and not the active replicas host.

This method is called periodically by the share manager. It will query for snapshot instances that track the parent snapshot across non-active replicas. Drivers can expect the status of the instance to be creating or deleting. If the driver sees that a snapshot instance has been removed from the replicas backend and the instance status was set to deleting, it is expected to raise a SnapshotResourceNotFound exception. All other exceptions will set the snapshot instance status to error. If the instance was not in deleting state, raising a SnapshotResourceNotFound will set the instance status to error.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
     'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'in_sync',
     ...
     'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
     'share_server': <models.ShareServer> or None,
    },
    {
     'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'active',
     ...
     'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
     'share_server': <models.ShareServer> or None,
    },
     ...
]
```

Parameters **share_replica** Share replica dictionary. This replica is associated with the snapshot instance whose status is being updated. Replicas in active replica_state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. This will include the snapshot instance being updated as well.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    ...
]
```

Parameters **replica_snapshot** Dictionary of the snapshot instance. This is the instance to be updated. It will be in creating or deleting state when sent via this parameter.

Example:

```
{
    'name': 'share-snapshot-18825630-574f-4912-93bb-af4611ef35a2',
    'share_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_name': 'share-d487b88d-e428-4230-a465-a800c2cce5f8',
```

(continues on next page)

(continued from previous page)

```
    'status': 'creating',
    'id': '18825630-574f-4912-93bb-af4611ef35a2',
    'deleted': False,
    'created_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share': <models.ShareInstance>,
    'updated_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'progress': '0%',
    'deleted_at': None,
    'provider_location': None,
}
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_snapshot_model_update: a dictionary. The dictionary must contain values that need to be updated on the database for the snapshot instance that represents the snapshot on the replica.

Raises exception.SnapshotResourceNotFound Raise this exception for snapshots that are not found on the backend and their status was deleting.

ensure_share_server_not_provided(*f*)

get_backend_configuration(*backend_name*)

The manila.share.drivers.zfsonlinux.utils Module

Module for storing ZFSonLinux driver utility stuff such as:

- Common ZFS code
- Share helpers

class ExecuteMixin

Bases: manila.share.driver.ExecuteMixin

execute(**cmd*, ***kwargs*)
Common interface for running shell commands.

execute_with_retry(**cmd*, ***kwargs*)
Retry wrapper over common shell interface.

get_zfs_option(*dataset_name*, *option_name*, ***kwargs*)
Returns value of requested zfs dataset option.

get_zpool_option(*zpool_name*, *option_name*, ***kwargs*)
Returns value of requested zpool option.

init_execute_mixin(**args*, ***kwargs*)
Init method for mixin called in the end of drivers __init__().

parse_zfs_answer(*string*)
Returns list of dicts with data returned by ZFS shell commands.

zfs(**cmd*, ***kwargs*)
ZFS shell commands executor.

zfs_with_retry(**cmd*, ***kwargs*)
ZFS shell commands executor.

class NASHelperBase(*configuration*)
Bases: `object`

Base class for share helpers of ZFS on Linux driver.

abstract create_exports(*dataset_name*, *executor*)
Creates share exports.

abstract get_exports(*dataset_name*, *service*, *executor*)
Gets/reads share exports.

abstract remove_exports(*dataset_name*, *executor*)
Removes share exports.

abstract update_access(*dataset_name*, *access_rules*, *add_rules*, *delete_rules*, *executor*)
Update access rules for specified ZFS dataset.

abstract verify_setup()
Performs checks for required stuff.

class NFSviaZFSHelper(*configuration*)
Bases: `manila.share.drivers.zfsonlinux.utils.ExecuteMixin`, `manila.share.drivers.zfsonlinux.utils.NASHelperBase`

Helper class for handling ZFS datasets as NFS shares.

Kernel and Fuse versions of ZFS have different syntax for setting up access rules, and this Helper designed to satisfy both making autodetection.

create_exports(*dataset_name*, *executor=None*)
Creates NFS share exports for given ZFS dataset.

get_exports(*dataset_name*, *executor=None*)
Gets/reads NFS share export for given ZFS dataset.

property is_kernel_version
Says whether Kernel version of ZFS is used or not.

remove_exports(**args*, ***kwargs*)
Removes share exports.

update_access(**args*, ***kwargs*)
Update access rules for specified ZFS dataset.

verify_setup()
Performs checks for required stuff.

get_remote_shell_executor(*ip*, *port*, *conn_timeout*, *login=None*, *password=None*, *privatekey=None*, *max_size=10*)

zfs_dataset_synchronized(*f*)

NetApp Clustered Data ONTAP

The Shared File Systems service can be configured to use NetApp Clustered Data ONTAP (cDOT) version 8.2 and later.

The driver can work with two types of pools: FlexGroup and FlexVol. By default, it only works with FlexVol, if desired, the FlexGroup pool can be enabled together or standalone.

FlexGroup pool requires ONTAP version 9.8 or later.

Supported Operations

The following operations are supported on Clustered Data ONTAP:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- Allow NFS Share access
  - IP access type is supported for NFS.
  - Read/write and read-only access are supported for NFS.
- Allow CIFS Share access
  - User access type is supported for CIFS.
  - Read/write access is supported for CIFS.
- Deny CIFS/NFS Share access
- Create snapshot
- Delete snapshot
- Create share from snapshot
- Extend share
- Shrink share
- Manage share
- Unmanage share
- Create consistency group
- Delete consistency group
- Create consistency group from CG snapshot
- Create CG snapshot
- Delete CG snapshot
- Create a replica (DHSS=False)
- Promote a replica (DHSS=False)
- Delete a replica (DHSS=False)
- Update a replica (DHSS=False)

- Create a replicated snapshot (DHSS=False)
- Delete a replicated snapshot (DHSS=False)
- Update a replicated snapshot (DHSS=False)
- Migrate share
- Migrate share server

Note: The operations are not fully supported configuring FlexGroup pool:

- Consistency group operations are only supported configuring the driver without any FlexGroup pool.
- For FlexGroup share, create more than one replica is only allowed with ONTAP 9.9.1 and newer.
- Migration of FlexGroup shares is not allowed.
- Migration of share servers containing FlexGroup share is not allowed.

Note: *DHSS* is abbreviated from *driver_handles_share_servers*.

Supported Operating Modes

The cDOT driver supports both driver_handles_share_servers (*DHSS*) modes.

If driver_handles_share_servers is True, the driver will create a storage virtual machine (SVM, previously known as vServers) for each unique tenant network and provision each of a tenants shares into that SVM. This requires the user to specify both a share network as well as a share type with the DHSS extra spec set to True when creating shares.

If driver_handles_share_servers is False, the manila admin must configure a single SVM, along with associated LIFs and protocol services, that will be used for provisioning shares. The SVM is specified in the manila config file.

Network approach

L3 connectivity between the storage cluster and manila host must exist, and VLAN segmentation may be configured. All of manilas network plug-ins are supported with the cDOT driver.

Supported shared filesystems

- NFS (access by IP address or subnet)
- CIFS (authentication by user)

Required licenses

- NFS
- CIFS
- FlexClone

Known restrictions

- For CIFS shares an external Active Directory (AD) service is required. The AD details should be provided via a manila security service that is attached to the specified share network.
- Share access rules for CIFS shares may be created only for existing users in Active Directory.
- The time on external security services and storage must be synchronized. The maximum allowed clock skew is 5 minutes.
- cDOT supports only flat and VLAN network segmentation types.

The `manila.share.drivers.netapp.common.py` Module

Unified driver for NetApp storage systems.

Supports multiple storage systems of different families and driver modes.

class NetAppDriver(**args*, ***kwargs*)

Bases: `object`

NetApp unified share storage driver.

Acts as a factory to create NetApp storage drivers based on the storage family and driver mode configured.

```
REQUIRED_FLAGS = ['netapp_storage_family', 'driver_handles_share_servers']
```

Isilon Driver

The EMC manila driver framework (EMCShareDriver) utilizes EMC storage products to provide shared filesystems to OpenStack. The EMC manila driver is a plugin based driver which is designed to use different plugins to manage different EMC storage products.

The Isilon manila driver is a plugin for the EMC manila driver framework which allows manila to interface with an Isilon backend to provide a shared filesystem. The EMC driver framework with the Isilon plugin is referred to as the Isilon Driver in this document.

This Isilon Driver interfaces with an Isilon cluster via the REST Isilon Platform API (PAPI) and the RESTful Access to Namespace API (RAN).

Requirements

- Isilon cluster running OneFS 7.2 or higher

Supported Operations

The following operations are supported on an Isilon cluster:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- **Allow CIFS/NFS Share access**
  - Only IP access type is supported for NFS and CIFS
  - Only RW access supported
- Deny CIFS/NFS Share access
- Create snapshot
- Delete snapshot
- Create share from snapshot
- Extend share

Backend Configuration

The following parameters need to be configured in the manila configuration file for the Isilon driver:

- share_driver = manila.share.drivers.dell_emc.driver.EMCShareDriver
- driver_handles_share_servers = False
- emc_share_backend = isilon
- emc_nas_server = <IP address of Isilon cluster>
- emc_nas_server_port = <port to use for Isilon cluster (optional)>
- emc_nas_login = <username>
- emc_nas_password = <password>
- emc_nas_root_dir = <root directory path to create shares (e.g./ifs/manila)>

Restart of *manila-share* service is needed for the configuration changes to take effect.

Restrictions

The Isilon driver has the following restrictions:

- Only IP access type is supported for NFS and CIFS.
- Only FLAT network is supported.

The manila.share.drivers.dell_emc.driver Module

EMC specific NAS storage driver. This driver is a pluggable driver that allows specific EMC NAS devices to be plugged-in as the underlying backend. Use the Manila configuration variable share_backend_name to specify, which backend plugins to use.

class EMCShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

EMC specific NAS driver. Allows for NFS and CIFS NAS storage usage.

allow_access(*context*, *share*, *access*, *share_server=None*)
Allow access to the share.

check_for_setup_error()
Check for setup error.

create_share(*context*, *share*, *share_server=None*)
Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)
Is called to create share from snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to create snapshot.

delete_share(*context*, *share*, *share_server=None*)
Is called to remove share.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to remove snapshot.

deny_access(*context*, *share*, *access*, *share_server=None*)
Deny access to the share.

do_setup(*context*)
Any initialization the share driver does while starting.

ensure_share(*context*, *share*, *share_server=None*)
Invoked to sure that share is exported.

extend_share(*share*, *new_size*, *share_server=None*)
Is called to extend share.

get_configured_ip_versions()
Get allowed IP versions.

The supported versions are returned with list, possible values are: [4], [6], or [4, 6]

Drivers that assert ipv6_implemented = True must override this method. If the returned list includes 4, then shares created by this driver must have an IPv4 export location. If the list includes 6, then shares created by the driver must have an IPv6 export location.

Drivers should check that their storage controller actually has IPv4/IPv6 enabled and configured properly.

get_default_filter_function()
Get the default filter_function string.

Each driver could overwrite the method to return a well-known default string if it is available.

Parameters **pool** pool name to get the filter or None

Returns None

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

get_share_server_network_info(*context*, *share_server*, *identifier*, *driver_options*)
Obtain network allocations used by share server.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier
- **driver_options** Dictionary of driver options to assist managing the share server

Returns A list containing IP addresses allocated in the backend.

Example:

```
['10.10.10.10', 'fd11::2000', '192.168.10.10']
```

manage_existing(*share*, *driver_options*)
manage an existing share

manage_existing_snapshot(*snapshot*, *driver_options*)
manage an existing share snapshot

manage_existing_snapshot_with_server(*snapshot*, *driver_options*, *share_server=None*)
manage an existing share snapshot

manage_existing_with_server(*share*, *driver_options*, *share_server=None*)
manage an existing share

manage_server(*context*, *share_server*, *identifier*, *driver_options*)
Manage the share server and return compiled back end details.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier

- **driver_options** Dictionary of driver options to assist managing the share server

Returns Identifier and dictionary with back end details to be saved in the database.

Example:

```
'my_new_server_identifier',{'server_name': 'my_old_server'}
```

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a share (in place) to the specified snapshot.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **snapshot** The snapshot to be restored
- **share_access_rules** List of all access rules for the affected share
- **snapshot_access_rules** List of all access rules for the affected snapshot
- **share_server** Optional Share server model or None

shrink_share(*share*, *new_size*, *share_server=None*)

Is called to shrink share.

unmanage(*share*)

Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to False.

unmanage_server(*server_details*, *security_services=None*)

Unmanages the share server.

If a driver supports unmanaging of share servers, the driver must override this method and return successfully.

Parameters

- **server_details** share server backend details.

- **security_services** list of security services configured with this share server.

unmanage_snapshot(*snapshot*)

Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to False.

unmanage_snapshot_with_server(*snapshot*, *share_server=None*)

Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to True.

unmanage_with_server(*share*, *share_server=None*)

Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to True.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)

Update access to the share.

The manila.share.drivers.dell_emc.plugins.isilon.isilon Module

Isilon specific NAS backend plugin.

class IsilonStorageConnection(*args, **kwargs)

Bases: manila.share.drivers.dell_emc.plugins.base.StorageConnection

Implements Isilon specific functionality for EMC Manila driver.

allow_access(*context*, *share*, *access*, *share_server*)
Allow access to the share.

check_for_setup_error()
Check for setup error.

connect(*emc_share_driver*, *context*)
Connect to an Isilon cluster.

create_share(*context*, *share*, *share_server*)
Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server*)
Creates a share from the snapshot.

create_snapshot(*context*, *snapshot*, *share_server*)
Is called to create snapshot.

delete_share(*context*, *share*, *share_server*)
Is called to remove share.

delete_snapshot(*context*, *snapshot*, *share_server*)
Is called to remove snapshot.

deny_access(*context*, *share*, *access*, *share_server*)
Deny access to the share.

ensure_share(*context*, *share*, *share_server*)
Invoked to ensure that share is exported.

extend_share(*share*, *new_size*, *share_server=None*)
Extends a share.

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

setup_server(*network_info*, *metadata=None*)
Set up and configures share server with given network parameters.

teardown_server(*server_details*, *security_services=None*)
Teardown share server.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update share access.

update_share_stats(*stats_dict*)
TODO.

VNX Driver

EMC manila driver framework (EMCShareDriver) utilizes the EMC storage products to provide the shared filesystems to OpenStack. The EMC manila driver is a plugin based driver which is designed to use different plugins to manage different EMC storage products.

VNX plugin is the plugin which manages the VNX to provide shared filesystems. EMC driver framework with VNX plugin is referred to as VNX driver in this document.

This driver performs the operations on VNX by XMLAPI and the File command line. Each backend manages one Data Mover of VNX. Multiple manila backends need to be configured to manage multiple Data Movers.

Requirements

- VNX OE for File version 7.1 or higher.
- VNX Unified, File only, or Gateway system with single storage backend.
- The following licenses should be activated on VNX for File: * CIFS * NFS * SnapSure (for snapshot) * ReplicationV2 (for create share from snapshot)

Supported Operations

The following operations will be supported on VNX array:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- Allow CIFS/NFS Share access * Only IP access type is supported for NFS. * Only user access type is supported for CIFS.
- Deny CIFS/NFS Share access
- Create snapshot
- Delete snapshot
- Create share from snapshot

While the generic driver creates shared filesystems based on Cinder volumes attached to Nova VMs, the VNX driver performs similar operations using the Data Movers on the array.

Pre-Configurations on VNX

1. Enable Unicode on Data mover

VNX driver requires that the Unicode is enabled on Data Mover.

CAUTION: After enabling Unicode, you cannot disable it. If there are some filesystems created before Unicode is enabled on the VNX, consult the storage administrator before enabling Unicode.

To check the Unicode status on Data Mover, use the following VNX File command on VNX control station:

server_cifs <mover_name> | head where: mover_name = <name of the Data Mover>

Check the value of *I18N mode* field. UNICODE mode is shown as *I18N mode = UNICODE*

To enable the Unicode for Data Mover:

uc_config -on -mover <mover_name> where: mover_name = <name of the Data Mover>

Refer to the document *Using International Character Sets on VNX for File* on [EMC support site](https://support.emc.com) for more information.

2. Enable CIFS service on Data Mover

Ensure the CIFS service is enabled on the Data Mover which is going to be managed by VNX driver.

To start the CIFS service, use the following command:

server_setup <mover_name> -Protocol cifs -option start [=<n>] where: <mover_name> = <name of the Data Mover> [=<n>] = <number of threads for CIFS users>

Note: If there is 1 GB of memory on the Data Mover, the default is 96 threads; however, if there is over 1 GB of memory, the default number of threads is 256.

To check the CIFS service status, use this command:

server_cifs <mover_name> | head where: <mover_name> = <name of the Data Mover>

The command output will show the number of CIFS threads started.

3. NTP settings on Data Mover

VNX driver only supports CIFS share creation with share network which has an Active Directory security-service associated.

Creating CIFS share requires that the time on the Data Mover is in sync with the Active Directory domain so that the CIFS server can join the domain. Otherwise, the domain join will fail when creating share with this security service. There is a limitation that the time of the domains used by security-services even for different tenants and different share networks should be in sync. Time difference should be less than 10 minutes.

It is recommended to set the NTP server to the same public NTP server on both the Data Mover and domains used in security services to ensure the time is in sync everywhere.

Check the date and time on Data Mover:

server_date <mover_name> where: mover_name = <name of the Data Mover>

Set the NTP server for Data Mover:

server_date <mover_name> timesvc start ntp <host> [<host>] where: mover_name = <name of the Data Mover> host = <IP address of the time server host>

Note: The host must be running the NTP protocol. Only 4 host entries are allowed.

4. Configure User Mapping on the Data Mover

Before creating CIFS share using VNX driver, you must select a method of mapping Windows SIDs to UIDs and GIDs. EMC recommends using usermapper in single protocol (CIFS) environment which is enabled on VNX by default.

To check usermapper status, use this command syntax:

server_usermapper <movername> where: <movername> = <name of the Data Mover>

If usermapper is not started, the following command can be used to start the usermapper:

> server_usermapper <movername> -enable where: <movername> = <name of the Data Mover>

For multiple protocol environment, refer to *Configuring VNX User Mapping* on [EMC support site](https://support.emc.com) for additional information.

5. Network Connection

In the current release, the share created by VNX driver uses the first network device (physical port on NIC) of Data Mover to access the network.

Go to Unisphere to check the device list: Settings -> Network -> Settings for File (Unified system only) -> Device.

Backend Configuration

The following parameters need to be configured in */etc/manila/manila.conf* for the VNX driver:

> emc_share_backend = vnx emc_nas_server = <IP address> emc_nas_password = <password> emc_nas_login = <user> emc_nas_server_container = <Data Mover name> emc_nas_pool_name = <pool name> emc_interface_ports = <Comma separated ports list> share_driver = manila.share.drivers.dell_emc.driver.EMCShareDriver driver_handles_share_servers = True

- *emc_share_backend* is the plugin name. Set it to *vnx* for the VNX driver.
- *emc_nas_server* is the control station IP address of the VNX system to be managed.
- *emc_nas_password* and *emc_nas_login* fields are used to provide credentials to the VNX system. Only local users of VNX File is supported.
- *emc_nas_server_container* field is the name of the Data Mover to serve the share service.
- *emc_nas_pool_name* is the pool name user wants to create volume from. The pools can be created using Unisphere for VNX.
- *emc_interface_ports* is comma separated list specifying the ports(devices) of Data Mover that can be used for share server interface. Members of the list can be Unix-style glob expressions (supports Unix shell-style wildcards). This list is optional. In the absence of this option, any of the ports on the Data Mover can be used.
- *driver_handles_share_servers* must be True, the driver will choose a port from port list which configured in emc_interface_ports.

Restart of *manila-share* service is needed for the configuration changes to take effect.

IPv6 support

IPv6 support for VNX driver is introduced in Queens release. The feature is divided into two parts:

1. The driver is able to manage share or snapshot in the Neutron IPv6 network.
2. The driver is able to connect VNX management interface using its IPv6 address.

Pre-Configurations for IPv6 support

The following parameters need to be configured in */etc/manila/manila.conf* for the VNX driver:

    network_plugin_ipv6_enabled = True

- *network_plugin_ipv6_enabled* indicates IPv6 is enabled.

If you want to connect VNX using IPv6 address, you should configure IPv6 address by *nas_cs* command for VNX and specify the address in */etc/manila/manila.conf*:

    emc_nas_server = <IPv6 address>

Snapshot support

In the Mitaka and Newton release of OpenStack, Snapshot support is enabled by default for a newly created share type. Starting with the Ocata release, the snapshot_support extra spec must be set to True in order to allow snapshots for a share type. If the snapshot_support extra_spec is omitted or if it is set to False, users would not be able to create snapshots on shares of this share type. The feature is divided into two parts:

1. The driver is able to create/delete snapshot of share.
2. The driver is able to create share from snapshot.

Pre-Configurations for Snapshot support

The following extra specifications need to be configured with share type.

- snapshot_support = True
- create_share_from_snapshot_support = True

For new share type, these extra specifications can be set directly when creating share type:

```
manila type-create --snapshot_support True --create_share_from_snapshot_
↪support True ${share_type_name} True
```

Or you can update already existing share type with command:

```
manila type-key ${share_type_name} set snapshot_support=True
manila type-key ${share_type_name} set create_share_from_snapshot_support=True
```

To snapshot a share and create share from the snapshot

Firstly, you need create a share from share type that has extra specifications(snapshot_support=True, create_share_from_snapshot_support=True). Then snapshot the share with command:

```
manila snapshot-create ${source_share_name} --name ${target_snapshot_name} --
↪description " "
```

After creating the snapshot from previous step, you can create share from that snapshot. Use command:

```
manila create nfs 1 --name ${target_share_name} --metadata source=snapshot --
↪description " " --snapshot-id ${source_snapshot_id}
```

Restrictions

The VNX driver has the following restrictions:

- Only IP access type is supported for NFS.
- Only user access type is supported for CIFS.
- Only FLAT network and VLAN network are supported.
- VLAN network is supported with limitations. The Neutron subnets in different VLANs that are used to create share networks cannot have overlapped address spaces. Otherwise, VNX may have a problem to communicate with the hosts in the VLANs. To create shares for different VLANs with same subnet address, use different Data Movers.
- The Active Directory security service is the only supported security service type and it is required to create CIFS shares.
- Only one security service can be configured for each share network.
- Active Directory domain name of the active_directory security service should be unique even for different tenants.
- The time on Data Mover and the Active Directory domains used in security services should be in sync (time difference should be less than 10 minutes). It is recommended to use same NTP server on both the Data Mover and Active Directory domains.
- On VNX the snapshot is stored in the SavVols. VNX system allows the space used by SavVol to be created and extended until the sum of the space consumed by all SavVols on the system exceeds the default 20% of the total space available on the system. If the 20% threshold value is reached, an alert will be generated on VNX. Continuing to create snapshot will cause the old snapshot to be inactivated (and the snapshot data to be abandoned). The limit percentage value can be changed manually by storage administrator based on the storage needs. Administrator is recommended to configure the notification on the SavVol usage. Refer to *Using VNX SnapSure* document on [EMC support site](https://support.emc.com) for more information.
- VNX has limitations on the overall numbers of Virtual Data Movers, filesystems, shares, checkpoints, and etc. Virtual Data Mover(VDM) is created by the VNX driver on the VNX to serve as the manila share server. Similarly, filesystem is created, mounted, and exported from the VDM over CIFS or NFS protocol to serve as the manila share. The VNX checkpoint serves as the manila share snapshot. Refer to the *NAS Support Matrix* document on [EMC support site](https://support.emc.com) for the limitations and configure the quotas accordingly.

The manila.share.drivers.dell_emc.driver Module

EMC specific NAS storage driver. This driver is a pluggable driver that allows specific EMC NAS devices to be plugged-in as the underlying backend. Use the Manila configuration variable share_backend_name to specify, which backend plugins to use.

class EMCShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

EMC specific NAS driver. Allows for NFS and CIFS NAS storage usage.

allow_access(*context*, *share*, *access*, *share_server=None*)
Allow access to the share.

check_for_setup_error()
Check for setup error.

create_share(*context*, *share*, *share_server=None*)
Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)
Is called to create share from snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to create snapshot.

delete_share(*context*, *share*, *share_server=None*)
Is called to remove share.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to remove snapshot.

deny_access(*context*, *share*, *access*, *share_server=None*)
Deny access to the share.

do_setup(*context*)
Any initialization the share driver does while starting.

ensure_share(*context*, *share*, *share_server=None*)
Invoked to sure that share is exported.

extend_share(*share*, *new_size*, *share_server=None*)
Is called to extend share.

get_configured_ip_versions()
Get allowed IP versions.

The supported versions are returned with list, possible values are: [4], [6], or [4, 6]

Drivers that assert ipv6_implemented = True must override this method. If the returned list includes 4, then shares created by this driver must have an IPv4 export location. If the list includes 6, then shares created by the driver must have an IPv6 export location.

Drivers should check that their storage controller actually has IPv4/IPv6 enabled and configured properly.

get_default_filter_function()
Get the default filter_function string.

Each driver could overwrite the method to return a well-known default string if it is available.

Parameters **pool** pool name to get the filter or None

Returns None

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

get_share_server_network_info(*context*, *share_server*, *identifier*, *driver_options*)
Obtain network allocations used by share server.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier
- **driver_options** Dictionary of driver options to assist managing the share server

Returns A list containing IP addresses allocated in the backend.

Example:

```
['10.10.10.10', 'fd11::2000', '192.168.10.10']
```

manage_existing(*share*, *driver_options*)
manage an existing share

manage_existing_snapshot(*snapshot*, *driver_options*)
manage an existing share snapshot

manage_existing_snapshot_with_server(*snapshot*, *driver_options*, *share_server=None*)
manage an existing share snapshot

manage_existing_with_server(*share*, *driver_options*, *share_server=None*)
manage an existing share

manage_server(*context*, *share_server*, *identifier*, *driver_options*)
Manage the share server and return compiled back end details.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier
- **driver_options** Dictionary of driver options to assist managing the share server

Returns Identifier and dictionary with back end details to be saved in the database.

Example:

```
'my_new_server_identifier',{'server_name': 'my_old_server'}
```

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a share (in place) to the specified snapshot.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **snapshot** The snapshot to be restored
- **share_access_rules** List of all access rules for the affected share
- **snapshot_access_rules** List of all access rules for the affected snapshot
- **share_server** Optional Share server model or None

shrink_share(*share*, *new_size*, *share_server=None*)

Is called to shrink share.

unmanage(*share*)

Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to False.

unmanage_server(*server_details*, *security_services=None*)

Unmanages the share server.

If a driver supports unmanaging of share servers, the driver must override this method and return successfully.

Parameters

- **server_details** share server backend details.
- **security_services** list of security services configured with this share server.

unmanage_snapshot(*snapshot*)

Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to False.

unmanage_snapshot_with_server(*snapshot*, *share_server=None*)
Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to True.

unmanage_with_server(*share*, *share_server=None*)
Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to True.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access to the share.

The `manila.share.drivers.dell_emc.plugins.vnx.connection` Module

VNX backend for the EMC Manila driver.

class VNXStorageConnection(**args*, ***kwargs*)
Bases: `manila.share.drivers.dell_emc.plugins.base.StorageConnection`

Implements VNX specific functionality for EMC Manila driver.

allow_access(*context*, *share*, *access*, *share_server=None*)
Allow access to a share.

check_for_setup_error()
Check for setup error.

clear_access(*share*, *share_server*, *white_list*)

connect(*emc_share_driver*, *context*)
Connect to VNX NAS server.

create_share(*context*, *share*, *share_server=None*)
Create a share and export it based on protocol used.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)
Create a share from a snapshot - clone a snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Create snapshot from share.

delete_share(*context*, *share*, *share_server=None*)
Delete a share.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Delete a snapshot.

deny_access(*context*, *share*, *access*, *share_server=None*)
Deny access to a share.

ensure_share(*context*, *share*, *share_server=None*)
Ensure that the share is exported.

extend_share(*share*, *new_size*, *share_server=None*)
Invoked to extend share.

get_managed_ports()

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

get_pool(*share*)
Get the pool name of the share.

setup_server(*network_info*, *metadata=None*)
Set up and configures share server with given network parameters.

teardown_server(*server_details*, *security_services=None*)
Teardown share server.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given share.

update_share_stats(*stats_dict*)
Communicate with EMCNASClient to get the stats.

Dell EMC Unity driver

The EMC Shared File Systems service driver framework (EMCShareDriver) utilizes the EMC storage products to provide the shared file systems to OpenStack. The EMC driver is a plug-in based driver which is designed to use different plug-ins to manage different EMC storage products.

The Unity plug-in manages the Unity system to provide shared filesystems. The EMC driver framework with the Unity plug-in is referred to as the Unity driver in this document.

This driver performs the operations on Unity through RESTful APIs. Each backend manages one Storage Processor of Unity. Configure multiple Shared File Systems service backends to manage multiple Unity systems.

Requirements

- Unity OE 4.1.x or higher.
- StorOps 1.1.0 or higher is installed on Manila node.
- Following licenses are activated on Unity:
  - CIFS/SMB Support
  - Network File System (NFS)
  - Thin Provisioning
  - Fiber Channel (FC)
  - Internet Small Computer System Interface (iSCSI)

Supported shared filesystems and operations

In detail, users are allowed to do following operation with EMC Unity Storage Systems.

- Create/delete a NFS share.
- Create/delete a CIFS share.
- Extend the size of a share.
- Shrink the size of a share.
- Modify the host access privilege of a NFS share.
- Modify the user access privilege of a CIFS share.
- Create/Delete snapshot of a share.
- Create a new share from snapshot.
- Revert a share to a snapshot.
- Manage/Unmanage a share server.
- Manage/Unmanage a share.
- Manage/Unmanage a snapshot.

Supported Network Topologies

- Flat

  This type is fully supported by Unity share driver, however flat networks are restricted due to the limited number of tenant networks that can be created from them.

- VLAN

  We recommend this type of network topology in Manila. In most use cases, VLAN is used to isolate the different tenants and provide an isolated network for each tenant. To support this function, an administrator needs to set a slot connected with Unity Ethernet port in `Trunk` mode or allow multiple VLANs from the slot.

- VXLAN

  Unity native VXLAN is still unavailable. However, with the HPB (Hierarchical Port Binding) in Networking and Shared file system services, it is possible that Unity co-exists with VXLAN enabled network environment.

Pre-Configurations

On Manila Node

Python library `storops` is required to run Unity driver. Install it with the `pip` command. You may need root privilege to install python libraries.

```
$ pip install storops
```

On Unity System

1. Configure system level NTP server.

   Open `Unisphere` of your Unity system and navigate to:

   ```
   Unisphere -> Settings -> Management -> System Time and NTP
   ```

   Select `Enable NTP synchronization` and add your NTP server(s).

   The time on the Unity system and the Active Directory domains used in security services should be in sync. We recommend using the same NTP server on both the Unity system and Active Directory domains.

2. Configure system level DNS server.

   Open `Unisphere` of your Unity system and navigate to:

   ```
   Unisphere -> Settings -> Management -> DNS Server
   ```

   Select `Configure DNS server address manually` and add your DNS server(s).

Backend configurations

Following configurations need to be configured in */etc/manila/manila.conf* for the Unity driver.

```
share_driver = manila.share.drivers.dell_emc.driver.EMCShareDriver
emc_share_backend = unity
emc_nas_server = <management IP address of the Unity system>
emc_nas_login = <user with administrator privilege>
emc_nas_password = <password>
unity_server_meta_pool = <pool name>
unity_share_data_pools = <comma separated pool names>
unity_ethernet_ports = <comma separated ports list>
driver_handles_share_servers = True/False
unity_share_server = <name of NAS server in Unity system>
report_default_filter_function = True/False
```

- ***emc_share_backend*** The plugin name. Set it to *unity* for the Unity driver.
- ***emc_nas_server*** The management IP for Unity.
- ***unity_server_meta_pool*** The name of the pool to persist the meta-data of NAS server. This option is required.
- ***unity_share_data_pools*** Comma separated list specifying the name of the pools to be used by this backend. Do not set this option if all storage pools on the system can be used. Wild card character is supported.

  Examples:

  ```
  # Only use pool_1
  unity_share_data_pools = pool_1
  # Only use pools whose name stars from pool_
  unity_share_data_pools = pool_*
  # Use all pools on Unity
  unity_share_data_pools = *
  ```

- ***unity_ethernet_ports*** Comma separated list specifying the ethernet ports of Unity system that can be used for share. Do not set this option if all ethernet ports can be used. Wild card character is supported. Both the normal ethernet port and link aggregation port can be used by Unity share driver.

  Examples:

  ```
  # Only use spa_eth1
  unity_ethernet_ports = spa_eth1
  # Use port whose name stars from spa_
  unity_ethernet_ports = spa_*
  # Use all Link Aggregation ports
  unity_ethernet_ports = sp*_la_*
  # Use all available ports
  unity_ethernet_ports = *
  ```

- ***driver_handles_share_servers*** Unity driver requires this option to be as *True* or *False*. Need to set *unity_share_server* when the value is *False*.

- ***unity_share_server*** One of NAS server names in Unity, it is used for share creation when the driver is in *DHSS=False* mode.
- ***report_default_filter_function*** Whether or not report default filter function. Default value is False. However, this value will be changed to True in a future release to ensure compliance with design expectations in Manila. So we recommend always setting this option in your deployment to True or False per your desired behavior.

Restart of *manila-share* service is needed for the configuration changes to take effect.

Supported MTU size

Unity currently only supports 1500 and 9000 as the mtu size, the user can change the above mtu size from Unity Unisphere:

1. In the Unisphere, go to *Settings*, *Access*, and then *Ethernet*.
2. Double click the ethernet port.
3. Select the *MTU* size from the drop down list.

The Unity driver will select the port where mtu is equal to the mtu of share network during share server creation.

IPv6 support

IPv6 support for Unity driver is introduced in Queens release. The feature is divided into two parts:

1. The driver is able to manage share or snapshot in the Neutron IPv6 network.
2. The driver is able to connect Unity management interface using its IPv6 address.

Pre-Configurations for IPv6 support

The following parameters need to be configured in */etc/manila/manila.conf* for the Unity driver:

```
network_plugin_ipv6_enabled = True
```

- *network_plugin_ipv6_enabled* indicates IPv6 is enabled.

If you want to connect Unity using IPv6 address, you should configure IPv6 address by */net/if/mgmt* uemcli command, *mgmtInterfaceSettings* RESTful api or the system settings of Unity GUI for Unity and specify the address in */etc/manila/manila.conf*:

```
emc_nas_server = <IPv6 address>
```

Supported share creation in mode that driver does not create and destroy share servers (DHSS=False)

To create a file share in this mode, you need to:

1. Create NAS server with network interface in Unity system.
2. Set driver_handles_share_servers=False and unity_share_server in `/etc/manila/manila.conf`:

```
driver_handles_share_servers = False
unity_share_server = <name of NAS server in Unity system>
```

3. Specify the share type with driver_handles_share_servers = False extra specification:

```
$ manila type-create ${share_type_name} False
```

4. Create share.

```
$ manila create ${share_protocol} ${size} --name ${share_name} --
↪share-type ${share_type_name}
```

Note: Do not specify the share network in share creation command because no share servers will be created. Driver will use the unity_share_server specified for share creation.

Snapshot support

In the Mitaka and Newton release of OpenStack, Snapshot support is enabled by default for a newly created share type. Starting with the Ocata release, the snapshot_support extra spec must be set to True in order to allow snapshots for a share type. If the snapshot_support extra_spec is omitted or if it is set to False, users would not be able to create snapshots on shares of this share type. The feature is divided into two parts:

1. The driver is able to create/delete snapshot of share.
2. The driver is able to create share from snapshot.

Pre-Configurations for Snapshot support

The following extra specifications need to be configured with share type.

- snapshot_support = True
- create_share_from_snapshot_support = True

For new share type, these extra specifications can be set directly when creating share type:

```
$ manila type-create --snapshot_support True --create_share_from_snapshot_
↪support True ${share_type_name} True
```

Or you can update already existing share type with command:

```
$ manila type-key ${share_type_name} set snapshot_support=True
$ manila type-key ${share_type_name} set create_share_from_snapshot_
↪support=True
```

To snapshot a share and create share from the snapshot

Firstly, you need create a share from share type that has extra specifications (snapshot_support=True, create_share_from_snapshot_support=True). Then snapshot the share with command:

```
$ manila snapshot-create ${source_share_name} --name ${target_snapshot_name} -
↪-description " "
```

After creating the snapshot from previous step, you can create share from that snapshot. Use command:

```
$ manila create nfs 1 --name ${target_share_name} --metadata source=snapshot -
↪-description " " --snapshot-id ${source_snapshot_id}
```

To manage an existing share server

To manage a share server existing in Unity System, you need to:

1. Create network, subnet, port (ip address of nas server in Unity system) and share network in OpenStack.

```
$ openstack network create ${network_name} --provider-network-
↪type ${network_type}
$ openstack subnet create ${subnet_name} --network ${network_
↪name} --subnet-range ${subnet_range}
$ openstack port create --network ${network_name} --fixed-ip␣
↪subnet=${subnet_name},ip-address=${ip address} \
  ${port_name} --device-owner=manila:share
$ manila share-network-create --name ${share_network_name} --
↪neutron-net-id ${network_name} \
  --neutron-subnet-id ${subnet_name}
```

2. Manage the share server in OpenStack:

```
$ manila share-server-manage ${host} ${share_network_name} $
↪{identifier}
```

Note: ${identifier} is the nas server name in Unity system.

To un-manage a Manila share server

To unmanage a share server existing in OpenStack:

```
$ manila share-server-unmanage ${share_server_id}
```

To manage an existing share

To manage a share existing in Unity System:

- In DHSS=True mode

  Need make sure the related share server is existing in OpenStack, otherwise need to manage share server first (check the step of Supported Manage share server).

  ```
  $ manila manage ${service_host} ${protocol} '${export_path}' --
  ↪name ${share_name} --driver_options size=${share_size} \
    --share_type ${share_type} --share_server_id ${share_server_id}
  ```

  Note: ${share_server_id} is the id of share server in OpenStack. ${share_type} should have the property driver_handles_share_servers=True.

- In DHSS=False mode

  ```
  $ manila manage ${service_host} ${protocol} '${export_path}' --
  ↪name ${share_name} --driver_options size=${share_size} \
    --share_type ${share_type}
  ```

  Note: ${share_type} should have the property driver_handles_share_servers=False.

To un-manage a Manila share

To unmanage a share existing in OpenStack:

```
$ manila unmanage ${share_id}
```

To manage an existing share snapshot

To manage a snapshot existing in Unity System, you need make sure the related share instance is existing in OpenStack, otherwise need to manage share first (check the step of Supported Manage share).

```
$ manila snapshot-manage --name ${name} ${share_name} ${provider_
↪location} --driver_options size=${snapshot_size}
```

Note: ${provider_location} is the snapshot name in Unity system. ${share_name} is the share name or id in OpenStack.

To un-manage a Manila share snapshot

To unmanage a snapshot existing in OpenStack:

```
$ manila snapshot-unmanage ${snapshot_id}
```

Supported security services

Unity share driver provides `IP` based authentication method support for `NFS` shares and `user` based authentication method for `CIFS` shares respectively. For `CIFS` share, Microsoft Active Directory is the only supported security service.

IO Load balance

The Unity driver automatically distributes the file interfaces per storage processor based on the option `unity_ethernet_ports`. This balances IO traffic. The recommended configuration for `unity_ethernet_ports` specifies balanced ports per storage processor. For example:

```
# Use eth2 from both SPs
unity_ethernet_ports = spa_eth2, spb_eth2
```

Default filter function

Unity does not support the file system creation with size smaller than 3GB, if the size of share user create is smaller than 3GB, Unity driver will supplement the size to 3GB in Unity.

Unity driver implemented the get_default_filter_function API to report the default filter function, if the share size is smaller than 3GB, Manila will not schedule the share creation to Unity backend.

Unity driver provides an option `report_default_filter_function` to disable or enable the filter function reporting, the default value is disabled.

Restrictions

The Unity driver has following restrictions.

- EMC Unity does not support the same IP in different VLANs.
- Only IP access type is supported for NFS.
- Only user access type is supported for CIFS.

API Implementations

Following driver features are implemented in the plugin.

- create_share: Create a share and export it based on the protocol used (NFS or CIFS).
- create_share_from_snapshot: Create a share from a snapshot - clone a snapshot.
- delete_share: Delete a share.
- extend_share: Extend the maximum size of a share.
- shrink_share: Shrink the minimum size of a share.
- create_snapshot: Create a snapshot for the specified share.
- delete_snapshot: Delete the snapshot of the share.
- update_access: recover, add or delete user/host access to a share.
- allow_access: Allow access (read write/read only) of a user to a CIFS share. Allow access (read write/read only) of a host to a NFS share.
- deny_access: Remove access (read write/read only) of a user from a CIFS share. Remove access (read write/read only) of a host from a NFS share.
- ensure_share: Check whether share exists or not.
- update_share_stats: Retrieve share related statistics from Unity.
- get_network_allocations_number: Returns number of network allocations for creating VIFs.
- setup_server: Set up and configures share server with given network parameters.
- teardown_server: Tear down the share server.
- revert_to_snapshot: Revert a share to a snapshot.
- get_default_filter_function: Report a default filter function.

Driver options

Configuration options specific to this driver:

Table 8: Description of Dell EMC Unity share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`unity_ethernet_ports` = `None`	(List) Comma separated list of ports that can be used for share server interfaces. Members of the list can be Unix-style glob expressions.
`unity_server_meta_pool` = `None`	(String) Pool to persist the meta-data of NAS server.
`unity_share_data_pools` = `None`	(List) Comma separated list of pools that can be used to persist share data.
`unity_share_server` = `None`	One of NAS server names in Unity, it is used for share creation when the driver is in `DHSS=False` mode..

Generic approach for share provisioning

The Shared File Systems service can be configured to use Nova VMs and Cinder volumes. There are two modules that handle them in manila: 1) service_instance module creates VMs in Nova with predefined image called service image. This module can be used by any backend driver for provisioning of service VMs to be able to separate share resources among tenants. 2) generic module operates with Cinder volumes and VMs created by service_instance module, then creates shared filesystems based on volumes attached to VMs.

Network configurations

Each backend driver can handle networking in its own way, see: https://wiki.openstack.org/wiki/Manila/Networking

One of two possible configurations can be chosen for share provisioning using service_instance module:

- Service VM has one net interface from net that is connected to public router. For successful creation of share, user network should be connected to public router too.
- Service VM has two net interfaces, first one connected to service network, second one connected directly to users network.

Requirements for service image

- Linux based distro
- NFS server
- Samba server >=3.2.0, that can be configured by data stored in registry
- SSH server
- Two net interfaces configured to DHCP (see network approaches)
- exportfs and net conf libraries used for share actions
- **Following files will be used, so if their paths differ one needs to create at** least symlinks for them:
  - /etc/exports (permanent file with NFS exports)
  - /var/lib/nfs/etab (temporary file with NFS exports used by exportfs)
  - /etc/fstab (permanent file with mounted filesystems)
  - /etc/mtab (temporary file with mounted filesystems used by mount)

Supported shared filesystems

- NFS (access by IP)
- CIFS (access by IP)

Known restrictions

- One of Novas configurations only allows 26 shares per server. This limit comes from the maximum number of virtual PCI interfaces that are used for block device attaching. There are 28 virtual PCI interfaces, in this configuration, two of them are used for server needs and other 26 are used for attaching block devices that are used for shares.
- Juno version works only with Neutron. Each share should be created with neutron-net and neutron-subnet IDs provided via share-network entity.
- Juno version handles security group, flavor, image, keypair for Nova VM and also creates service networks, but does not use availability zones for Nova VMs and volume types for Cinder block devices.
- Juno version does not use security services data provided with share-network. These data will be just ignored.
- Liberty version adds a share extend capability. Share access will be briefly interrupted during an extend operation.
- Liberty version adds a share shrink capability, but this capability is not effective because generic driver shrinks only filesystem size and doesnt shrink the size of Cinder volume.
- Modifying network-related configuration options, such as `service_network_cidr` or `service_network_division_mask`, after manila has already created some shares using those options is not supported.

Using Windows instances

While the generic driver only supports Linux instances, you may use the Windows SMB driver when Windows VMs are preferred.

For more details, please check out the following page: *Windows SMB driver*.

The `manila.share.drivers.generic` Module

Generic Driver for shares.

class GenericShareDriver(**args*, ***kwargs*)

Bases: `manila.share.driver.ExecuteMixin`, *manila.share.driver.ShareDriver*

Executes commands relating to Shares.

check_for_setup_error()

Returns an error if prerequisites arent met.

create_share(*context*, **args*, ***kwargs*)

Is called to create share.

create_share_from_snapshot(*context*, **args*, ***kwargs*)
Is called to create share from snapshot.

Creating a share from snapshot can take longer than a simple clone operation if data copy is required from one host to another. For this reason driver will be able complete this creation asynchronously, by providing a creating_from_snapshot status in the model update.

When answering asynchronously, drivers must implement the call get_share_status in order to provide updates for shares with creating_from_snapshot status.

It is expected that the driver returns a model update to the share manager that contains: share status and a list of export_locations. A list of export_locations is mandatory only for share in available status. The current supported status are available and creating_from_snapshot.

Parameters

- **context** Current context
- **share** Share instance model with share data.
- **snapshot** Snapshot instance model .
- **share_server** Share server model or None.
- **parent_share** Share model from parent snapshot with share data and share server model.

Returns

a dictionary of updates containing current share status and its export_location (if available).

Example:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance to error and the operation will end.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Creates a snapshot.

delete_share(*context*, *share*, *share_server=None*)
Deletes share.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Deletes a snapshot.

do_setup(*context*)
Any initialization the generic driver does while starting.

ensure_share(*context*, **args*, ***kwargs*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

extend_share(*context*, **args*, ***kwargs*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

get_network_allocations_number()
Get number of network interfaces to be created.

manage_existing(*share*, *driver_options*)
Manage existing share to manila.

Generic driver accepts only one driver_option volume_id. If an administrator provides this option, then appropriate Cinder volume will be managed by Manila as well.

Parameters

- **share** share data
- **driver_options** Empty dict or dict with volume_id option.

Returns dict with share size, example: {size: 1}

manage_existing_snapshot(*snapshot*, *driver_options*)
Manage existing share snapshot with manila.

Parameters

- **snapshot** Snapshot data
- **driver_options** Not used by the Generic driver currently

Returns dict with share snapshot size, example: {size: 1}

shrink_share(*context*, **args*, ***kwargs*)
Shrinks size of existing share.

If consumed space on share larger than new_size driver should raise ShareShrinkingPossibleDataLoss exception: raise ShareShrinkingPossibleDataLoss(share_id=share[id])

Parameters

- **share** Share model
- **new_size** New size of share (new_size < share[size])
- **share_server** Optional Share server model

:raises ShareShrinkingPossibleDataLoss, NotImplementedError

unmanage_snapshot(*snapshot*)
Unmanage share snapshot with manila.

update_access(*context*, **args*, ***kwargs*)
Update access rules for given share.

`access_rules` contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the `add_rules` and `delete_rules` parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in `add_rules` and rules that need to be removed to be present in `delete_rules`.

When a rule in `delete_rules` was never applied, drivers must not raise an exception, or attempt to set the rule to `error` state.

`add_rules` and `delete_rules` can be empty lists, in this situation, drivers should ensure that the rules present in `access_rules` are the same as those on the back end. One scenario where this situation is forced is when the access_level is changed for all existing rules (share migration and for readable replicas).

Drivers must be mindful of this call for share replicas. When update_access is called on one of the replicas, the call is likely propagated to all replicas belonging to the share, especially when individual rules are added or removed. If a particular access rule does not make sense to the driver in the context of a given replica, the driver should be careful to report a correct behavior, and take meaningful action. For example, if R/W access is requested on a replica that is part of a readable type replication; R/O access may be added by the driver instead of R/W. Note that raising an exception *will* result in the access_rules_status on the replica, and the share itself being out_of_sync. Drivers can sync on the valid access rules that are provided on the `create_replica` and `promote_replica` calls.

Parameters

- **`context`** Current context
- **`share`** Share model with share data.
- **`access_rules`** A list of access rules for given share
- **`add_rules`** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **`delete_rules`** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **`share_server`** None or Share server model

Returns

None, or a dictionary of updates in the format:

```
{
```

09960614-8574-4e03-89cf-7cf267b0bd08: {

access_key: alice31493e5441b8171d2310d80e37e, state: error,

},

28f6eabb-4342-486a-a7f4-45688f0c0295: {

access_key: bob0078aa042d5a7325480fd13228b, state: active,

},

}

The top level keys are access_id fields of the access rules that need to be updated. `access_key``s are credentials (str) of the entities granted access. Any rule in the ``access_rules` parameter can be updated.

Important: Raising an exception in this method will force *all* rules in applying and denying states to error.

An access rule can be set to error state, either explicitly via this return parameter or because of an exception raised in this method. Such an access rule will no longer be sent to the driver on subsequent access rule updates. When users deny that rule however, the driver will be asked to deny access to the client/s represented by the rule. We expect that a rule that was error-ed at the driver should never exist on the back end. So, do not fail the deletion request.

Also, it is possible that the driver may receive a request to add a rule that is already present on the back end. This can happen if the share manager service goes down while the driver is committing access rule changes. Since we cannot determine if the rule was applied successfully by the driver before the disruption, we will treat all applying transitional rules as new rules and repeat the request.

ensure_server(*f*)

The manila.share.drivers.service_instance Module

Module for managing nova instances for share drivers.

class BaseNetworkhelper(*service_instance_manager*)
Bases: object

abstract property NAME
Returns code name of network helper.

abstract get_network_name(*network_info*)
Returns name of network for service instance.

abstract setup_connectivity_with_service_instances()
Sets up connectivity between Manila host and service instances.

abstract setup_network(*network_info*)
Sets up network for service instance.

abstract teardown_network(*server_details*)
Teardowns network resources provided for service instance.

class NeutronNetworkHelper(*service_instance_manager*)
Bases: manila.share.drivers.service_instance.BaseNetworkhelper

property NAME
Returns code name of network helper.

property admin_project_id

get_network_name(*network_info*)
Returns name of network for service instance.

property neutron_api

property service_network_id

setup_connectivity_with_service_instances()
Sets up connectivity with service instances.

Creates host port in service network and/or admin network, creating and setting up required network devices.

setup_network(*network_info*)
Sets up network for service instance.

teardown_network(*server_details*)
Teardowns network resources provided for service instance.

class ServiceInstanceManager(*driver_config=None*)
Bases: `object`

Manages nova instances for various share drivers.

This class provides following external methods:

1. set_up_service_instance: creates instance and sets up share infrastructure.
2. ensure_service_instance: ensure service instance is available.
3. delete_service_instance: removes service instance and network infrastructure.

delete_service_instance(*context*, *server_details*)
Removes share infrastructure.

Deletes service vm and subnet, associated to share network.

ensure_service_instance(*context*, *server*)
Ensures that server exists and active.

get_common_server()

get_config_option(*key*)
Returns value of config option.

Parameters **key** key of config option.

Returns str value of configs option. first priority is drivers config, second priority is global config.

property network_helper

reboot_server(*server*, *soft_reboot=False*)

set_up_service_instance(*context*, *network_info*)
Finds or creates and sets up service vm.

Parameters

- **context** defines context, that should be used
- **network_info** network info for getting allocations

Returns dict with service instance details

Raises exception.ServiceInstanceException

wait_for_instance_to_be_active(*instance_id*, *timeout*)

GlusterFS driver

GlusterFS driver uses GlusterFS, an open source distributed file system, as the storage backend for serving file shares to manila clients.

Supported shared filesystems

- NFS (access by IP)

Supported Operations

- Create share
- Delete share
- Allow share access (rw)
- Deny share access
- With volume layout:
  - Create snapshot
  - Delete snapshot
  - Create share from snapshot

Requirements

- Install glusterfs-server package, version >= 3.5.x, on the storage backend.
- Install NFS-Ganesha, version >=2.1, if using NFS-Ganesha as the NFS server for the GlusterFS backend.
- Install glusterfs and glusterfs-fuse package, version >=3.5.x, on the manila host.
- Establish network connection between the manila host and the storage backend.

Manila driver configuration setting

The following parameters in the manilas configuration file need to be set:

- *share_driver* = manila.share.drivers.glusterfs.GlusterfsShareDriver

The following configuration parameters are optional:

- ***glusterfs_nfs_server_type* = <NFS server type used by the GlusterFS** backend, *Gluster* or *Ganesha*. *Gluster* is the default type>
- *glusterfs_share_layout* = <share layout used>; cf. *Layouts*
- *glusterfs_path_to_private_key* = <path to manila hosts private key file>
- *glusterfs_server_password* = <password of remote GlusterFS server machine>

If Ganesha NFS server is used (`glusterfs_nfs_server_type = Ganesha`), then by default the Ganesha server is supposed to run on the manila host and is managed by local commands. If its deployed somewhere else, then its managed via ssh, which can be configured by the following parameters:

- *glusterfs_ganesha_server_ip*
- *glusterfs_ganesha_server_username*
- *glusterfs_ganesha_server_password*

In lack of `glusterfs_ganesha_server_password` ssh access will fall back to key based authentication, using the key specified by `glusterfs_path_to_private_key`, or, in lack of that, a key at one of the OpenSSH-style default key locations (*~/.ssh/id_{r,d,ecd}sa*).

Layouts have also their set of parameters, see *Layouts* about that.

Layouts

New in Liberty, multiple share layouts can be used with glusterfs driver. A layout is a strategy of allocating storage from GlusterFS backends for shares. Currently there are two layouts implemented:

- *directory mapped layout* (or *directory layout*, or *dir layout* for short): a share is backed by top-level subdirectories of a given GlusterFS volume.

  Directory mapped layout is the default and backward compatible with Kilo. The following setting explicitly specifies its usage: `glusterfs_share_layout = layout_directory.GlusterfsDirectoryMappedLayout`.

  Options:

  - *glusterfs_target*: address of the volume that hosts the directories. If its of the format <*glustervolserver*>*:/*<*glustervolid*>, then the manila host is expected to be part of the GlusterFS cluster of the volume and GlusterFS management happens through locally calling the `gluster` utility. If its of the format <*username*>@<*glustervolserver*>*:/*<*glustervolid*>, then we ssh to <*username*>@<*glustervolserver*> to execute `gluster` (<*username*> is supposed to have administrative privileges on <*glustervolserver*>).
  - ***glusterfs_mount_point_base* = <base path of GlusterFS volume mounted on** manila host> (optional; defaults to *$state_path*`/mnt`, where *$state_path* defaults to `/var/lib/manila`)

  Limitations:

  - directory layout does not support snapshot operations.

- *volume mapped layout* (or *volume layout*, or *vol layout* for short): a share is backed by a whole GlusterFS volume.

  Volume mapped layout is new in Liberty. It can be chosen by setting `glusterfs_share_layout = layout_volume.GlusterfsVolumeMappedLayout`.

  Options (required):

  - *glusterfs_servers*
  - *glusterfs_volume_pattern*

Volume mapped layout is implemented as a common backend of the glusterfs and glusterfs-native drivers; see the description of these options in *GlusterFS Native driver*: *Manila driver configuration setting*.

Gluster NFS with volume mapped layout

A special configuration choice is

```
glusterfs_nfs_server_type = Gluster
glusterfs_share_layout = layout_volume.GlusterfsVolumeMappedLayout
```

that is, Gluster NFS used to export whole volumes.

All other GlusterFS backend configurations (including GlusterFS set up with glusterfs-native) require the `nfs.export-volumes = off` GlusterFS setting. Gluster NFS with volume layout requires `nfs.export-volumes = on`. `nfs.export-volumes` is a *cluster-wide* setting, so a given GlusterFS cluster cannot host a share backend with Gluster NFS + volume layout and other share backend configurations at the same time.

There is another caveat with `nfs.export-volumes`: setting it to `on` without enough care is a security risk, as the default access control for the volume exports is allow all. For this reason, while the `nfs.export-volumes = off` setting is automatically set by manila for all other share backend configurations, `nfs.export-volumes = on` is *not* set by manila in case of a Gluster NFS with volume layout setup. Its left to the GlusterFS admin to make this setting in conjunction with the associated safeguards (that is, for those volumes of the cluster which are not used by manila, access restrictions have to be manually configured through the `nfs.rpc-auth-{allow,reject}` options).

Known Restrictions

- The driver does not support network segmented multi-tenancy model, but instead works over a flat network, where the tenants share a network.
- If NFS Ganesha is the NFS server used by the GlusterFS backend, then the shares can be accessed by NFSv3 and v4 protocols. However, if Gluster NFS is used by the GlusterFS backend, then the shares can only be accessed by NFSv3 protocol.
- All manila shares, which map to subdirectories within a GlusterFS volume, are currently created within a single GlusterFS volume of a GlusterFS storage pool.
- The driver does not provide read-only access level for shares.
- Assume that share S is exported through Gluster NFS, and tenant machine T has mounted S. If at this point access of T to S is revoked through *access-deny*, the pre-existing mount will be still usable and T will still be able to access the data in S as long as that mount is in place. (This violates the principle *Access deny should always result in immediate loss of access to the share*, see http://lists.openstack.org/pipermail/openstack-dev/2015-July/069109.html.)

The manila.share.drivers.glusterfs Module

Flat network GlusterFS Driver.

Manila shares are subdirectories within a GlusterFS volume. The backend, a GlusterFS cluster, uses one of the two NFS servers, Gluster-NFS or NFS-Ganesha, based on a configuration option, to mediate access to the shares. NFS-Ganesha server supports NFSv3 and v4 protocols, while Gluster-NFS server supports only NFSv3 protocol.

TODO(rraja): support SMB protocol.

class GaneshaNFSHelper(*execute*, *config_object*, ***kwargs*)

Bases: manila.share.drivers.ganesha.GaneshaNASHelper

get_export(*share*)

init_helper()

Initializes protocol-specific NAS drivers.

shared_data = {}

class GlusterNFSHelper(*execute*, *config_object*, ***kwargs*)

Bases: manila.share.drivers.ganesha.NASHelperBase

Manage shares with Gluster-NFS server.

get_export(*share*)

supported_access_levels = ('rw',)

supported_access_types = ('ip',)

update_access(*base_path*, *share*, *add_rules*, *delete_rules*, *recovery=False*)

Update access rules.

class GlusterNFSVolHelper(*execute*, *config_object*, ***kwargs*)

Bases: manila.share.drivers.glusterfs.GlusterNFSHelper

Manage shares with Gluster-NFS server, volume mapped variant.

update_access(*base_path*, *share*, *add_rules*, *delete_rules*, *recovery=False*)

Update access rules.

class GlusterfsShareDriver(**args*, ***kwargs*)

Bases: manila.share.driver.ExecuteMixin, manila.share.driver.GaneshaMixin, manila.share.drivers.glusterfs.layout.GlusterfsShareDriverBase

Execute commands relating to Shares.

GLUSTERFS_VERSION_MIN = (3, 5)

check_for_setup_error()

Check for setup error.

do_setup(*context*)

Any initialization the share driver does while starting.

get_network_allocations_number()

Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

property supported_access_levels

property supported_access_types

supported_layouts = ('layout_directory.GlusterfsDirectoryMappedLayout', 'layout_volume.GlusterfsVolumeMappedLayout')

supported_protocols = ('NFS',)

GlusterFS Native driver

GlusterFS Native driver uses GlusterFS, an open source distributed file system, as the storage backend for serving file shares to manila clients.

A manila share is a GlusterFS volume. This driver uses flat-network (share-server-less) model. Instances directly talk with the GlusterFS backend storage pool. The instances use glusterfs protocol to mount the GlusterFS shares. Access to each share is allowed via TLS Certificates. Only the instance which has the TLS trust established with the GlusterFS backend can mount and hence use the share. Currently only rw access is supported.

Network Approach

L3 connectivity between the storage backend and the host running the manila share service should exist.

Supported shared filesystems

- GlusterFS (share protocol: `glusterfs`, access by TLS certificates (`cert` access type))

Multi-tenancy model

The driver does not support network segmented multi-tenancy model. Instead multi-tenancy is supported using tenant specific TLS certificates.

Supported Operations

- Create share
- Delete share
- Allow share access (rw)
- Deny share access
- Create snapshot
- Delete snapshot

- Create share from snapshot

Requirements

- Install glusterfs-server package, version >= 3.6.x, on the storage backend.
- Install glusterfs and glusterfs-fuse package, version >=3.6.x, on the manila host.
- Establish network connection between the manila host and the storage backend.

Manila driver configuration setting

The following parameters in manilas configuration file need to be set:

- ***share_driver*** = manila.share.drivers.glusterfs.glusterfs_native.GlusterfsNativeShareDriver
- ***glusterfs_servers*** = **List of GlusterFS servers which provide volumes** that can be used to create shares. The servers are expected to be of distinct Gluster clusters (ie. should not be gluster peers). Each server should be of the form `[<remoteuser>@]<glustervolserver>`.

  The optional `<remoteuser>@` part of the server URI indicates SSH access for cluster management (see related optional parameters below). If it is not given, direct command line management is performed (ie. manila host is assumed to be part of the GlusterFS cluster the server belongs to).
- ***glusterfs_volume_pattern*** = **Regular expression template** used to filter GlusterFS volumes for share creation. The regex template can contain the #{size} parameter which matches a number (sequence of digits) and the value shall be interpreted as size of the volume in GB. Examples: `manila-share-volume-\d+$`, `manila-share-volume-#{size}G-\d+$`; with matching volume names, respectively: *manila-share-volume-12*, *manila-share-volume-3G-13*. In latter example, the number that matches `#{size}`, that is, 3, is an indication that the size of volume is 3G.

The following configuration parameters are optional:

- ***glusterfs_mount_point_base*** = **<base path of GlusterFS volume mounted on** manila host>
- *glusterfs_path_to_private_key* = <path to manila hosts private key file>
- *glusterfs_server_password* = <password of remote GlusterFS server machine>

Host and backend configuration

- SSL/TLS should be enabled on the I/O path for GlusterFS servers and volumes involved (ie. ones specified in `glusterfs_servers`), as described in https://docs.gluster.org/en/latest/Administrator%20Guide/SSL/. (Enabling SSL/TLS for the management path is also possible but not recommended currently.)
- The manila host should be also configured for GlusterFS SSL/TLS (ie. */etc/ssl/glusterfs.{pem,key,ca}* files has to be deployed as the above document specifies).
- There is a further requirement for the CA-s used: the set of CA-s involved should be consensual, ie. */etc/ssl/glusterfs.ca* should be identical across all the servers and the manila host.

- There is a further requirement for the common names (CN-s) of the certificates used: the certificates of the servers should have a common name starting with *glusterfs-server*, and the certificate of the host should have common name starting with *manila-host*.
- To support snapshots, bricks that consist the GlusterFS volumes used by manila should be thinly provisioned LVM ones (cf. https://gluster.readthedocs.org/en/latest/Administrator%20Guide/Managing%20Snapshots/).

Known Restrictions

- GlusterFS volumes are not created on demand. A pre-existing set of GlusterFS volumes should be supplied by the GlusterFS cluster(s), conforming to the naming convention encoded by `glusterfs_volume_pattern`. However, the GlusterFS endpoint is allowed to extend this set any time (so manila and GlusterFS endpoints are expected to communicate volume supply/demand out-of-band). `glusterfs_volume_pattern` can include a size hint (with `#{size}` syntax), which, if present, requires the GlusterFS end to indicate the size of the shares in GB in the name. (On share creation, manila picks volumes *at least* as big as the requested one.)
- Certificate setup (aka trust setup) between instance and storage backend is out of band of manila.
- For manila to use GlusterFS volumes, the name of the trashcan directory in GlusterFS volumes must not be changed from the default.

The `manila.share.drivers.glusterfs.glusterfs_native.GlusterfsNativeShareDriver` Module

GlusterFS native protocol (glusterfs) driver for shares.

Manila share is a GlusterFS volume. Unlike the generic driver, this does not use service VM approach. Instances directly talk with the GlusterFS backend storage pool. Instance use the glusterfs protocol to mount the GlusterFS share. Access to the share is allowed via SSL Certificates. Only the instance which has the SSL trust established with the GlusterFS backend can mount and hence use the share.

Supports working with multiple glusterfs volumes.

class GlusterfsNativeShareDriver(**args*, ***kwargs*)

Bases: `manila.share.driver.ExecuteMixin`, `manila.share.drivers.glusterfs.layout.GlusterfsShareDriverBase`

GlusterFS native protocol (glusterfs) share driver.

Executes commands relating to Shares. Supports working with multiple glusterfs volumes.

API version history:

1.0 - Initial version. 1.1 - Support for working with multiple gluster volumes.

GLUSTERFS_VERSION_MIN = (3, 6)

get_network_allocations_number()

Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

```
supported_layouts = ('layout_volume.GlusterfsVolumeMappedLayout',)
supported_protocols = ('GLUSTERFS',)
```

CephFS driver

The CephFS driver enables manila to export shared filesystems backed by Cephs File System (CephFS) using either the Ceph network protocol or NFS protocol. Guests require a native Ceph client or an NFS client in order to mount the filesystem.

When guests access CephFS using the native Ceph protocol, access is controlled via Cephs cephx authentication system. If a user requests share access for an ID, Ceph creates a corresponding Ceph auth ID and a secret key if they do not already exist, and authorizes the ID to access the share. The client can then mount the share using the ID and the secret key. To learn more about configuring Ceph clients to access the shares created using this driver, please see the Ceph documentation

And when guests access CephFS through NFS, an NFS-Ganesha server mediates access to CephFS. The driver enables access control by managing the NFS-Ganesha servers exports.

Supported Operations

The following operations are supported with CephFS backend:

- Create, delete, update and list share
- Allow/deny access to share
  - Only `cephx` access type is supported for CephFS native protocol.
  - Only `ip` access type is supported for NFS protocol.
  - `read-only` and `read-write` access levels are supported.
- Extend/shrink share
- Create, delete, update and list snapshot
- Create, delete, update and list share groups
- Delete and list share group snapshots

Important: Share group snapshot creation is no longer supported in mainline CephFS. This feature has been removed from manila W release.

Prerequisites

Important: A manila share backed by CephFS is only as good as the underlying filesystem. Take care when configuring your Ceph cluster, and consult the latest guidance on the use of CephFS in the Ceph documentation.

Ceph testing matrix

As Ceph and Manila continue to grow, it is essential to test and support combinations of releases supported by both projects. However, there is little community bandwidth to cover all of them. For simplicity sake, we are focused on testing (and therefore supporting) the current Ceph active releases. Check out the list of Ceph active releases here.

Below is the current state of testing for Ceph releases with this project. Adjacent components such as devstack-plugin-ceph and tripleo are added to the table below. Contributors to those projects can determine what versions of ceph are tested and supported with manila by those components; however, their state is presented here for ease of access.

Important: From the Victoria cycle, the Manila CephFS driver is not tested or supported with Ceph clusters older than Nautilus. Future releases of Manila may be incompatible with Nautilus too! We suggest always running the latest version of Manila with the latest release of Ceph.

OpenStack release	manila	devstack-plugin-ceph	tripleo
Queens	Luminous	Luminous	Luminous
Rocky	Luminous	Luminous	Luminous
Stein	Nautilus	Luminous, Nautilus	Nautilus
Train	Nautilus	Luminous, Nautilus	Nautilus
Ussuri	Nautilus	Luminous, Nautilus	Nautilus
Victoria	Nautilus	Nautilus, Octopus	Nautilus
Wallaby	Octopus	Nautilus, Octopus	Pacific

Additionally, it is expected that the version of the Ceph client available to manila is aligned with the Ceph server version. Mixing server and client versions is strongly unadvised.

In case of using the NFS Ganesha driver, its also a good practice to use the versions that align with the Ceph version of choice.

Important: Its recommended to install the latest stable version of Ceph Nautilus/Octopus/Pacific release. See, Ceph releases

Prior to upgrading to Wallaby, please ensure that youre running at least the following versions of Ceph:

Release	Minimum version
Nautilus	14.2.20
Octopus	15.2.11
Pacific	16.2.1

Common Prerequisites

- A Ceph cluster with a filesystem configured (See Create ceph filesystem on how to create a filesystem.)
- `python3-rados` and `python3-ceph-argparse` packages installed in the servers running the *manila-share* service.
- Network connectivity between your Ceph clusters public network and the servers running the *manila-share* service.

For CephFS native shares

- Ceph client installed in the guest
- Network connectivity between your Ceph clusters public network and guests. See *Security with CephFS native share backend*.

For CephFS NFS shares

- 3.0 or later versions of NFS-Ganesha.
- NFS client installed in the guest.
- Network connectivity between your Ceph clusters public network and NFS-Ganesha server.
- Network connectivity between your NFS-Ganesha server and the manila guest.

Authorizing the driver to communicate with Ceph

Capabilities required for the Ceph manila identity have changed from the Wallaby release. The Ceph manila identity configured no longer needs any MDS capability. The MON and OSD capabilities can be reduced as well. However new MGR capabilities are now required. If not accorded, the driver cannot communicate to the Ceph Cluster.

Important: The driver in the Wallaby (or later) release requires a Ceph identity with a different set of Ceph capabilities when compared to the driver in a pre-Wallaby release.

When upgrading to Wallaby youll also have to update the capabilities of the Ceph identity used by the driver (refer to Ceph user capabilities docs) E.g. a native driver that already uses *client.manila* Ceph identity, issue command *ceph auth caps client.manila mon allow r mgr allow rw*

For the CephFS Native driver, the auth ID should be set as follows:

```
ceph auth get-or-create client.manila -o manila.keyring \
  mgr 'allow rw' \
  mon 'allow r'
```

For the CephFS NFS driver, we use a specific pool to store exports (configurable with the config option ganesha_rados_store_pool_name). We also need to specify osd caps for it. So, the auth ID should be set as follows:

```
ceph auth get-or-create client.manila -o manila.keyring \
  osd 'allow rw pool=<ganesha_rados_store_pool_name>" \
  mgr 'allow rw' \
  mon 'allow r'
```

`manila.keyring`, along with your `ceph.conf` file, will then need to be placed on the server running the *manila-share* service.

Important: To communicate with the Ceph backend, a CephFS driver instance (represented as a backend driver section in manila.conf) requires its own Ceph auth ID that is not used by other CephFS driver instances running in the same controller node.

In the server running the *manila-share* service, you can place the `ceph.conf` and `manila.keyring` files in the /etc/ceph directory. Set the same owner for the *manila-share* process and the `manila.keyring` file. Add the following section to the `ceph.conf` file.

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

It is advisable to modify the Ceph clients admin socket file and log file locations so that they are co-located with manila servicess pid files and log files respectively.

Enabling snapshot support in Ceph backend

CephFS Snapshots were experimental prior to the Nautilus release of Ceph. There may be some limitations on snapshots based on the Ceph version you use.

From Ceph Nautilus, all new filesystems created on Ceph have snapshots enabled by default. If youve upgraded your ceph cluster and want to enable snapshots on a pre-existing filesystem, you can do so:

```
ceph fs set {fs_name} allow_new_snaps true
```

Configuring CephFS backend in manila.conf

Configure CephFS native share backend in manila.conf

Add CephFS to `enabled_share_protocols` (enforced at manila api layer). In this example we leave NFS and CIFS enabled, although you can remove these if you will only use a CephFS backend:

```
enabled_share_protocols = NFS,CIFS,CEPHFS
```

Create a section like this to define a CephFS native backend:

```
[cephfsnative1]
driver_handles_share_servers = False
share_backend_name = CEPHFSNATIVE1
share_driver = manila.share.drivers.cephfs.driver.CephFSDriver
cephfs_conf_path = /etc/ceph/ceph.conf
cephfs_protocol_helper_type = CEPHFS
cephfs_auth_id = manila
cephfs_cluster_name = ceph
cephfs_filesystem_name = cephfs
```

Set `driver-handles-share-servers` to `False` as the driver does not manage the lifecycle of `share-servers`. For the driver backend to expose shares via the native Ceph protocol, set `cephfs_protocol_helper_type` to `CEPHFS`.

Then edit `enabled_share_backends` to point to the drivers backend section using the section name. In this example we are also including another backend (generic1), you would include whatever other backends you have configured.

Finally, edit `cephfs_filesystem_name` with the name of the Ceph filesystem (also referred to as a CephFS volume) you want to use. If you have more than one Ceph filesystem in the cluster, you need to set this option.

```
enabled_share_backends = generic1, cephfsnative1
```

Configure CephFS NFS share backend in manila.conf

Note: Prior to configuring the Manila CephFS driver to use NFS, you must have installed and configured NFS-Ganesha. For guidance on configuration, refer to the NFS-Ganesha setup guide.

Add NFS to `enabled_share_protocols` if its not already there:

```
enabled_share_protocols = NFS,CIFS,CEPHFS
```

Create a section to define a CephFS NFS share backend:

```
[cephfsnfs1]
driver_handles_share_servers = False
share_backend_name = CEPHFSNFS1
share_driver = manila.share.drivers.cephfs.driver.CephFSDriver
cephfs_protocol_helper_type = NFS
cephfs_conf_path = /etc/ceph/ceph.conf
cephfs_auth_id = manila
cephfs_cluster_name = ceph
cephfs_filesystem_name = cephfs
cephfs_ganesha_server_is_remote= False
cephfs_ganesha_server_ip = 172.24.4.3
ganesha_rados_store_enable = True
ganesha_rados_store_pool_name = cephfs_data
```

The following options are set in the driver backend section above:

- `driver-handles-share-servers` to `False` as the driver does not manage the lifecycle of `share-servers`.
- `cephfs_protocol_helper_type` to `NFS` to allow NFS protocol access to the CephFS backed shares.
- `ceph_auth_id` to the ceph auth ID created in *Authorizing the driver to communicate with Ceph*.
- `cephfs_ganesha_server_is_remote` to False if the NFS-ganesha server is co-located with the *manila-share* service. If the NFS-Ganesha server is remote, then set the options to `True`, and set other options such as `cephfs_ganesha_server_ip`, `cephfs_ganesha_server_username`, and `cephfs_ganesha_server_password` (or `cephfs_ganesha_path_to_private_key`) to allow the driver to manage the NFS-Ganesha export entries over SSH.
- `cephfs_ganesha_server_ip` to the ganesha server IP address. It is recommended to set this option even if the ganesha server is co-located with the *manila-share* service.
- `ganesha_rados_store_enable` to True or False. Setting this option to True allows NFS Ganesha to store exports and its export counter in Ceph RADOS objects. We recommend setting this to True and using a RADOS object since it is useful for highly available NFS-Ganesha deployments to store their configuration efficiently in an already available distributed storage system.
- `ganesha_rados_store_pool_name` to the name of the RADOS pool you have created for use with NFS-Ganesha. Set this option only if also setting the `ganesha_rados_store_enable` option to True. If you want to use one of the backend CephFSs RADOS pools, then using CephFSs data pool is preferred over using its metadata pool.

Edit `enabled_share_backends` to point to the drivers backend section using the section name, `cephfsnfs1`.

Finally, edit `cephfs_filesystem_name` with the name of the Ceph filesystem (also referred to as a CephFS volume) you want to use. If you have more than one Ceph filesystem in the cluster, you need to set this option.

```
enabled_share_backends = generic1, cephfsnfs1
```

Space considerations

The CephFS driver reports total and free capacity available across the Ceph cluster to manila to allow provisioning. All CephFS shares are thinly provisioned, i.e., empty shares do not consume any significant space on the cluster. The CephFS driver does not allow controlling oversubscription via manila. So, as long as there is free space, provisioning will continue, and eventually this may cause your Ceph cluster to be over provisioned and you may run out of space if shares are being filled to capacity. It is advised that you use Cephs monitoring tools to monitor space usage and add more storage when required in order to honor space requirements for provisioned manila shares. You may use the driver configuration option `reserved_share_percentage` to prevent manila from filling up your Ceph cluster, and allow existing shares to grow.

Creating shares

Create CephFS native share

The default share type may have `driver_handles_share_servers` set to True. Configure a share type suitable for CephFS native share:

```
manila type-create cephfsnativetype false
manila type-key cephfsnativetype set vendor_name=Ceph storage_protocol=CEPHFS
```

Then create a share,

```
manila create --share-type cephfsnativetype --name cephnativeshare1 cephfs 1
```

Note the export location of the share:

```
manila share-export-location-list cephnativeshare1
```

The export location of the share contains the Ceph monitor (mon) addresses and ports, and the path to be mounted. It is of the form, `{mon ip addr:port}[,{mon ip addr:port}]:{path to be mounted}`

Create CephFS NFS share

Configure a share type suitable for CephFS NFS share:

```
manila type-create cephfsnfstype false
manila type-key cephfsnfstype set vendor_name=Ceph storage_protocol=NFS
```

Then create a share:

```
manila create --share-type cephfsnfstype --name cephnfsshare1 nfs 1
```

Note the export location of the share:

```
manila share-export-location-list cephnfsshare1
```

The export location of the share contains the IP address of the NFS-Ganesha server and the path to be mounted. It is of the form, `{NFS-Ganesha server address}:{path to be mounted}`

Allowing access to shares

Allow access to CephFS native share

Allow Ceph auth ID `alice` access to the share using `cephx` access type.

```
manila access-allow cephnativeshare1 cephx alice
```

Note the access status, and the access/secret key of `alice`.

```
manila access-list cephnativeshare1
```

Allow access to CephFS NFS share

Allow a guest access to the share using `ip` access type.

```
manila access-allow cephnfsshare1 ip 172.24.4.225
```

Mounting CephFS shares

Mounting CephFS native share using FUSE client

Using the secret key of the authorized ID `alice` create a keyring file, `alice.keyring` like:

```
[client.alice]
        key = AQA8+ANW/4ZWNRAAOtWJMFPEihBA1unFImJczA==
```

Using the mon IP addresses from the shares export location, create a configuration file, `ceph.conf` like:

```
[client]
        client quota = true
        mon host = 192.168.1.7:6789, 192.168.1.8:6789, 192.168.1.9:6789
```

Finally, mount the filesystem, substituting the filenames of the keyring and configuration files you just created, and substituting the path to be mounted from the shares export location:

```
sudo ceph-fuse ~/mnt \
--id=alice \
--conf=./ceph.conf \
--keyring=./alice.keyring \
--client-mountpoint=/volumes/_nogroup/4c55ad20-9c55-4a5e-9233-8ac64566b98c
```

Mounting CephFS native share using Kernel client

If you have the `ceph-common` package installed in the client host, you can use the kernel client to mount CephFS shares.

Important: If you choose to use the kernel client rather than the FUSE client the share size limits set in manila may not be obeyed in versions of kernel older than 4.17 and Ceph versions older than mimic. See the quota limitations documentation to understand CephFS quotas.

The mount command is as follows:

```
mount -t ceph {mon1 ip addr}:6789,{mon2 ip addr}:6789,{mon3 ip addr}:6789:/ \
    {mount-point} -o name={access-id},secret={access-key}
```

With our earlier examples, this would be:

```
mount -t ceph 192.168.1.7:6789, 192.168.1.8:6789, 192.168.1.9:6789:/ \
    /volumes/_nogroup/4c55ad20-9c55-4a5e-9233-8ac64566b98c \
    -o name=alice,secret='AQA8+ANW/4ZWNRAAOtWJMFPEihBA1unFImJczA=='
```

Mount CephFS NFS share using NFS client

In the guest, mount the share using the NFS client and knowing the shares export location.

```
sudo mount -t nfs 172.24.4.3:/volumes/_nogroup/6732900b-32c1-4816-a529-
↪4d6d3f15811e /mnt/nfs/
```

Known restrictions

- A CephFS driver instance, represented as a backend driver section in manila.conf, requires a Ceph auth ID unique to the backend Ceph Filesystem. Using a non-unique Ceph auth ID will result in the driver unintentionally evicting other CephFS clients using the same Ceph auth ID to connect to the backend.
- Snapshots are read-only. A user can read a snapshots contents from the `.snap/{manila-snapshot-id}_{unknown-id}` folder within the mounted share.

Security

- Each shares data is mapped to a distinct Ceph RADOS namespace. A guest is restricted to access only that particular RADOS namespace. https://docs.ceph.com/docs/nautilus/cephfs/file-layouts/
- An additional level of resource isolation can be provided by mapping a shares contents to a separate RADOS pool. This layout would be preferred only for cloud deployments with a limited number of shares needing strong resource separation. You can do this by setting a share type specification, `cephfs:data_isolated` for the share type used by the cephfs driver.

  ```
  manila type-key cephfstype set cephfs:data_isolated=True
  ```

Security with CephFS native share backend

As the guests need direct access to Cephs public network, CephFS native share backend is suitable only in private clouds where guests can be trusted.

The manila.share.drivers.cephfs.driver Module

class CephFSDriver(*args, **kwargs)

Bases: manila.share.driver.ExecuteMixin, manila.share.driver.GaneshaMixin, *manila.share.driver.ShareDriver*

Driver for the Ceph Filesystem.

property ceph_mon_version

check_for_setup_error()

Returns an error if prerequisites arent met.

create_share(*context*, *share*, *share_server=None*)

Create a CephFS volume.

Parameters

- **context** A RequestContext.
- **share** A Share.
- **share_server** Always None for CephFS native.

Returns The export locations dictionary.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Create a CephFS subvolume from a snapshot

create_share_group(*context*, *sg_dict*, *share_server=None*)

Create a share group.

Parameters

- **context**
- **share_group_dict** The share group details EXAMPLE: { status: creating, project_id: 13c0be6290934bd98596cfa004650049, user_id: a0314a441ca842019b0952224aa39192, description: None, deleted: False, created_at: datetime.datetime(2015, 8, 10, 15, 14, 6), updated_at: None, source_share_group_snapshot_id: some_fake_uuid, share_group_type_id: some_fake_uuid, host: hostname@backend_name, share_network_id: None, share_server_id: None, deleted_at: None, share_types: [<models.ShareGroupShareTypeMapping>], id: some_fake_uuid, name: None }

Returns (share_group_model_update, share_update_list) share_group_model_update - a dict containing any values to be updated for the SG in the database. This value may be None.

create_share_group_snapshot(*context*, *snap_dict*, *share_server=None*)

Create a share group snapshot.

Parameters

- **context**
- **snap_dict** The share group snapshot details EXAMPLE: .. code:

```
{
'status': 'available',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': '0',
'created_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'updated_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'share_group_id': 'some_fake_uuid',
'share_group_snapshot_members': [
    {
     'status': 'available',
     'share_type_id': 'some_fake_uuid',
     'user_id': 'a0314a441ca842019b0952224aa39192',
     'deleted': 'False',
     'created_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share': <models.Share>,
     'updated_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share_proto': 'NFS',
     'share_name': 'share_some_fake_uuid',
     'name': 'share-snapshot-some_fake_uuid',
     'project_id': '13c0be6290934bd98596cfa004650049',
     'share_group_snapshot_id': 'some_fake_uuid',
     'deleted_at': None,
     'share_id': 'some_fake_uuid',
     'id': 'some_fake_uuid',
     'size': 1,
     'provider_location': None,
    }
],
'deleted_at': None,
'id': 'some_fake_uuid',
'name': None
}
```

Returns

(share_group_snapshot_update, member_update_list) share_group_snapshot_update - a dict containing any values to be updated for the CGSnapshot in the database. This value may be None.

member_update_list - a list of dictionaries containing for every member of the share group snapshot. Each dict should contains values to be updated for the ShareGroupSnapshotMember in the database. This list may be empty or None.

create_snapshot(*context*, *snapshot*, *share_server=None*)

Is called to create snapshot.

Parameters

- **context** Current context

- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

Returns None or a dictionary with key export_locations containing a list of export locations, if snapshots can be mounted.

delete_share(*context*, *share*, *share_server=None*)
Is called to remove share.

delete_share_group(*context*, *sg_dict*, *share_server=None*)
Delete a share group

Parameters

- **context** The request context
- **share_group_dict** The share group details EXAMPLE: .. code:

```
{
'status': 'creating',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': 'False',
'created_at': datetime.datetime(2015, 8, 10, 15, 14, 6),
'updated_at': None,
'source_share_group_snapshot_id': 'some_fake_uuid',
'share_share_group_type_id': 'some_fake_uuid',
'host': 'hostname@backend_name',
'deleted_at': None,
'shares': [<models.Share>], # The new shares being␣
↪created
'share_types': [<models.ShareGroupShareTypeMapping>],
'id': 'some_fake_uuid',
'name': None
}
```

Returns share_group_model_update share_group_model_update - a dict containing any values to be updated for the group in the database. This value may be None.

delete_share_group_snapshot(*context*, *snap_dict*, *share_server=None*)
Delete a share group snapshot

Parameters

- **context**
- **snap_dict** The share group snapshot details EXAMPLE: .. code:

```
{
'status': 'available',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
```

(continues on next page)

(continued from previous page)

```
'description': None,
'deleted': '0',
'created_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'updated_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'share_group_id': 'some_fake_uuid',
'share_group_snapshot_members': [
    {
     'status': 'available',
     'share_type_id': 'some_fake_uuid',
     'share_id': 'some_fake_uuid',
     'user_id': 'a0314a441ca842019b0952224aa39192',
     'deleted': 'False',
     'created_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share': <models.Share>,
     'updated_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share_proto': 'NFS',
     'share_name':'share_some_fake_uuid',
     'name': 'share-snapshot-some_fake_uuid',
     'project_id': '13c0be6290934bd98596cfa004650049',
     'share_group_snapshot_id': 'some_fake_uuid',
     'deleted_at': None,
     'id': 'some_fake_uuid',
     'size': 1,
     'provider_location': 'fake_provider_location_value',
    }
],
'deleted_at': None,
'id': 'f6aa3b59-57eb-421e-965c-4e182538e36a',
'name': None
}
```

Returns (share_group_snapshot_update, member_update_list) share_group_snapshot_update - a dict containing any values to be updated for the ShareGroupSnapshot in the database. This value may be None.

delete_snapshot(*context*, *snapshot*, *share_server=None*)

Is called to remove snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

do_setup(*context*)

Any initialization the share driver does while starting.

ensure_share(*context*, *share*, *share_server=None*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

extend_share(*share*, *new_size*, *share_server=None*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

get_configured_ip_versions()
Get allowed IP versions.

The supported versions are returned with list, possible values are: [4], [6], or [4, 6]

Drivers that assert ipv6_implemented = True must override this method. If the returned list includes 4, then shares created by this driver must have an IPv4 export location. If the list includes 6, then shares created by the driver must have an IPv6 export location.

Drivers should check that their storage controller actually has IPv4/IPv6 enabled and configured properly.

get_share_status(*share*, *share_server=None*)
Returns the current status for a share.

Parameters

- **share** a manila share.
- **share_server** a manila share server (not currently supported).

Returns manila share status.

property rados_client

setup_default_ceph_cmd_target()

shrink_share(*share*, *new_size*, *share_server=None*)
Shrinks size of existing share.

If consumed space on share larger than new_size driver should raise ShareShrinkingPossibleDataLoss exception: raise ShareShrinkingPossibleDataLoss(share_id=share[id])

Parameters

- **share** Share model
- **new_size** New size of share (new_size < share[size])
- **share_server** Optional Share server model

:raises ShareShrinkingPossibleDataLoss, NotImplementedError

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given share.

access_rules contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the add_rules and delete_rules parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in add_rules and rules that need to be removed to be present in delete_rules.

When a rule in delete_rules was never applied, drivers must not raise an exception, or attempt to set the rule to error state.

add_rules and delete_rules can be empty lists, in this situation, drivers should ensure that the rules present in access_rules are the same as those on the back end. One scenario where this situation is forced is when the access_level is changed for all existing rules (share migration and for readable replicas).

Drivers must be mindful of this call for share replicas. When update_access is called on one of the replicas, the call is likely propagated to all replicas belonging to the share, especially when individual rules are added or removed. If a particular access rule does not make sense to the driver in the context of a given replica, the driver should be careful to report a correct behavior, and take meaningful action. For example, if R/W access is requested on a replica that is part of a readable type replication; R/O access may be added by the driver instead of R/W. Note that raising an exception *will* result in the access_rules_status on the replica, and the share itself being out_of_sync. Drivers can sync on the valid access rules that are provided on the create_replica and promote_replica calls.

Parameters

- **context** Current context
- **share** Share model with share data.
- **access_rules** A list of access rules for given share
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

Returns

None, or a dictionary of updates in the format:

```
{
  09960614-8574-4e03-89cf-7cf267b0bd08: {
    access_key: alice31493e5441b8171d2310d80e37e, state: error,
  },
  28f6eabb-4342-486a-a7f4-45688f0c0295: {
    access_key: bob0078aa042d5a7325480fd13228b, state: active,
  },
}
```

The top level keys are access_id fields of the access rules that need to be updated. `access_key``s are credentials (str) of the entities granted access. Any rule in the ``access_rules` parameter can be updated.

Important: Raising an exception in this method will force *all* rules in applying and denying states to error.

An access rule can be set to error state, either explicitly via this return parameter or because of an exception raised in this method. Such an access rule will no longer be sent to the driver on subsequent access rule updates. When users deny that rule however, the driver will be asked to deny access to the client/s represented by the rule. We expect that a rule that was error-ed at the driver should never exist on the back end. So, do not fail the deletion request.

Also, it is possible that the driver may receive a request to add a rule that is already present on the back end. This can happen if the share manager service goes down while the driver is committing access rule changes. Since we cannot determine if the rule was applied successfully by the driver before the disruption, we will treat all applying transitional rules as new rules and repeat the request.

---

property volname

class NFSProtocolHelper(*execute*, *config_object*, ***kwargs*)

Bases: `manila.share.drivers.ganesha.GaneshaNASHelper2`

check_for_setup_error()

Returns an error if prerequisites arent met.

get_configured_ip_versions()

get_export_locations(*share*, *subvolume_path*)

shared_data = {}

supported_protocols = ('NFS',)

class NativeProtocolHelper(*execute*, *config*, ***kwargs*)

Bases: `manila.share.drivers.ganesha.NASHelperBase`

Helper class for native CephFS protocol

check_for_setup_error()

Returns an error if prerequisites arent met.

get_configured_ip_versions()

get_export_locations(*share*, *subvolume_path*)

get_mon_addrs()

supported_access_levels = ('rw', 'ro')

supported_access_types = ('cephx',)

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)

Update access rules of share.

exception RadosError

Bases: `Exception`

Something went wrong talking to Ceph with librados

rados_command(*rados_client*, *prefix=None*, *args=None*, *json_obj=False*, *target=None*)
Safer wrapper for ceph_argparse.json_command

Raises error exception instead of relying on caller to check return codes.

Error exception can result from: * Timeout * Actual legitimate errors * Malformed JSON output

return: If json_obj is True, return the decoded JSON object from ceph, or None if empty string returned. If json is False, return a decoded string (the data returned by ceph command)

setup_json_command()

setup_rados()

GPFS Driver

GPFS driver uses IBM General Parallel File System (GPFS), a high-performance, clustered file system, developed by IBM, as the storage backend for serving file shares to the manila clients.

Supported shared filesystems

- NFS (access by IP)

Supported Operations

- Create NFS Share
- Delete NFS Share
- Create Share Snapshot
- Delete Share Snapshot
- Create Share from a Share Snapshot
- Allow NFS Share access
  - Currently only rw access level is supported
- Deny NFS Share access

Requirements

- Install GPFS with server license, version >= 2.0, on the storage backend.
- Install Kernel NFS or Ganesha NFS server on the storage backend servers.
- If using Ganesha NFS, currently NFS Ganesha v1.5 and v2.0 are supported.
- Create a GPFS cluster and create a filesystem on the cluster, that will be used to create the manila shares.
- Enable quotas for the GPFS file system (*mmchfs -Q yes*).
- Establish network connection between the manila host and the storage backend.

Manila driver configuration setting

The following parameters in the manila configuration file need to be set:

- *share_driver* = manila.share.drivers.ibm.gpfs.GPFSShareDriver
- *gpfs_share_export_ip* = <IP to be added to GPFS export string>
- If the backend GPFS server is not running on the manila host machine, the following options are required to SSH to the remote GPFS backend server:
  - *gpfs_ssh_login* = <GPFS server SSH login name>

    and one of the following settings is required to execute commands over SSH:
  - *gpfs_ssh_private_key* = <path to GPFS server SSH private key for login>
  - *gpfs_ssh_password* = <GPFS server SSH login password>

The following configuration parameters are optional:

- *gpfs_mount_point_base* = <base folder where exported shares are located>
- *gpfs_nfs_server_type* = <KNFS|GNFS>
- *gpfs_nfs_server_list* = <list of the fully qualified NFS server names>
- *gpfs_ssh_port* = <ssh port number>

Restart of *manila-share* service is needed for the configuration changes to take effect.

Known Restrictions

- The driver does not support a segmented-network multi-tenancy model but instead works over a flat network where the tenants share a network.
- While using remote GPFS node, with Ganesha NFS, gpfs_ssh_private_key for remote login to the GPFS node must be specified and there must be a passwordless authentication already setup between the manila share service and the remote GPFS node.

The `manila.share.drivers.ibm.gpfs` Module

GPFS Driver for shares.

Config Requirements: GPFS file system must have quotas enabled (*mmchfs -Q yes*).

Notes: GPFS independent fileset is used for each share.

TODO(nileshb): add support for share server creation/deletion/handling.

Limitation: While using remote GPFS node, with Ganesha NFS, gpfs_ssh_private_key for remote login to the GPFS node must be specified and there must be a passwordless authentication already setup between the Manila share service and the remote GPFS node.

`class CESHelper(`*execute*, *config_object*`)`

Bases: `manila.share.drivers.ibm.gpfs.NASHelperBase`

Wrapper for NFS by Spectrum Scale CES

allow_access(*local_path*, *share*, *access*)
Allow access to the host.

deny_access(*local_path*, *share*, *access*, *force=False*)
Deny access to the host.

get_access_option(*access*)
Get access option string based on access level.

remove_export(*local_path*, *share*)
Remove export.

resync_access(*local_path*, *share*, *access_rules*)
Re-sync all access rules for given share.

class GPFSShareDriver(**args*, ***kwargs*)
Bases: `manila.share.driver.ExecuteMixin`, `manila.share.driver.GaneshaMixin`, *manila.share.driver.ShareDriver*

GPFS Share Driver.

Executes commands relating to Shares. Supports creation of shares on a GPFS cluster.

API version history:

1.0 - Initial version. 1.1 - Added extend_share functionality 2.0 - Added CES support for NFS Ganesha

check_for_setup_error()
Returns an error if prerequisites arent met.

create_share(*ctx*, *share*, *share_server=None*)
Create GPFS directory that will be represented as share.

create_share_from_snapshot(*ctx*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)
Is called to create share from a snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Creates a snapshot.

delete_share(*ctx*, *share*, *share_server=None*)
Remove and cleanup share storage.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Deletes a snapshot.

do_setup(*context*)
Any initialization the share driver does while starting.

ensure_share(*ctx*, *share*, *share_server=None*)
Ensure that storage are mounted and exported.

extend_share(*share*, *new_size*, *share_server=None*)
Extends the quota on the share fileset.

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle network-

ing itself should calculate it according to their own requirements. It can have 1+ network interfaces.

manage_existing(*share*, *driver_options*)
Brings an existing share under Manila management.

If the provided share is not valid, then raise a ManageInvalidShare exception, specifying a reason for the failure.

If the provided share is not in a state that can be managed, such as being replicated on the backend, the driver *MUST* raise ManageInvalidShare exception with an appropriate message.

The share has a share_type, and the driver can inspect that and compare against the properties of the referenced backend share. If they are incompatible, raise a ManageExistingShareTypeMismatch, specifying a reason for the failure.

This method is invoked when the share is being managed with a share type that has `driver_handles_share_servers` extra-spec set to False.

Parameters

- **share** Share model
- **driver_options** Driver-specific options provided by admin.

Returns share_update dictionary with required key size, which should contain size of the share.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given share.

class KNFSHelper(*execute*, *config_object*)
Bases: `manila.share.drivers.ibm.gpfs.NASHelperBase`

Wrapper for Kernel NFS Commands.

allow_access(*local_path*, *share*, *access*, *error_on_exists=True*)
Allow access to one or more vm instances.

deny_access(*local_path*, *share*, *access*)
Remove access for one or more vm instances.

get_access_option(*access*)
Get access option string based on access level.

remove_export(*local_path*, *share*)
Remove export.

resync_access(*local_path*, *share*, *access_rules*)
Re-sync all access rules for given share.

class NASHelperBase(*execute*, *config_object*)
Bases: `object`

Interface to work with share.

abstract allow_access(*local_path*, *share*, *access*)
Allow access to the host.

create_export(*local_path*)
Construct location of new export.

abstract deny_access(*local_path*, *share*, *access*)
Deny access to the host.

abstract get_access_option(*access*)
Get access option string based on access level.

get_export_options(*share*, *access*, *helper*)
Get the export options.

abstract remove_export(*local_path*, *share*)
Remove export.

abstract resync_access(*local_path*, *share*, *access_rules*)
Re-sync all access rules for given share.

Huawei Driver

Huawei NAS Driver is a plugin based the OpenStack manila service. The Huawei NAS Driver can be used to provide functions such as the share and snapshot for virtual machines(instances) in OpenStack. Huawei NAS Driver enables the OceanStor V3 series V300R002 storage system to provide only network filesystems for OpenStack.

Requirements

- The OceanStor V3 series V300R002 storage system.
- The following licenses should be activated on V3 for File:
  - CIFS
  - NFS
  - HyperSnap License (for snapshot)

Supported Operations

The following operations is supported on V3 storage:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- Allow CIFS/NFS Share access
  - IP and USER access types are supported for NFS(ro/rw).
  - Only USER access type is supported for CIFS(ro/rw).
- Deny CIFS/NFS Share access
- Create snapshot
- Delete snapshot
- Manage CIFS/NFS share
- Support pools in one backend

- Extend share
- Shrink share
- Support multi RestURLs(<RestURL>)
- Support multi-tenancy
- Ensure share
- Create share from snapshot
- Support QoS

Pre-Configurations on Huawei

1. Create a driver configuration file. The driver configuration file name must be the same as the manila_huawei_conf_file item in the manila_conf configuration file.

2. Configure Product. Product indicates the storage system type. For the OceanStor V3 series V300R002 storage systems, the driver configuration file is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<Config>
    <Storage>
        <Product>V3</Product>
        <LogicalPortIP>x.x.x.x</LogicalPortIP>
        <Port>abc;CTE0.A.H1</Port>
        <RestURL>https://x.x.x.x:8088/deviceManager/rest/;
        https://x.x.x.x:8088/deviceManager/rest/</RestURL>
        <UserName>xxxxxxxxx</UserName>
        <UserPassword>xxxxxxxxx</UserPassword>
    </Storage>
    <Filesystem>
        <StoragePool>xxxxxxxxx</StoragePool>
        <SectorSize>64</SectorSize>
        <WaitInterval>3</WaitInterval>
        <Timeout>60</Timeout>
        <NFSClient>
            <IP>x.x.x.x</IP>
        </NFSClient>
        <CIFSClient>
            <UserName>xxxxxxxxx</UserName>
            <UserPassword>xxxxxxxxx</UserPassword>
        </CIFSClient>
    </Filesystem>
</Config>
```

- *Product* is a type of a storage product. Set it to *V3*.
- *LogicalPortIP* is an IP address of the logical port.
- *Port* is a port name list of bond port or ETH port, used to create vlan and logical port. Multi Ports can be configured in <Port>(separated by ;). If <Port> is not configured, then will choose an online port on the array.

- *RestURL* is an access address of the REST interface. Multi RestURLs can be configured in <RestURL>(separated by ;). When one of the RestURL failed to connect, driver will retry another automatically.
- *UserName* is a user name of an administrator.
- *UserPassword* is a password of an administrator.
- *StoragePool* is a name of a storage pool to be used.
- *SectorSize* is the size of the disk blocks, optional value can be 4, 8, 16, 32 or 64, and the units is KB. If sectorsize is configured in both share_type and xml file, the value of sectorsize in the share_type will be used. If sectorsize is configured in neither share_type nor xml file, huawei storage backends will provide a default value(64) when creating a new share.
- *WaitInterval* is the interval time of querying the file system status.
- *Timeout* is the timeout period for waiting command execution of a device to complete.
- *NFSClientIP* is the backend IP in admin network to use for mounting NFS share.
- *CIFSClientUserName* is the backend user name in admin network to use for mounting CIFS share.
- *CIFSClientUserPassword* is the backend password in admin network to use for mounting CIFS share.

Backend Configuration

Modify the *manila.conf* manila configuration file and add share_driver and manila_huawei_conf_file items. Example for configuring a storage system:

- *share_driver* = manila.share.drivers.huawei.huawei_nas.HuaweiNasDriver
- *manila_huawei_conf_file* = /etc/manila/manila_huawei_conf.xml
- *driver_handles_share_servers* = True or False

Note:

- If *driver_handles_share_servers* is True, the driver will choose a port in <Port> to create vlan and logical port for each tenant network. And the share type with the DHSS extra spec should be set to True when creating shares.
- If *driver_handles_share_servers* is False, then will use the IP in <LogicalPortIP>. Also the share type with the DHSS extra spec should be set to False when creating shares.

Restart of manila-share service is needed for the configuration changes to take effect.

Share Types

When creating a share, a share type can be specified to determine where and how the share will be created. If a share type is not specified, the *default_share_type* set in the manila configuration file is used.

Manila requires that the share type includes the *driver_handles_share_servers* extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers (share networks) capability. For the Huawei driver, this must be set to False.

To create a share on a backend with a specific type of disks, include the *huawei_disk_type* extra-spec in the share type. Valid values for this extra-spec are ssd, sas, nl_sas or mix. This share will be created on a backend with a matching disk type.

Another common manila extra-spec used to determine where a share is created is *share_backend_name*. When this extra-spec is defined in the share type, the share will be created on a backend with a matching share_backend_name.

Manila share types may contain qualified extra-specs, -extra-specs that have significance for the backend driver and the CapabilityFilter. This commit makes the Huawei driver report the following boolean capabilities:

- capabilities:dedupe
- capabilities:compression
- capabilities:thin_provisioning
- capabilities:huawei_smartcache
  - huawei_smartcache:cachename
- capabilities:huawei_smartpartition
  - huawei_smartpartition:partitionname
- capabilities:qos
  - qos:maxIOPS
  - qos:minIOPS
  - qos:minbandwidth
  - qos:maxbandwidth
  - qos:latency
  - qos:iotype
- capabilities:huawei_sectorsize

The scheduler will choose a host that supports the needed capability when the CapabilityFilter is used and a share type uses one or more of the following extra-specs:

- capabilities:dedupe=<is> True or <is> False
- capabilities:compression=<is> True or <is> False
- capabilities:thin_provisioning=<is> True or <is> False
- capabilities:huawei_smartcache=<is> True or <is> False
  - huawei_smartcache:cachename=test_cache_name

- capabilities:huawei_smartpartition=<is> True or <is> False
  - huawei_smartpartition:partitionname=test_partition_name
- capabilities:qos=<is> True or <is> False
  - qos:maxIOPS=100
  - qos:minIOPS=10
  - qos:maxbandwidth=100
  - qos:minbandwidth=10
  - qos:latency=10
  - qos:iotype=0
- capabilities:huawei_sectorsize=<is> True or <is> False
  - huawei_sectorsize:sectorsize=4
- huawei_disk_type=ssd or sas or nl_sas or mix

thin_provisioning will be reported as [True, False] for Huawei backends.

dedupe will be reported as [True, False] for Huawei backends.

compression will be reported as [True, False] for Huawei backends.

huawei_smartcache will be reported as [True, False] for Huawei backends. Adds SSDs into a high-speed cache pool and divides the pool into multiple cache partitions to cache hotspot data in random and small read I/Os.

huawei_smartpartition will be reported as [True, False] for Huawei backends. Add share to the smartpartition named test_partition_name. Allocates cache resources based on service characteristics, ensuring the quality of critical services.

qos will be reported as True for backends that use QoS (Quality of Service) specification.

huawei_sectorsize will be reported as [True, False] for Huawei backends.

huawei_disk_type will be reported as ssd, sas, nl_sas or mix for Huawei backends.

Restrictions

The Huawei driver has the following restrictions:

- IP and USER access types are supported for NFS.
- Only LDAP domain is supported for NFS.
- Only USER access type is supported for CIFS.
- Only AD domain is supported for CIFS.

The manila.share.drivers.huawei.huawei_nas Module

Huawei Nas Driver for Huawei storage arrays.

class HuaweiNasDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

Huawei Share Driver.

Executes commands relating to Shares. Driver version history:

```
1.0 - Initial version.
1.1 - Add shrink share.
      Add extend share.
      Add manage share.
      Add share level(ro).
      Add smartx capabilities.
      Support multi pools in one backend.
1.2 - Add share server support.
      Add ensure share.
      Add QoS support.
      Add create share from snapshot.
1.3 - Add manage snapshot.
      Support reporting disk type of pool.
      Add replication support.
```

allow_access(*context*, *share*, *access*, *share_server=None*)

Allow access to the share.

check_for_setup_error()

Returns an error if prerequisites arent met.

create_replica(*context*, *replica_list*, *new_replica*, *access_rules*, *replica_snapshots*, *share_server=None*)

Replicate the active replica to a new replica on this backend.

create_share(*context*, *share*, *share_server=None*)

Create a share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Create a share from snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)

Create a snapshot.

delete_replica(*context*, *replica_list*, *replica_snapshots*, *replica*, *share_server=None*)

Delete a replica.

delete_share(*context*, *share*, *share_server=None*)

Delete a share.

delete_snapshot(*context*, *snapshot*, *share_server=None*)

Delete a snapshot.

deny_access(*context*, *share*, *access*, *share_server=None*)

Deny access to the share.

do_setup(*context*)
Any initialization the huawei nas driver does while starting.

ensure_share(*context*, *share*, *share_server=None*)
Ensure that share is exported.

extend_share(*share*, *new_size*, *share_server=None*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

get_backend_driver()

get_network_allocations_number()
Get number of network interfaces to be created.

get_pool(*share*)
Return pool name where the share resides on.

manage_existing(*share*, *driver_options*)
Manage existing share.

manage_existing_snapshot(*snapshot*, *driver_options*)
Manage existing snapshot.

promote_replica(*context*, *replica_list*, *replica*, *access_rules*, *share_server=None*)
Promote a replica to active replica state..

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)
Reverts a share (in place) to the specified snapshot.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **snapshot** The snapshot to be restored
- **share_access_rules** List of all access rules for the affected share
- **snapshot_access_rules** List of all access rules for the affected snapshot
- **share_server** Optional Share server model or None

shrink_share(*share*, *new_size*, *share_server=None*)
Shrinks size of existing share.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules list.

update_replica_state(*context*, *replica_list*, *replica*, *access_rules*, *replica_snapshots*, *share_server=None*)

Update the replica_state of a replica.

HDFS native driver

HDFS native driver is a plugin based on the OpenStack manila service, which uses Hadoop distributed file system (HDFS), a distributed file system designed to hold very large amounts of data, and provide high-throughput access to the data.

A manila share in this driver is a subdirectory in hdfs root directory. Instances talk directly to the HDFS storage backend with hdfs protocol. And access to each share is allowed by user based access type, which is aligned with HDFS ACLs to support access control of multiple users and groups.

Network configuration

The storage backend and manila hosts should be in a flat network, otherwise, the L3 connectivity between them should exist.

Supported shared filesystems

- HDFS (authentication by user)

Supported Operations

- Create HDFS share
- Delete HDFS share
- Allow HDFS Share access * Only support user access type * Support level of access (ro/rw)
- Deny HDFS Share access
- Create snapshot
- Delete snapshot
- Create share from snapshot
- Extend share

Requirements

- Install HDFS package, version >= 2.4.x, on the storage backend
- To enable access control, the HDFS file system must have ACLs enabled
- Establish network connection between the manila host and storage backend

Manila driver configuration

- *share_driver* = manila.share.drivers.hdfs.hdfs_native.HDFSNativeShareDriver
- ***hdfs_namenode_ip* = the IP address of the HDFS namenode, and only single** namenode is supported now
- *hdfs_namenode_port* = the port of the HDFS namenode service
- *hdfs_ssh_port* = HDFS namenode SSH port
- *hdfs_ssh_name* = HDFS namenode SSH login name
- ***hdfs_ssh_pw* = HDFS namenode SSH login password, this parameter is not** necessary, if the following *hdfs_ssh_private_key* is configured
- *hdfs_ssh_private_key* = Path to the HDFS namenode private key to ssh login

Known Restrictions

- This driver does not support network segmented multi-tenancy model. Instead multi-tenancy is supported by the tenant specific user authentication
- Only support for single HDFS namenode in Kilo release

The `manila.share.drivers.hdfs.hdfs_native` Module

HDFS native protocol (hdfs) driver for manila shares.

Manila share is a directory in HDFS. And this share does not use service VM instance (share server). The instance directly talks to the HDFS cluster.

The initial version only supports single namenode and flat network.

Configuration Requirements: To enable access control, HDFS file system must have ACLs enabled.

class HDFSNativeShareDriver(**args*, ***kwargs*)

Bases: `manila.share.driver.ExecuteMixin`, *manila.share.driver.ShareDriver*

HDFS Share Driver.

Executes commands relating to shares. API version history:

1.0 - Initial Version

allow_access(*context*, *share*, *access*, *share_server=None*)

Allows access to the share for a given user.

check_for_setup_error()

Return an error if the prerequisites are met.

create_share(*context*, *share*, *share_server=None*)

Create a HDFS directory which acted as a share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Creates a snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Creates a snapshot.

delete_share(*context*, *share*, *share_server=None*)
Deletes share storage.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Deletes a snapshot.

deny_access(*context*, *share*, *access*, *share_server=None*)
Denies the access to the share for a given user.

do_setup(*context*)
Do initialization while the share driver starts.

ensure_share(*context*, *share*, *share_server=None*)
Ensure the storage are exported.

extend_share(*share*, *new_size*, *share_server=None*)
Extend share storage.

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

Hitachi NAS Platform File Services Driver for OpenStack

Driver Version 3.0

Hitachi NAS Platform Storage Requirements

This Hitachi NAS Platform File Services Driver for OpenStack provides support for Hitachi NAS Platform (HNAS) models 3080, 3090, 4040, 4060, 4080 and 4100 with NAS OS 12.2 or higher. Before configuring the driver, ensure the HNAS has at least:

- 1 storage pool (span) configured.
- 1 EVS configured.
- 1 file system in this EVS, created without replication target option and should be in mounted state. It is recommended to disable auto-expansion, because the scheduler uses the current free space reported by the file system when creating shares.
- 1 Management User configured with supervisor permission level.
- Hitachi NAS Management interface should be reachable from manila-share node.

Also, if the driver is going to create CIFS shares, either LDAP servers or domains must be configured previously in HNAS to provide the users and groups.

Supported Operations

The following operations are supported in this version of Hitachi NAS Platform File Services Driver for OpenStack:

- Create and delete CIFS and NFS shares;
- Extend and shrink shares;
- Manage rules to shares (allow/deny access);
- Allow and deny share access;
  - `IP` access type supported for `NFS` shares;
  - `User` access type supported for `CIFS` shares;
  - Both `RW` and `RO` access level are supported for NFS and CIFS shares;
- Manage and unmanage shares;
- Create and delete snapshots;
- Create shares from snapshots.

Driver Configuration

This document contains the installation and user guide of the Hitachi NAS Platform File Services Driver for OpenStack. Although mentioning some Shared File Systems service operations and HNAS commands, both are not in the scope of this document. Please refer to their own guides for details.

Before configuring the driver, make sure that the nodes running the manila-share service have access to the HNAS management port, and compute and network nodes have access to the data ports (EVS IPs or aggregations).

The driver configuration can be summarized in the following steps:

1. Configure HNAS parameters on `manila.conf`;
2. Prepare the network ensuring all OpenStack-HNAS connections mentioned above;
3. Configure/create share type;
4. Restart the services;
5. Configure OpenStack networks.

Step 1 - HNAS Parameters Configuration

The following parameters need to be configured in the [DEFAULT] section of `/etc/manila/manila.conf`:

Option	Description
enabled_share_backends	Name of the section on `manila.conf` used to specify a backend. For example: *enabled_share_backends = hnas1*
enabled_share_protocols	Specify a list of protocols to be allowed for share creation. This driver version supports NFS and/or CIFS.

The following parameters need to be configured in the [backend] section of `/etc/manila/manila.conf`:

Option	Description
share_backend_name	A name for the backend.
share_driver	Python module path. For this driver **this must be**: *manila.share.drivers.hitachi.hnas.driver.HitachiHNASDriver*
driver_handles_share_servers	Driver working mode. For this driver **this must be**: *False*.
hitachi_hnas_ip	HNAS management interface IP for communication between manila-share node and HNAS.
hitachi_hnas_user	This field is used to provide user credential to HNAS. Provided management user must have supervisor level.
hitachi_hnas_password*	This field is used to provide password credential to HNAS. Either hitachi_hnas_password or hitachi_hnas_ssh_private_key must be set.
hitachi_hnas_ssh_private_key*	Set this parameter with RSA/DSA private key path to allow the driver to connect into HNAS.
hitachi_hnas_evs_id	ID from EVS which this backend is assigned to (ID can be listed by CLI evs list or EVS Management in HNAS Interface).
hitachi_hnas_evs_ip	EVS IP for mounting shares (this can be listed by CLI evs list or EVS Management in HNAS interface).
hitachi_hnas_file_system_name	Name of the file system in HNAS, located in the specified EVS.
hitachi_hnas_cluster_admin_ip0*	If HNAS is in a multi-farm (one SMU managing multiple HNAS) configuration, set this parameter with the IP of the clusters admin node.
hitachi_hnas_stalled_job_timeout*	Tree-clone-job commands are used to create snapshots and create shares from snapshots. This parameter sets a timeout (in seconds) to wait for jobs to complete. Default value is 30 seconds.
hitachi_hnas_driver_helper*	Python module path for the driver helper. For this driver, it should use (default value): *manila.share.drivers.hitachi.hnas.ssh.HNASSSHBackend*
hitachi_hnas_allow_cifs_snapshot_while_mounted*	By default, CIFS snapshots are not allowed to be taken while the share has clients connected because consistent snapshots cannot be guaranteed for all files. This parameter can be set to *True* to allow snapshots to be taken while the share has clients connected. **WARNING**: Setting this parameter to *True* might cause inconsistent snapshots on CIFS shares. Default value is *False*.

* Non mandatory parameters.

Below is an example of a valid configuration of HNAS driver:

```
[DEFAULT]``
...
enabled_share_backends = hitachi1
enabled_share_protocols = CIFS,NFS
...

[hitachi1]
share_backend_name = HITACHI1
share_driver = manila.share.drivers.hitachi.hnas.driver.HitachiHNASDriver
driver_handles_share_servers = False
hitachi_hnas_ip = 172.24.44.15
```

(continues on next page)

(continued from previous page)

```
hitachi_hnas_user = supervisor
hitachi_hnas_password = supervisor
hitachi_hnas_evs_id = 1
hitachi_hnas_evs_ip = 10.0.1.20
hitachi_hnas_file_system_name = FS-Manila
```

Step 2 - Prepare the Network

In the driver mode used by Hitachi NAS Platform File Services Driver for OpenStack, driver_handles_share_servers (DHSS) as False, the driver does not handle network configuration, it is up to the administrator to configure it. It is mandatory that HNAS management interface is reachable from a manila-share node through admin network, while the selected EVS data interface is reachable from OpenStack Cloud, such as through neutron flat networking. Here is a step-by-step of an example configuration:

Manila-Share Node:

eth0: Admin Network, can ping HNAS management interface.

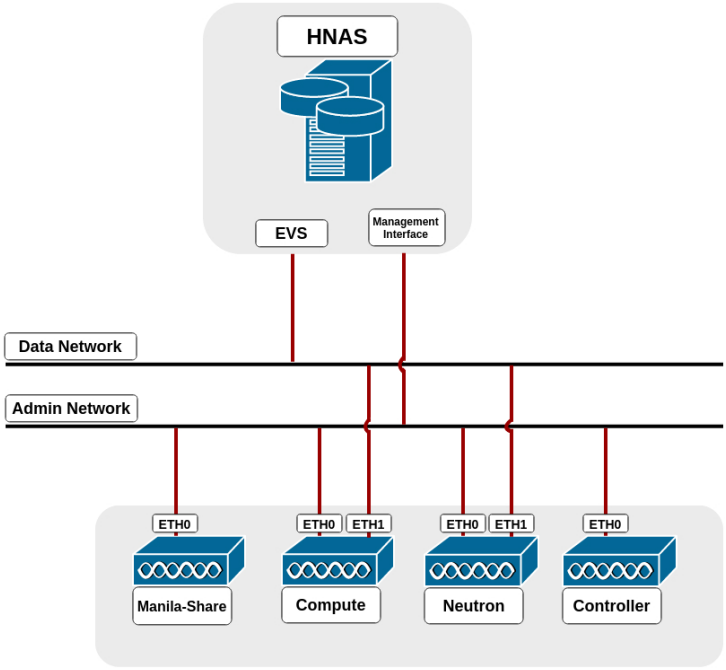
eth1: Data Network, can ping HNAS EVS IP (data interface). This interface is only required if you plan to use Share Migration.

Network Node and Compute Nodes:

eth0: Admin Network, can ping HNAS management interface.

eth1: Data Network, can ping HNAS EVS IP (data interface).

The following image represents the described scenario:



Run in **Network Node**:

```
$ sudo ifconfig eth1 0
$ sudo ovs-vsctl add-br br-eth1
$ sudo ovs-vsctl add-port br-eth1 eth1
$ sudo ifconfig eth1 up
```

Edit */etc/neutron/plugins/ml2/ml2_conf.ini* (default directory), change the following settings as follows in their respective tags:

```
[ml2]
type_drivers = flat,vlan,vxlan,gre
mechanism_drivers = openvswitch

[ml2_type_flat]
flat_networks = physnet1,physnet2

[ml2_type_vlan]
network_vlan_ranges = physnet1:1000:1500,physnet2:2000:2500

[ovs]
bridge_mappings = physnet1:br-ex,physnet2:br-eth1
```

You may have to repeat the last line above in another file in the Compute Node, if it exists is located in: */etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini*.

Create a route in HNAS to the tenant network. Please make sure multi-tenancy is enabled and routes are configured per EVS. Use the command route-net-add in HNAS console, where the network parameter should be the tenants private network, while the gateway parameter should be the flat network gateway and the console-context evs parameter should be the ID of EVS in use, such as in the following example:

```
$ console-context --evs 3 route-net-add --gateway 192.168.1.1 10.0.0.0/24
```

Step 3 - Share Type Configuration

Shared File Systems service requires that the share type includes the driver_handles_share_servers extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers capability. For the Hitachi NAS Platform File Services Driver for OpenStack this must be set to False.

```
$ manila type-create hitachi False
```

Additionally, the driver also reports the following common capabilities that can be specified in the share type:

Capability	Description
thin_provisioning = True	All shares created on HNAS are always thin provisioned. So, if you set it, the value **must be**: *True*.
dedupe = True/False	HNAS supports deduplication on its file systems and the driver will report *dedupe=True* if it is enabled on the file system being used. To use it, go to HNAS and enable the feature on the file system used.

To specify a common capability on the share type, use the *type-key* command, for example:

```
$ manila type-key hitachi set dedupe=True
```

Step 4 - Restart the Services

Restart all Shared File Systems services (manila-share, manila-scheduler and manila-api) and neutron services (neutron-*). This step is specific to your environment. If you are running in devstack for example, you have to log into screen (`screen -r`), stop the process (`Ctrl^C`) and run it again. If you are running it in a distro like RHEL or SUSE, a service command (for example *service manila-api restart*) is used to restart the service.

Step 5 - Configure OpenStack Networks

In Neutron Controller it is necessary to create a network, a subnet and to add this subnet interface to a router:

Create a network to the given tenant (demo), providing the DEMO_ID (this can be fetched using *keystone tenant-list*), a name for the network, the name of the physical network over which the virtual network is implemented and the type of the physical mechanism by which the virtual network is implemented:

```
$ neutron net-create --tenant-id <DEMO_ID> hnas_network
--provider:physical_network=physnet2 --provider:network_type=flat
```

Create a subnet to same tenant (demo), providing the DEMO_ID (this can be fetched using *keystone tenant-list*), the gateway IP of this subnet, a name for the subnet, the network ID created on previously step (this can be fetched using *neutron net-list*) and CIDR of subnet:

```
$ neutron subnet-create --tenant-id <DEMO_ID> --gateway <GATEWAY>
--name hnas_subnet <NETWORK_ID> <SUBNET_CIDR>
```

Finally, add the subnet interface to a router, providing the router ID and subnet ID created on previously step (can be fetched using *neutron subnet-list*):

```
$ neutron router-interface-add <ROUTER_ID> <SUBNET_ID>
```

Manage and Unmanage Shares

Manila has the ability to manage and unmanage shares. If there is a share in the storage and it is not in OpenStack, you can manage that share and use it as a manila share. Hitachi NAS Platform File Services Driver for OpenStack use virtual-volumes (V-VOLs) to create shares. Only V-VOLs with a quota limit can be used by the driver, also, they must be created or moved inside the directory /shares/ and exported (as NFS or CIFS shares). The unmanage operation only unlinks the share from OpenStack, preserving all data in the share.

To **manage** shares use:

```
$ manila manage [--name <name>] [--description <description>]
[--share_type <share_type>] [--driver_options [<key=value> [<key=value> ...]]]
<service_host> <protocol> <export_path>
```

Where:

Parameter	Description
service_host	Manila host, backend and share name. For example ubuntu@hitachi1#HITACHI1. The available hosts can be listed with the command: *manila pool-list* (admin only).
protocol	NFS or CIFS protocols are currently supported.
export_path	The export path of the share. For example: *172.24.44.31:/shares/some_share_id*

To **unmanage** a share use:

```
$ manila unmanage <share_id>
```

Where:

Parameter	Description
share_id	Manila ID of the share to be unmanaged. This list can be fetched with: *manila list*.

Additional Notes

- HNAS has some restrictions about the number of EVSs, file systems, virtual-volumes and simultaneous SSC connections. Check the manual specification for your system.
- Shares and snapshots are thin provisioned. It is reported to manila only the real used space in HNAS. Also, a snapshot does not initially take any space in HNAS, it only stores the difference between the share and the snapshot, so it grows when share data is changed.
- Admins should manage the tenants quota (*manila quota-update*) to control the backend usage.
- By default, CIFS snapshots are disabled when the share is mounted, since it uses tree-clone to create snapshots and does not guarantee point-in-time replicas when the source directory tree is changing, also, changing permissions to *read-only* does not affect already mounted shares. So, enable it if your source directory can be static while taking snapshots. Currently, it affects only CIFS protocol. For more information check the tree-clone feature in HNAS with *man tree-clone*.

The manila.share.drivers.hitachi.hnas.driver Module

class HitachiHNASDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

Manila HNAS Driver implementation.

Driver versions:

```
1.0.0 - Initial Version.
2.0.0 - Refactoring, bugfixes, implemented Share Shrink and
        Update Access.
3.0.0 - New driver location, implemented support for CIFS protocol.
3.1.0 - Added admin network export location support.
4.0.0 - Added mountable snapshots, revert-to-snapshot and
        manage snapshots features support.
```

create_share(*context*, *share*, *share_server=None*)

Creates share.

Parameters

- **context** The *context.RequestContext* object for the request
- **share** Share that will be created.
- **share_server** Data structure with share server information. Not used by this driver.

Returns

Returns a list of dicts containing the EVS IP concatenated with the path of share in the filesystem.

Example for NFS:

```
[
  {
    path: 172.24.44.10:/shares/id, metadata: {}, is_admin_only: False
  },
  {
    path: 192.168.0.10:/shares/id, metadata: {}, is_admin_only: True
  }
]
```

Example for CIFS:

```
[
  {
    path: \172.24.44.10id, metadata: {}, is_admin_only: False
```

},

{

path: \192.168.0.10id, metadata: {}, is_admin_only: True

}

]

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Creates a new share from snapshot.

Parameters

- **context** The *context.RequestContext* object for the request
- **share** Information about the new share.
- **snapshot** Information about the snapshot that will be copied to new share.
- **share_server** Data structure with share server information. Not used by this driver.

Returns

Returns a list of dicts containing the EVS IP concatenated with the path of share in the filesystem.

Example for NFS:

[

{

path: 172.24.44.10:/shares/id, metadata: {}, is_admin_only: False

},

{

path: 192.168.0.10:/shares/id, metadata: {}, is_admin_only: True

}

]

Example for CIFS:

[

{

path: \172.24.44.10id, metadata: {}, is_admin_only: False

},

{

path: \192.168.0.10id, metadata: {}, is_admin_only: True

```
        }

    ]
```

create_snapshot(*context*, *snapshot*, *share_server=None*)

Creates snapshot.

Parameters

- **context** The *context.RequestContext* object for the request
- **snapshot** Snapshot that will be created.
- **share_server** Data structure with share server information. Not used by this driver.

delete_share(*context*, *share*, *share_server=None*)

Deletes share.

Parameters

- **context** The *context.RequestContext* object for the request
- **share** Share that will be deleted.
- **share_server** Data structure with share server information. Not used by this driver.

delete_snapshot(*context*, *snapshot*, *share_server=None*)

Deletes snapshot.

Parameters

- **context** The *context.RequestContext* object for the request
- **snapshot** Snapshot that will be deleted.
- **share_server** Data structure with share server information. Not used by this driver.

ensure_share(*context*, *share*, *share_server=None*)

Ensure that share is exported.

Parameters

- **context** The *context.RequestContext* object for the request
- **share** Share that will be checked.
- **share_server** Data structure with share server information. Not used by this driver.

Returns

Returns a list of dicts containing the EVS IP concatenated with the path of share in the filesystem.

Example for NFS:

```
[
```

```
    {
```

path: 172.24.44.10:/shares/id, metadata: {}, is_admin_only: False

},

{

path: 192.168.0.10:/shares/id, metadata: {}, is_admin_only: True

}

]

Example for CIFS:

```
[
```

{

path: \172.24.44.10id, metadata: {}, is_admin_only: False

},

{

path: \192.168.0.10id, metadata: {}, is_admin_only: True

}

]

ensure_snapshot(*context*, *snapshot*, *share_server=None*)

Ensure that snapshot is exported.

Parameters

- **context** The *context.RequestContext* object for the request.
- **snapshot** Snapshot that will be checked.
- **share_server** Data structure with share server information. Not used by this driver.

Returns

Returns a list of dicts containing the EVS IP concatenated with the path of snapshot in the filesystem or None if mount_snapshot_support is False.

Example for NFS:

```
[
```

{

path: 172.24.44.10:/snapshots/id, metadata: {}, is_admin_only: False

},

{

path: 192.168.0.10:/snapshots/id, metadata: {}, is_admin_only: True

}

]

Example for CIFS:

```
[
```

{

path: \172.24.44.10id, metadata: {}, is_admin_only: False

},

{

path: \192.168.0.10id, metadata: {}, is_admin_only: True

}

]

extend_share(*share*, *new_size*, *share_server=None*)

Extends a share to new size.

Parameters

- **share** Share that will be extended.
- **new_size** New size of share.
- **share_server** Data structure with share server information. Not used by this driver.

get_network_allocations_number()

Track allocations_number in DHSS = true.

When using the setting driver_handles_share_server = false does not require to track allocations_number because we do not handle network stuff.

manage_existing(*share*, *driver_options*)

Manages a share that exists on backend.

Parameters

- **share** Share that will be managed.
- **driver_options** Empty dict or dict with volume_id option.

Returns

Returns a dict with size of the share managed and a list of dicts containing its export locations.

Example for NFS:

```
{
```

size: 10, export_locations: [

{

path: 172.24.44.10:/shares/id, metadata: {}, is_admin_only: False

},

{

path: 192.168.0.10:/shares/id, metadata: {}, is_admin_only: True

}

]

}

Example for CIFS:

```
{
```

size: 10, export_locations: [

{

path: \172.24.44.10id, metadata: {}, is_admin_only: False

},

{

path: \192.168.0.10id, metadata: {}, is_admin_only: True

}

]

}

manage_existing_snapshot(*snapshot*, *driver_options*)

Manages a snapshot that exists only in HNAS.

The snapshot to be managed should be in the path /snapshots/SHARE_ID/SNAPSHOT_ID. Also, the size of snapshot should be provided as driver_options size=<size>. :param snapshot: snapshot that will be managed. :param driver_options: expects only one key size. It must be provided in order to manage a snapshot.

Returns Returns a dict with size of snapshot managed

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a share to a given snapshot.

Parameters

- **context** The *context.RequestContext* object for the request
- **snapshot** The snapshot to which the share is to be reverted to.
- **share_access_rules** List of all access rules for the affected share. Not used by this driver.

- **snapshot_access_rules** List of all access rules for the affected snapshot. Not used by this driver.
- **share_server** Data structure with share server information. Not used by this driver.

shrink_share(*share*, *new_size*, *share_server=None*)
Shrinks a share to new size.

Parameters

- **share** Share that will be shrunk.
- **new_size** New size of share.
- **share_server** Data structure with share server information. Not used by this driver.

snapshot_update_access(*context*, *snapshot*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given snapshot.

Drivers should support 2 different cases in this method: 1. Recovery after error - access_rules contains all access rules, add_rules and delete_rules shall be empty. Driver should clear any existent access rules and apply all access rules for given snapshot. This recovery is made at driver start up.

2. Adding/Deleting of several access rules - access_rules contains all access rules, add_rules and delete_rules contain rules which should be added/deleted. Driver can ignore rules in access_rules and apply only rules from add_rules and delete_rules. All snapshots rules should be read only.

Parameters

- **context** Current context
- **snapshot** Snapshot model with snapshot data.
- **access_rules** All access rules for given snapshot
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

unmanage(*share*)
Unmanages a share.

Parameters **share** Share that will be unmanaged.

unmanage_snapshot(*snapshot*)
Unmanage a share snapshot

Parameters **snapshot** Snapshot that will be unmanaged.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given share.

Parameters

- **context** The *context.RequestContext* object for the request
- **share** Share that will have its access rules updated.
- **access_rules** All access rules for given share.
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** Data structure with share server information. Not used by this driver.

HPE 3PAR Driver for OpenStack Manila

The HPE 3PAR manila driver provides NFS and CIFS shared file systems to OpenStack using HPE 3PARs File Persona capabilities.

Note: In OpenStack releases prior to Mitaka this driver was called the HP 3PAR driver. The Liberty configuration reference can be found at: http://docs.openstack.org/liberty/config-reference/content/hp-3par-share-driver.html

---

For information on HPE 3PAR Driver for OpenStack Manila, refer to content kit page.

Supported Operations

The following operations are supported with HPE 3PAR File Persona:

- Create/delete NFS and CIFS shares
  - Shares are not accessible until access rules allow access
- Allow/deny NFS share access
  - IP access rules are required for NFS share access
- Allow/deny CIFS share access
  - CIFS shares require user access rules.
  - User access requires a 3PAR local or AD user (LDAP is not yet supported)
- Create/delete snapshots
- Create shares from snapshots

Share networks are not supported. Shares are created directly on the 3PAR without the use of a share server or service VM. Network connectivity is setup outside of manila.

Requirements

On the system running the manila share service:

- python-3parclient 4.2.0 or newer from PyPI.

On the HPE 3PAR array:

- HPE 3PAR Operating System software version 3.2.1 MU3 or higher
- The array class and hardware configuration must support File Persona

Pre-Configuration on the HPE 3PAR

- HPE 3PAR File Persona must be initialized and started (`startfs`)
- A File Provisioning Group (FPG) must be created for use with manila
- A Virtual File Server (VFS) must be created for the FPG
- The VFS must be configured with an appropriate share export IP address
- A local user in the Administrators group is needed for CIFS shares

Backend Configuration

The following parameters need to be configured in the manila configuration file for the HPE 3PAR driver:

- *share_backend_name* = <backend name to enable>
- *share_driver* = manila.share.drivers.hpe.hpe_3par_driver.HPE3ParShareDriver
- *driver_handles_share_servers* = False
- *hpe3par_fpg* = <FPG to use for share creation>
- *hpe3par_share_ip_address* = <IP address to use for share export location>
- *hpe3par_san_ip* = <IP address for SSH access to the SAN controller>
- *hpe3par_api_url* = <3PAR WS API Server URL>
- *hpe3par_username* = <3PAR username with the edit role>
- *hpe3par_password* = <3PAR password for the user specified in hpe3par_username>
- *hpe3par_san_login* = <Username for SSH access to the SAN controller>
- *hpe3par_san_password* = <Password for SSH access to the SAN controller>
- *hpe3par_debug* = <False or True for extra debug logging>
- *hpe3par_cifs_admin_access_username* = <CIFS admin user name>
- *hpe3par_cifs_admin_access_password* = <CIFS admin password>
- *hpe3par_cifs_admin_access_domain* = <CIFS admin domain>
- *hpe3par_share_mount_path* = <Full path to mount shares>

The *hpe3par_share_ip_address* must be a valid IP address for the configured FPGs VFS. This IP address is used in export locations for shares that are created. Networking must be configured to allow connectivity from clients to shares.

hpe3par_cifs_admin_access_username and *hpe3par_cifs_admin_access_password* must be provided to delete nested CIFS shares. If they are not, the share contents will not be deleted. *hpe3par_cifs_admin_access_domain* and *hpe3par_share_mount_path* can be provided for additional configuration.

Restart of *manila-share* service is needed for the configuration changes to take effect.

Backend Configuration for AD user

The following parameters need to be configured through HPE 3PAR CLI to access file share using AD.

Set authentication parameters:

```
$ setauthparam ldap-server IP_ADDRESS_OF_AD_SERVER
$ setauthparam binding simple
$ setauthparam user-attr AD_DOMAIN_NAME\\
$ setauthparam accounts-dn CN=Users,DC=AD,DC=DOMAIN,DC=NAME
$ setauthparam account-obj user
$ setauthparam account-name-attr sAMAccountName
$ setauthparam memberof-attr memberOf
$ setauthparam super-map CN=AD_USER_GROUP,DC=AD,DC=DOMAIN,DC=NAME
```

Verify new authentication parameters set as expected:

```
$ showauthparam
```

Verify AD users set as expected:

```
$ checkpassword AD_USER
```

Command result should show `user AD_USER is authenticated and authorized` message on successful configuration.

Add ActiveDirectory in authentication providers list:

```
$ setfs auth ActiveDirectory Local
```

Verify authentication provider list shows ActiveDirectory:

```
$ showfs -auth
```

Set/Add AD user on FS:

```
$ setfs ad passwd PASSWORD AD_USER AD_DOMAIN_NAME
```

Verify FS user details:

```
$ showfs -ad
```

Example of using AD user to access CIFS share

Pre-requisite:

- Share type should be configured for 3PAR backend

Create a CIFS file share with 2GB of size:

```
$ manila create --name FILE_SHARE_NAME --share-type SHARE_TYPE CIFS 2
```

Check file share created as expected:

```
$ manila show FILE_SHARE_NAME
```

Configuration to provide share access to AD user:

```
$ manila access-allow FILE_SHARE_NAME user AD_DOMAIN_NAME\\\\AD_USER
  --access-level rw
```

Check users permission set as expected:

```
$ manila access-list FILE_SHARE_NAME
```

The AD_DOMAIN_NAME\AD_USER must be listed in access_to column and should show active in its state column as result of this command.

Network Approach

Connectivity between the storage array (SSH/CLI and WSAPI) and the manila host is required for share management.

Connectivity between the clients and the VFS is required for mounting and using the shares. This includes:

- Routing from the client to the external network
- Assigning the client an external IP address (e.g., a floating IP)
- Configuring the manila host networking properly for IP forwarding
- Configuring the VFS networking properly for client subnets

Share Types

When creating a share, a share type can be specified to determine where and how the share will be created. If a share type is not specified, the *default_share_type* set in the manila configuration file is used.

Manila requires that the share type includes the *driver_handles_share_servers* extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers (share networks) capability. For the HPE 3PAR driver, this must be set to False.

Another common manila extra-spec used to determine where a share is created is *share_backend_name*. When this extra-spec is defined in the share type, the share will be created on a backend with a matching share_backend_name.

The HPE 3PAR driver automatically reports capabilities based on the FPG used for each backend. Share types with extra specs can be created by an administrator to control which share types are allowed to use FPGs with or without specific capabilities. The following extra-specs are used with the capabilities filter and the HPE 3PAR driver:

- *hpe3par_flash_cache* = <is> True or <is> False
- *thin_provisioning* = <is> True or <is> False
- *dedupe* = <is> True or <is> False

hpe3par_flash_cache will be reported as True for backends that have 3PARs Adaptive Flash Cache enabled.

thin_provisioning will be reported as True for backends that use thin provisioned volumes. FPGs that use fully provisioned volumes will report False. Backends that use thin provisioning also support manilas over-subscription feature.

dedupe will be reported as True for backends that use deduplication technology.

Scoped extra-specs are used to influence vendor-specific implementation details. Scoped extra-specs use a prefix followed by a colon. For HPE 3PAR these extra-specs have a prefix of *hpe3par*. For HP 3PAR these extra-specs have a prefix of *hp3par*.

The following HPE 3PAR extra-specs are used when creating CIFS (SMB) shares:

- *hpe3par:smb_access_based_enum* = true or false
- *hpe3par:smb_continuous_avail* = true or false
- *hpe3par:smb_cache* = off, manual, optimized or auto

smb_access_based_enum (Access Based Enumeration) specifies if users can see only the files and directories to which they have been allowed access on the shares. The default is *false*.

smb_continuous_avail (Continuous Availability) specifies if SMB3 continuous availability features should be enabled for this share. If not specified, the default is *true*. This setting will be ignored with hp3parclient 3.2.1 or earlier.

smb_cache specifies client-side caching for offline files. Valid values are:

- *off*: The client must not cache any files from this share. The share is configured to disallow caching.
- *manual*: The client must allow only manual caching for the files open from this share.
- *optimized*: The client may cache every file that it opens from this share. Also, the client may satisfy the file requests from its local cache. The share is configured to allow automatic caching of programs and documents.
- *auto*: The client may cache every file that it opens from this share. The share is configured to allow automatic caching of documents.
- If this is not specified, the default is *manual*.

The following HPE 3PAR extra-specs are used when creating NFS shares:

- *hpe3par:nfs_options* = Comma separated list of NFS export options

The NFS export options have the following limitations:

- *ro* and *rw* are not allowed (manila will determine the read-only option)
- *no_subtree_check* and *fsid* are not allowed per HPE 3PAR CLI support

- *(in)secure* and *(no_)root_squash* are not allowed because the HPE 3PAR driver controls those settings

All other NFS options are forwarded to the HPE 3PAR as part of share creation. The HPE 3PAR will do additional validation at share creation time. Refer to HPE 3PAR CLI help for more details.

Delete Nested Shares

When a nested share is deleted (nested shares will be created when `hpe_3par_fstore_per_share` is set to `False`), the file tree also attempts to be deleted.

With NFS shares, there is no additional configuration that needs to be done.

For CIFS shares, `hpe3par_cifs_admin_access_username` and `hpe3par_cifs_admin_access_password` must be provided. If they are omitted, the original functionality is honored and the file tree remains untouched. `hpe3par_cifs_admin_access_domain` and `hpe3par_share_mount_path` can also be specified to create further customization.

The `manila.share.drivers.hpe.hpe_3par_driver` Module

HPE 3PAR Driver for OpenStack Manila.

class FPG(*min_ip=0*, *max_ip=4*, *type_name='FPG'*)

Bases: `oslo_config.types.String`, `oslo_config.types.IPAddress`

FPG type.

Used to represent multiple pools per backend values. Converts configuration value to an FPGs value. FPGs value format:

```
FPG name, IP address 1, IP address 2, ..., IP address 4
```

where FPG name is a string value, IP address is of type types.IPAddress

Optionally doing range checking. If value is whitespace or empty string will raise error

Parameters

- **min_ip** Optional check that number of min IP address of VFS.
- **max_ip** Optional check that number of max IP address of VFS.
- **type_name** Type name to be used in the sample config file.

MAX_SUPPORTED_IP_PER_VFS = 4

class HPE3ParShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

HPE 3PAR driver for Manila.

Supports NFS and CIFS protocols on arrays with File Persona.

Version history:

```
1.0.0 - Begin Liberty development (post-Kilo)
1.0.1 - Report thin/dedup/hp_flash_cache capabilities
1.0.2 - Add share server/share network support
2.0.0 - Rebranded HP to HPE
2.0.1 - Add access_level (e.g. read-only support)
2.0.2 - Add extend/shrink
2.0.3 - Remove file tree on delete when using nested shares #1538800
2.0.4 - Reduce the fsquota by share size
        when a share is deleted #1582931
2.0.5 - Add update_access support
2.0.6 - Multi pool support per backend
2.0.7 - Fix get_vfs() to correctly validate conf IP addresses at
        boot up #1621016
2.0.8 - Replace ConsistencyGroup with ShareGroup
```

VERSION = '2.0.8'

static build_share_comment(*share*)
Create an informational only comment to help admins and testers.

check_for_setup_error()
Check for setup error.

choose_share_server_compatible_with_share(*context*, *share_servers*, *share*, *snapshot=None*, *share_group=None*)
Method that allows driver to choose share server for provided share.

If compatible share-server is not found, method should return None.

Parameters

- **context** Current context
- **share_servers** list with share-server models
- **share** share model
- **snapshot** snapshot model
- **share_group** ShareGroup model with shares

Returns share-server or None

create_share(*context*, *share*, *share_server=None*)
Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)
Is called to create share from snapshot.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Creates a snapshot of a share.

delete_share(*context*, *share*, *share_server=None*)
Deletes share and its fstore.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Deletes a snapshot of a share.

do_setup(*context*)
Any initialization the share driver does while starting.

ensure_share(*context*, *share*, *share_server=None*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

extend_share(*share*, *new_size*, *share_server=None*)
Extends size of existing share.

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

static sha1_hash(*clazz*)
Get the SHA1 hash for the source of a class.

shrink_share(*share*, *new_size*, *share_server=None*)
Shrinks size of existing share.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access to the share.

to_list(*var*)
Convert var to list type if not

Infortrend Driver for OpenStack Manila

The Infortrend Manila driver provides NFS and CIFS shared file systems to Openstack.

Requirements

- The EonStor GS/GSe series Fireware version 139A23

Supported shared filesystems and operations

This driver supports NFS and CIFS shares.

The following operations are supported:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- Allow CIFS/NFS Share access
  - Only IP access type is supported for NFS (ro/rw).

  - Only USER access type is supported for CIFS (ro/rw).
- Deny CIFS/NFS Share access
- Manage a share.
- Unmanage a share.
- Extend a share.
- Shrink a share.

Backend Configuration

The following parameters need to be configured in the manila configuration file for the Infortrend driver:

- *share_backend_name* = <backend name to enable>
- *share_driver* = manila.share.drivers.infortrend.driver.InfortrendNASDriver
- *driver_handles_share_servers* = False
- *infortrend_nas_ip* = <IP address for SSH access to the SAN controller>
- *infortrend_nas_user* = <username with the edit role>
- *infortrend_nas_password* = <password for the user specified in infortrend_nas_user>
- *infortrend_share_pools* = <Poolname of the SAN controller>
- *infortrend_share_channels* = <Data channel for file service in SAN controller>

Share Types

When creating a share, a share type can be specified to determine where and how the share will be created. If a share type is not specified, the *default_share_type* set in the manila configuration file is used.

Manila requires that the share type includes the *driver_handles_share_servers* extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers (share networks) capability. For the Infortrend driver, this must be set to False.

Back-end configuration example

```
[DEFAULT]
enabled_share_backends = ift-manila
enabled_share_protocols = NFS, CIFS

[ift-manila]
share_backend_name = ift-manila
share_driver = manila.share.drivers.infortrend.driver.InfortrendNASDriver
driver_handles_share_servers = False
infortrend_nas_ip = FAKE_IP
infortrend_nas_user = FAKE_USER
infortrend_nas_password = FAKE_PASS
```

(continues on next page)

(continued from previous page)

```
infortrend_share_pools = pool-1, pool-2
infortrend_share_channels = 0, 1
```

Pure Storage FlashBlade Driver for OpenStack Manila

The Pure Storage FlashBlade Manila driver provides NFS shared file systems to OpenStack using Pure Storages FlashBlade native filesystem capabilities.

Supported Operations

The following operations are supported with Pure Storage FlashBlade:

- Create/delete NFS shares
  - Shares are not accessible until access rules allow access
- Allow/deny NFS share access
  - IP access rules are required for NFS share access
- Create/delete snapshots
- Expand and Shrink shares
- Revert to Snapshot

Share networks are not supported. Shares are created directly on the FlashBlade without the use of a share server or service VM. Network connectivity is setup outside of Manila.

General Requirements

On the system running the Manila share service:

- purity_fb 1.12.1 or newer from PyPI.

On the Pure Storage FlashBlade:

- Purity//FB Operating System software version 2.3.0 or higher

Network Requirements

Connectivity between the FlashBlade (REST) and the manila host is required for share management.

Connectivity between the clients and the FlashBlade is required for mounting and using the shares. This includes:

- Routing from the client to the external network
- Assigning the client an external IP address (e.g., a floating IP)
- Configuring the manila host networking properly for IP forwarding
- Configuring the FlashBlade networking properly for client subnets

Driver Configuration

Before configuring the driver, make sure the following networking requirements have been met:

- A management subnet must be accessible from the system running the Manila share services
- A data subnet must be accessible from the system running the Nova compute services
- An API token must be available for a user with administrative privileges

Perform the following steps:

1. Configure the Pure Storage FlashBlade parameters in *manila.conf*
2. Configure/create a share type
3. Restart the services

It is also assumed that the OpenStack networking has been confiured correctly.

Step 1 - FlashBlade Parameters configuration

The following parameters need to be configured in the [DEFAULT] section of */etc/manila/manila.conf*:

Option	Description
enabled_share_backends	Name of the section on `manila.conf` used to specify a backend. For example: *enabled_share_backends = flashblade*
enabled_share_protocols	Specify a list of protocols to be allowed for share creation. This driver version only supports NFS

The following parameters need to be configured in the [backend] section of `/etc/manila/manila.conf`:

Option	Description
share_backend_name	A name for the backend.
share_driver	Python module path. For this driver **this must be**: *manila.share.drivers.purestorage.flashblade.FlashBladeShareDriver*
driver_handles_share_servers	Driver share server mode. For this driver **this must be**: *False*.
flashblade_mgmt_vip	The name (or IP address) for the Pure Storage FlashBlade storage system management VIP
flashblade_data_vip	The name (or IP address) for the Pure Storage FlashBlade storage system data VIP.
flashblade_api	API token for an administrative user account
flashblade_eradicate (Optional)	When enabled, all FlashBlade file systems and snapshots will be eradicated at the time of deletion in Manila. Data will NOT be recoverable after a delete with this set to True! When disabled, file systems and snapshots will go into pending eradication state and can be recovered. Default value is *True*.

Below is an example of a valid configuration of the FlashBlade driver:

```
[DEFAULT]
...
enabled_share_backends = flashblade
enabled_share_protocols = NFS
...

[flashblade]
share_backend_name = flashblade
share_driver = manila.share.drivers.purestorage.flashblade.
↪FlashBladeShareDriver
driver_handles_share_servers = False
flashblade_mgmt_vip = 1.2.3.4
flashblade_data_vip = 1.2.3.5
flashblade_api = T-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Restart of *manila-share* service is needed for the configuration changes to take effect.

Step 2 - Share Type Configuration

Shared File Systems service requires that the share type includes the driver_handles_share_servers extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers capability. For the Pure Storage FlashBlade Driver for OpenStack this must be set to False.

```
$ manila type-create flashblade False
```

Additionally, the driver also reports the following common capabilities that can be specified in the share type:

Capability	Description
thin_provisioning = True	All shares created on FlashBlade are always thin provisioned. If you set it this, the value **must be**: *True*.
snapshot_support = True/False	FlashBlade supports share snapshots. If you set this, the value **must be**: *True*.
revert_to_snapshot = True/False	FlashBlade supports reverting a share to the latest available snapshot. If you set this, the value **must be**: *True*.

To specify a common capability on the share type, use the *type-key* command, for example:

```
$ manila type-key flashblade set snapshot_support=True
$ manila type-key flashblade set revert_to_snapshot=True
```

Step 3 - Restart the Services

Restart all Shared File Systems services (manila-share, manila-scheduler and manila-api). This step is specific to your environment. for example, *systemctl restart <controller>@manila-shr* is used to restart the share service.

The `manila.share.drivers.purestorage.flashblade` Module

Pure Storage FlashBlade Share Driver

class FlashBladeShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

USER_AGENT_BASE = 'OpenStack Manila'

VERSION = '2.0'

create_share(*context*, *share*, *share_server=None*)

Create a share and export it based on protocol used.

create_snapshot(*context*, *snapshot*, *share_server=None*)

Called to create a snapshot

delete_share(*context*, *share*, *share_server=None*)

Called to delete a share

delete_snapshot(*context*, *snapshot*, *share_server=None*)

Called to delete a snapshot

do_setup(*context*)

Driver initialization

ensure_share(*context*, *share*, *share_server=None*)

Dummy - called to ensure share is exported.

All shares created on a FlashBlade are guaranteed to be exported so this check is redundant

extend_share(*share*, *new_size*, *share_server=None*)

uses resize_share to extend a share

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a share (in place) to the specified snapshot.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **snapshot** The snapshot to be restored
- **share_access_rules** List of all access rules for the affected share

- **snapshot_access_rules** List of all access rules for the affected snapshot
- **share_server** Optional Share server model or None

shrink_share(*share*, *new_size*, *share_server=None*)
uses resize_share to shrink a share

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access of share

purity_fb_to_manila_exceptions(*func*)

Tegile Driver

The Tegile Manila driver uses Tegile IntelliFlash Arrays to provide shared filesystems to OpenStack.

The Tegile Driver interfaces with a Tegile Array via the REST API.

Requirements

- Tegile IntelliFlash version 3.5.1
- For using CIFS, Active Directory must be configured in the Tegile Array.

Supported Operations

The following operations are supported on a Tegile Array:

- Create CIFS/NFS Share
- Delete CIFS/NFS Share
- **Allow CIFS/NFS Share access**
  - Only IP access type is supported for NFS
  - USER access type is supported for NFS and CIFS
  - RW and RO access supported
- **Deny CIFS/NFS Share access**
  - IP access type is supported for NFS
  - USER access type is supported for NFS and CIFS
- Create snapshot
- Delete snapshot
- Extend share
- Shrink share
- Create share from snapshot

Backend Configuration

The following parameters need to be configured in the [DEFAULT] section of */etc/manila/manila.conf*:

[DEFAULT]	
Option	**Description**
enabled_share_backends	Name of the section on manila.conf used to specify a backend. E.g. *enabled_share_backends = tegileNAS*
enabled_share_protocols	Specify a list of protocols to be allowed for share creation. For Tegile driver this can be: *NFS* or *CIFS* or *NFS, CIFS*.

The following parameters need to be configured in the [backend] section of */etc/manila/manila.conf*:

[tegileNAS]	
Option	**Description**
share_backend_name	A name for the backend.
share_driver	Python module path. For Tegile driver this must be: *manila.share.drivers.tegile.tegile.TegileShareDriver*.
driver_handles_share_servers	DHSS, Driver working mode. For Tegile driver **this must be**: *False*.
tegile_nas_server	Tegile array IP to connect from the Manila node.
tegile_nas_login	This field is used to provide username credential to Tegile array.
tegile_nas_password	This field is used to provide password credential to Tegile array.
tegile_default_project	This field can be used to specify the default project in Tegile array where shares are created. This field is optional.

Below is an example of a valid configuration of Tegile driver:

```
[DEFAULT]
enabled_share_backends = tegileNAS
enabled_share_protocols = NFS,CIFS


[tegileNAS]
driver_handles_share_servers = False
share_backend_name = tegileNAS
share_driver = manila.share.drivers.tegile.tegile.TegileShareDriver
tegile_nas_server = 10.12.14.16
tegile_nas_login = admin
tegile_nas_password = password
tegile_default_project = financeshares
```

Restart of *manila-share* service is needed for the configuration changes to take effect.

Restrictions

The Tegile driver has the following restrictions:

- IP access type is supported only for NFS.
- Only FLAT network is supported.

The manila.share.drivers.tegile.tegile Module

Share driver for Tegile storage.

class TegileShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

Tegile NAS driver. Allows for NFS and CIFS NAS storage usage.

create_share(***kwds*)

Is called to create share.

create_share_from_snapshot(***kwds*)

Is called to create share from snapshot.

Creating a share from snapshot can take longer than a simple clone operation if data copy is required from one host to another. For this reason driver will be able complete this creation asynchronously, by providing a creating_from_snapshot status in the model update.

When answering asynchronously, drivers must implement the call get_share_status in order to provide updates for shares with creating_from_snapshot status.

It is expected that the driver returns a model update to the share manager that contains: share status and a list of export_locations. A list of export_locations is mandatory only for share in available status. The current supported status are available and creating_from_snapshot.

Parameters

- **context** Current context
- **share** Share instance model with share data.
- **snapshot** Snapshot instance model .
- **share_server** Share server model or None.
- **parent_share** Share model from parent snapshot with share data and share server model.

Returns

a dictionary of updates containing current share status and its export_location (if available).

Example:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance to error and the operation will end.

`create_snapshot`(***kwds*)
Is called to create snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

Returns None or a dictionary with key export_locations containing a list of export locations, if snapshots can be mounted.

`delete_share`(***kwds*)
Is called to remove share.

`delete_snapshot`(***kwds*)
Is called to remove snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

`ensure_share`(***kwds*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

`extend_share`(***kwds*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

`get_network_allocations_number`(***kwds*)
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

`get_pool`(***kwds*)
Return pool name where the share resides on.

Parameters **share** The share hosted by the driver.

shrink_share(***kwds*)
Shrinks size of existing share.

If consumed space on share larger than new_size driver should raise ShareShrinkingPossibleDataLoss exception: raise ShareShrinkingPossibleDataLoss(share_id=share[id])

Parameters

- **share** Share model
- **new_size** New size of share (new_size < share[size])
- **share_server** Optional Share server model

:raises ShareShrinkingPossibleDataLoss, NotImplementedError

update_access(***kwds*)
Update access rules for given share.

`access_rules` contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the `add_rules` and `delete_rules` parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in `add_rules` and rules that need to be removed to be present in `delete_rules`.

When a rule in `delete_rules` was never applied, drivers must not raise an exception, or attempt to set the rule to `error` state.

`add_rules` and `delete_rules` can be empty lists, in this situation, drivers should ensure that the rules present in `access_rules` are the same as those on the back end. One scenario where this situation is forced is when the access_level is changed for all existing rules (share migration and for readable replicas).

Drivers must be mindful of this call for share replicas. When update_access is called on one of the replicas, the call is likely propagated to all replicas belonging to the share, especially when individual rules are added or removed. If a particular access rule does not make sense to the driver in the context of a given replica, the driver should be careful to report a correct behavior, and take meaningful action. For example, if R/W access is requested on a replica that is part of a readable type replication; R/O access may be added by the driver instead of R/W. Note that raising an exception *will* result in the access_rules_status on the replica, and the share itself being out_of_sync. Drivers can sync on the valid access rules that are provided on the `create_replica` and `promote_replica` calls.

Parameters

- **context** Current context
- **share** Share model with share data.
- **access_rules** A list of access rules for given share
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

Returns

None, or a dictionary of updates in the format:

```
{
    09960614-8574-4e03-89cf-7cf267b0bd08: {
        access_key: alice31493e5441b8171d2310d80e37e, state: error,
    },
    28f6eabb-4342-486a-a7f4-45688f0c0295: {
        access_key: bob0078aa042d5a7325480fd13228b, state: active,
    },
}
```

The top level keys are access_id fields of the access rules that need to be updated. `access_key``s are credentials (str) of the entities granted access. Any rule in the ``access_rules` parameter can be updated.

Important: Raising an exception in this method will force *all* rules in applying and denying states to error.

An access rule can be set to error state, either explicitly via this return parameter or because of an exception raised in this method. Such an access rule will no longer be sent to the driver on subsequent access rule updates. When users deny that rule however, the driver will be asked to deny access to the client/s represented by the rule. We expect that a rule that was error-ed at the driver should never exist on the back end. So, do not fail the deletion request.

Also, it is possible that the driver may receive a request to add a rule that is already present on the back end. This can happen if the share manager service goes down while the driver is committing access rule changes. Since we cannot determine if the rule was applied successfully by the driver before the disruption, we will treat all applying transitional rules as new rules and repeat the request.

NexentaStor5 Driver for OpenStack Manila

The NexentaStor5 Manila driver provides NFS shared file systems to OpenStack.

Requirements

- The NexentaStor 5.1 or newer

Supported shared filesystems and operations

This driver supports NFS shares.

The following operations are supported:

- Create NFS Share
- Delete NFS Share
- Allow NFS Share access
  - Only IP access type is supported for NFS (ro/rw).
- Deny NFS Share access
- Manage a share.
- Unmanage a share.
- Extend a share.
- Shrink a share.
- Create snapshot
- Revert to snapshot
- Delete snapshot
- Create share from snapshot

Backend Configuration

The following parameters need to be configured in the manila configuration file for the NexentaStor5 driver:

- *share_backend_name* = <backend name to enable>
- *share_driver* = manila.share.drivers.nexenta.ns5.nexenta_nas.NexentaNasDriver
- *driver_handles_share_servers* = False
- *nexenta_nas_host* = <Data address to NAS shares>
- *nexenta_user* = <username for management operations>
- *nexenta_password* = <password for management operations>
- *nexenta_pool* = <Pool name where NAS shares are created>
- *nexenta_rest_addresses* = <Management address for Rest API access>
- *nexenta_folder* = <Parent filesystem where all Manila shares are kept>
- *nexenta_nfs* = True

Share Types

When creating a share, a share type can be specified to determine where and how the share will be created. If a share type is not specified, the *default_share_type* set in the manila configuration file is used.

Manila requires that the share type includes the *driver_handles_share_servers* extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers (share networks) capability. For the NexentaStor driver, this extra-specs value must be set to False.

Restrictions

- Only IP share access control is allowed for NFS shares.

Back-end configuration example

```
[DEFAULT]
enabled_share_backends = NexentaStor5

[NexentaStor5]
share_backend_name = NexentaStor5
driver_handles_share_servers = False
nexenta_folder = manila
share_driver = manila.share.drivers.nexenta.ns5.nexenta_nas.NexentaNasDriver
nexenta_rest_addresses = 10.3.1.1,10.3.1.2
nexenta_nas_host = 10.3.1.10
nexenta_rest_port = 8443
nexenta_pool = pool1
nexenta_nfs = True
nexenta_user = admin
nexenta_password = secret_password
nexenta_thin_provisioning = True
```

Windows SMB driver

While the generic driver only supports Linux instances, you may use the Windows SMB driver when Windows VMs are preferred.

This driver extends the generic one in order to provide Windows instance support. It can integrate with Active Directory domains through the Manila security service feature, which can ease access control.

Although Samba is a great SMB share server, Windows instances may provide improved SMB 3 support.

Limitations

- ip access rules are not supported at the moment, only user based ACLs may be used
- SMB (also known as CIFS) is the only supported share protocol
- although it can handle Windows VMs, Manila cannot run on Windows at the moment. The VMs on the other hand may very well run on Hyper-V, KVM or any other hypervisor supported by Nova.

Prerequisites

This driver requires a Windows Server image having cloudbase-init installed. Cloudbase-init is the de-facto standard tool for initializing Windows VMs running on OpenStack. The driver relies on it to do tasks such as:

- configuring WinRM access using password or certificate based authentication
- network configuration
- setting the host name

Note: This driver was initially developed with Windows Nano Server in mind. Unfortunately, Microsoft no longer supports running Nano Servers on bare metal or virtual machines, for which reason you may want to use Windows Server Core images.

Configuring

Below is a config sample that enables the Windows SMB driver.

```
[DEFAULT]
manila_service_keypair_name = manila-service
enabled_share_backends = windows_smb
enabled_share_protocols = CIFS

[windows_smb]
service_net_name_or_ip = private
tenant_net_name_or_ip = private

share_mount_path = C:/shares
# The driver can either create share servers by itself
# or use existing ones.
driver_handles_share_servers = True
service_instance_user = Admin
service_image_name = ws2016

# nova get-password may be used to retrieve passwords generated
# by cloudbase-init and encrypted with the public key.
path_to_private_key = /etc/manila/ssh/id_rsa
path_to_public_key = /etc/manila/ssh/id_rsa.pub
winrm_cert_pem_path = /etc/manila/ssl/winrm_client_cert.pem
```

(continues on next page)

(continued from previous page)

```
winrm_cert_key_pem_path = /etc/manila/ssl/winrm_client_cert.key
# If really needed, you can use password based authentication as well.
winrm_use_cert_based_auth = True
winrm_conn_timeout = 40
max_time_to_build_instance = 900

share_backend_name = windows_smb
share_driver = manila.share.drivers.windows.windows_smb_driver.
↪WindowsSMBDriver
service_instance_flavor_id = 100
```

Zadara VPSA Driver for OpenStack Manila

Zadaras Virtual Private Storage Array (VPSA) is the first software defined, Enterprise-Storage-as-a-Service. It is an elastic and private block and file storage system which provides enterprise-grade data protection and data management storage services.

Manila VPSA driver provides a seamless management capabilities for VPSA volumes, in this case, NFS & SMB volumes without losing the added value provided by the VPSA Storage Array/Flash-Array.

Requirements

- VPSA Storage Array/Flash-Array running version 20.12 or higher.
- Networking preparation - the Zadara VPSA driver for Manila support DHSS=False (driver_handles_share_servers), the driver does not handle the network configuration, it is up to the administrator to ensure connectivity from a manila-share node and the Openstack cloud to the VPSA Front-End network (such as neutron flat/VLAN network).

Supported shared filesystems and operations

Share file system supported

- SMB (CIFS)
- NFS

Supported operations

The following operations are supported:

- Create a share.
- Delete a share.
- Extend a share.
- Create a snapshot.
- Delete a snapshot.

- Create a share from snapshot.
- Allow share access.
- Manage a share.

Note:

- Only IP access type is supported
- Both RW and RO access levels supported

Backend Configuration

The following parameters need to be configured in the [DEFAULT] section of manila configuration (/etc/manila/manila.conf):

- *enabled_share_backends* = Name of the section on manila.conf used to specify a backend i.e. *enabled_share_backends = zadaravpsa*
- *enabled_share_protocols* - Specify a list of protocols to be allowed for share creation. The VPSA driver support the following options: *NFS* or *CIFS* or *NFS, CIFS*

The following parameters need to be configured in the [backend] section of manila configuration (/etc/manila/manila.conf):

Driver options

- *zadara_vpsa_host* = <VPSA - Management Host name or IP address>
- *zadara_vpsa_port* = <VPSA - Port number>
- *zadara_vpsa_use_ssl* = <VPSA - Use SSL connection (default=False)
- *zadara_ssl_cert_verify* = <If set to True the http client will validate the SSL certificate of the VPSA endpoint (default=True)>
- *zadara_driver_ssl_cert_path* = <Can be used to specify a non default path to a CA_BUNDLE file or directory with certificates of trusted CAs (default=None)
- *zadara_access_key* - <VPSA access key>
- *zadara_vpsa_poolname* - <VPSA - Storage Pool assigned for volumes>
- *zadara_vol_encrypt* = <VPSA - Default encryption policy for volumes (default = True)
- *zadara_gen3_vol_dedupe* = <VPSA - Default encryption policy for volumes (default = True)>
- *zadara_gen3_vol_compress* = <VPSA - Enable compression for volumes (default=False)>
- *zadara_share_name_template* = <VPSA - Default template for VPSA share names (default=OS_share-%s>
- *zadara_share_snap_name_template* = <VPSA - Default template for VPSA share snapshot names (default=OS_share-snapshot-%s)
- *zadara_default_snap_policy* = <VPSA - Attach snapshot policy for volumes (default=False)>

- *driver_handles_share_servers* = <DHSS, driver working mode (must be set to False)>
- *share_driver* = manila.share.drivers.zadara.zadara.ZadaraVPSAShareDriver

Back-end configuration example

```
[DEFAULT]
enabled_share_backends = zadaravpsa
enabled_share_protocols = NFS,CIFS

[zadaravpsa]
driver_handles_share_servers = False
zadara_vpsa_host = vsa-00000010-mycloud.zadaravpsa.com
zadara_vpsa_port = 443
zadara_access_key = MYSUPERSECRETACCESSKEY
zadara_vpsa_poolname = pool-00010001
share_backend_name = zadaravpsa
zadara_vpsa_use_ssl = true
share_driver = manila.share.drivers.zadara.zadara.ZadaraVPSAShareDriver
```

3.3 Reference

Contents

3.3.1 Configuration

Introduction to the Shared File Systems service

The Shared File Systems service provides shared file systems that Compute instances can consume.

The overall Shared File Systems service is implemented via the following specific services:

manila-api A WSGI app that authenticates and routes requests throughout the Shared File Systems service. It supports the OpenStack APIs.

manila-data A standalone service whose purpose is to receive requests, process data operations with potentially long running time such as copying, share migration or backup.

manila-scheduler Schedules and routes requests to the appropriate share service. The scheduler uses configurable filters and weighers to route requests. The Filter Scheduler is the default and enables filters on things like Capacity, Availability Zone, Share Types, and Capabilities as well as custom filters.

manila-share Manages back-end devices that provide shared file systems. A manila-share service can run in one of two modes, with or without handling of share servers. Share servers export file shares via share networks. When share servers are not used, the networking requirements are handled outside of Manila.

The Shared File Systems service contains the following components:

Back-end storage devices The Shared File Services service requires some form of back-end shared file system provider that the service is built on. The reference implementation uses the Block Storage service (Cinder) and a service VM to provide shares. Additional drivers are used to access shared file systems from a variety of vendor solutions.

Users and tenants (projects) The Shared File Systems service can be used by many different cloud computing consumers or customers (tenants on a shared system), using role-based access assignments. Roles control the actions that a user is allowed to perform. In the default configuration, most actions do not require a particular role unless they are restricted to administrators, but this can be configured by the system administrator in the appropriate `policy.yaml` file that maintains the rules. A users access to manage particular shares is limited by tenant. Guest access to mount and use shares is secured by IP and/or user access rules. Quotas used to control resource consumption across available hardware resources are per tenant.

For tenants, quota controls are available to limit:

- The number of shares that can be created.
- The number of gigabytes that can be provisioned for shares.
- The number of share snapshots that can be created.
- The number of gigabytes that can be provisioned for share snapshots.
- The number of share networks that can be created.
- The number of share groups that can be created.
- The number of share group snapshots that can be created.
- The number of share replicas that can be created.
- The number of gigabytes that can be provisioned for share replicas.
- The number of gigabytes that can be provisioned for each share.

You can revise the default quota values with the Shared File Systems CLI, so the limits placed by quotas are editable by admin users.

Shares, snapshots, and share networks The basic resources offered by the Shared File Systems service are shares, snapshots and share networks:

Shares A share is a unit of storage with a protocol, a size, and an access list. Shares are the basic primitive provided by Manila. All shares exist on a backend. Some shares are associated with share networks and share servers. The main protocols supported are NFS and CIFS, but other protocols are supported as well.

Snapshots A snapshot is a point in time copy of a share. Snapshots can only be used to create new shares (containing the snapshotted data). Shares cannot be deleted until all associated snapshots are deleted.

Share networks A share network is a tenant-defined object that informs Manila about the security and network configuration for a group of shares. Share networks are only relevant for backends that manage share servers. A share network contains a security service and network/subnet.

Shared File Systems API configuration

Configuration options

The following options allow configuration of the APIs that Shared File Systems service supports.

Table 9: Description of API configuration options

Configuration option = Default value	Description
[DEFAULT]	
admin_network_config_group = None	(String) If share driver requires to setup admin network for share, then define network plugin config options in some separate config group and set its name here. Used only with another option driver_handles_share_servers set to True.
admin_network_id = None	(String) ID of neutron network used to communicate with admin network, to create additional admin export locations on.
admin_subnet_id = None	(String) ID of neutron subnet used to communicate with admin network, to create additional admin export locations on. Related to admin_network_id.
api_paste_config = api-paste.ini	(String) File name for the paste.deploy config for manila-api.
api_rate_limit = True	(Boolean) Whether to rate limit the API.
db_backend = sqlalchemy	(String) The backend to use for database.
max_header_line = 16384	(Integer) Maximum line size of message headers to be accepted. Option max_header_line may need to be increased when using large tokens (typically those generated by the Keystone v3 API with big service catalogs).
osapi_max_limit = 1000	(Integer) The maximum number of items returned in a single response from a collection resource.
osapi_share_base_URL = None	(String) Base URL to be presented to users in links to the Share API
osapi_share_ext_list =	(List) Specify list of extensions to load when using osapi_share_extension option with manila.api.contrib.select_extensions.
osapi_share_extension = manila.api.contrib.standard_extensions	(List) The osapi share extensions to load.
osapi_share_listen = ::	(String) IP address for OpenStack Share API to listen on.
osapi_share_listen_port = 8786	(Port number) Port for OpenStack Share API to listen on.
osapi_share_workers = 1	(Integer) Number of workers for OpenStack Share API service.
share_api_class = manila.share.api.API	(String) The full class name of the share API class to use.
volume_api_class = manila.volume.cinder.API	(String) The full class name of the Volume API class to use.
volume_name_template = manila-share-%s	(String) Volume name template.
volume_snapshot_name_template = manila-snapshot-%s	(String) Volume snapshot name template.
[oslo_middleware]	
enable_proxy_headers_parsing = False	(Boolean) Whether the application is behind a proxy or not. This determines if the middleware should parse the headers or not.
max_request_body_size = 114688	(Integer) The maximum body size for each request, in bytes.
secure_proxy_ssl_header =	(String) DEPRECATED: The HTTP Header that will be used to determine

Share drivers

Generic approach for share provisioning

The Shared File Systems service can be configured to use Compute VMs and Block Storage service volumes. There are two modules that handle them in the Shared File Systems service:

- The `service_instance` module creates VMs in Compute with a predefined image called `service image`. This module can be used by any driver for provisioning of service VMs to be able to separate share resources among tenants.
- The `generic` module operates with Block Storage service volumes and VMs created by the `service_instance` module, then creates shared filesystems based on volumes attached to VMs.

Network configurations

Each driver can handle networking in its own way, see: https://wiki.openstack.org/wiki/manila/Networking.

One of the two possible configurations can be chosen for share provisioning using the `service_instance` module:

- Service VM has one network interface from a network that is connected to a public router. For successful creation of a share, the user network should be connected to a public router, too.
- Service VM has two network interfaces, the first one is connected to the service network, the second one is connected directly to the users network.

Requirements for service image

- Linux based distro
- NFS server
- Samba server >= 3.2.0, that can be configured by data stored in registry
- SSH server
- Two network interfaces configured to DHCP (see network approaches)
- `exportfs` and `net conf` libraries used for share actions
- The following files will be used, so if their paths differ one needs to create at least symlinks for them:
  - `/etc/exports`: permanent file with NFS exports.
  - `/var/lib/nfs/etab`: temporary file with NFS exports used by `exportfs`.
  - `/etc/fstab`: permanent file with mounted filesystems.
  - `/etc/mtab`: temporary file with mounted filesystems used by `mount`.

Supported shared filesystems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS and CIFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.
- Extend a share.
- Shrink a share.

Known restrictions

- One of novas configurations only allows 26 shares per server. This limit comes from the maximum number of virtual PCI interfaces that are used for block device attaching. There are 28 virtual PCI interfaces, in this configuration, two of them are used for server needs and the other 26 are used for attaching block devices that are used for shares.

Using Windows instances

While the generic driver only supports Linux instances, you may use the Windows SMB driver when Windows instances are preferred.

For more details, please check out the following page: *Windows SMB driver*.

Driver options

The following table contains the configuration options specific to this driver.

Configuration option = Default value
[DEFAULT]
`connect_share_server_to_tenant_network = False`
`container_volume_group = manila_docker_volumes`
`driver_handles_share_servers = None`

Configuration option = Default value
goodness_function = None
interface_driver = manila.network.linux.interface.OVSInterfaceDriver
manila_service_keypair_name = manila-service
max_time_to_attach = 120
max_time_to_build_instance = 300
max_time_to_create_volume = 180
max_time_to_extend_volume = 180
ovs_integration_bridge = br-int
path_to_private_key = None
path_to_public_key = ~/.ssh/id_rsa.pub
protocol_access_mapping = {'ip': ['nfs'], 'user': ['cifs']}
service_image_name = manila-service-image
service_instance_flavor_id = 100
service_instance_name_or_id = None
service_instance_name_template = manila_service_instance_%s
service_instance_network_helper_type = neutron
service_instance_password = None
service_instance_security_group = manila-service
service_instance_smb_config_path = $share_mount_path/smb.conf
service_instance_user = None
service_net_name_or_ip = None
service_network_cidr = 10.254.0.0/16
service_network_division_mask = 28
service_network_name = manila_service_network
share_helpers = CIFS=manila.share.drivers.helpers.CIFSHelperIPAccess, NFS=manila.share.driv
share_mount_path = /shares
share_mount_template = mount -vt %(proto)s %(options)s %(export)s %(path)s
share_unmount_template = umount -v %(path)s
share_volume_fstype = ext4
tenant_net_name_or_ip = None
volume_name_template = manila-share-%s
volume_snapshot_name_template = manila-snapshot-%s
[glance]
api_microversion = 2
region_name = RegionOne
auth_url = None
auth_type = None
cafile = None
certfile = None
collect_timing = false
default_domain_id = None
default_domain_name = None
domain_id = None
domain_name = None
insecure = false
keyfile = None
password = None

Configuration option = Default value
project_domain_id = None
project_domain_name = None
project_id = None
project_name = None
split_loggers = false
system_scope = None
timeout = None
trust_id = None
user_domain_id = None
user_domain_name = None
user_id = None
username = None
[cinder]
cross_az_attach = True
http_retries = 3
endpoint_type = publicURL
region_name = RegionOne
auth_url = None
auth_type = None
cafile = None
certfile = None
collect_timing = false
default_domain_id = None
default_domain_name = None
domain_id = None
domain_name = None
insecure = false
keyfile = None
password = None
project_domain_id = None
project_domain_name = None
project_id = None
project_name = None
split_loggers = false
system_scope = None
timeout = None
trust_id = None
user_domain_id = None
user_domain_name = None
user_id = None
username = None
[neutron]
url = http://127.0.0.1:9696
url_timeout = 30
auth_strategy = keystone
endpoint_type = publicURL
region_name = None

Configuration option = Default value
auth_url = None
auth_type = None
cafile = None
certfile = None
collect_timing = false
default_domain_id = None
default_domain_name = None
domain_id = None
domain_name = None
insecure = false
keyfile = None
password = None
project_domain_id = None
project_domain_name = None
project_id = None
project_name = None
split_loggers = false
system_scope = None
timeout = None
trust_id = None
user_domain_id = None
user_domain_name = None
user_id = None
username = None
[nova]
api_microversion = 2.10
endpoint_type = publicURL
region_name = None
auth_url = None
auth_type = None
cafile = None
certfile = None
collect_timing = false
default_domain_id = None
default_domain_name = None
domain_id = None
domain_name = None
insecure = false
keyfile = None
password = None
project_domain_id = None
project_domain_name = None
project_id = None
project_name = None
split_loggers = false
system_scope = None
timeout = None

Configuration option = Default value
`trust_id` = `None`
`user_domain_id` = `None`
`user_domain_name` = `None`
`user_id` = `None`
`username` = `None`

CephFS Native driver

The CephFS Native driver enables the Shared File Systems service to export shared file systems to guests using the Ceph network protocol. Guests require a Ceph client in order to mount the file system.

Access is controlled via Cephs cephx authentication system. When a user requests share access for an ID, Ceph creates a corresponding Ceph auth ID and a secret key, if they do not already exist, and authorizes the ID to access the share. The client can then mount the share using the ID and the secret key.

To learn more about configuring Ceph clients to access the shares created using this driver, please see the Ceph documentation (http://docs.ceph.com/docs/master/cephfs/). If you choose to use the kernel client rather than the FUSE client, the share size limits set in the Shared File Systems service may not be obeyed.

Supported shared file systems and operations

The driver supports CephFS shares.

The following operations are supported with CephFS back end:

- Create a share.
- Delete a share.
- Allow share access.
  - `read-only` access level is supported.
  - `read-write` access level is supported.

  Note the following limitation for CephFS shares:
  - Only `cephx` access type is supported.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a consistency group (CG).
- Delete a CG.
- Create a CG snapshot.
- Delete a CG snapshot.

Requirements

- Mitaka or later versions of manila.
- Jewel or later versions of Ceph.
- A Ceph cluster with a file system configured (http://docs.ceph.com/docs/master/cephfs/createfs/)
- `ceph-common` package installed in the servers running the `manila-share` service.
- Ceph client installed in the guest, preferably the FUSE based client, `ceph-fuse`.
- Network connectivity between your Ceph clusters public network and the servers running the `manila-share` service.
- Network connectivity between your Ceph clusters public network and guests.

Important: A manila share backed onto CephFS is only as good as the underlying file system. Take care when configuring your Ceph cluster, and consult the latest guidance on the use of CephFS in the Ceph documentation (http://docs.ceph.com/docs/master/cephfs/).

Authorize the driver to communicate with Ceph

Run the following commands to create a Ceph identity for the Shared File Systems service to use:

```
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' \
mgr 'allow r' \
mon "$MON_CAPS"
```

`manila.keyring`, along with your `ceph.conf` file, then needs to be placed on the server running the `manila-share` service.

Enable snapshots in Ceph if you want to use them in the Shared File Systems service:

```
ceph mds set allow_new_snaps true --yes-i-really-mean-it
```

In the server running the `manila-share` service, you can place the `ceph.conf` and `manila.keyring` files in the `/etc/ceph` directory. Set the same owner for the `manila-share` process and the `manila.keyring` file. Add the following section to the `ceph.conf` file.

```
[client.manila]
client mount uid = 0
```

(continues on next page)

(continued from previous page)

```
client mount gid = 0
log file = /opt/stack/logs/ceph-client.manila.log
admin socket = /opt/stack/status/stack/ceph-$name.$pid.asok
keyring = /etc/ceph/manila.keyring
```

It is advisable to modify the Ceph clients admin socket file and log file locations so that they are co-located with the Shared File Systems services pid files and log files respectively.

Configure CephFS back end in `manila.conf`

1. Add CephFS to `enabled_share_protocols` (enforced at the Shared File Systems services API layer). In this example we leave NFS and CIFS enabled, although you can remove these if you only use CephFS:

```
enabled_share_protocols = NFS,CIFS,CEPHFS
```

2. Refer to the following table for the list of all the `cephfs_native` driver-specific configuration options.

Table 11: Description of CephFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`cephfs_auth_id = manila`	(String) The name of the ceph auth identity to use.
`cephfs_cluster_name = None`	(String) The name of the cluster in use, if it is not the default (ceph).
`cephfs_conf_path =`	(String) Fully qualified path to the ceph.conf file.

Create a section to define a CephFS back end:

```
[cephfs1]
driver_handles_share_servers = False
share_backend_name = CEPHFS1
share_driver = manila.share.drivers.cephfs.cephfs_native.
↪CephFSNativeDriver
cephfs_conf_path = /etc/ceph/ceph.conf
cephfs_auth_id = manila
cephfs_cluster_name = ceph
```

Also set the `driver-handles-share-servers` to `False` as the driver does not manage the lifecycle of `share-servers`.

3. Edit `enabled_share_backends` to point to the drivers back-end section using the section name. In this example we are also including another back end (`generic1`), you would include whatever other back ends you have configured.

```
enabled_share_backends = generic1,cephfs1
```

Creating shares

The default share type may have `driver_handles_share_servers` set to `True`. Configure a share type suitable for CephFS:

```
manila type-create cephfstype false

manila type-set cephfstype set share_backend_name='CEPHFS1'
```

Then create a share:

```
manila create --share-type cephfstype --name cephshare1 cephfs 1
```

Note the export location of the share:

```
manila share-export-location-list cephshare1
```

The export location of the share contains the Ceph monitor (mon) addresses and ports, and the path to be mounted. It is of the form, `{mon ip addr:port}[,{mon ip addr:port}]:{path to be mounted}`

Allowing access to shares

Allow Ceph auth ID `alice` access to the share using `cephx` access type.

```
manila access-allow cephshare1 cephx alice
```

Note the access status and the secret access key of `alice`.

```
manila access-list cephshare1
```

Mounting shares using FUSE client

Using the secret key of the authorized ID `alice`, create a keyring file `alice.keyring`.

```
[client.alice]
        key = AQA8+ANW/4ZWNRAAOtWJMFPEihBA1unFImJczA==
```

Using the monitor IP addresses from the shares export location, create a configuration file, `ceph.conf`:

```
[client]
        client quota = true
        mon host = 192.168.1.7:6789, 192.168.1.8:6789, 192.168.1.9:6789
```

Finally, mount the file system, substituting the file names of the keyring and configuration files you just created, and substituting the path to be mounted from the shares export location:

```
sudo ceph-fuse ~/mnt \
--id=alice \
--conf=./ceph.conf \
```

(continues on next page)

(continued from previous page)

```
--keyring=./alice.keyring \
--client-mountpoint=/volumes/_nogroup/4c55ad20-9c55-4a5e-9233-8ac64566b98c
```

Known restrictions

Consider the driver as a building block for supporting multi-tenant workloads in the future. However, it can be used in private cloud deployments.

- The guests have direct access to Cephs public network.
- Snapshots are read-only. A user can read a snapshots contents from the `.snap/{manila-snapshot-id}_{unknown-id}` folder within the mounted share.
- To restrict share sizes, CephFS uses quotas that are enforced in the client side. The CephFS clients are relied on to respect quotas.

Security

- Each shares data is mapped to a distinct Ceph RADOS namespace. A guest is restricted to access only that particular RADOS namespace.
- An additional level of resource isolation can be provided by mapping a shares contents to a separate RADOS pool. This layout would be preferred only for cloud deployments with a limited number of shares needing strong resource separation. You can do this by setting a share type specification, `cephfs:data_isolated` for the share type used by the cephfs driver.

  ```
  manila type-key cephfstype set cephfs:data_isolated=True
  ```

- Untrusted manila guests pose security risks to the Ceph storage cluster as they would have direct access to the clusters public network.

Dell EMC PowerMax Plugin

The Dell EMC Shared File Systems service driver framework (EMCShareDriver) utilizes the Dell EMC storage products to provide the shared file systems to OpenStack. The Dell EMC driver is a plug-in based driver which is designed to use different plug-ins to manage different Dell EMC storage products.

The PowerMax plug-in manages the PowerMax to provide shared file systems. The Dell EMC driver framework with the PowerMax plug-in is referred to as the PowerMax driver in this document.

This driver performs the operations on PowerMax eNAS by XMLAPI and the file command line. Each back end manages one Data Mover of PowerMax. Multiple Shared File Systems service back ends need to be configured to manage multiple Data Movers.

Requirements

- PowerMax eNAS OE for File version 8.1 or higher
- PowerMax Unified or File only
- The following licenses should be activated on PowerMax for File:
  - CIFS
  - NFS
  - SnapSure (for snapshot)
  - ReplicationV2 (for create share from snapshot)

Supported shared file systems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
  - Only user access type is supported for CIFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.

While the generic driver creates shared file systems based on cinder volumes attached to nova VMs, the PowerMax driver performs similar operations using the Data Movers on the array.

Pre-configurations on PowerMax

1. Configure a storage pool

   There is a one to one relationship between a storage pool in embedded NAS to a storage group on the PowerMax. The best way to provision storage for file is from the Unisphere for PowerMax UI rather than eNAS UI. Go to *{array} > SYSTEM > FIle* and under *Actions* click *PROVISION STORAGE FOR FILE*

   Note: When creating a new storage group you have the ability to assign a service level e.g. Diamond and disable compression/deduplication which is enabled by default.

To pick up the newly created storage pool in the eNAS UI, go to *{Control Station}* > *Storage* > *Storage Configuration* > *Storage Pools* and under *File Storage* click *Rescan Storage Systems*

or on the command line:

```
$ nas_diskmark -mark -all -discovery y -monitor y
```

The new storage pool should now appear in the eNAS UI

2. Make sure you have the appropriate licenses

```
$ nas_license -l
key                 status    value
site_key            online    xx xx xx xx
nfs                 online
cifs                online
snapsure            online
replicatorV2        online
filelevelretention  online
```

3. Enable CIFS service on Data Mover.

   Ensure the CIFS service is enabled on the Data Mover which is going to be managed by PowerMax driver.

   To start the CIFS service, use the following command:

```
$ server_setup <movername> -Protocol cifs -option start [=<n>]
  # movername = name of the Data Mover
  # n = number of threads for CIFS users
```

Note: If there is 1 GB of memory on the Data Mover, the default is 96 threads. However, if there is over 1 GB of memory, the default number of threads is 256.

   To check the CIFS service status, use the following command:

```
$ server_cifs <movername> | head
  # movername = name of the Data Mover
```

   The command output will show the number of CIFS threads started.

4. NTP settings on Data Mover.

   PowerMax driver only supports CIFS share creation with share network which has an Active Directory security-service associated.

   Creating CIFS share requires that the time on the Data Mover is in sync with the Active Directory domain so that the CIFS server can join the domain. Otherwise, the domain join will fail when creating a share with this security service. There is a limitation that the time of the domains used by security-services, even for different tenants and different share networks, should be in sync. Time difference should be less than 5 minutes.

Note: If there is a clock skew then you may see the following error The local machine and the remote machine are not synchronized. Kerberos protocol requires a synchronization of both

participants within the same 5 minutes. To fix this error you must make sure the times of the eNas controller host and the Domain Controller or within 5 minutes of each other. You must be root to change the date of the eNas control station. Check also that your time zones coincide.

We recommend setting the NTP server to the same public NTP server on both the Data Mover and domains used in security services to ensure the time is in sync everywhere.

Check the date and time on Data Mover with the following command:

```
$ server_date <movername>
  # movername = name of the Data Mover
```

Set the NTP server for Data Mover with the following command:

```
$ server_date <movername> timesvc start ntp <host> [<host> ...]
  # movername = name of the Data Mover
  # host = IP address of the time server host
```

Note: The host must be running the NTP protocol. Only 4 host entries are allowed.

---

5. Configure User Mapping on the Data Mover.

   Before creating CIFS share using PowerMax driver, you must select a method of mapping Windows SIDs to UIDs and GIDs. DELL EMC recommends using usermapper in single protocol (CIFS) environment which is enabled on PowerMax eNAS by default.

   To check usermapper status, use the following command syntax:

   ```
   $ server_usermapper <movername>
     # movername = name of the Data Mover
   ```

   If usermapper does not start, use the following command to start the usermapper:

   ```
   $ server_usermapper <movername> -enable
     # movername = name of the Data Mover
   ```

   For a multiple protocol environment, refer to Configuring PowerMax eNAS User Mapping on EMC support site for additional information.

6. Configure network connection.

   Find the network devices (physical port on NIC) of the Data Mover that has access to the share network.

   To check the device list on the eNAS UI go to *{Control Station}* > *Settings* > *Network* > *Devices*.

   or on the command line:

   ```
   $ server_sysconfig server_2 -pci
   server_2 : PCI DEVICES:

   On Board:
     VendorID=0x1120 DeviceID=0x1B00  Controller
   ```

(continues on next page)

(continued from previous page)

```
    0:  scsi-0  IRQ: 32

    0:  scsi-16  IRQ: 33

    0:  scsi-32  IRQ: 34

    0:  scsi-48  IRQ: 35

  Broadcom 10 Gigabit Ethernet Controller
    0:  fxg-3-0  IRQ: 36
    speed=10000 duplex=full txflowctl=disable rxflowctl=disable
    Link: Up

     0:  fxg-3-1  IRQ: 38
    speed=10000 duplex=full txflowctl=disable rxflowctl=disable
    Link: Down
```

Back-end configurations

Note: The following deprecated tags will be removed in the T release:

- emc_nas_server_container
- emc_nas_pool_names
- emc_interface_ports

The following parameters need to be configured in the `/etc/manila/manila.conf` file for the PowerMax driver:

```
emc_share_backend = powermax
emc_nas_server = <IP address>
emc_nas_password = <password>
emc_nas_login = <user>
driver_handles_share_servers = True
powermax_server_container = <Data Mover name>
powermax_share_data_pools = <Comma separated pool names>
share_driver = manila.share.drivers.dell_emc.driver.EMCShareDriver
powermax_ethernet_ports = <Comma separated ports list>
emc_ssl_cert_verify = True
emc_ssl_cert_path = <path to cert>
share_backend_name = <Backend>
```

- ***emc_share_backend*** The plug-in name. Set it to `powermax` for the PowerMax driver. Other values are `isilon`, `vnx` and `unity`.
- ***emc_nas_server*** The control station IP address of the PowerMax system to be managed.
- ***emc_nas_password*** **and** ***emc_nas_login*** The fields that are used to provide credentials to the PowerMax system. Only local users of PowerMax File is supported.

- ***driver_handles_share_servers*** PowerMax only supports True, where the share driver handles the provisioning and management of the share servers.
- ***powermax_server_container*** Name of the Data Mover to serve the share service.
- ***powermax_share_data_pools*** Comma separated list specifying the name of the pools to be used by this back end. Do not set this option if all storage pools on the system can be used. Wild card character is supported.

  Examples: pool_1, pool_*, *
- ***powermax_ethernet_ports (optional)*** Comma-separated list specifying the ports (devices) of Data Mover that can be used for share server interface. Do not set this option if all ports on the Data Mover can be used. Wild card character is supported.

  Examples: fxg-9-0, fxg-_*, *
- ***emc_ssl_cert_verify (optional)*** By default this is True, setting it to False is not recommended
- ***emc_ssl_cert_path (optional)*** The path to the This must be set if emc_ssl_cert_verify is True which is the recommended configuration. See `SSL Support` section for more details.
- ***share_backend_name*** The backend name for a given driver implementation.

Restart of the `manila-share` service is needed for the configuration changes to take effect.

SSL Support

1. Run the following on eNas Control Station, to display the CA certification for the active CS.

```
$ /nas/sbin/nas_ca_certificate -display
```

> **Warning:** This cert will be different for the secondary CS so if there is a failover a different certificate must be used.

2. Copy the contents and create a file with a .pem extention on your manila host.

```
-----BEGIN CERTIFICATE-----
the cert contents are here
-----END CERTIFICATE-----
```

3. To verify the cert by running the following and examining the output:

```
$ openssl x509 -in test.pem -text -noout
```

```
Certificate:
 Data:
     Version: 3 (0x2)
     Serial Number: xxxxxx
 Signature Algorithm: sha1WithRSAEncryption
     Issuer: O=VNX Certificate Authority, CN=xxx
     Validity
         Not Before: Feb 27 16:02:41 2019 GMT
```

(continues on next page)

(continued from previous page)

```
        Not After : Mar  4 16:02:41 2024 GMT
    Subject: O=VNX Certificate Authority, CN=xxxxxx
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                xxxxxx
            Exponent: xxxxxx
    X509v3 extensions:
        X509v3 Subject Key Identifier:
            xxxxxx
        X509v3 Authority Key Identifier:
            keyid:xxxxx
            DirName:/O=VNX Certificate Authority/CN=xxxxxx
            serial:xxxxx

        X509v3 Basic Constraints:
            CA:TRUE
        X509v3 Subject Alternative Name:
            DNS:xxxxxx, DNS:xxxxxx.localdomain, DNS:xxxxxxx, DNS:xxxxx
Signature Algorithm: sha1WithRSAEncryption
        xxxxxx
```

4. As it is the capath and not the cafile that is expected, copy the file to either new directory or an existing directory (where other .pem files exist).

5. Run the following on the directory

```
$ c_rehash $PATH_TO_CERTS
```

6. Update manila.conf with the directory where the .pem exists.

```
emc_ssl_cert_path = /path_to_certs/
```

7. Restart manila services.

Snapshot Support

Snapshot support is disabled by default, so in order to allow shapshots for a share type, the `snapshot_support` extra spec must be set to True. Creating a share from a snapshot is also disabled by default so `create_share_from_snapshot_support` must also be set to True if this functionality is required.

For a new share type:

```
$ manila type-create --snapshot_support True \
                     --create_share_from_snapshot_support True \
                     ${share_type_name} True
```

For an existing share type:

```
$ manila type-key ${share_type_name} \
                  set snapshot_support=True
$ manila type-key ${share_type_name} \
                  set create_share_from_snapshot_support=True
```

To create a snapshot from a share where snapshot_support=True:

```
$ manila snapshot-create ${source_share_name} --name ${target_snapshot_name}
```

To create a target share from a shapshot where create_share_from_snapshot_support=True:

```
$ manila create cifs 3 --name ${target_share_name} \
                       --share-network ${share_network} \
                       --share-type ${share_type_name} \
                       --metadata source=snapshot \
                       --snapshot-id ${snapshot_id}
```

IPv6 support

IPv6 support for PowerMax Manila driver was introduced in Rocky release. The feature is divided into two parts:

1. The driver is able to manage share or snapshot in the Neutron IPv6 network.
2. The driver is able to connect PowerMax management interface using its IPv6 address.

Pre-Configurations for IPv6 support

The following parameters need to be configured in `/etc/manila/manila.conf` for the PowerMax driver:

```
network_plugin_ipv6_enabled = True
```

If you want to connect to the eNAS controller using IPv6 address specify the address in `/etc/manila/manila.conf`:

```
emc_nas_server = <IPv6 address>
```

Restrictions

The PowerMax driver has the following restrictions:

- Only `driver_handles_share_servers` equals True is supported.
- Only IP access type is supported for NFS.
- Only user access type is supported for CIFS.
- Only FLAT network and VLAN network are supported.

- VLAN network is supported with limitations. The neutron subnets in different VLANs that are used to create share networks cannot have overlapped address spaces. Otherwise, PowerMax may have a problem to communicate with the hosts in the VLANs. To create shares for different VLANs with same subnet address, use different Data Movers.
- The **Active Directory** security service is the only supported security service type and it is required to create CIFS shares.
- Only one security service can be configured for each share network.
- The domain name of the `active_directory` security service should be unique even for different tenants.
- The time on the Data Mover and the Active Directory domains used in security services should be in sync (time difference should be less than 10 minutes). We recommended using same NTP server on both the Data Mover and Active Directory domains.
- On eNAS, the snapshot is stored in the SavVols. eNAS system allows the space used by SavVol to be created and extended until the sum of the space consumed by all SavVols on the system exceeds the default 20% of the total space available on the system. If the 20% threshold value is reached, an alert will be generated on eNAS. Continuing to create snapshot will cause the old snapshot to be inactivated (and the snapshot data to be abandoned). The limit percentage value can be changed manually by storage administrator based on the storage needs. We recommend the administrator configures the notification on the SavVol usage. Refer to Using eNAS SnapSure document on EMC support site for more information.
- eNAS has limitations on the overall numbers of Virtual Data Movers, filesystems, shares, and checkpoints. Virtual Data Mover(VDM) is created by the eNAS driver on the eNAS to serve as the Shared File Systems service share server. Similarly, the filesystem is created, mounted, and exported from the VDM over CIFS or NFS protocol to serve as the Shared File Systems service share. The eNAS checkpoint serves as the Shared File Systems service share snapshot. Refer to the NAS Support Matrix document on EMC support site for the limitations and configure the quotas accordingly.

Other Remarks

- eNAS `nas_quotas` should not be confused with OpenStack manila quotas. The former edits quotas for mounted file systems, and displays a listing of quotas and disk usage at the file system level (by the user, group, or tree), or at the quota-tree level (by the user or group). `nas_quotas` also turns quotas on and off, and clears quotas records for a file system, quota tree, or a Data Mover. Refer to PowerMax eNAS CLI Reference guide on EMC support site for additional information. `OpenStack manila quotas` delimit the number of shares, snapshots etc. a user can create.

```
$ manila quota-show --tenant <project_id> --user <user_id>
+-----------------------+-------+
| Property              | Value |
+-----------------------+-------+
| share_groups          | 50    |
| gigabytes             | 1000  |
| snapshot_gigabytes    | 1000  |
| share_group_snapshots | 50    |
| snapshots             | 50    |
| shares                | 50    |
```

(continues on next page)

(continued from previous page)

```
| share_networks          | 10     |
+-------------------------+--------+
```

Driver options

Configuration options specific to this driver:

Table 12: Description of Dell EMC PowerMax share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`powermax_ethernet_ports` = `None`	(List) Comma separated list of ports that can be used for share server interfaces. Members of the list can be Unix-style glob expressions.
`powermax_server_container` = `None`	(String) Data mover to host the NAS server.
`powermax_share_data_pools` = `None`	(List) Comma separated list of pools that can be used to persist share data.

Dell EMC VNX driver

The EMC Shared File Systems service driver framework (EMCShareDriver) utilizes the EMC storage products to provide the shared file systems to OpenStack. The EMC driver is a plug-in based driver which is designed to use different plug-ins to manage different EMC storage products.

The VNX plug-in is the plug-in which manages the VNX to provide shared filesystems. The EMC driver framework with the VNX plug-in is referred to as the VNX driver in this document.

This driver performs the operations on VNX by XMLAPI and the file command line. Each back end manages one Data Mover of VNX. Multiple Shared File Systems service back ends need to be configured to manage multiple Data Movers.

Requirements

- VNX OE for File version 7.1 or higher
- VNX Unified, File only, or Gateway system with a single storage back end
- The following licenses should be activated on VNX for File:
  - CIFS
  - NFS
  - SnapSure (for snapshot)
  - ReplicationV2 (for create share from snapshot)

Supported shared filesystems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
  - Only user access type is supported for CIFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.

While the generic driver creates shared filesystems based on cinder volumes attached to nova VMs, the VNX driver performs similar operations using the Data Movers on the array.

Pre-configurations on VNX

1. Enable unicode on Data Mover.

   The VNX driver requires that the unicode is enabled on Data Mover.

   > **Warning:** After enabling Unicode, you cannot disable it. If there are some filesystems created before Unicode is enabled on the VNX, consult the storage administrator before enabling Unicode.

   To check the Unicode status on Data Mover, use the following VNX File command on the VNX control station:

   ```
   server_cifs <mover_name> | head
   # mover_name = <name of the Data Mover>
   ```

   Check the value of I18N mode field. UNICODE mode is shown as `I18N mode = UNICODE`.

   To enable the Unicode for Data Mover:

   ```
   uc_config -on -mover <mover_name>
   # mover_name = <name of the Data Mover>
   ```

   Refer to the document Using International Character Sets on VNX for File on EMC support site for more information.

2. Enable CIFS service on Data Mover.

   Ensure the CIFS service is enabled on the Data Mover which is going to be managed by VNX driver.

   To start the CIFS service, use the following command:

```
server_setup <mover_name> -Protocol cifs -option start [=<n>]
# mover_name = <name of the Data Mover>
# n = <number of threads for CIFS users>
```

   Note: If there is 1 GB of memory on the Data Mover, the default is 96 threads; however, if there is over 1 GB of memory, the default number of threads is 256.

   To check the CIFS service status, use this command:

```
server_cifs <mover_name> | head
# mover_name = <name of the Data Mover>
```

   The command output will show the number of CIFS threads started.

3. NTP settings on Data Mover.

   VNX driver only supports CIFS share creation with share network which has an Active Directory security-service associated.

   Creating CIFS share requires that the time on the Data Mover is in sync with the Active Directory domain so that the CIFS server can join the domain. Otherwise, the domain join will fail when creating share with this security service. There is a limitation that the time of the domains used by security-services even for different tenants and different share networks should be in sync. Time difference should be less than 10 minutes.

   It is recommended to set the NTP server to the same public NTP server on both the Data Mover and domains used in security services to ensure the time is in sync everywhere.

   Check the date and time on Data Mover:

```
server_date <mover_name>
# mover_name = <name of the Data Mover>
```

   Set the NTP server for Data Mover:

```
server_date <mover_name> timesvc start ntp <host> [<host> ...]
# mover_name = <name of the Data Mover>
# host = <IP address of the time server host>
```

   Note: The host must be running the NTP protocol. Only 4 host entries are allowed.

4. Configure User Mapping on the Data Mover.

   Before creating CIFS share using VNX driver, you must select a method of mapping Windows SIDs to UIDs and GIDs. EMC recommends using usermapper in single protocol (CIFS) environment which is enabled on VNX by default.

To check usermapper status, use this command syntax:

```
server_usermapper <movername>
# movername = <name of the Data Mover>
```

If usermapper is not started, the following command can be used to start the usermapper:

```
server_usermapper <movername> -enable
# movername = <name of the Data Mover>
```

For a multiple protocol environment, refer to Configuring VNX User Mapping on EMC support site for additional information.

5. Network Connection.

   Find the network devices (physical port on NIC) of Data Mover that has access to the share network.

   Go to *Unisphere* to check the device list: *Settings* > *Network* > *Settings for File (Unified system only)* > *Device*.

Back-end configurations

The following parameters need to be configured in the `/etc/manila/manila.conf` file for the VNX driver:

```
emc_share_backend = vnx
emc_nas_server = <IP address>
emc_nas_password = <password>
emc_nas_login = <user>
vnx_server_container = <Data Mover name>
vnx_share_data_pools = <Comma separated pool names>
share_driver = manila.share.drivers.emc.driver.EMCShareDriver
vnx_ethernet_ports = <Comma separated ports list>
```

- ***emc_share_backend*** The plug-in name. Set it to `vnx` for the VNX driver.
- ***emc_nas_server*** The control station IP address of the VNX system to be managed.
- ***emc_nas_password*** **and** ***emc_nas_login*** The fields that are used to provide credentials to the VNX system. Only local users of VNX File is supported.
- ***vnx_server_container*** Name of the Data Mover to serve the share service.
- ***vnx_share_data_pools*** Comma separated list specifying the name of the pools to be used by this back end. Do not set this option if all storage pools on the system can be used. Wild card character is supported.

  Examples: pool_1, pool_*, *
- ***vnx_ethernet_ports*** Comma separated list specifying the ports (devices) of Data Mover that can be used for share server interface. Do not set this option if all ports on the Data Mover can be used. Wild card character is supported.

  Examples: spa_eth1, spa_*, *

Restart of the `manila-share` service is needed for the configuration changes to take effect.

Restrictions

The VNX driver has the following restrictions:

- Only IP access type is supported for NFS.
- Only user access type is supported for CIFS.
- Only FLAT network and VLAN network are supported.
- VLAN network is supported with limitations. The neutron subnets in different VLANs that are used to create share networks cannot have overlapped address spaces. Otherwise, VNX may have a problem to communicate with the hosts in the VLANs. To create shares for different VLANs with same subnet address, use different Data Movers.
- The `Active Directory` security service is the only supported security service type and it is required to create CIFS shares.
- Only one security service can be configured for each share network.
- Active Directory domain name of the active_directory security service should be unique even for different tenants.
- The time on Data Mover and the Active Directory domains used in security services should be in sync (time difference should be less than 10 minutes). It is recommended to use same NTP server on both the Data Mover and Active Directory domains.
- On VNX the snapshot is stored in the SavVols. VNX system allows the space used by SavVol to be created and extended until the sum of the space consumed by all SavVols on the system exceeds the default 20% of the total space available on the system. If the 20% threshold value is reached, an alert will be generated on VNX. Continuing to create snapshot will cause the old snapshot to be inactivated (and the snapshot data to be abandoned). The limit percentage value can be changed manually by storage administrator based on the storage needs. Administrator is recommended to configure the notification on the SavVol usage. Refer to Using VNX SnapSure document on EMC support site for more information.
- VNX has limitations on the overall numbers of Virtual Data Movers, filesystems, shares, checkpoints, etc. Virtual Data Mover(VDM) is created by the VNX driver on the VNX to serve as the Shared File Systems service share server. Similarly, filesystem is created, mounted, and exported from the VDM over CIFS or NFS protocol to serve as the Shared File Systems service share. The VNX checkpoint serves as the Shared File Systems service share snapshot. Refer to the NAS Support Matrix document on EMC support site for the limitations and configure the quotas accordingly.

Driver options

Configuration options specific to this driver:

Table 13: Description of Dell EMC VNX share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
vnx_ethernet_ports = None	(List) Comma separated list of ports that can be used for share server interfaces. Members of the list can be Unix-style glob expressions.
vnx_server_container = None	(String) Data mover to host the NAS server.
vnx_share_data_pools = None	(List) Comma separated list of pools that can be used to persist share data.

GlusterFS driver

GlusterFS driver uses GlusterFS, an open source distributed file system, as the storage back end for serving file shares to the Shared File Systems clients.

Supported shared filesystems and operations

The driver supports NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported
  - Only read-write access is supported.
- Deny share access.

Requirements

- Install glusterfs-server package, version >= 3.5.x, on the storage back end.
- Install NFS-Ganesha, version >=2.1, if using NFS-Ganesha as the NFS server for the GlusterFS back end.
- Install glusterfs and glusterfs-fuse package, version >=3.5.x, on the Shared File Systems service host.
- Establish network connection between the Shared File Systems service host and the storage back end.

Shared File Systems service driver configuration setting

The following parameters in the Shared File Systems services configuration file `manila.conf` need to be set:

```
share_driver = manila.share.drivers.glusterfs.GlusterfsShareDriver
```

If the back-end GlusterFS server runs on the Shared File Systems service host machine:

```
glusterfs_target = <glustervolserver>:/<glustervolid>
```

If the back-end GlusterFS server runs remotely:

```
glusterfs_target = <username>@<glustervolserver>:/<glustervolid>
```

Known restrictions

- The driver does not support network segmented multi-tenancy model, but instead works over a flat network, where the tenants share a network.
- If NFS Ganesha is the NFS server used by the GlusterFS back end, then the shares can be accessed by NFSv3 and v4 protocols. However, if Gluster NFS is used by the GlusterFS back end, then the shares can only be accessed by NFSv3 protocol.
- All Shared File Systems service shares, which map to subdirectories within a GlusterFS volume, are currently created within a single GlusterFS volume of a GlusterFS storage pool.
- The driver does not provide read-only access level for shares.

Driver options

The following table contains the configuration options specific to the share driver.

Table 14: Description of GlusterFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
glusterfs_ganesha_server_ip = None	(String) Ganesha server IP address.
glusterfs_ganesha_server_password = None	(String) Ganesha server password. This is not required if glusterfs_path_to_private_key is configured.
glusterfs_ganesha_server_username = root	(String) Ganesha server username.
glusterfs_mount_point_base = $state_path/mnt	(String) Base directory containing mount points for Gluster volumes.
glusterfs_nfs_server_type = Gluster	(String) Type of NFS server that mediate access to the Gluster volumes (Gluster or Ganesha).
glusterfs_path_to_private_key = None	(String) Path of Manila hosts private SSH key file.
glusterfs_server_password = None	(String) GlusterFS server nodes login password. This is not required if glusterfs_path_to_private_key is configured.
glusterfs_servers =	(List) List of GlusterFS servers that can be used to create shares. Each GlusterFS server should be of the form [remoteuser@]<volserver>, and they are assumed to belong to distinct Gluster clusters.
glusterfs_share_layout = None	(String) Specifies GlusterFS share layout, that is, the method of associating backing GlusterFS resources to shares.
glusterfs_target = None	(String) Specifies the GlusterFS volume to be mounted on the Manila host. It is of the form [remoteuser@]<volserver>:<volid>.
glusterfs_volume_pattern = None	(String) Regular expression template used to filter GlusterFS volumes for share creation. The regex template can optionally (ie. with support of the GlusterFS backend) contain the #{size} parameter which matches an integer (sequence of digits) in which case the value shall be interpreted as size of the volume in GB. Examples: manila-share-volume-d+$, manila-share-volume-#{size}G-d+$; with matching volume names, respectively: manila-share-volume-12, manila-share-volume-3G-13. In latter example, the number that matches #{size}, that is, 3, is an indication that the size of volume is 3G.

GlusterFS Native driver

GlusterFS Native driver uses GlusterFS, an open source distributed file system, as the storage back end for serving file shares to Shared File Systems service clients.

A Shared File Systems service share is a GlusterFS volume. This driver uses flat-network (share-server-less) model. Instances directly talk with the GlusterFS back end storage pool. The instances use `glusterfs` protocol to mount the GlusterFS shares. Access to each share is allowed via TLS Certificates. Only the instance which has the TLS trust established with the GlusterFS back end can mount and hence use the share. Currently only `read-write (rw)` access is supported.

Network approach

L3 connectivity between the storage back end and the host running the Shared File Systems share service should exist.

Multi-tenancy model

The driver does not support network segmented multi-tenancy model. Instead multi-tenancy is supported using tenant specific TLS certificates.

Supported shared filesystems and operations

The driver supports GlusterFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only access by TLS Certificates (`cert` access type) is supported.
  - Only read-write access is supported.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.

Requirements

- Install glusterfs-server package, version >= 3.6.x, on the storage back end.
- Install glusterfs and glusterfs-fuse package, version >= 3.6.x, on the Shared File Systems service host.
- Establish network connection between the Shared File Systems service host and the storage back end.

Shared File Systems service driver configuration setting

The following parameters in the Shared File Systems services configuration file need to be set:

```
share_driver = manila.share.drivers.glusterfs_native.
↪GlusterfsNativeShareDriver
glusterfs_servers = glustervolserver
glusterfs_volume_pattern = manila-share-volume-\d+$
```

The parameters are:

glusterfs_servers List of GlusterFS servers which provide volumes that can be used to create shares. The servers are expected to be of distinct Gluster clusters, so they should not be Gluster peers. Each server should be of the form `[<remoteuser>@]<glustervolserver>`.

The optional `<remoteuser>@` part of the server URI indicates SSH access for cluster management (see related optional parameters below). If it is not given, direct command line management is performed (the Shared File Systems service host is assumed to be part of the GlusterFS cluster the server belongs to).

glusterfs_volume_pattern Regular expression template used to filter GlusterFS volumes for share creation. The regular expression template can contain the `#{size}` parameter which matches a number and the value will be interpreted as size of the volume in GB. Examples: `manila-share-volume-\d+$`, `manila-share-volume-#{size}G-\d+$`; with matching volume names, respectively: `manila-share-volume-12`, `manila-share-volume-3G-13`. In the latter example, the number that matches `#{size}`, which is 3, is an indication that the size of volume is 3 GB. On share creation, the Shared File Systems service picks volumes at least as large as the requested one.

When setting up GlusterFS shares, note the following:

- GlusterFS volumes are not created on demand. A pre-existing set of GlusterFS volumes should be supplied by the GlusterFS cluster(s), conforming to the naming convention encoded by `glusterfs_volume_pattern`. However, the GlusterFS endpoint is allowed to extend this set any time, so the Shared File Systems service and GlusterFS endpoints are expected to communicate volume supply and demand out-of-band.
- Certificate setup, also known as trust setup, between instance and storage back end is out of band of the Shared File Systems service.
- For the Shared File Systems service to use GlusterFS volumes, the name of the trashcan directory in GlusterFS volumes must not be changed from the default.

HDFS native driver

The HDFS native driver is a plug-in for the Shared File Systems service. It uses Hadoop distributed file system (HDFS), a distributed file system designed to hold very large amounts of data, and provide high-throughput access to the data.

A Shared File Systems service share in this driver is a subdirectory in the hdfs root directory. Instances talk directly to the HDFS storage back end using the `hdfs` protocol. Access to each share is allowed by user based access type, which is aligned with HDFS ACLs to support access control of multiple users and groups.

Network configuration

The storage back end and Shared File Systems service hosts should be in a flat network, otherwise L3 connectivity between them should exist.

Supported shared filesystems and operations

The driver supports HDFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only user access type is supported.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.

Requirements

- Install HDFS package, version >= 2.4.x, on the storage back end.
- To enable access control, the HDFS file system must have ACLs enabled.
- Establish network connection between the Shared File Systems service host and storage back end.

Shared File Systems service driver configuration

To enable the driver, set the `share_driver` option in file `manila.conf` and add other options as appropriate.

```
share_driver = manila.share.drivers.hdfs.hdfs_native.HDFSNativeShareDriver
```

Known restrictions

- This driver does not support network segmented multi-tenancy model. Instead multi-tenancy is supported by the tenant specific user authentication.
- Only support for single HDFS namenode in Kilo release.

Driver options

The following table contains the configuration options specific to the share driver.

Table 15: Description of HDFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`hdfs_namenode_ip = None`	(String) The IP of the HDFS namenode.
`hdfs_namenode_port = 9000`	(Port number) The port of HDFS namenode service.
`hdfs_ssh_name = None`	(String) HDFS namenode ssh login name.
`hdfs_ssh_port = 22`	(Port number) HDFS namenode SSH port.
`hdfs_ssh_private_key = None`	(String) Path to HDFS namenode SSH private key for login.
`hdfs_ssh_pw = None`	(String) HDFS namenode SSH login password, This parameter is not necessary, if hdfs_ssh_private_key is configured.

LVM share driver

The Shared File Systems service can be configured to use LVM share driver. LVM share driver relies solely on LVM running on the same host with manila-share service. It does not require any services not related to the Shared File Systems service to be present to work.

Prerequisites

The following packages must be installed on the same host with manila-share service:

- NFS server
- Samba server >= 3.2.0
- LVM2 >= 2.02.66

Services must be up and running, ports used by the services must not be blocked. A node with manila-share service should be accessible to share service users.

LVM should be preconfigured. By default, LVM driver expects to find a volume group named `lvm-shares`. This volume group will be used by the driver for share provisioning. It should be managed by node administrator separately.

Shared File Systems service driver configuration setting

To use the driver, one should set up a corresponding back end. A driver must be explicitly specified as well as export IP address. A minimal back-end specification that will enable LVM share driver is presented below:

```
[LVM_sample_backend]
driver_handles_share_servers = False
share_driver = manila.share.drivers.lvm.LVMShareDriver
lvm_share_export_ips = 1.2.3.4
```

In the example above, `lvm_share_export_ips` is the address to be used by clients for accessing shares. In the simplest case, it should be the same as hosts address. The option allows configuring more than one IP address as a comma separated string.

Supported shared file systems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.
- Extend a share.

Known restrictions

- LVM driver should not be used on a host running Neutron agents, simultaneous usage might cause issues with share deletion (shares will not get deleted from volume groups).

Driver options

The following table contains the configuration options specific to this driver.

Table 16: Description of LVM share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`lvm_share_export_ips = None`	(String) List of IPs to export shares belonging to the LVM storage driver.
`lvm_share_export_root = $state_path/mnt`	(String) Base folder where exported shares are located.
`lvm_share_helpers = CIFS=manila.share.drivers.helpers.CIFSHelperUserAccess, NFS=manila.share.drivers.helpers.NFSHelper`	(List) Specify list of share export helpers.
`lvm_share_mirrors = 0`	(Integer) If set, create LVMs with multiple mirrors. Note that this requires lvm_mirrors + 2 PVs with available space.
`lvm_share_volume_group = lvm-shares`	(String) Name for the VG that will contain exported shares.

ZFS (on Linux) driver

Manila ZFSonLinux share driver uses ZFS file system for exporting NFS shares. Written and tested using Linux version of ZFS.

Requirements

- NFS daemon that can be handled through `exportfs` app.
- ZFS file system packages, either Kernel or FUSE versions.
- ZFS zpools that are going to be used by Manila should exist and be configured as desired. Manila will not change zpool configuration.
- For remote ZFS hosts according to manila-share service host SSH should be installed.
- For ZFS hosts that support replication:
  - SSH access for each other should be passwordless.
  - Service IP addresses should be available by ZFS hosts for each other.

Supported shared filesystems and operations

The driver supports NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.
  - Only IP access type is supported.
  - Both access levels are supported - `RW` and `RO`.
- Deny share access.
- Bring an existing ZFSOnLinux share under the shared file system service (Managing a share)
- Remove a ZFSOnLinux share from the shared file system service without deleting it (Unmanaging a share)
- Create a snapshot.
- Delete a snapshot.
- Bring an existing ZFSOnLinux snapshot under the shared file system service (Managing a snapshot)
- Remove a ZFSOnLinux snapshot from the shared file system service without deleting it (Unmanaging a snapshot)
- Create a share from snapshot.
- Extend a share.
- Shrink a share.
- Share replication (experimental):
  - Create, update, delete, and promote replica operations are supported.

Possibilities

- Any amount of ZFS zpools can be used by share driver.
- Allowed to configure default options for ZFS datasets that are used for share creation.
- Any amount of nested datasets is allowed to be used.
- All share replicas are read-only, only active one is read-write.
- All share replicas are synchronized periodically, not continuously. Status `in_sync` means latest sync was successful. Time range between syncs equals to the value of the `replica_state_update_interval` configuration global option.
- Driver can use qualified extra spec `zfsonlinux:compression`. It can contain any value that ZFS app supports. But if it is disabled through the configuration option with the value `compression=off`, then it will not be used.

Restrictions

The ZFSonLinux share driver has the following restrictions:

- Only IP access type is supported for NFS.
- Only FLAT network is supported.
- `Promote share replica` operation will switch roles of current `secondary` replica and `active`. It does not make more than one active replica available.
- The below items are not yet implemented:
  - `SaMBa` based sharing.
  - `Thick provisioning` capability.

Known problems

- `Promote share replica` operation will make ZFS file system that became secondary as RO only on NFS level. On ZFS level system will stay mounted as was - RW.

Back-end configuration

The following parameters need to be configured in the manila configuration file for back-ends that use the ZFSonLinux driver:

- `share_driver` = manila.share.drivers.zfsonlinux.driver.ZFSonLinuxShareDriver
- `driver_handles_share_servers` = False
- `replication_domain` = custom_str_value_as_domain_name
  - If empty, then replication will be disabled.
  - If set, then will be able to be used as replication peer for other back ends with the same value.
- `zfs_share_export_ip` = <user_facing IP address of ZFS host>
- `zfs_service_ip` = <IP address of service network interface of ZFS host>
- `zfs_zpool_list` = zpoolname1,zpoolname2/nested_dataset_for_zpool2
  - Can be one or more zpools.
  - Can contain nested datasets.
- `zfs_dataset_creation_options` = <list of ZFS dataset options>
  - readonly, quota, sharenfs and sharesmb options will be ignored.
- `zfs_dataset_name_prefix` = <prefix>
  - Prefix to be used in each dataset name.
- `zfs_dataset_snapshot_name_prefix` = <prefix>
  - Prefix to be used in each dataset snapshot name.
- `zfs_use_ssh` = <boolean_value>

  - Set `False` if ZFS located on the same host as *manila-share* service.
  - Set `True` if *manila-share* service should use SSH for ZFS configuration.
- `zfs_ssh_username` = <ssh_username>
  - Required for replication operations.
  - Required for SSH“ing to ZFS host if `zfs_use_ssh` is set to `True`.
- `zfs_ssh_user_password` = <ssh_user_password>
  - Password for `zfs_ssh_username` of ZFS host.
  - Used only if `zfs_use_ssh` is set to `True`.
- `zfs_ssh_private_key_path` = <path_to_private_ssh_key>
  - Used only if `zfs_use_ssh` is set to `True`.
- `zfs_share_helpers` = NFS=manila.share.drivers.zfsonlinux.utils.NFSviaZFSHelper
  - Approach for setting up helpers is similar to various other share drivers.
  - At least one helper should be used.
- `zfs_replica_snapshot_prefix` = <prefix>
  - Prefix to be used in dataset snapshot names that are created by `update replica` operation.

Driver options

Table 17: Description of ZFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
zfs_dataset_creation_options = None	(List) Define here list of options that should be applied for each dataset creation if needed. Example: compression=gzip,dedup=off. Note that, for secondary replicas option readonly will be set to on and for active replicas to off in any way. Also, quota will be equal to share size. Optional.
zfs_dataset_name_prefix = manila_share_	(String) Prefix to be used in each dataset name. Optional.
zfs_dataset_snapshot_name_prefix = manila_share_snapshot_	(String) Prefix to be used in each dataset snapshot name. Optional.
zfs_migration_snapshot_prefix = tmp_snapshot_for_share_migration_	(String) Set snapshot prefix for usage in ZFS migration. Required.
zfs_replica_snapshot_prefix = tmp_snapshot_for_replication_	(String) Set snapshot prefix for usage in ZFS replication. Required.
zfs_service_ip = None	(String) IP to be added to admin-facing export location. Required.
zfs_share_export_ip = None	(String) IP to be added to user-facing export location. Required.
zfs_share_helpers = NFS=manila.share.drivers.zfsonlinux.utils.NFSviaZFSHelper	(List) Specify list of share export helpers for ZFS storage. It should look like following: FOO_protocol=foo.FooClass,BAR_protocol=bar.BarClass. Required.
zfs_ssh_private_key_path = None	(String) Path to SSH private key that should be used for SSHing ZFS storage host. Not used for replication operations. Optional.
zfs_ssh_user_password = None	(String) Password for user that is used for SSHing ZFS storage host. Not used for replication operations. They require passwordless SSH access. Optional.
zfs_ssh_username = None	(String) SSH user that will be used in 2 cases: 1) By manila-share service in case it is located on different host than its ZFS storage. 2) By manila-share services with other ZFS backends that perform replication. It is expected that SSHing will be key-based, passwordless. This user should be passwordless sudoer. Optional.
zfs_use_ssh = False	(Boolean) Remote ZFS storage hostname that should be used for SSHing. Optional.
zfs_zpool_list = None	(List) Specify list of zpools that are allowed to be used by backend. Can contain nested datasets. Examples: Without nested dataset: zpool_name. With nested dataset: zpool_name/nested_dataset_name. Required.

Oracle ZFS Storage Appliance driver

The Oracle ZFS Storage Appliance driver, version 1.0.0, enables the Oracle ZFS Storage Appliance (ZFSSA) to be used seamlessly as a shared storage resource for the OpenStack File System service (manila). The driver provides the ability to create and manage NFS and CIFS shares on the appliance, allowing virtual machines to access the shares simultaneously and securely.

Requirements

Oracle ZFS Storage Appliance Software version 2013.1.2.0 or later.

Supported operations

- Create NFS and CIFS shares.
- Delete NFS and CIFS shares.
- Allow or deny IP access to NFS shares.
- Create snapshots of a share.
- Delete snapshots of a share.
- Create share from snapshot.

Restrictions

- Access to CIFS shares are open and cannot be changed from manila.
- Version 1.0.0 of the driver only supports Single SVM networking mode.

Appliance configuration

1. Enable RESTful service on the ZFSSA Storage Appliance.
2. Create a new user on the appliance with the following authorizations:

```
scope=stmf - allow_configure=true
scope=nas - allow_clone=true, allow_createProject=true, allow_
↪createShare=true, allow_changeSpaceProps=true, allow_
↪changeGeneralProps=true, allow_destroy=true, allow_rollback=true, allow_
↪takeSnap=true
```

You can create a role with authorizations as follows:

```
zfssa:> configuration roles
zfssa:configuration roles> role OpenStackRole
zfssa:configuration roles OpenStackRole (uncommitted)> set description=
↪"OpenStack Manila Driver"
zfssa:configuration roles OpenStackRole (uncommitted)> commit
```

(continues on next page)

(continued from previous page)

```
zfssa:configuration roles> select OpenStackRole
zfssa:configuration roles OpenStackRole> authorizations create
zfssa:configuration roles OpenStackRole auth (uncommitted)> set scope=stmf
zfssa:configuration roles OpenStackRole auth (uncommitted)> set allow_
↪configure=true
zfssa:configuration roles OpenStackRole auth (uncommitted)> commit
```

You can create a user with a specific role as follows:

```
zfssa:> configuration users
zfssa:configuration users> user cinder
zfssa:configuration users cinder (uncommitted)> set fullname="OpenStack␣
↪Manila Driver"
zfssa:configuration users cinder (uncommitted)> set initial_password=12345
zfssa:configuration users cinder (uncommitted)> commit
zfssa:configuration users> select cinder set roles=OpenStackRole
```

3. Create a storage pool.

   An existing pool can also be used if required. You can create a pool as follows:

```
zfssa:> configuration storage
zfssa:configuration storage> config pool
zfssa:configuration storage verify> set data=2
zfssa:configuration storage verify> done
zfssa:configuration storage config> done
```

4. Create a new project.

   You can create a project as follows:

```
zfssa:> shares
zfssa:shares> project proj
zfssa:shares proj (uncommitted)> commit
```

5. Create a new or use an existing data IP address.

   You can create an interface as follows:

```
zfssa:> configuration net interfaces ip
zfssa:configuration net interfaces ip (uncommitted)> set v4addrs=127.0.0.
↪1/24
                     v4addrs = 127.0.0.1/24 (uncommitted)
zfssa:configuration net interfaces ip (uncommitted)> set links=vnic1
                       links = vnic1 (uncommitted)
zfssa:configuration net interfaces ip (uncommitted)> set admin=false
                       admin = false (uncommitted)
zfssa:configuration net interfaces ip (uncommitted)> commit
```

   It is required that both interfaces used for data and management are configured properly. The data interface must be different from the management interface.

6. Configure the cluster.

If a cluster is used as the manila storage resource, the following verifications are required:

- Verify that both the newly created pool and the network interface are of type singleton and are not locked to the current controller. This approach ensures that the pool and the interface used for data always belong to the active controller, regardless of the current state of the cluster.
- Verify that the management IP, data IP and storage pool belong to the same head.

Note: A short service interruption occurs during failback or takeover, but once the process is complete, manila should be able to access the pool through the data IP.

Driver options

The Oracle ZFSSA driver supports these options:

Table 18: Description of ZFSSA share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
zfssa_auth_password = None	(String) ZFSSA management authorized userpassword.
zfssa_auth_user = None	(String) ZFSSA management authorized username.
zfssa_data_ip = None	(String) IP address for data.
zfssa_host = None	(String) ZFSSA management IP address.
zfssa_manage_policy = loose	(String) Driver policy for share manage. A strict policy checks for a schema named manila_managed, and makes sure its value is true. A loose policy does not check for the schema.
zfssa_nas_checksum = fletcher4	(String) Controls checksum used for data blocks.
zfssa_nas_compression = off	(String) Data compression-off, lzjb, gzip-2, gzip, gzip-9.
zfssa_nas_logbias = latency	(String) Controls behavior when servicing synchronous writes.
zfssa_nas_mountpoint =	(String) Location of project in ZFS/SA.
zfssa_nas_quota_snap = true	(String) Controls whether a share quota includes snapshot.
zfssa_nas_rstchown = true	(String) Controls whether file ownership can be changed.
zfssa_nas_vscan = false	(String) Controls whether the share is scanned for viruses.
zfssa_pool = None	(String) ZFSSA storage pool name.
zfssa_project = None	(String) ZFSSA project name.
zfssa_rest_timeout = None	(String) REST connection timeout (in seconds).

EMC Isilon driver

The EMC Shared File Systems driver framework (EMCShareDriver) utilizes EMC storage products to provide shared file systems to OpenStack. The EMC driver is a plug-in based driver which is designed to use different plug-ins to manage different EMC storage products.

The Isilon driver is a plug-in for the EMC framework which allows the Shared File Systems service to interface with an Isilon back end to provide a shared filesystem. The EMC driver framework with the Isilon plug-in is referred to as the `Isilon Driver` in this document.

This Isilon Driver interfaces with an Isilon cluster via the REST Isilon Platform API (PAPI) and the RESTful Access to Namespace API (RAN).

Requirements

- Isilon cluster running OneFS 7.2 or higher

Supported shared filesystems and operations

The drivers supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported.
  - Only read-write access is supported.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.

Back end configuration

The following parameters need to be configured in the Shared File Systems service configuration file for the Isilon driver:

```
share_driver = manila.share.drivers.emc.driver.EMCShareDriver
emc_share_backend = isilon
emc_nas_server = <IP address of Isilon cluster>
emc_nas_login = <username>
emc_nas_password = <password>
```

Restrictions

The Isilon driver has the following restrictions:

- Only IP access type is supported for NFS and CIFS.
- Only FLAT network is supported.
- Quotas are not yet supported.

Driver options

The following table contains the configuration options specific to the share driver.

Table 19: Description of EMC share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
emc_nas_login = None	(String) User name for the EMC server.
emc_nas_password = None	(String) Password for the EMC server.
emc_nas_root_dir = None	(String) The root directory where shares will be located.
emc_nas_server = None	(String) EMC server hostname or IP address.
emc_nas_server_container = None	(String) DEPRECATED: Storage processor to host the NAS server. Obsolete. Unity driver supports nas server auto load balance.
emc_nas_server_port = 8080	(Port number) Port number for the EMC server.
emc_nas_server_secure = True	(Boolean) Use secure connection to server.
emc_share_backend = None	(String) Share backend.

Hitachi NAS (HNAS) driver

The HNAS driver provides NFS Shared File Systems to OpenStack.

Requirements

- Hitachi NAS Platform Models 3080, 3090, 4040, 4060, 4080, and 4100.
- HNAS/SMU software version is 12.2 or higher.
- HNAS configuration and management utilities to create a storage pool (span) and an EVS.
  - GUI (SMU).
  - SSC CLI.

Supported shared filesystems and operations

The driver supports NFS and CIFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.
- Revert a share to a snapshot.
- Extend a share.
- Manage a share.
- Unmanage a share.
- Shrink a share.
- Mount snapshots.
- Allow snapshot access.
- Deny snapshot access.
- Manage a snapshot.
- Unmanage a snapshot.

Driver options

This table contains the configuration options specific to the share driver.

Table 20: Description of HDS NAS share driver configuration options

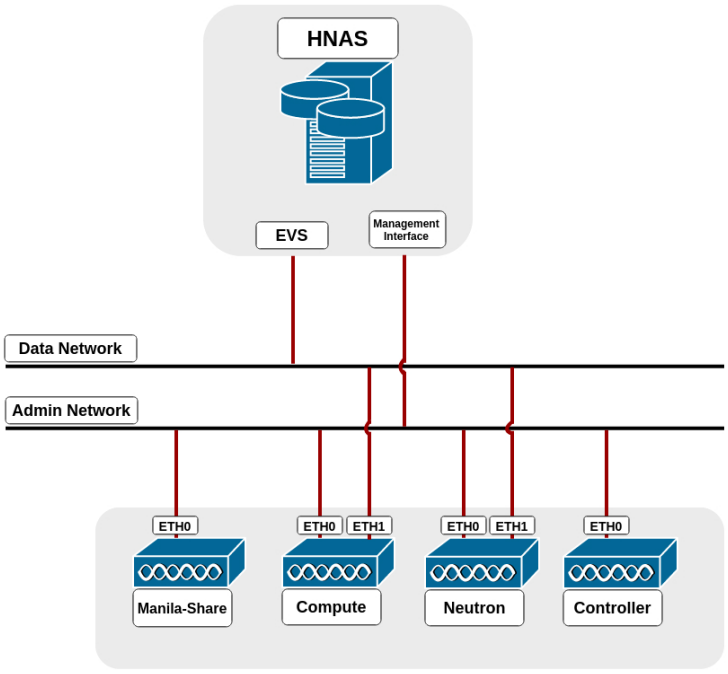
Configuration option = Default value	Description
[DEFAULT]	
hitachi_hnas_admin_network_ip = None	(String) Specify IP for mounting shares in the Admin network.
hitachi_hnas_allow_cifs_snapshot_while_mounted = False	(Boolean) By default, CIFS snapshots are not allowed to be taken when the share has clients connected because consistent point-in-time replica cannot be guaranteed for all files. Enabling this might cause inconsistent snapshots on CIFS shares.
hitachi_hnas_cluster_admin_ip0 = None	(String) The IP of the clusters admin node. Only set in HNAS multinode clusters.
hitachi_hnas_driver_helper = manila.share.drivers.hitachi.hnas.ssh.HNASSSHBackend	(String) Python class to be used for driver helper.
hitachi_hnas_evs_id = None	(Integer) Specify which EVS this backend is assigned to.
hitachi_hnas_evs_ip = None	(String) Specify IP for mounting shares.
hitachi_hnas_file_system_name = None	(String) Specify file-system name for creating shares.
hitachi_hnas_ip = None	(String) HNAS management interface IP for communication between Manila controller and HNAS.
hitachi_hnas_password = None	(String) HNAS user password. Required only if private key is not provided.
hitachi_hnas_ssh_private_key = None	(String) RSA/DSA private key value used to connect into HNAS. Required only if password is not provided.
hitachi_hnas_stalled_job_timeout = 30	(Integer) The time (in seconds) to wait for stalled HNAS jobs before aborting.
hitachi_hnas_user = None	(String) HNAS username Base64 String in order to perform tasks such as create file-systems and network interfaces.
[hnas1]	
share_backend_name = None	(String) The backend name for a given driver implementation.
share_driver = manila.share.drivers.generic.GenericShareDriver	(String) Driver to use for share creation.

Pre-configuration on OpenStack deployment

1. Install the OpenStack environment with manila. See the OpenStack installation guide.
2. Configure the OpenStack networking so it can reach HNAS Management interface and HNAS EVS Data interface.

Note: In the driver mode used by HNAS Driver (DHSS = `False`), the driver does not handle network configuration, it is up to the administrator to configure it.

- Configure the network of the manila-share node network to reach HNAS management interface through the admin network.
- Configure the network of the Compute and Networking nodes to reach HNAS EVS data interface through the data network.
- Example of networking architecture:



- Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini` file and update the following settings in their respective tags. In case you use linuxbridge, update bridge mappings at linuxbridge section:

Important: It is mandatory that HNAS management interface is reachable from the Shared File System node through the admin network, while the selected EVS data interface is reachable from OpenStack Cloud, such as through Neutron flat networking.

```
[ml2]
type_drivers = flat,vlan,vxlan,gre
mechanism_drivers = openvswitch
[ml2_type_flat]
```

(continues on next page)

(continued from previous page)

```
flat_networks = physnet1,physnet2
[ml2_type_vlan]
network_vlan_ranges = physnet1:1000:1500,physnet2:2000:2500
[ovs]
bridge_mappings = physnet1:br-ex,physnet2:br-eth1
```

You may have to repeat the last line above in another file on the Compute node, if it exists it is located in: `/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini`.

- In case openvswitch for neutron agent, run in network node:

```
# ifconfig eth1 0
# ovs-vsctl add-br br-eth1
# ovs-vsctl add-port br-eth1 eth1
# ifconfig eth1 up
```

- Restart all neutron processes.

3. Create the data HNAS network in OpenStack:

   - List the available projects:

```
$ openstack project list
```

   - Create a network to the given project (DEMO), providing the project name, a name for the network, the name of the physical network over which the virtual network is implemented, and the type of the physical mechanism by which the virtual network is implemented:

```
$ openstack network create --project DEMO \
  --provider-network-type flat \
  --provider-physical-network physnet2 hnas_network
```

   - Optional: List available networks:

```
$ openstack network list
```

   - Create a subnet to the same project (DEMO), the gateway IP of this subnet, a name for the subnet, the network name created before, and the CIDR of subnet:

```
$ openstack subnet create --project DEMO --gateway GATEWAY \
  --subnet-range SUBNET_CIDR --network NETWORK HNAS_SUBNET
```

   - Optional: List available subnets:

```
$ openstack subnet list
```

   - Add the subnet interface to a router, providing the router name and subnet name created before:

```
$ openstack router add subnet SUBNET ROUTER
```

Pre-configuration on HNAS

1. Create a file system on HNAS. See the Hitachi HNAS reference.

   Important: Make sure that the filesystem is not created as a replication target. For more information, refer to the official HNAS administration guide.

2. Prepare the HNAS EVS network.

   - Create a route in HNAS to the project network:

   ```
   $ console-context --evs <EVS_ID_IN_USE> route-net-add \
     --gateway <FLAT_NETWORK_GATEWAY> <TENANT_PRIVATE_NETWORK>
   ```

   Important: Make sure multi-tenancy is enabled and routes are configured per EVS.

   ```
   $ console-context --evs 3 route-net-add --gateway 192.168.1.1 \
   10.0.0.0/24
   ```

3. Configure the CIFS security.

   - Before using CIFS shares with the HNAS driver, make sure to configure a security service in the back end. For details, refer to the Hitachi HNAS reference.

Back end configuration

1. Configure HNAS driver.

   - Configure HNAS driver according to your environment. This example shows a minimal HNAS driver configuration:

   ```
   [DEFAULT]
   enabled_share_backends = hnas1
   enabled_share_protocols = NFS,CIFS

   [hnas1]
   share_backend_name = HNAS1
   share_driver = manila.share.drivers.hitachi.hnas.driver.
   ↪HitachiHNASDriver
   driver_handles_share_servers = False
   hitachi_hnas_ip = 172.24.44.15
   hitachi_hnas_user = supervisor
   hitachi_hnas_password = supervisor
   hitachi_hnas_evs_id = 1
   hitachi_hnas_evs_ip = 10.0.1.20
   hitachi_hnas_file_system_name = FS-Manila
   hitachi_hnas_cifs_snapshot_while_mounted = True
   ```

Note: The `hds_hnas_cifs_snapshot_while_mounted` parameter allows snapshots to be taken while CIFS shares are mounted. This parameter is set to `False` by default, which prevents a snapshot from being taken if the share is mounted or in use.

2. Optional. HNAS multi-backend configuration.

   - Update the `enabled_share_backends` flag with the names of the back ends separated by commas.
   - Add a section for every back end according to the example bellow:

```
[DEFAULT]
enabled_share_backends = hnas1,hnas2
enabled_share_protocols = NFS,CIFS

[hnas1]
share_backend_name = HNAS1
share_driver = manila.share.drivers.hitachi.hnas.driver.
↪HitachiHNASDriver
driver_handles_share_servers = False
hitachi_hnas_ip = 172.24.44.15
hitachi_hnas_user = supervisor
hitachi_hnas_password = supervisor
hitachi_hnas_evs_id = 1
hitachi_hnas_evs_ip = 10.0.1.20
hitachi_hnas_file_system_name = FS-Manila1
hitachi_hnas_cifs_snapshot_while_mounted = True

[hnas2]
share_backend_name = HNAS2
share_driver = manila.share.drivers.hitachi.hnas.driver.
↪HitachiHNASDriver
driver_handles_share_servers = False
hitachi_hnas_ip = 172.24.44.15
hitachi_hnas_user = supervisor
hitachi_hnas_password = supervisor
hitachi_hnas_evs_id = 1
hitachi_hnas_evs_ip = 10.0.1.20
hitachi_hnas_file_system_name = FS-Manila2
hitachi_hnas_cifs_snapshot_while_mounted = True
```

3. Disable DHSS for HNAS share type configuration:

Note: Shared File Systems requires that the share type includes the `driver_handles_share_servers` extra-spec. This ensures that the share will be created on a back end that supports the requested `driver_handles_share_servers` capability.

```
$ manila type-create hitachi False
```

4. Optional: Add extra-specs for enabling HNAS-supported features:

- These commands will enable various snapshot-related features that are supported in HNAS.

```
$ manila type-key hitachi set snapshot_support=True
$ manila type-key hitachi set mount_snapshot_support=True
$ manila type-key hitachi set revert_to_snapshot_support=True
$ manila type-key hitachi set create_share_from_snapshot_support=True
```

- To specify which HNAS back end will be created by the share, in case of multiple back end setups, add an extra-spec for each share-type to match a specific back end. Therefore, it is possible to specify which back end the Shared File System service will use when creating a share.

```
$ manila type-key hitachi set share_backend_name=hnas1
$ manila type-key hitachi2 set share_backend_name=hnas2
```

5. Restart all Shared File Systems services (`manila-share`, `manila-scheduler` and `manila-api`).

Share migration

Extra configuration is needed for allowing shares to be migrated from or to HNAS. In the OpenStack deployment, the manila-share node needs an additional connection to the EVS data interface. Furthermore, make sure to add `hitachi_hnas_admin_network_ip` to the configuration. This should match the value of `data_node_access_ips`. For more in-depth documentation, refer to the share migration documents

Manage and unmanage shares

Shared File Systems has the ability to manage and unmanage shares. If there is a share in the storage and it is not in OpenStack, you can manage that share and use it as a Shared File Systems share. Administrators have to make sure the exports are under the `/shares` folder beforehand. HNAS drivers use virtual-volumes (V-VOL) to create shares. Only V-VOL shares can be used by the driver, and V-VOLs must have a quota limit. If the NFS export is an ordinary FS export, it is not possible to use it in Shared File Systems. The unmanage operation only unlinks the share from Shared File Systems, all data is preserved. Both manage and unmanage operations are non-disruptive by default, until access rules are modified.

To **manage** a share, use:

```
$ manila manage [--name <name>] [--description <description>]
                [--share_type <share-type>]
                [--driver_options [<key=value> [<key=value> ...]]]
                [--public]
                <service_host> <protocol> <export_path>
```

Where:

Parameter	Description
service_host	Manila host, back end and share name. For example, ubuntu@hitachi1#hsp1. The available hosts can be listed with the command: manila pool-list (admin only).
protocol	Protocol of share to manage, such as NFS or CIFS.
export_path	**Share export path.** For NFS: 10.0.0.1:/shares/share_name For CIFS: \\10.0.0.1\share_name

Note: For NFS exports, export_path **must** include /shares/ after the target address. Trying to reference the share name directly or under another path will fail.

Note: For CIFS exports, although the shares will be created under the /shares/ folder in the back end, only the share name is needed in the export path. It should also be noted that the backslash \ character has to be escaped when entered in Linux terminals.

For additional details, refer to manila help manage.

To **unmanage** a share, use:

```
$ manila unmanage <share>
```

Where:

Parameter	Description
share	ID or name of the share to be unmanaged. A list of shares can be fetched with manila list.

Manage and unmanage snapshots

The Shared File Systems service also has the ability to manage share snapshots. Existing HNAS snapshots can be managed, as long as the snapshot directory is located in /snapshots/share_ID. New snapshots created through the Shared File Systems service are also created according to this specific folder structure.

To **manage** a snapshot, use:

```
$ manila snapshot-manage [--name <name>] [--description <description>]
                         [--driver_options [<key=value> [<key=value> ...]]]
                         <share> <provider_location>
```

Where:

Parameter	Description
`share`	ID or name of the share to be managed. A list of shares can be fetched with `manila list`.
`provider_location`	Location of the snapshot on the back end, such as `/snapshots/share_ID/snapshot_ID`.
`--driver_options`	Driver-related configuration, passed such as `size=10`.

Note: The mandatory `provider_location` parameter uses the same syntax for both NFS and CIFS shares. This is only the case for snapshot management.

Note: The `--driver_options` parameter `size` is **required** for the HNAS driver. Administrators need to know the size of the to-be-managed snapshot beforehand.

Note: If the `mount_snapshot_support=True` extra-spec is set in the share type, the HNAS driver will automatically create an export when managing a snapshot if one does not already exist.

To **unmanage** a snapshot, use:

```
$ manila snapshot-unmanage <snapshot>
```

Where:

Parameter	Description
`snapshot`	Name or ID of the snapshot(s).

Additional notes

- HNAS has some restrictions about the number of EVSs, filesystems, virtual-volumes, and simultaneous SSC connections. Check the manual specification for your system.
- Shares and snapshots are thin provisioned. It is reported to Shared File System only the real used space in HNAS. Also, a snapshot does not initially take any space in HNAS, it only stores the difference between the share and the snapshot, so it grows when share data is changed.
- Administrators should manage the projects quota (**`manila quota-update`**) to control the back end usage.
- Shares will need to be remounted after a revert-to-snapshot operation.

Hitachi Hyper Scale-Out Platform File Services Driver for OpenStack

The Hitachi Hyper Scale-Out Platform File Services Driver for OpenStack provides the management of file shares, supporting NFS shares with IP based rules to control access. It has a layer that handles the complexity of the protocol used to communicate to Hitachi Hyper Scale-Out Platform via a RESTful API, formatting and sending requests to the backend.

Requirements

- Hitachi Hyper Scale-Out Platform (HSP) version 1.1.
- HSP user with `file-system-full-access` role.
- Established network connection between the HSP interface and OpenStack nodes.

Supported shared filesystems and operations

The driver supports NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Extend a share.
- Shrink a share.
- Allow share access.
- Deny share access.
- Manage a share.
- Unmanage a share.

Note:

- Only `IP` access type is supported
- Both `RW` and `RO` access levels supported

Known restrictions

- The Hitachi HSP allows only 1024 virtual file systems per cluster. This determines the limit of shares the driver can provide.
- The Hitachi HSP file systems must have at least 128 GB. This means that all shares created by Shared File Systems service should have 128 GB or more.

Note: The driver has an internal filter function that accepts only requests for shares size greater than or equal to 128 GB, otherwise the request will fail or be redirected to another available storage backend.

Driver options

The following table contains the configuration options specific to the share driver.

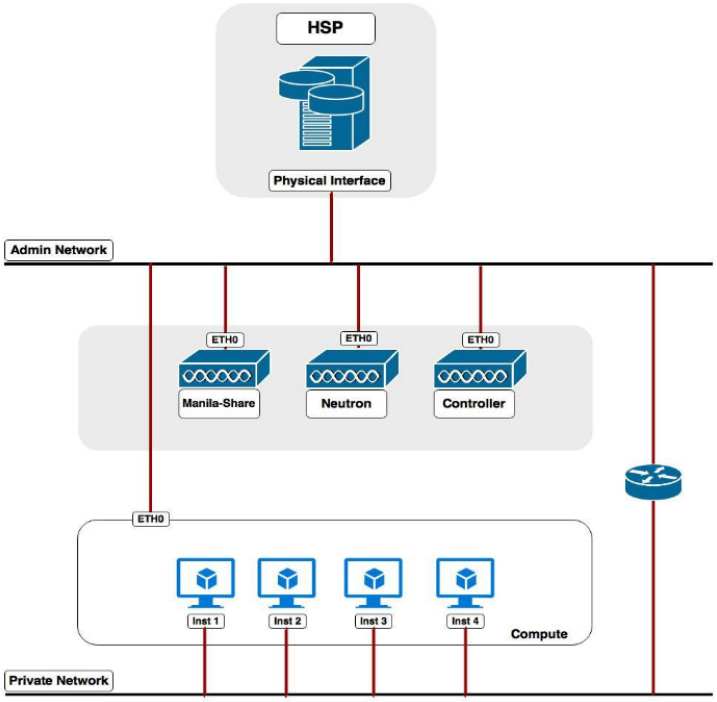
Table 21: Description of HDS HSP share driver configuration options

Configuration option = Default value	Description
[hsp1]	
`share_backend_name = None`	(String) The backend name for a given driver implementation.
`share_driver = manila.share.drivers.generic.GenericShareDriver`	(String) Driver to use for share creation.

Network approach

Note: In the driver mode used by HSP Driver (DHSS = `False`), the driver does not handle network configuration, it is up to the administrator to configure it.

- Configure the network of the manila-share, Compute and Networking nodes to reach HSP interface. For this, your provider network should be capable of reaching HSP Cluster-Virtual-IP. These connections are mandatory so nova instances are capable of accessing shares provided by the backend.
- The following image represents a valid scenario:



Note: To HSP, the Virtual IP is the address through which clients access shares and the Shared File Systems service sends commands to the management interface. This IP can be checked in HSP using its CLI:

```
$ hspadm ip-address list
```

Back end configuration

1. Configure HSP driver according to your environment. This example shows a valid HSP driver configuration:

```
[DEFAULT]
# ...
enabled_share_backends = hsp1
enabled_share_protocols = NFS
# ...

[hsp1]
share_backend_name = HITACHI1
share_driver = manila.share.drivers.hitachi.hsp.driver.HitachiHSPDriver
driver_handles_share_servers = False
hitachi_hsp_host = 172.24.47.190
hitachi_hsp_username = admin
hitachi_hsp_password = admin_password
```

2. Configure HSP share type.

Note: Shared File Systems service requires that the share type includes the

driver_handles_share_servers extra-spec. This ensures that the share will be created on a backend that supports the requested driver_handles_share_servers capability. Also, snapshot_support extra-spec should be provided if its value differs from the default value (True), as this driver version that currently does not support snapshot operations. For this driver both extra-specs must be set to False.

```
$ manila type-create --snapshot_support False hsp False
```

3. Restart all Shared File Systems services (manila-share, manila-scheduler and manila-api).

Manage and unmanage shares

The Shared File Systems service has the ability to manage and unmanage shares. If there is a share in the storage and it is not in OpenStack, you can manage that share and use it as a Shared File Systems share. Previous access rules are not imported by manila. The unmanage operation only unlinks the share from OpenStack, preserving all data in the share.

In order to manage a HSP share, it must adhere to the following rules:

- File system and share name must not contain spaces.
- Share name must not contain backslashes (\).

To **manage** a share use:

```
$ manila manage [--name <name>] [--description <description>]
[--share_type <share_type>] [--driver_options [<key=value>
[<key=value> ...]]] <service_host> <protocol> <export_path>
```

Where:

Parameter	Description
service_host	Manila host, backend and share name. For example, ubuntu@hitachi1#hsp1. The available hosts can be listed with the command: manila pool-list (admin only).
protocol	Must be **NFS**, the only supported protocol in this driver version.
export_path	The Hitachi Hyper Scale-Out Platform export path of the share, for example: 172.24.47.190:/some_share_name

To **unmanage** a share use:

```
$ manila unmanage <share>
```

Where:

Parameter	Description
share	ID or name of the share to be unmanaged. This list can be fetched with: manila list.

Additional notes

- Shares are thin provisioned. It is reported to manila only the real used space in HSP.
- Administrators should manage the tenants quota (`manila quota-update`) to control the backend usage.

HPE 3PAR Driver for OpenStack Manila

The HPE 3PAR driver provides NFS and CIFS shared file systems to OpenStack using HPE 3PARs File Persona capabilities.

For information on HPE 3PAR Driver for OpenStack Manila, refer to content kit page.

HPE 3PAR File Persona Software Suite concepts and terminology

The software suite comprises the following managed objects:

- File Provisioning Groups (FPGs)
- Virtual File Servers (VFSs)
- File Stores
- File Shares

The File Persona Software Suite is built upon the resilient mesh-active architecture of HPE 3PAR StoreServ and benefits from HPE 3PAR storage foundation of wide-striped logical disks and autonomic `Common Provisioning Groups (CPGs)`. A CPG can be shared between file and block to create the File Shares or the logical unit numbers (LUNs) to provide true convergence.

`A File Provisioning Group (FPG)` is an instance of the HPE intellectual property Adaptive File System. It controls how files are stored and retrieved. Each FPG is transparently constructed from one or multiple Virtual Volumes (VVs) and is the unit for replication and disaster recovery for File Persona Software Suite. There are up to 16 FPGs supported on a node pair.

`A Virtual File Server (VFS)` is conceptually like a server. As such, it presents virtual IP addresses to clients, participates in user authentication services, and can have properties for such things as user/group quota management and antivirus policies. Up to 16 VFSs are supported on a node pair, one per FPG.

`File Stores` are the slice of a VFS and FPG at which snapshots are taken, capacity quota management can be performed, and antivirus scan service policies customized. There are up to 256 File Stores supported on a node pair, 16 File Stores per VFS.

`File Shares` are what provide data access to clients via SMB, NFS, and the Object Access API, subject to the share permissions applied to them. Multiple File Shares can be created for a File Store and at different directory levels within a File Store.

Supported shared filesystems

The driver supports CIFS and NFS shares.

Operations supported

- Create a share.

  Share is not accessible until access rules allow access.
- Delete a share.
- Allow share access.

  Note the following limitations:

  IP access rules are required for NFS share access.

  User access rules are not allowed for NFS shares.

  User access rules are required for SMB share access.

  User access requires a File Persona local user for SMB shares.

  Shares are read/write (and subject to ACLs).
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.
- Extend a share.
- Shrink a share.
- Share networks.

HPE 3PAR File Persona driver can be configured to work with or without share networks. When using share networks, the HPE 3PAR driver allocates an FSIP on the back end FPG (VFS) to match the share networks subnet and segmentation ID. Security groups associated with share networks are ignored.

Operations not supported

- Manage and unmanage
- Manila Experimental APIs (consistency groups, replication, and migration) were added in Mitaka but have not yet been implemented by the HPE 3PAR File Persona driver.

Requirements

On the OpenStack host running the Manila share service:

- python-3parclient version 4.2.0 or newer from PyPI.

On the HPE 3PAR array:

- HPE 3PAR Operating System software version 3.2.1 MU3 or higher.
- The array class and hardware configuration must support File Persona.

Pre-configuration on the HPE 3PAR StoreServ

The following HPE 3PAR CLI commands show how to set up the HPE 3PAR StoreServ to use File Persona with OpenStack Manila. HPE 3PAR File Persona must be initialized, and started on the HPE 3PAR storage.

```
cli% startfs 0:2:1 1:2:1
cli% setfs nodeip -ipaddress 10.10.10.11 -subnet 255.255.240.0 0
cli% setfs nodeip -ipaddress 10.10.10.12 -subnet 255.255.240.0 1
cli% setfs dns 192.168.8.80,127.127.5.50 foo.com,bar.com
cli% setfs gw 10.10.10.10
```

- A File Provisioning Group (FPG) must be created for use with the Shared File Systems service.

```
cli% createfpg examplecpg examplefpg 18T
```

- A Virtual File Server (VFS) must be created on the FPG.
- The VFS must be configured with an appropriate share export IP address.

```
cli% createvfs -fpg examplefpg 10.10.10.101 255.255.0.0 examplevfs
```

- A local user in the Administrators group is needed for CIFS (SMB) shares.

```
cli% createfsgroup fsusers
cli% createfsuser passwd <password> -enable true -grplist
Users,Administrators primarygroup fsusers fsadmin
```

- The WSAPI with HTTP and/or HTTPS must be enabled and started.

```
cli% setwsapi -https enable
cli% startwsapi
```

HPE 3PAR shared file system driver configuration

- Install the python-3parclient python package on the OpenStack Block Storage system:

```
$ pip install 'python-3parclient>=4.0,<5.0'
```

- Manila configuration file

  The Manila configuration file (typically `/etc/manila/manila.conf`) defines and configures the Manila drivers and backends. After updating the configuration file, the Manila share service must be restarted for changes to take effect.

- Enable share protocols

  To enable share protocols, an optional list of supported protocols can be specified using the `enabled_share_protocols` setting in the `DEFAULT` section of the `manila.conf` file. The default is `NFS, CIFS` which allows both protocols supported by HPE 3PAR (NFS and SMB). Where Manila uses the term `CIFS`, HPE 3PAR uses the term `SMB`. Use the `enabled_share_protocols` option if you want to only provide one type of share (for example, only NFS) or if you want to explicitly avoid the introduction of other protocols that can be added for other drivers in the future.

- Enable share back ends

  In the `[DEFAULT]` section of the Manila configuration file, use the `enabled_share_backends` option to specify the name of one or more back-end configuration sections to be enabled. To enable multiple back ends, use a comma-separated list.

  Note: The name of the backends configuration section is used (which may be different from the `share_backend_name` value)

- Configure each back end

  For each back end, a configuration section defines the driver and back end options. These include common Manila options, as well as driver-specific options. The following `Driver options` section describes the parameters that need to be configured in the Manila configuration file for the HPE 3PAR driver.

Driver options

The following table contains the configuration options specific to the share driver.

Table 22: Description of HPE 3PAR share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
hpe3par_api_url =	(String) 3PAR WSAPI Server Url like https://<3par ip>:8080/api/v1
hpe3par_cifs_admin_access_domain = LOCAL_CLUSTER	(String) File system domain for the CIFS admin user.
hpe3par_cifs_admin_access_password =	(String) File system admin password for CIFS.
hpe3par_cifs_admin_access_username =	(String) File system admin user name for CIFS.
hpe3par_debug = False	(Boolean) Enable HTTP debugging to 3PAR
hpe3par_fpg = None	(Unknown) The File Provisioning Group (FPG) to use
hpe3par_fstore_per_share = False	(Boolean) Use one filestore per share
hpe3par_password =	(String) 3PAR password for the user specified in hpe3par_username
hpe3par_require_cifs_ip = False	(Boolean) Require IP access rules for CIFS (in addition to user)
hpe3par_san_ip =	(String) IP address of SAN controller
hpe3par_san_login =	(String) Username for SAN controller
hpe3par_san_password =	(String) Password for SAN controller
hpe3par_san_ssh_port = 22	(Port number) SSH port to use with SAN
hpe3par_share_mount_path = /mnt/	(String) The path where shares will be mounted when deleting nested file trees.
hpe3par_username =	(String) 3PAR username with the edit role

HPE 3PAR Manila driver configuration example

The following parameters shows a sample subset of the `manila.conf` file, which configures two backends and the relevant `[DEFAULT]` options. A real configuration would include additional `[DEFAULT]` options and additional sections that are not discussed in this document. In this example, the backends are using different FPGs on the same array:

```
[DEFAULT]
enabled_share_backends = HPE1,HPE2
enabled_share_protocols = NFS,CIFS
default_share_type = default
[HPE1]
share_backend_name = HPE3PAR1
share_driver = manila.share.drivers.hpe.hpe_3par_driver.HPE3ParShareDriver
driver_handles_share_servers = False
max_over_subscription_ratio = 1
hpe3par_fpg = examplefpg,10.10.10.101
```

(continues on next page)

(continued from previous page)

```
hpe3par_san_ip = 10.20.30.40
hpe3par_api_url = https://10.20.30.40:8080/api/v1
hpe3par_username = <username>
hpe3par_password = <password>
hpe3par_san_login = <san_username>
hpe3par_san_password = <san_password>
hpe3par_debug = False
hpe3par_cifs_admin_access_username = <fs_admin>
hpe3par_cifs_admin_access_password = <fs_password>
[HPE2]
share_backend_name = HPE3PAR2
share_driver = manila.share.drivers.hpe.hpe_3par_driver.HPE3ParShareDriver
driver_handles_share_servers = False
max_over_subscription_ratio = 1
hpe3par_fpg = examplefpg2,10.10.10.102
hpe3par_san_ip = 10.20.30.40
hpe3par_api_url = https://10.20.30.40:8080/api/v1
hpe3par_username = <username>
hpe3par_password = <password>
hpe3par_san_login = <san_username>
hpe3par_san_password = <san_password>
hpe3par_debug = False
hpe3par_cifs_admin_access_username = <fs_admin>
hpe3par_cifs_admin_access_password = <password>
```

Network approach

Network connectivity between the storage array (SSH/CLI and WSAPI) and the Manila host is required for share management. Network connectivity between the clients and the VFS is required for mounting and using the shares. This includes:

- Routing from the client to the external network.
- Assigning the client an external IP address, for example a floating IP.
- Configuring the Shared File Systems service host networking properly for IP forwarding.
- Configuring the VFS networking properly for client subnets.
- Configuring network segmentation, if applicable.

In the OpenStack Kilo release, the HPE 3PAR driver did not support share networks. Share access from clients to HPE 3PAR shares required external network access (external to OpenStack) and was set up and configured outside of Manila.

In the OpenStack Liberty release, the HPE 3PAR driver could run with or without share networks. The configuration option `driver_handles_share_servers`` ( ``True` or `False` ) indicated whether share networks could be used. When set to `False`, the HPE 3PAR driver behaved as described earlier for Kilo. When set to `True`, the share networks subnet, segmentation ID and IP address range were used to allocate an FSIP on the HPE 3PAR. There is a limit of four FSIPs per VFS. For clients to communicate with shares via this FSIP, the client must have access to the external network using the subnet and segmentation ID of the share network.

For example, the client must be routed to the neutron provider network with external access. The Manila host networking configuration and network switches must support the subnet routing. If the VLAN segmentation ID is used, communication with the share will use the FSIP IP address. Neutron networking is required for HPE 3PAR share network support. Flat and VLAN provider networks are supported, but the HPE 3PAR driver does not support share network security groups.

Share access

A share that is mounted before access is allowed can appear to be an empty read-only share. After granting access, the share must be remounted.

- IP access rules are required for NFS.
- SMB shares require user access rules.

With the proper access rules, share access is not limited to the OpenStack environment. Access rules added via Manila or directly in HPE 3PAR CLI can be used to allow access to clients outside of the stack. The HPE 3PAR VFS/FSIP settings determine the subnets visible for HPE 3PAR share access.

- IP access rules

  To allow IP access to a share in the horizon UI, find the share in the Project|Manage Compute|Shares view. Use the `Manage Rules` action to add a rule. Select IP as the access type, and enter the external IP address (for example, the floating IP) of the client in the `Access to` box.

  You can also use the command line to allow IP access to a share in the horizon UI with the command:

  ```
  $ manila access-allow <share-id> ip <ip-address>
  ```

- User access rules

  To allow user access to a share in the horizon UI, find the share in the Project|Manage Compute|Shares view. Use the `Manage Rules` action to add a rule. Select user as the access type and enter user name in the `Access to` box.

  You can also use the command line to allow user access to a share in the horizon UI with the command:

  ```
  $ manila access-allow <share-id> user <user name>
  ```

  The user name must be an HPE 3PAR user.

  Share access is different from file system permissions, for example, ACLs on files and folders. If a user wants to read a file, the user must have at least read permissions on the share and an ACL that grants him read permissions on the file or folder. Even with full control share access, it does not mean every user can do everything due to the additional restrictions of the folder ACLs.

  To modify the file or folder ACLs, allow access to an HPE 3PAR File Persona local user that is in the administrators group and connect to the share using that users credentials. Then, use the appropriate mechanism to modify the ACL or permissions to allow different access than what is provided by default.

Share types

When creating a share, a share type can be specified to determine where and how the share will be created. If a share type is not specified, the `default_share_type` set in the Shared File Systems service configuration file is used.

Manila share types are a type or label that can be selected at share creation time in OpenStack. These types can be created either in the `Admin` horizon UI or using the command line, as follows:

```
$ manila --os-username admin --os-tenant-name demo type-create
is_public false <name> false
```

The `<name>` is the name of the new share type. False at the end specifies `driver_handles_share_servers=False`. The `driver_handles_share_servers` setting in the share type needs to match the setting configured for the back end in the `manila.conf` file.

`is_public` is used to indicate whether this share type is applicable to all tenants or will be assigned to specific tenants.

`--os-username admin --os-tenant-name demo` are only needed if your environment variables do not specify the desired user and tenant.

For share types that are not public, use Manila `type-access-add` to assign the share type to a tenant.

- Using share types to require share networks

  The Shared File Systems service requires that the share type include the `driver_handles_share_servers` extra-spec. This ensures that the share is created on a back end that supports the requested `driver_handles_share_servers` (share networks) capability. From the Liberty release forward, both `True` and `False` are supported.

  The `driver_handles_share_servers` setting in the share type must match the setting in the back end configuration.

- Using share types to select backends by name

  Administrators can optionally specify that a particular share type be explicitly associated with a single back end (or group of backends) by including the extra spec share_backend_name to match the name specified within the `share_backend_name` option in the back end configuration.

  When a share type is not selected during share creation, the default share type is used. To prevent creating these shares on any back end, the default share type needs to be specific enough to find appropriate default backends (or to find none if the default should not be used). The following example shows how to set share_backend_name for a share type.

  ```
  $ manila --os-username admin --os-tenant-name demo type-key <share-type>
  set share_backend_name=HPE3PAR2
  ```

- Using share types to select backends with capabilities

  The HPE 3PAR driver automatically reports capabilities based on the FPG used for each back end. An administrator can create share types with extra specs, which controls share types that can use FPGs with or without specific capabilities.

  With the OpenStack Liberty release or later, below section shows the extra specs used with the capabilities filter and the HPE 3PAR driver:

hpe3par_flash_cache When the value is set to <is> True (or <is> False), shares of this type are only created on a back end that uses HPE 3PAR Adaptive Flash Cache. For Adaptive Flash Cache, the HPE 3PAR StoreServ Storage array must meet the following requirements:

- Adaptive Flash Cache enabled
- Available SSDs
- Adaptive Flash Cache must be enabled on the HPE 3PAR StoreServ Storage array. This is done with the following CLI command:

```
cli% createflashcache <size>
```

<size> must be in 16 GB increments. For example, the below command creates 128 GB of Flash Cache for each node pair in the array.

```
cli% createflashcache 128g
```

- Adaptive Flash Cache must be enabled for the VV set used by an FPG. For example, setflashcache vvset:<fpgname>. The VV set name is the same as the FPG name.

Note: This setting affects all shares in that FPG (on that back end).

Dedupe When the value is set to <is> True (or <is> False), shares of this type are only created on a back end that uses deduplication. For HPE 3PAR File Persona, the provisioning type is determined when the FPG is created. Using the createfpg tdvv option creates an FPG that supports both dedupe and thin provisioning. The thin deduplication must be enabled to use the tdvv option.

thin_provisioning When the value is set to <is> True (or <is> False), shares of this type are only created on a back end that uses thin (or full) provisioning. For HPE 3PAR File Persona, the provisioning type is determined when the FPG is created. By default, FPGs are created with thin provisioning. The capacity filter uses the total provisioned space and configured max_oversubscription_ratio when filtering and weighing backends that use thin provisioning.

- Using share types to influence share creation options

Scoped extra-specs are used to influence vendor-specific implementation details. Scoped extra-specs use a prefix followed by a colon. For HPE 3PAR, these extra specs have a prefix of hpe3par.

The following HPE 3PAR extra-specs are used when creating CIFS (SMB) shares:

hpe3par:smb_access_based_enum smb_access_based_enum (Access Based Enumeration) specifies if users can see only the files and directories to which they have been allowed access on the shares. Valid values are True or False. The default is False.

hpe3par:smb_continuous_avail smb_continuous_avail (Continuous Availability) specifies if continuous availability features of SMB3 should be enabled for this share. Valid values are True or False. The default is True.

hpe3par:smb_cache smb_cache specifies client-side caching for offline files. The default value is manual. Valid values are:

- off the client must not cache any files from this share. The share is configured to disallow caching.

  - `manual` the client must allow only manual caching for the files open from this share.
  - `optimized` the client may cache every file that it opens from this share. Also, the client may satisfy the file requests from its local cache. The share is configured to allow automatic caching of programs and documents.
  - `auto` the client may cache every file that it opens from this share. The share is configured to allow automatic caching of documents.

When creating NFS shares, the following HPE 3PAR extra-specs are used:

`hpe3par:nfs_options` Comma separated list of NFS export options.

The NFS export options have the following limitations:

`ro` and `rw` are not allowed (will be determined by the driver)

`no_subtree_check` and `fsid` are not allowed per HPE 3PAR CLI support

`(in)secure` and `(no_)root_squash` are not allowed because the HPE 3PAR driver controls those settings

All other NFS options are forwarded to the HPE 3PAR as part of share creation. The HPE 3PAR performs additional validation at share creation time. For details, see the HPE 3PAR CLI help.

Implementation characteristics

- Shares from snapshots
  - When a share is created from a snapshot, the share must be deleted before the snapshot can be deleted. This is enforced by the driver.
  - A snapshot of an empty share will appear to work correctly, but attempting to create a share from an empty share snapshot may fail with an `NFS Create export` error.
  - HPE 3PAR File Persona snapshots are for an entire File Store. In Manila, they appear as snapshots of shares. A share sub-directory is used to give the appearance of a share snapshot when using `create share from snapshot` .
- Snapshots
  - For HPE 3PAR File Persona, snapshots are per File Store and not per share. So, the HPE 3PAR limit of 1024 snapshots per File Store results in a Manila limit of 1024 snapshots per tenant on each back end FPG.
  - Before deleting a share, you must delete its snapshots. This is enforced by Manila. For HPE 3PAR File Persona, this also kicks off a snapshot reclamation.
- Size enforcement

  Manila users create shares with size limits. HPE 3PAR enforces size limits by using File Store quotas. When using `hpe3par_fstore_per_share``= ``True``(the non-default setting) there is only one share per File Store, so the size enforcement acts as expected. When using ``hpe3par_fstore_per_share = False` (the default), the HPE 3PAR Manila driver uses one File Store for multiple shares. In this case, the size of the File Store limit is set to the cumulative limit of its Manila share sizes. This can allow one tenant share to exceed the limit and affect the space available for the same tenants other shares. One tenant cannot use another tenants File Store.

- File removal

  When shares are removed and the `hpe3par_fstore_per_share``=``False` setting is used (the default), files may be left behind in the File Store. Prior to Mitaka, removal of obsolete share directories and files that have been stranded would require tools outside of OpenStack/Manila. In Mitaka and later, the driver mounts the File Store to remove the deleted shares subdirectory and files. For SMB/CIFS share, it requires the `hpe3par_cifs_admin_access_username` and `hpe3par_cifs_admin_access_password` configuration. If the mount and delete cannot be performed, an error is logged and the share is deleted in Manila. Due to the potential space held by leftover files, File Store quotas are not reduced when shares are removed.

- Multi-tenancy

  – Network

    The `driver_handles_share_servers` configuration setting determines whether share networks are supported. When `driver_handles_share_servers` is set to `True`, a share network is required to create a share. The administrator creates share networks with the desired network, subnet, IP range, and segmentation ID. The HPE 3PAR is configured with an FSIP using the same subnet and segmentation ID and an IP address allocated from the neutron network. Using share network-specific IP addresses, subnets, and segmentation IDs give the appearance of better tenant isolation. Shares on an FPG, however, are accessible via any of the FSIPs (subject to access rules). Back end filtering should be used for further separation.

  – Back end filtering

    A Manila HPE 3PAR back end configuration refers to a specific array and a specific FPG. With multiple backends and multiple tenants, the scheduler determines where shares will be created. In a scenario where an array or back end needs to be restricted to one or more specific tenants, share types can be used to influence the selection of a back end. For more information on using share types, see *Share types* .

  – Tenant limit

    The HPE 3PAR driver uses one File Store per tenant per protocol in each configured FPG. When only one back end is configured, this results in a limit of eight tenants (16 if only using one protocol). Use multiple back end configurations to introduce additional FPGs on the same array to increase the tenant limit.

    When using share networks, an FSIP is created for each share network (when its first share is created on the back end). The HPE 3PAR supports 4 FSIPs per FPG (VFS). One of those 4 FSIPs is reserved for the initial VFS IP, so the share network limit is 48 share networks per node pair.

Huawei driver

Huawei NAS driver is a plug-in based on the Shared File Systems service. The Huawei NAS driver can be used to provide functions such as the share and snapshot for virtual machines, or instances, in OpenStack. Huawei NAS driver enables the OceanStor V3 series V300R002 storage system to provide only network filesystems for OpenStack.

Requirements

- The OceanStor V3 series V300R002 storage system.
- The following licenses should be activated on V3 for File: CIFS, NFS, HyperSnap License (for snapshot).

Supported shared filesystems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
  - Only user access is supported for CIFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Support pools in one backend.
- Extend a share.
- Shrink a share.
- Create a replica.
- Delete a replica.
- Promote a replica.
- Update a replica state.

Pre-configurations on Huawei

1. Create a driver configuration file. The driver configuration file name must be the same as the `manila_huawei_conf_file` item in the `manila_conf` configuration file.
2. Configure the product. Product indicates the storage system type. For the OceanStor V3 series V300R002 storage systems, the driver configuration file is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<Config>
    <Storage>
        <Product>V3</Product>
        <LogicalPortIP>x.x.x.x</LogicalPortIP>
```

(continues on next page)

(continued from previous page)

```
        <RestURL>https://x.x.x.x:8088/deviceManager/rest/</RestURL>
        <UserName>xxxxxxxxx</UserName>
        <UserPassword>xxxxxxxxx</UserPassword>
    </Storage>
    <Filesystem>
        <Thin_StoragePool>xxxxxxxxx</Thin_StoragePool>
        <Thick_StoragePool>xxxxxxxxx</Thick_StoragePool>
        <WaitInterval>3</WaitInterval>
        <Timeout>60</Timeout>
    </Filesystem>
</Config>
```

The options are:

- `Product` is a type of storage product. Set it to `V3`.
- `LogicalPortIP` is the IP address of the logical port.
- `RestURL` is an access address of the REST interface. Multiple RestURLs can be configured in <`RestURL`>, separated by ;. The driver will automatically retry another `RestURL` if one fails to connect.
- `UserName` is the user name of an administrator.
- `UserPassword` is the password of an administrator.
- `Thin_StoragePool` is the name of a thin storage pool to be used.
- `Thick_StoragePool` is the name of a thick storage pool to be used.
- `WaitInterval` is the interval time of querying the file system status.
- `Timeout` is the timeout period for waiting command execution of a device to complete.

Back end configuration

Modify the `manila.conf` Shared File Systems service configuration file and add `share_driver` and `manila_huawei_conf_file` items. Here is an example for configuring a storage system:

```
share_driver = manila.share.drivers.huawei.huawei_nas.HuaweiNasDriver
manila_huawei_conf_file = /etc/manila/manila_huawei_conf.xml
driver_handles_share_servers = False
```

Driver options

The following table contains the configuration options specific to the share driver.

Table 23: Description of Huawei share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`manila_huawei_conf_file = /etc/manila/manila_huawei_conf.xml`	(String) The configuration file for the Manila Huawei driver.

IBM Spectrum Scale share driver

IBM Spectrum Scale is a flexible software-defined storage product that can be deployed as high-performance file storage or a cost optimized large-scale content repository. IBM Spectrum Scale, previously known as IBM General Parallel File System (GPFS), is designed to scale performance and capacity with no bottlenecks. IBM Spectrum Scale is a cluster file system that provides concurrent access to file systems from multiple nodes. The storage provided by these nodes can be direct attached, network attached, SAN attached, or a combination of these methods. Spectrum Scale provides many features beyond common data access, including data replication, policy based storage management, and space efficient file snapshot and clone operations.

Supported shared filesystems and operations (NFS shares only)

The Spectrum Scale share driver supports NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.
  - Only IP access type is supported.
  - Both RW & RO access level is supported.
- Deny share access.
- Create a share snapshot.
- Delete a share snapshot.
- Create a share from a snapshot.
- Extend a share.
- Manage a share.
- Unmanage a share.

Requirements

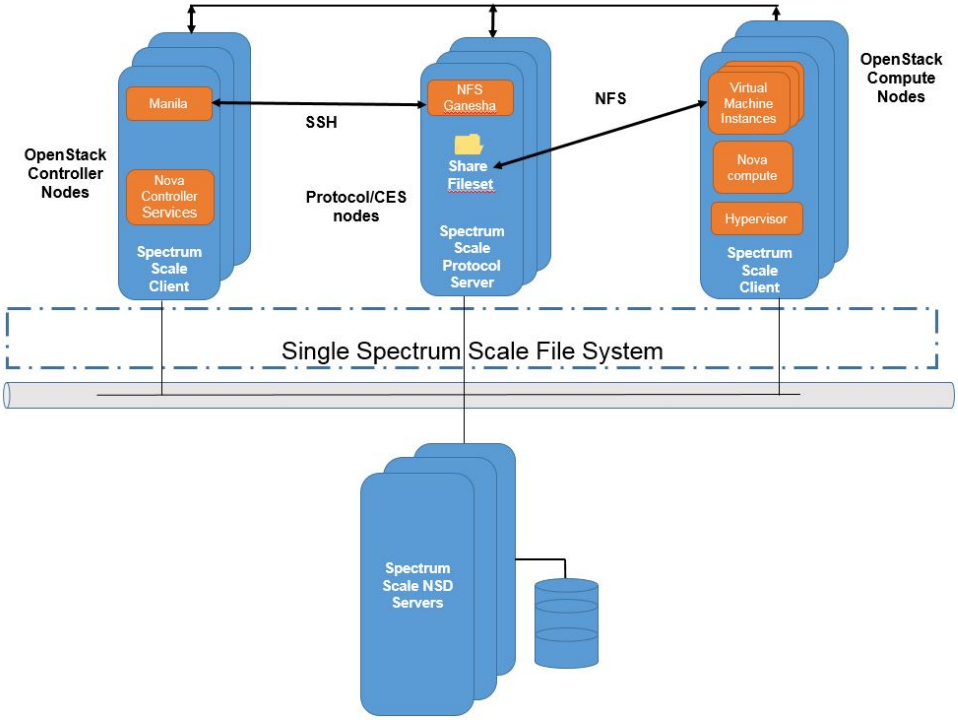
Spectrum Scale must be installed and a cluster must be created that includes one or more storage nodes and protocol server nodes. The NFS server running on these nodes is used to export shares to storage consumers in OpenStack virtual machines or even to bare metal storage consumers in the OpenStack environment. A file system must also be created and mounted on these nodes before configuring the manila service to use Spectrum Scale storage. For more details, refer to Spectrum Scale product documentation.

Spectrum Scale supports two ways of exporting data through NFS with high availability.

1. CES (which uses Ganesha NFS)
   - This is provided inherently by the protocol support in Spectrum Scale and is a recommended method for NFS access.
2. CNFS (which uses kernel NFS)

For more information on NFS support in Spectrum Scale, refer to Protocol support in Spectrum Scale and NFS Support overview in Spectrum Scale.

The following figure is an example of Spectrum Scale architecture with OpenStack services:



Quotas should be enabled for the Spectrum Scale filesystem to be exported through NFS using Spectrum Scale share driver. Use the following command to enable quota for a filesystem:

```
$ mmchfs <filesystem> -Q yes
```

Limitation

Spectrum Scale share driver currently supports creation of NFS shares in the flat network space only. For example, the Spectrum Scale storage node exporting the data should be in the same network as that of the Compute VMs which mount the shares acting as NFS clients.

Driver configuration

Spectrum Scale share driver supports creation of shares using both NFS servers (Ganesha using Spectrum Scale CES/Kernel NFS).

For both the NFS server types, you need to set the `share_driver` in the `manila.conf` as:

```
share_driver = manila.share.drivers.ibm.gpfs.GPFSShareDriver
```

Spectrum Scale CES (NFS Ganesha server)

To use Spectrum Scale share driver in this mode, set the `gpfs_share_helpers` in the `manila.conf` as:

```
gpfs_share_helpers = CES=manila.share.drivers.ibm.gpfs.CESHelper
```

Following table lists the additional configuration options which are used with this driver configuration.

Table 24: Description of IBM Spectrum Scale CES share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`gpfs_mount_point_base = $state_path/mnt`	(String) Base folder where exported shares are located.
`gpfs_nfs_server_type = CES`	(String) NFS Server type. Valid choices are CES (Ganesha NFS) or KNFS (Kernel NFS).
`gpfs_share_export_ip = None`	(Host address) IP to be added to GPFS export string.
`gpfs_share_helpers = KNFS=manila.share.drivers.ibm.gpfs.KNFSHelper, CES=manila.share.drivers.ibm.gpfs.CESHelper`	(List) Specify list of share export helpers.
`gpfs_ssh_login = None`	(String) GPFS server SSH login name.
`gpfs_ssh_password = None`	(String) GPFS server SSH login password. The password is not needed, if gpfs_ssh_private_key is configured.
`gpfs_ssh_port = 22`	(Port number) GPFS server SSH port.
`gpfs_ssh_private_key = None`	(String) Path to GPFS server SSH private key for login.
`is_gpfs_node = False`	(Boolean) True:when Manila services are running on one of the Spectrum Scale node. False:when Manila services are not running on any of the Spectrum Scale node.

Note: Configuration options related to ssh are required only if `is_gpfs_node` is set to `False`.

Spectrum Scale Clustered NFS (Kernel NFS server)

To use Spectrum Scale share driver in this mode, set the `gpfs_share_helpers` in the `manila.conf` as:

```
gpfs_share_helpers = KNFS=manila.share.drivers.ibm.gpfs.KNFSHelper
```

Following table lists the additional configuration options which are used with this driver configuration.

Table 25: Description of IBM Spectrum Scale KNFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
gpfs_mount_point_base = $state_path/mnt	(String) Base folder where exported shares are located.
gpfs_nfs_server_list = None	(List) A list of the fully qualified NFS server names that make up the OpenStack Manila configuration.
gpfs_nfs_server_type = CES	(String) NFS Server type. Valid choices are CES (Ganesha NFS) or KNFS (Kernel NFS).
gpfs_share_export_ip = None	(Host address) IP to be added to GPFS export string.
gpfs_share_helpers = KNFS=manila.share.drivers.ibm.gpfs.KNFSHelper, CES=manila.share.drivers.ibm.gpfs.CESHelper	(List) Specify list of share export helpers.
gpfs_ssh_login = None	(String) GPFS server SSH login name.
gpfs_ssh_password = None	(String) GPFS server SSH login password. The password is not needed, if gpfs_ssh_private_key is configured.
gpfs_ssh_port = 22	(Port number) GPFS server SSH port.
gpfs_ssh_private_key = None	(String) Path to GPFS server SSH private key for login.
is_gpfs_node = False	(Boolean) True:when Manila services are running on one of the Spectrum Scale node. False:when Manila services are not running on any of the Spectrum Scale node.

Note: Configuration options related to ssh are required only if `is_gpfs_node` is set to `False`.

Share creation steps

Sample configuration

```
[gpfs]
share_driver = manila.share.drivers.ibm.gpfs.GPFSShareDriver
gpfs_share_export_ip = x.x.x.x
gpfs_mount_point_base = /ibm/gpfs0
gpfs_nfs_server_type = CES
is_gpfs_node = True
gpfs_share_helpers = CES=manila.share.drivers.ibm.gpfs.CESHelper
share_backend_name = GPFS
driver_handles_share_servers = False
```

Create GPFS share type and set extra spec

```
$ manila type-create --snapshot_support True \
  --create_share_from_snapshot_support True gpfs False

$ manila type-key gpfs set share_backend_name=GPFS
```

INFINIDAT InfiniBox Share driver

The INFINIDAT Share driver provides support for managing filesystem shares on the INFINIDAT InfiniBox storage systems.

This section explains how to configure the INFINIDAT driver.

Supported operations

- Create and delete filesystem shares.
- Ensure filesystem shares.
- Extend a share.
- Create and delete filesystem snapshots.
- Create a share from a share snapshot.
- Revert a share to its snapshot.
- Mount a snapshot.
- Set access rights to shares and snapshots.

  Note the following limitations:

  - Only IP access type is supported.
  - Both RW & RO access levels are supported.

External package installation

The driver requires the `infinisdk` package for communicating with InfiniBox systems. Install the package from PyPI using the following command:

```
$ pip install infinisdk
```

Setting up the storage array

Create a storage pool object on the InfiniBox array in advance. The storage pool will contain shares managed by OpenStack. Refer to the InfiniBox manuals for details on pool management.

Driver configuration

Edit the `manila.conf` file, which is usually located under the following path `/etc/manila/manila.conf`.

- Add a section for the INFINIDAT driver back end.
- Under the `[DEFAULT]` section, set the `enabled_share_backends` parameter with the name of the new back-end section.

Configure the driver back-end section with the parameters below.

- Configure the driver name by setting the following parameter:

```
share_driver = manila.share.drivers.infinidat.infinibox.
↪InfiniboxShareDriver
```

- Configure the management IP of the InfiniBox array by adding the following parameter:

```
infinibox_hostname = InfiniBox management IP
```

- Configure SSL support for InfiniBox management API:

  We recommend enabling SSL support for InfiniBox management API. Refer to the InfiniBox manuals for details on security management. Configure SSL options by adding the following parameters:

```
infinidat_use_ssl = true/false
infinidat_suppress_ssl_warnings = true/false
```

  These parameters defaults to `false`.

- Configure user credentials:

  The driver requires an InfiniBox user with administrative privileges. We recommend creating a dedicated OpenStack user account that holds an administrative user role. Refer to the InfiniBox manuals for details on user account management. Configure the user credentials by adding the following parameters:

```
infinibox_login = Infinibox management login
infinibox_password = Infinibox management password
```

- Configure the name of the InfiniBox pool by adding the following parameter:

```
infinidat_pool_name = Pool as defined in the InfiniBox
```

- Configure the name of the InfiniBox NAS network space by adding the following parameter:

```
infinidat_nas_network_space_name = Network space as defined in the␣
↪InfiniBox
```

- The back-end name is an identifier for the back end. We recommend using the same name as the name of the section. Configure the back-end name by adding the following parameter:

```
share_backend_name = back-end name
```

- Thin provisioning:

  The INFINIDAT driver supports creating thin or thick provisioned filesystems. Configure thin or thick provisioning by adding the following parameter:

```
infinidat_thin_provision = true/false
```

  This parameter defaults to `true`.

- Controls access to the `.snapshot` directory:

```
infinidat_snapdir_accessible = true/false
```

  By default, each share allows access to its own `.snapshot` directory, which contains files and directories of each snapshot taken. To restrict access to the `.snapshot` directory on the client side, this option should be set to `false`.

  This parameter defaults to `true`.

- Controls visibility of the `.snapshot` directory:

```
infinidat_snapdir_visible = true/false
```

  By default, each share contains the `.snapshot` directory, which is hidden on the client side. To make the `.snapshot` directory visible, this option should be set to `true`.

  This parameter defaults to `false`.

Configuration example

```
[DEFAULT]
enabled_share_backends = infinidat-pool-a

[infinidat-pool-a]
share_driver = manila.share.drivers.infinidat.infinibox.InfiniboxShareDriver
share_backend_name = infinidat-pool-a
driver_handles_share_servers = false
infinibox_hostname = 10.1.2.3
infinidat_use_ssl = true
infinidat_suppress_ssl_warnings = true
infinibox_login = openstackuser
infinibox_password = openstackpass
infinidat_pool_name = pool-a
infinidat_nas_network_space_name = nas_space
infinidat_thin_provision = true
infinidat_snapdir_accessible = true
infinidat_snapdir_visible = false
```

Driver options

Configuration options specific to this driver:

Table 26: Description of INFINIDAT InfiniBox share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`infinibox_hostname = None`	(String) The name (or IP address) for the INFINIDAT Infinibox storage system.
`infinidat_use_ssl = False`	(Boolean) Enable SSL communication to access the INFINIDAT Infinibox storage system.
`infinidat_suppress_ssl_warnings = False`	(Boolean) Suppress requests library SSL certificate warnings.
`infinibox_login = None`	(String) Administrative user account name used to access the INFINIDAT Infinibox storage system.
`infinibox_password = None`	(String) Password for the administrative user account specified in the infinibox_login option.
`infinidat_pool_name = None`	(String) Name of the pool from which volumes are allocated.
`infinidat_nas_network_space_name = None`	(String) Name of the NAS network space on the INFINIDAT InfiniBox.
`infinidat_thin_provisioning = True`	(Boolean) Use thin provisioning.
`infinidat_snapdir_accessible = True`	(Boolean) Enables access to the `.snapshot` directory. By default, each share allows access to its own `.snapshot` directory, which contains files and directories of each snapshot taken. To restrict access to the `.snapshot` directory, this option should be set to `False`.
`infinidat_snapdir_visible = False`	(Boolean) Controls visibility of the `.snapshot` directory. By default, each share contains the `.snapshot` directory, which is hidden on the client side. To make the `.snapshot` directory visible, this option should be set to `True`.

Infortrend Manila driver

The Infortrend Manila driver provides NFS and CIFS shared file systems to OpenStack.

Requirements

To use the Infortrend Manila driver, the following items are required:

- GS/GSe Family firmware version v73.1.0-4 and later.
- Configure at least one channel for shared file systems.

Supported shared filesystems and operations

This driver supports NFS and CIFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
  - Only user access type is supported for CIFS.
- Deny share access.
- Manage a share.
- Unmanage a share.
- Extend a share.
- Shrink a share.

Restrictions

The Infortrend manila driver has the following restrictions:

- Only IP access type is supported for NFS.
- Only user access type is supported for CIFS.
- Only file-level data service channel can offer the NAS service.

Driver configuration

On `manila-share` nodes, set the following in your `/etc/manila/manila.conf`, and use the following options to configure it:

Driver options

Table 27: Description of Infortrend Manila driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
infortrend_nas_ip = None	(String) Infortrend NAS ip. It is the ip for management.
infortrend_nas_user = manila	(String) Infortrend NAS username.
infortrend_nas_password = None	(String) Password for the Infortrend NAS server. This is not necessary if infortrend_nas_ssh_key is set.
infortrend_nas_ssh_key = None	(String) SSH key for the Infortrend NAS server. This is not necessary if infortrend_nas_password is set.
infortrend_share_pools = None	(String) Infortrend nas pool name list. It is separated with comma.
infortrend_share_channels = None	(String) Infortrend channels for file service. It is separated with comma.
infortrend_cli_timeout = 30	(Integer) CLI timeout in seconds.

Back-end configuration example

```
[DEFAULT]
enabled_share_backends = ift-manila
enabled_share_protocols = NFS, CIFS

[ift-manila]
share_backend_name = ift-manila
share_driver = manila.share.drivers.infortrend.driver.InfortrendNASDriver
driver_handles_share_servers = False
infortrend_nas_ip = FAKE_IP
infortrend_nas_user = FAKE_USER
infortrend_nas_password = FAKE_PASS
infortrend_share_pools = pool-1, pool-2
infortrend_share_channels = 0, 1
```

MapRFS native driver

MapR-FS native driver is a plug-in based on the Shared File Systems service and provides high-throughput access to the data on MapR-FS distributed file system, which is designed to hold very large amounts of data.

A Shared File Systems service share in this driver is a volume in MapR-FS. Instances talk directly to the MapR-FS storage backend via the (mapr-posix) client. To mount a MapR-FS volume, the MapR POSIX client is required. Access to each share is allowed by user and group based access type, which is aligned

with MapR-FS ACEs to support access control for multiple users and groups. If user name and group name are the same, the group access type will be used by default.

For more details, see MapR documentation.

Network configuration

The storage backend and Shared File Systems service hosts should be in a flat network. Otherwise, the L3 connectivity between them should exist.

Supported shared filesystems and operations

The driver supports MapR-FS shares.

The following operations are supported:

- Create MapR-FS share.
- Delete MapR-FS share.
- Allow MapR-FS Share access.
  - Only support user and group access type.
  - Support level of access (ro/rw).
- Deny MapR-FS Share access.
- Update MapR-FS Share access.
- Create snapshot.
- Delete snapshot.
- Create share from snapshot.
- Extend share.
- Shrink share.
- Manage share.
- Unmanage share.
- Manage snapshot.
- Unmanage snapshot.
- Ensure share.

Requirements

- Install MapR core packages, version >= 5.2.x, on the storage backend.
- To enable snapshots, the MapR cluster should have at least M5 license.
- Establish network connection between the Shared File Systems service hosts and storage backend.
- Obtain a ticket for user who will be used to access MapR-FS.

Back end configuration (manila.conf)

Add MapR-FS protocol to `enabled_share_protocols`:

```
enabled_share_protocols = MAPRFS
```

Create a section for MapR-FS backend. Example:

```
[maprfs]
driver_handles_share_servers = False
share_driver =
manila.share.drivers.maprfs.maprfs_native.MapRFSNativeShareDriver
maprfs_clinode_ip = example
maprfs_ssh_name = mapr
maprfs_ssh_pw = mapr
share_backend_name = maprfs
```

Set `driver-handles-share-servers` to `False` as the driver does not manage the lifecycle of `share-servers`.

Add driver backend to `enabled_share_backends`:

```
enabled_share_backends = maprfs
```

Driver options

The following table contains the configuration options specific to this driver.

Table 28: Description of MapRFS share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
maprfs_base_volume_dir = /	(String) Path in MapRFS where share volumes must be created.
maprfs_cldb_ip = None	(List) The list of IPs or hostnames of CLDB nodes.
maprfs_clinode_ip = None	(List) The list of IPs or hostnames of nodes where mapr-core is installed.
maprfs_rename_managed_volume = True	(Boolean) Specify whether existing volume should be renamed when start managing.
maprfs_ssh_name = mapr	(String) Cluster admin user ssh login name.
maprfs_ssh_port = 22	(Port number) CLDB node SSH port.
maprfs_ssh_private_key = None	(String) Path to SSH private key for login.
maprfs_ssh_pw = None	(String) Cluster node SSH login password, This parameter is not necessary, if maprfs_ssh_private_key is configured.
maprfs_zookeeper_ip = None	(List) The list of IPs or hostnames of ZooKeeper nodes.

Known restrictions

This driver does not handle user authentication, no tickets or users are created by this driver. This means that when access_allow or update_access is calling, this will have no effect without providing tickets to users.

Share metadata

MapR-FS shares can be created by specifying additional options. Metadata is used for this purpose. Every metadata option with - prefix is passed to MapR-FS volume. For example, to specify advisory volume quota add `_advisoryquota=10G` option to metadata:

```
$ manila create MAPRFS 1 --metadata _advisoryquota=10G
```

If you need to create a share with your custom backend name or export location instead if uuid, you can specify `_name` and `_path` options:

```
$ manila create MAPRFS 1 --metadata _name=example _path=/example
```

Warning: Specifying invalid options will cause an error.

The list of allowed options depends on mapr-core version. See volume create for more information.

NetApp Clustered Data ONTAP driver

The Shared File Systems service can be configured to use NetApp clustered Data ONTAP version 8.

Network approach

L3 connectivity between the storage cluster and Shared File Systems service host should exist, and VLAN segmentation should be configured.

The clustered Data ONTAP driver creates storage virtual machines (SVM, previously known as vServers) as representations of the Shared File Systems service share server interface, configures logical interfaces (LIFs) and stores shares there.

Supported shared filesystems and operations

The driver supports CIFS and NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported for NFS.
  - Only user access type is supported for CIFS.
- Deny share access.
- Create a snapshot.
- Delete a snapshot.
- Create a share from a snapshot.
- Extend a share.
- Shrink a share.
- Create a consistency group.
- Delete a consistency group.
- Create a consistency group snapshot.
- Delete a consistency group snapshot.

Required licenses

- NFS
- CIFS
- FlexClone

Known restrictions

- For CIFS shares an external active directory service is required. Its data should be provided via security-service that is attached to used share-network.
- Share access rule by user for CIFS shares can be created only for existing user in active directory.
- To be able to configure clients to security services, the time on these external security services and storage should be synchronized. The maximum allowed clock skew is 5 minutes.

Driver options

The following table contains the configuration options specific to the share driver.

Table 29: Description of NetApp share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
netapp_aggregate_name_search_pattern = (.*)	(String) Pattern for searching available aggregates for provisioning.
netapp_enabled_share_protocols = nfs3, nfs4.0	(List) The NFS protocol versions that will be enabled. Supported values include nfs3, nfs4.0, nfs4.1. This option only applies when the option driver_handles_share_servers is set to True.
netapp_lif_name_template = os_%(net_allocation_id)s	(String) Logical interface (LIF) name template
netapp_login = None	(String) Administrative user account name used to access the storage system.
netapp_password = None	(String) Password for the administrative user account specified in the netapp_login option.
netapp_port_name_search_pattern = (.*)	(String) Pattern for overriding the selection of network ports on which to create Vserver LIFs.
netapp_root_volume = root	(String) Root volume name.
netapp_root_volume_aggregate = None	(String) Name of aggregate to create Vserver root volumes on. This option only applies when the option driver_handles_share_servers is set to True.
netapp_server_hostname = None	(String) The hostname (or IP address) for the storage system.
netapp_server_port = None	(Port number) The TCP port to use for communication with the storage system or proxy server. If not specified, Data ONTAP drivers will use 80 for HTTP and 443 for HTTPS.
netapp_snapmirror_quiesce_timeout = 3600	(Integer) The maximum time in seconds to wait for existing snapmirror transfers to complete before aborting when promoting a replica.
netapp_storage_family = ontap_cluster	(String) The storage family type used on the storage system; valid values include ontap_cluster for using clustered Data ONTAP.
netapp_trace_flags = None	(String) Comma-separated list of options that control which trace info is written to the debug logs. Values include method and api.
netapp_transport_type = http	(String) The transport protocol used when communicating with the storage system or proxy server. Valid values are http or https.
netapp_volume_move_cutover_timeout = 3600	(Integer) The maximum time in seconds to wait for the completion of a volume move operation after the cutover was triggered.
netapp_volume_name_template = share_%(share_id)s	(String) NetApp volume name template.
netapp_volume_snapshot_reserve_percent = 5	(Integer) The percentage of share space set aside as reserve for snapshot usage; valid values range from 0 to 90.
netapp_vserver_name_template = os_%s	(String) Name template to use for new Vserver. When using CIFS protocol make sure to not configure characters illegal in DNS hostnames.

Quobyte Driver

Quobyte can be used as a storage back end for the OpenStack Shared File System service. Shares in the Shared File System service are mapped 1:1 to Quobyte volumes. Access is provided via NFS protocol and IP-based authentication. The Quobyte driver uses the Quobyte API service.

Supported shared filesystems and operations

The drivers supports NFS shares.

The following operations are supported:

- Create a share.
- Delete a share.
- Allow share access.

  Note the following limitations:

  - Only IP access type is supported.
- Deny share access.

Driver options

The following table contains the configuration options specific to the share driver.

Table 30: Description of Quobyte share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
quobyte_api_ca = None	(String) The X.509 CA file to verify the server cert.
quobyte_api_password = quobyte	(String) Password for Quobyte API server
quobyte_api_url = None	(String) URL of the Quobyte API server (http or https)
quobyte_api_username = admin	(String) Username for Quobyte API server.
quobyte_default_volume_group = root	(String) Default owning group for new volumes.
quobyte_default_volume_user = root	(String) Default owning user for new volumes.
quobyte_delete_shares = False	(Boolean) Actually deletes shares (vs. unexport)
quobyte_volume_configuration = BASE	(String) Name of volume configuration used for new shares.

Configuration

To configure Quobyte access for the Shared File System service, a back end configuration section has to be added in the `manila.conf` file. Add the name of the configuration section to `enabled_share_backends` in the `manila.conf` file. For example, if the section is named `Quobyte`:

```
enabled_share_backends = Quobyte
```

Create the new back end configuration section, in this case named `Quobyte`:

```
[Quobyte]

share_driver = manila.share.drivers.quobyte.quobyte.QuobyteShareDriver
share_backend_name = QUOBYTE
quobyte_api_url = http://api.myserver.com:1234/
quobyte_delete_shares = False
quobyte_volume_configuration = BASE
quobyte_default_volume_user = myuser
quobyte_default_volume_group = mygroup
```

The section name must match the name used in the `enabled_share_backends` option described above. The `share_driver` setting is required as shown, the other options should be set according to your local Quobyte setup.

Other security-related options are:

```
quobyte_api_ca = /path/to/API/server/verification/certificate
quobyte_api_username = api_user
quobyte_api_password = api_user_pwd
```

Quobyte support can be found at the Quobyte support webpage.

NexentaStor5 Driver

Nexentastor5 can be used as a storage back end for the OpenStack Shared File System service. Shares in the Shared File System service are mapped 1:1 to Nexentastor5 filesystems. Access is provided via NFS protocol and IP-based authentication.

Network approach

L3 connectivity between the storage back end and the host running the Shared File Systems share service should exist.

Supported shared filesystems and operations

The drivers supports NFS shares.

The following operations are supported:

- Create NFS share
- Delete share
- Extend share
- Shrink share
- Allow share access

  Note the following limitation:

  - Only IP based access is supported (ro/rw).
- Deny share access
- Create snapshot
- Revert to snapshot
- Delete snapshot
- Create share from snapshot
- Manage share
- Unmanage share

Requirements

- NexentaStor 5.x Appliance pre-provisioned and licensed
- Pool and parent filesystem configured (this filesystem will contain all manila shares)

Restrictions

- Only IP share access control is allowed for NFS shares.

Configuration

```
enabled_share_backends = NexentaStor5
```

Create the new back end configuration section, in this case named `NexentaStor5`:

```
[NexentaStor5]

 share_backend_name = NexentaStor5
 driver_handles_share_servers = False
 nexenta_folder = manila
```

(continues on next page)

(continued from previous page)

```
share_driver = manila.share.drivers.nexenta.ns5.nexenta_nas.NexentaNasDriver
nexenta_rest_addresses = 10.3.1.1,10.3.1.2
nexenta_nas_host = 10.3.1.10
nexenta_rest_port = 8443
nexenta_pool = pool1
nexenta_nfs = True
nexenta_user = admin
nexenta_password = secret_password
nexenta_thin_provisioning = True
```

More information can be found at the *Nexenta documentation webpage <https://nexenta.github.io>*.

Driver options

The following table contains the configuration options specific to the share driver.

Table 31: Description of NexentaStor5 configuration options

Configuration option = Default value	Description
[DEFAULT]	
nexenta_rest_addresses = None	(List) One or more comma delimited IP addresses for management communication with NexentaStor appliance.
nexenta_rest_port = 8443	(Integer) Port to connect to Nexenta REST API server.
nexenta_use_https = True	(Boolean) Use HTTP secure protocol for NexentaStor management REST API connections.
nexenta_user = admin	(String) User name to connect to Nexenta SA.
nexenta_password = None	(String) Password to connect to Nexenta SA.
nexenta_pool = pool1	(String) Pool name on NexentaStor.
nexenta_nfs = True	(Boolean) Defines whether share over NFS is enabled.
nexenta_ssl_cert_verify = False	(Boolean) Defines whether the driver should check ssl cert.
nexenta_rest_connect_timeout = 30	(Float) Specifies the time limit (in seconds), within which the connection to NexentaStor management REST API server must be established.
nexenta_rest_read_timeout = 300	(Float) Specifies the time limit (in seconds), within which NexentaStor management REST API server must send a response.
nexenta_rest_backoff_factor = 1	(Float) Specifies the backoff factor to apply between connection attempts to NexentaStor management REST API server.
nexenta_rest_retry_count = 5	(Integer) Specifies the number of times to repeat NexentaStor management REST API call in case of connection errors and NexentaStor appliance EBUSY or ENOENT errors.
nexenta_nas_host = None	(Hostname) Data IP address of Nexenta storage appliance.
nexenta_mount_point_base = $state_path/mnt	(String) Base directory that contains NFS share mount points.
nexenta_share_name_prefix = share-	(String) Nexenta share name prefix.
nexenta_folder = folder	(String) Parent folder on NexentaStor.
nexenta_dataset_compression = on	(String) Compression value for new ZFS folders.
nexenta_thin_provisioning = True	(Boolean) If True shares will not be space guaranteed and overprovisioning will be enabled.
nexenta_dataset_record_size = 131072	(Integer) Specifies a suggested block size in for files in a file system. (bytes)

Pure Storage FlashBlade driver

The Pure Storage FlashBlade driver provides support for managing filesystem shares on the Pure Storage FlashBlade storage systems.

The driver is compatible with Pure Storage FlashBlades that support REST API version 1.6 or higher (Purity//FB v2.3.0 or higher). This section explains how to configure the FlashBlade driver.

Supported operations

- Create and delete NFS shares.
- Extend/Shrink a share.
- Create and delete filesystem snapshots (No support for create-from or mount).
- Revert to Snapshot.
- Both RW and RO access levels are supported.
- Set access rights to NFS shares.

  Note the following limitations:

  - Only IP (for NFS shares) access types are supported.

External package installation

The driver requires the `purity_fb` package for communicating with FlashBlade systems. Install the package from PyPI using the following command:

```
$ pip install purity_fb
```

Driver configuration

Edit the `manila.conf` file, which is usually located under the following path `/etc/manila/manila.conf`.

- Add a section for the FlashBlade driver back end.
- Under the `[DEFAULT]` section, set the `enabled_share_backends` parameter with the name of the new back-end section.

Configure the driver back-end section with the parameters below.

- Configure the driver name by setting the following parameter:

  ```
  share_driver = manila.share.drivers.purestorage.flashblade.
  ↪FlashBladeShareDriver
  ```

- Configure the management and data VIPs of the FlashBlade array by adding the following parameters:

```
flashblade_mgmt_vip = FlashBlade management VIP
flashblade_data_vip = FlashBlade data VIP
```

- Configure user credentials:

  The driver requires a FlashBlade user with administrative privileges. We recommend creating a dedicated OpenStack user account that holds an administrative user role. Refer to the FlashBlade manuals for details on user account management. Configure the user credentials by adding the following parameters:

```
flashblade_api = FlashBlade API token for admin-privileged user
```

- (Optional) Configure File System and Snapshot Eradication:

  The option, when enabled, all FlashBlade file systems and snapshots will be eradicated at the time of deletion in Manila. Data will NOT be recoverable after a delete with this set to True! When disabled, file systems and snapshots will go into pending eradication state and can be recovered. Recovery of these pending eradication snapshots cannot be accomplished through Manila. These snapshots will self-eradicate after 24 hours unless manually restored. The default setting is True.

```
flashblade_eradicate = { True | False }
```

- The back-end name is an identifier for the back end. We recommend using the same name as the name of the section. Configure the back-end name by adding the following parameter:

```
share_backend_name = back-end name
```

Configuration example

```
[DEFAULT]
enabled_share_backends = flashblade-1

[flashblade-1]
share_driver = manila.share.drivers.purestorage.flashblade.
↪FlashBladeShareDriver
share_backend_name = flashblade-1
driver_handles_share_servers = false
flashblade_mgmt_vip = 10.1.2.3
flashblade_data_vip = 10.1.2.4
flashblade_api = pureuser API
```

Driver options

Configuration options specific to this driver:

Table 32: Description of Pure Storage FlashBlade share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
`flashblade_mgmt_vip = None`	(String) The name (or IP address) for the Pure Storage FlashBlade storage system management port.
`flashblade_data_vip = None`	(String) The name (or IP address) for the Pure Storage FlashBlade storage system data port.
`flashblade_api = None`	(String) API token for an administrative level user account.
`flashblade_eradicate = True`	(Boolean) Enable or disable filesystem and snapshot eradication on delete.

To use different share drivers for the Shared File Systems service, use the parameters described in these sections.

The Shared File Systems service can handle multiple drivers at once. The configuration for all of them follows a common paradigm:

1. In the configuration file `manila.conf`, configure the option `enabled_backends` with the list of names for your configuration.

   For example, if you want to enable two drivers and name them `Driver1` and `Driver2`:

   ```
   [Default]
   # ...
   enabled_backends = Driver1 Driver2
   ```

2. Configure a separate section for each driver using these names. You need to define in each section at least the option `share_driver` and assign it the value of your driver. In this example it is the generic driver:

   ```
   [Driver1]
   share_driver = manila.share.drivers.generic.GenericShareDriver
   # ...

   [Driver2]
   share_driver = manila.share.drivers.generic.GenericShareDriver
   # ...
   ```

The share drivers are included in the Shared File Systems repository.

Log files used by Shared File Systems

The corresponding log file of each Shared File Systems service is stored in the `/var/log/manila/` directory of the host on which each service runs.

Table 33: Log files used by Shared File Systems services

Log file	Service/interface (for CentOS, Fedora, openSUSE, Red Hat Enterprise Linux, and SUSE Linux Enterprise)	Service/interface (for Ubuntu and Debian)
`api.log`	`openstack-manila-api`	`manila-api`
`manila-manage.log`	`manila-manage`	`manila-manage`
`scheduler.log`	`openstack-manila-scheduler`	`manila-scheduler`
`share.log`	`openstack-manila-share`	`manila-share`
`data.log`	`openstack-manila-data`	`manila-data`

Additional options

These options can also be set in the `manila.conf` file.

Table 34: Description of Certificate Authority configuration options

Configuration option = Default value	Description
[DEFAULT]	
`osapi_share_use_ssl = False`	(Boolean) Wraps the socket in a SSL context if True is set.

Configuration option = Default value	Description
[DEFAULT]	
`check_hash = False`	(Boolean) Chooses whether hash of each file should be
`client_socket_timeout = 900`	(Integer) Timeout for client connections socket operatio
`compute_api_class = manila.compute.nova.API`	(String) The full class name of the Compute API class t
`data_access_wait_access_rules_timeout = 180`	(Integer) Time to wait for access rules to be allowed/de
`data_manager = manila.data.manager.DataManager`	(String) Full class name for the data manager.
`data_node_access_admin_user = None`	(String) The admin user name registered in the security
`data_node_access_cert = None`	(String) The certificate installed in the data node in ord
`data_node_access_ips = None`	(String) A list of the IPs of the node interface connecte
`data_node_mount_options = {}`	(Dict) Mount options to be included in the mount comm
`data_topic = manila-data`	(String) The topic data nodes listen on.
`enable_new_services = True`	(Boolean) Services to be added to the available pool on
`fatal_exception_format_errors = False`	(Boolean) Whether to make exception message format e
`filter_function = None`	(String) String representation for an equation that will b
`host = <your_hostname>`	(String) Name of this node. This can be an opaque iden
`max_over_subscription_ratio = 20.0`	(Floating point) Float representation of the over subscri
`memcached_servers = None`	(List) Memcached servers or None for in process cache
`monkey_patch = False`	(Boolean) Whether to log monkey patching.

Configuration option = Default value	Description
monkey_patch_modules =	(List) List of modules or decorators to monkey patch.
mount_tmp_location = /tmp/	(String) Temporary path to create and mount shares dur
my_ip = <your_ip>	(String) IP address of this host.
num_shell_tries = 3	(Integer) Number of times to attempt to run flakey shell
periodic_fuzzy_delay = 60	(Integer) Range of seconds to randomly delay when sta
periodic_hooks_interval = 300.0	(Floating point) Interval in seconds between execution
periodic_interval = 60	(Integer) Seconds between running periodic tasks.
replica_state_update_interval = 300	(Integer) This value, specified in seconds, determines h
replication_domain = None	(String) A string specifying the replication domain that
report_interval = 10	(Integer) Seconds between nodes reporting state to data
reserved_share_percentage = 0	(Integer) The percentage of backend capacity reserved.
rootwrap_config = None	(String) Path to the rootwrap configuration file to use fo
service_down_time = 60	(Integer) Maximum time since last check-in for up servi
smb_template_config_path = $state_path/smb.conf	(String) Path to smb config.
sql_idle_timeout = 3600	(Integer) Timeout before idle SQL connections are reap
sql_max_retries = 10	(Integer) Maximum database connection retries during
sql_retry_interval = 10	(Integer) Interval between retries of opening a SQL con
sqlite_clean_db = clean.sqlite	(String) File name of clean sqlite database.
sqlite_db = manila.sqlite	(String) The filename to use with sqlite.
sqlite_synchronous = True	(Boolean) If passed, use synchronous mode for sqlite.
state_path = /var/lib/manila	(String) Top-level directory for maintaining manilas sta
storage_availability_zone = nova	(String) Availability zone of this node.
tcp_keepalive = True	(Boolean) Sets the value of TCP_KEEPALIVE (True/F
tcp_keepalive_count = None	(Integer) Sets the value of TCP_KEEPCNT for each ser
tcp_keepalive_interval = None	(Integer) Sets the value of TCP_KEEPINTVL in secon
tcp_keepidle = 600	(Integer) Sets the value of TCP_KEEPIDLE in seconds
until_refresh = 0	(Integer) Count of reservations until usage is refreshed.
use_forwarded_for = False	(Boolean) Treat X-Forwarded-For as the canonical rem
wsgi_keep_alive = True	(Boolean) If False, closes the client socket connection e
[coordination]	
backend_url = file://$state_path	(String) The back end URL to use for distributed coord
[healthcheck]	
backends =	(List) Additional backends that can perform health chec
detailed = False	(Boolean) Show more detailed information as part of th
disable_by_file_path = None	(String) Check the presence of a file to determine if an
disable_by_file_paths =	(List) Check the presence of a file based on a port to de
path = /healthcheck	(String) DEPRECATED: The path to respond to healtch

Table 36: Description of Compute configuration options

Configuration option = Default value	Description
[nova]	
api_microversion = 2.10	(String) Version of Nova API to be used.
endpoint_type = publicURL	(String) Endpoint type to be used with nova client calls.
region_name = None	(String) Region name for connecting to nova.

Table 37: Description of Ganesha configuration options

Configuration option = Default value	Description
[DEFAULT]	
ganesha_config_dir = /etc/ganesha	(String) Directory where Ganesha config files are stored.
ganesha_config_path = $ganesha_config_dir/ganesha.conf	(String) Path to main Ganesha config file.
ganesha_db_path = $state_path/manila-ganesha.db	(String) Location of Ganesha database file. (Ganesha module only.)
ganesha_export_dir = $ganesha_config_dir/export.d	(String) Path to directory containing Ganesha export configuration. (Ganesha module only.)
ganesha_export_template_dir = /etc/manila/ganesha-export-templ.d	(String) Path to directory containing Ganesha export block templates. (Ganesha module only.)
ganesha_service_name = ganesha.nfsd	(String) Name of the ganesha nfs service.

Table 38: Description of hnas configuration options

Configuration option = Default value	Description
[DEFAULT]	
hitachi_hnas_driver_helper = manila.share.drivers.hitachi.hnas.ssh.HNASSSHBackend	(String) Python class to be used for driver helper.

Table 39: Description of Quota configuration options

Configuration option = Default value	Description
[DEFAULT]	
max_age = 0	(Integer) Number of seconds between subsequent usage refreshes.
max_gigabytes = 10000	(Integer) Maximum number of volume gigabytes to allow per host.
quota_driver = manila.quota.DbQuotaDriver	(String) Default driver to use for quota checks.
quota_gigabytes = 1000	(Integer) Number of share gigabytes allowed per project.
quota_share_networks = 10	(Integer) Number of share-networks allowed per project.
quota_shares = 50	(Integer) Number of shares allowed per project.
quota_snapshot_gigabytes = 1000	(Integer) Number of snapshot gigabytes allowed per project.
quota_snapshots = 50	(Integer) Number of share snapshots allowed per project.
quota_share_groups = 50	(Integer) Number of share groups allowed.
quota_share_group_snapshots = 50	(Integer) Number of share group snapshots allowed.
reservation_expire = 86400	(Integer) Number of seconds until a reservation expires.

Table 40: Description of Redis configuration options

Configuration option = Default value	Description
[matchmaker_redis]	
check_timeout = 20000	(Integer) Time in ms to wait before the transaction is killed.
host = 127.0.0.1	(String) DEPRECATED: Host to locate redis. Replaced by [DEFAULT]/transport_url
password =	(String) DEPRECATED: Password for Redis server (optional). Replaced by [DEFAULT]/transport_url
port = 6379	(Port number) DEPRECATED: Use this port to connect to redis host. Replaced by [DEFAULT]/transport_url
sentinel_group_name = oslo-messaging-zeromq	(String) Redis replica set name.
sentinel_hosts =	(List) DEPRECATED: List of Redis Sentinel hosts (fault tolerance mode), e.g., [host:port, host1:port] Replaced by [DEFAULT]/transport_url
socket_timeout = 10000	(Integer) Timeout in ms on blocking socket operations.
wait_timeout = 2000	(Integer) Time in ms to wait between connection attempts.

Table 41: Description of SAN configuration options

Configuration option = Default value	Description
[DEFAULT]	
ssh_conn_timeout = 60	(Integer) Backend server SSH connection timeout.
ssh_max_pool_conn = 10	(Integer) Maximum number of connections in the SSH pool.
ssh_min_pool_conn = 1	(Integer) Minimum number of connections in the SSH pool.

Table 42: Description of Scheduler configuration options

Configuration option = Default value	Description
[DEFAULT]	
capacity_weight_multiplier = 1.0	(Floating point) Multiplier used for weighing share capacity. Negative numbers mean to stack vs spread.
pool_weight_multiplier = 1.0	(Floating point) Multiplier used for weighing pools which have existing share servers. Negative numbers mean to spread vs stack.
scheduler_default_filters = AvailabilityZoneFilter, CapacityFilter, CapabilitiesFilter, DriverFilter, ShareReplicationFilter	(List) Which filter class names to use for filtering hosts when not specified in the request.
scheduler_default_weighers = CapacityWeigher, GoodnessWeigher	(List) Which weigher class names to use for weighing hosts.
scheduler_driver = manila.scheduler.drivers.filter.FilterScheduler	(String) Default scheduler driver to use.
scheduler_host_manager = manila.scheduler.host_manager.HostManager	(String) The scheduler host manager class to use.
scheduler_json_config_location =	(String) Absolute path to scheduler configuration JSON file.
scheduler_manager = manila.scheduler.manager.SchedulerManager	(String) Full class name for the scheduler manager.
scheduler_max_attempts = 3	(Integer) Maximum number of attempts to schedule a share.
scheduler_topic = manila-scheduler	(String) The topic scheduler nodes listen on.

Table 43: Description of Share configuration options

Configuration option = Default value	Description
[DEFAULT]	
automatic_share_server_cleanup = True	(Boolean) If set to True, then Manila will delete all share servers which were unused more than specified time .If set to False - automatic deletion of share servers will be disabled.
backlog = 4096	(Integer) Number of backlog requests to configure the socket with.
default_share_group_type = None	(String) Default share group type to use.
default_share_type = None	(String) Default share type to use.
delete_share_server_with_last_share = False	(Boolean) Whether share servers will be deleted on deletion of the last share.
driver_handles_share_servers = None	(Boolean) There are two possible approaches for share drivers in Manila. First is when share driver is able to handle share-servers and second when not. Drivers can support either both or only one of these approaches. So, set this opt to True if share driver is able to handle share servers and it is desired mode else set False. It is set to None by default to make this choice intentional.
enable_periodic_hooks = False	(Boolean) Whether to enable periodic hooks or not.
enable_post_hooks = False	(Boolean) Whether to enable post hooks or not.
enable_pre_hooks = False	(Boolean) Whether to enable pre hooks or not.
enabled_share_backends = None	(List) A list of share backend names to use. These backend names should be backed by a unique [CONFIG] group with its options.
enabled_share_protocols = NFS, CIFS	(List) Specify list of protocols to be allowed for share creation. Available values are (NFS, CIFS, GLUSTERFS, HDFS, CEPHFS, MAPRFS)
executor_thread_pool_size = 64	(Integer) Size of executor thread pool.
hook_drivers =	(List) Driver(s) to perform some additional actions before and after share driver actions and on a periodic basis. Default is [].
migration_create_delete_share_timeout = 300	(Integer) Time to wait for creating and deleting share instances when performing share migration (seconds).
migration_driver_continue_update_interval = 60	(Integer) This value, specified in seconds, determines how often the share manager will poll the driver to perform the next step of migration in the storage backend, for a migrating share.
migration_ignore_files = lost+found	(List) List of files and folders to be ignored when migrating shares. Items should be names (not including any path).
migration_readonly_rules_support = True	(Boolean) DEPRECATED: Specify whether read only access rule mode is supported in this backend. Obsolete. All drivers are now required to support read-only access rules.
migration_wait_access_rules_timeout = 180	(Integer) Time to wait for access rules to be allowed/denied on backends when migrating shares using generic approach (seconds).
network_config_group = None	(String) Name of the configuration group in the Manila conf file to look for network config options.If not set, the share backends config group will be used.If an option is not found within provided group, thenDEFAULT group will be used for search of option.
share_manager = manila.	(String) Full class name for the share manager.

Table 44: Description of Tegile share driver configuration options

Configuration option = Default value	Description
[DEFAULT]	
tegile_default_project = None	(String) Create shares in this project
tegile_nas_login = None	(String) User name for the Tegile NAS server.
tegile_nas_password = None	(String) Password for the Tegile NAS server.
tegile_nas_server = None	(String) Tegile NAS server hostname or IP address.

Table 45: Description of WinRM configuration options

Configuration option = Default value	Description
[DEFAULT]	
winrm_cert_key_pem_path = ~/.ssl/key.pem	(String) Path to the x509 certificate key.
winrm_cert_pem_path = ~/.ssl/cert.pem	(String) Path to the x509 certificate used for accessing the serviceinstance.
winrm_conn_timeout = 60	(Integer) WinRM connection timeout.
winrm_operation_timeout = 60	(Integer) WinRM operation timeout.
winrm_retry_count = 3	(Integer) WinRM retry count.
winrm_retry_interval = 5	(Integer) WinRM retry interval in seconds
winrm_use_cert_based_auth = False	(Boolean) Use x509 certificates in order to authenticate to theservice instance.

Shared File Systems service sample configuration files

All the files in this section can be found in `/etc/manila`.

manila.conf

The `manila.conf` file is installed in `/etc/manila` by default. When you manually install the Shared File Systems service, the options in the `manila.conf` file are set to default values.

The `manila.conf` file contains most of the options needed to configure the Shared File Systems service.

See the online version of this documentation for the full config file example.

api-paste.ini

The shared file systems service stores its API configuration settings in the `api-paste.ini` file.

```
#############
# OpenStack #
#############

[composite:osapi_share]
use = call:manila.api:root_app_factory
```

(continues on next page)

(continued from previous page)

```
/: apiversions
/healthcheck: healthcheck
/v1: openstack_share_api
/v2: openstack_share_api_v2

[composite:openstack_share_api]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler noauth api
keystone = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler authtoken␣
↪keystonecontext api
keystone_nolimit = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler␣
↪authtoken keystonecontext api

[composite:openstack_share_api_v2]
use = call:manila.api.middleware.auth:pipeline_factory
noauth = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler noauth apiv2
noauthv2 = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler noauthv2␣
↪apiv2
keystone = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler authtoken␣
↪keystonecontext apiv2
keystone_nolimit = cors faultwrap http_proxy_to_wsgi sizelimit osprofiler␣
↪authtoken keystonecontext apiv2

[filter:faultwrap]
paste.filter_factory = manila.api.middleware.fault:FaultWrapper.factory

[filter:noauth]
paste.filter_factory = manila.api.middleware.auth:NoAuthMiddleware.factory

[filter:noauthv2]
paste.filter_factory = manila.api.middleware.auth:NoAuthMiddlewarev2_60.
↪factory

[filter:sizelimit]
paste.filter_factory = oslo_middleware.sizelimit:RequestBodySizeLimiter.
↪factory

[filter:osprofiler]
paste.filter_factory = osprofiler.web:WsgiMiddleware.factory

[filter:http_proxy_to_wsgi]
paste.filter_factory = oslo_middleware.http_proxy_to_wsgi:HTTPProxyToWSGI.
↪factory

[app:api]
paste.app_factory = manila.api.v1.router:APIRouter.factory

[app:apiv2]
paste.app_factory = manila.api.v2.router:APIRouter.factory
```

(continues on next page)

(continued from previous page)

```
[pipeline:apiversions]
pipeline = cors faultwrap http_proxy_to_wsgi osshareversionapp

[app:osshareversionapp]
paste.app_factory = manila.api.versions:VersionsRouter.factory

##########
# Shared #
##########

[filter:keystonecontext]
paste.filter_factory = manila.api.middleware.auth:ManilaKeystoneContext.
↪factory

[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory

[filter:cors]
paste.filter_factory = oslo_middleware.cors:filter_factory
oslo_config_project = manila

[app:healthcheck]
paste.app_factory = oslo_middleware:Healthcheck.app_factory
backends = disable_by_file
disable_by_file_path = /etc/manila/healthcheck_disable
```

rootwrap.conf

The `rootwrap.conf` file defines configuration values used by the `rootwrap` script when the Shared File Systems service must escalate its privileges to those of the root user.

```
# Configuration for manila-rootwrap
# This file should be owned by (and only-writeable by) the root user

[DEFAULT]
# List of directories to load filter definitions from (separated by ',').
# These directories MUST all be only writeable by root !
filters_path=/etc/manila/rootwrap.d,/usr/share/manila/rootwrap

# List of directories to search executables in, in case filters do not
# explicitly specify a full path (separated by ',')
# If not specified, defaults to system PATH environment variable.
# These directories MUST all be only writeable by root !
exec_dirs=/sbin,/usr/sbin,/bin,/usr/bin,/usr/local/sbin,/usr/local/bin,/usr/
↪lpp/mmfs/bin

# Enable logging to syslog
```

(continues on next page)

(continued from previous page)

```
# Default value is False
use_syslog=False

# Which syslog facility to use.
# Valid values include auth, authpriv, syslog, user0, user1...
# Default value is 'syslog'
syslog_log_facility=syslog

# Which messages to log.
# INFO means log all usage
# ERROR means only log unsuccessful attempts
syslog_log_level=ERROR
```

Policy configuration

Warning: JSON formatted policy file is deprecated since Manila 12.0.0 (Wallaby). This oslopolicy-convert-json-to-yaml tool will migrate your existing JSON-formatted policy file to YAML in a backward-compatible way.

Configuration

See the online version of this documentation for the list of available policies in Manila.

Manila Sample Policy

Warning: JSON formatted policy file is deprecated since Manila 12.0.0 (Wallaby). This oslopolicy-convert-json-to-yaml tool will migrate your existing JSON-formatted policy file to YAML in a backward-compatible way.

The following is a sample Manila policy file that has been auto-generated from default policy values in code. If youre using the default policies, then the maintenance of this file is not necessary. It is here to help explain which policy operations protect specific Manila API, but it is not suggested to copy and paste into a deployment unless youre planning on providing a different policy for an operation that is not the default. For instance, if you want to change the default value of share:create, you only need to keep this single rule in your policy config file (**/etc/manila/policy.yaml**).

See the online version of this documentation for the sample file (`manila.policy.yaml.sample`).

The Shared File Systems service works with many different drivers that you can configure by using these instructions.

3.3.2 Command Line Interface

Shared File Systems service (manila) command-line client

The manila client is the command-line interface (CLI) for the Shared File Systems service (manila) API and its extensions.

This chapter documents **manila** version 1.16.0.

For help on a specific **manila** command, enter:

```
$ manila help COMMAND
```

manila usage

```
usage: manila [--version] [-d] [--os-cache] [--os-reset-cache]
              [--os-user-id <auth-user-id>] [--os-username <auth-user-name>]
              [--os-password <auth-password>]
              [--os-tenant-name <auth-tenant-name>]
              [--os-project-name <auth-project-name>]
              [--os-tenant-id <auth-tenant-id>]
              [--os-project-id <auth-project-id>]
              [--os-user-domain-id <auth-user-domain-id>]
              [--os-user-domain-name <auth-user-domain-name>]
              [--os-project-domain-id <auth-project-domain-id>]
              [--os-project-domain-name <auth-project-domain-name>]
              [--os-auth-url <auth-url>] [--os-region-name <region-name>]
              [--os-token <token>] [--bypass-url <bypass-url>]
              [--service-type <service-type>] [--service-name <service-name>]
              [--share-service-name <share-service-name>]
              [--endpoint-type <endpoint-type>]
              [--os-share-api-version <share-api-ver>]
              [--os-cacert <ca-certificate>] [--retries <retries>]
              [--os-cert <certificate>]
              <subcommand> ...
```

Subcommands:

absolute-limits Print a list of absolute limits for a user.

access-allow Allow access to the share.

access-deny Deny access to a share.

access-list Show access list for share.

api-version Display the API version information.

availability-zone-list List all availability zones.

create Creates a new share (NFS, CIFS, CephFS, GlusterFS or HDFS).

credentials Show user credentials returned from auth.

delete Remove one or more shares.

endpoints Discover endpoints that get returned from the authenticate services.

extend Increases the size of an existing share.

extra-specs-list Print a list of current share types and extra specs (Admin Only).

force-delete Attempt force-delete of share, regardless of state (Admin only).

list List NAS shares with filters.

manage Manage share not handled by Manila (Admin only).

message-delete Remove one or more messages.

message-list Lists all messages.

message-show Show messages details.

metadata Set or delete metadata on a share.

metadata-show Show metadata of given share.

metadata-update-all Update all metadata of a share.

migration-cancel Cancels migration of a given share when copying (Admin only, Experimental).

migration-complete Completes migration for a given share (Admin only, Experimental).

migration-get-progress Gets migration progress of a given share when copying (Admin only, Experimental).

migration-start Migrates share to a new host (Admin only, Experimental).

pool-list List all backend storage pools known to the scheduler (Admin only).

quota-class-show List the quotas for a quota class.

quota-class-update Update the quotas for a quota class (Admin only).

quota-defaults List the default quotas for a tenant.

quota-delete Delete quota for a tenant/user. The quota will revert back to default (Admin only).

quota-show List the quotas for a tenant/user.

quota-update Update the quotas for a tenant/user (Admin only).

rate-limits Print a list of rate limits for a user.

reset-state Explicitly update the state of a share (Admin only).

reset-task-state Explicitly update the task state of a share (Admin only, Experimental).

revert-to-snapshot Revert a share to the specified snapshot.

security-service-create Create security service used by tenant.

security-service-delete Delete one or more security services.

security-service-list Get a list of security services.

security-service-show Show security service.

security-service-update Update security service.

service-disable Disables manila-share or manila-scheduler services (Admin only).

service-enable Enables manila-share or manila-scheduler services (Admin only).

service-list List all services (Admin only).

share-export-location-list List export locations of a given share.

share-export-location-show Show export location of the share.

share-group-create Creates a new share group (Experimental).

share-group-delete Remove one or more share groups (Experimental).

share-group-list List share groups with filters (Experimental).

share-group-reset-state Explicitly update the state of a share group (Admin only, Experimental).

share-group-show Show details about a share group (Experimental).

share-group-snapshot-create Creates a new share group snapshot (Experimental).

share-group-snapshot-delete Remove one or more share group snapshots (Experimental).

share-group-snapshot-list List share group snapshots with filters (Experimental).

share-group-snapshot-list-members List members of a share group snapshot (Experimental).

share-group-snapshot-reset-state Explicitly update the state of a share group snapshot (Admin only, Experimental).

share-group-snapshot-show Show details about a share group snapshot (Experimental).

share-group-snapshot-update Update a share group snapshot (Experimental).

share-group-type-access-add Adds share group type access for the given project (Admin only).

share-group-type-access-list Print access information about a share group type (Admin only).

share-group-type-access-remove Removes share group type access for the given project (Admin only).

share-group-type-create Create a new share group type (Admin only).

share-group-type-delete Delete a specific share group type (Admin only).

share-group-type-key Set or unset group_spec for a share group type (Admin only).

share-group-type-list Print a list of available share group types.

share-group-type-specs-list Print a list of share group types specs (Admin Only).

share-group-update Update a share group (Experimental).

share-instance-export-location-list List export locations of a given share instance.

share-instance-export-location-show Show export location for the share instance.

share-instance-force-delete Force-delete the share instance, regardless of state (Admin only).

share-instance-list List share instances (Admin only).

share-instance-reset-state Explicitly update the state of a share instance (Admin only).

share-instance-show Show details about a share instance (Admin only).

share-network-create Create description for network used by the tenant.

share-network-delete Delete one or more share networks.

share-network-list Get a list of network info.

share-network-security-service-add Associate security service with share network.

share-network-security-service-list Get list of security services associated with a given share network.

share-network-security-service-remove Dissociate security service from share network.

share-network-show Get a description for network used by the tenant.

share-network-update Update share network data.

share-replica-create Create a share replica (Experimental).

share-replica-delete Remove one or more share replicas (Experimental).

share-replica-list List share replicas (Experimental).

share-replica-promote Promote specified replica to active replica_state (Experimental).

share-replica-reset-replica-state Explicitly update the replica_state of a share replica (Experimental).

share-replica-reset-state Explicitly update the status of a share replica (Experimental).

share-replica-resync Attempt to update the share replica with its active mirror (Experimental).

share-replica-show Show details about a replica (Experimental).

share-server-delete Delete one or more share servers (Admin only).

share-server-details Show share server details (Admin only).

share-server-list List all share servers (Admin only).

share-server-show Show share server info (Admin only).

show Show details about a NAS share.

shrink Decreases the size of an existing share.

snapshot-access-allow Allow read only access to a snapshot.

snapshot-access-deny Deny access to a snapshot.

snapshot-access-list Show access list for a snapshot.

snapshot-create Add a new snapshot.

snapshot-delete Remove one or more snapshots.

snapshot-export-location-list List export locations of a given snapshot.

snapshot-export-location-show Show export location of the share snapshot.

snapshot-force-delete Attempt force-deletion of one or more snapshots. Regardless of the state (Admin only).

snapshot-instance-export-location-list List export locations of a given snapshot instance.

snapshot-instance-export-location-show Show export location of the share instance snapshot.

snapshot-instance-list List share snapshot instances.

snapshot-instance-reset-state Explicitly update the state of a share snapshot instance.

snapshot-instance-show Show details about a share snapshot instance.

snapshot-list List all the snapshots.

snapshot-manage Manage share snapshot not handled by Manila (Admin only).

snapshot-rename Rename a snapshot.

snapshot-reset-state Explicitly update the state of a snapshot (Admin only).

snapshot-show Show details about a snapshot.

snapshot-unmanage Unmanage one or more share snapshots (Admin only).

type-access-add Adds share type access for the given project (Admin only).

type-access-list Print access information about the given share type (Admin only).

type-access-remove Removes share type access for the given project (Admin only).

type-create Create a new share type (Admin only).

type-delete Delete one or more specific share types (Admin only).

type-key Set or unset extra_spec for a share type (Admin only).

type-list Print a list of available share types.

unmanage Unmanage share (Admin only).

update Rename a share.

bash-completion Print arguments for bash_completion. Prints all of the commands and options to stdout so that the manila.bash_completion script doesnt have to hard code them.

help Display help about this program or one of its subcommands.

list-extensions List all the os-api extensions that are available.

manila optional arguments

--version show programs version number and exit

-d, --debug Print debugging output.

--os-cache Use the auth token cache. Defaults to env[OS_CACHE].

--os-reset-cache Delete cached password and auth token.

--os-user-id <auth-user-id> Defaults to env [OS_USER_ID].

--os-username <auth-user-name> Defaults to env[OS_USERNAME].

--os-password <auth-password> Defaults to env[OS_PASSWORD].

--os-tenant-name <auth-tenant-name> Defaults to env[OS_TENANT_NAME].

--os-project-name <auth-project-name> Another way to specify tenant name. This option is mutually exclusive with os-tenant-name. Defaults to env[OS_PROJECT_NAME].

--os-tenant-id <auth-tenant-id> Defaults to env[OS_TENANT_ID].

--os-project-id <auth-project-id> Another way to specify tenant ID. This option is mutually exclusive with os-tenant-id. Defaults to env[OS_PROJECT_ID].

--os-user-domain-id <auth-user-domain-id> OpenStack user domain ID. Defaults to env[OS_USER_DOMAIN_ID].

--os-user-domain-name <auth-user-domain-name> OpenStack user domain name. Defaults to env[OS_USER_DOMAIN_NAME].

--os-project-domain-id <auth-project-domain-id> Defaults to env[OS_PROJECT_DOMAIN_ID].

--os-project-domain-name <auth-project-domain-name> Defaults to env[OS_PROJECT_DOMAIN_NAME].

--os-auth-url <auth-url> Defaults to env[OS_AUTH_URL].

--os-region-name <region-name> Defaults to env[OS_REGION_NAME].

--os-token <token> Defaults to env[OS_TOKEN].

--bypass-url <bypass-url> Use this API endpoint instead of the Service Catalog. Defaults to env[OS_MANILA_BYPASS_URL].

--service-type <service-type> Defaults to compute for most actions.

--service-name <service-name> Defaults to env[OS_MANILA_SERVICE_NAME].

--share-service-name <share-service-name> Defaults to env[OS_MANILA_SHARE_SERVICE_NAME].

--endpoint-type <endpoint-type> Defaults to env[OS_MANILA_ENDPOINT_TYPE] or publicURL.

--os-share-api-version <share-api-ver> Accepts 1.x to override default to env[OS_SHARE_API_VERSION].

--os-cacert <ca-certificate> Specify a CA bundle file to use in verifying a TLS (https) server certificate. Defaults to env[OS_CACERT].

--retries <retries> Number of retries.

--os-cert <certificate> Defaults to env[OS_CERT].

manila absolute-limits

```
usage: manila absolute-limits
```

Print a list of absolute limits for a user.

manila access-allow

```
usage: manila access-allow [--access-level <access_level>]
                           <share> <access_type> <access_to>
```

Allow access to the share.

Positional arguments:

<share> Name or ID of the NAS share to modify.

<access_type> Access rule type (only ip, user(user or group), cert or cephx are supported).

<access_to> Value that defines access.

Optional arguments:

--access-level <access_level>, --access_level <access_level> Share access level (rw and ro access levels are supported). Defaults to rw.

manila access-deny

```
usage: manila access-deny <share> <id>
```

Deny access to a share.

Positional arguments:

<share> Name or ID of the NAS share to modify.

<id> ID of the access rule to be deleted.

manila access-list

```
usage: manila access-list [--columns <columns>] <share>
```

Show access list for share.

Positional arguments:

<share> Name or ID of the share.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns access_type,access_to.

manila api-version

```
usage: manila api-version
```

Display the API version information.

manila availability-zone-list

```
usage: manila availability-zone-list [--columns <columns>]
```

List all availability zones.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila create

```
usage: manila create [--snapshot-id <snapshot-id>] [--name <name>]
                     [--metadata [<key=value> [<key=value> ...]]]
                     [--share-network <network-info>]
                     [--description <description>] [--share-type <share-type>]
                     [--public] [--availability-zone <availability-zone>]
                     [--share-group <share-group>]
                     <share_protocol> <size>
```

Creates a new share (NFS, CIFS, CephFS, GlusterFS or HDFS).

Positional arguments:

<share_protocol> Share protocol (NFS, CIFS, CephFS, GlusterFS or HDFS).

<size> Share size in GiB.

Optional arguments:

--snapshot-id <snapshot-id>, --snapshot_id <snapshot-id> Optional snapshot ID to create the share from. (Default=None)

--name <name> Optional share name. (Default=None)

--metadata [<key=value> [<key=value> ...]] Metadata key=value pairs (Optional, Default=None).

--share-network <network-info>, --share_network <network-info> Optional network info ID or name.

--description <description> Optional share description. (Default=None)

--share-type <share-type>, --share_type <share-type>, --volume-type <share-type>, --volume_t Optional share type. Use of optional volume type is deprecated. (Default=None)

--public Level of visibility for share. Defines whether other tenants are able to see it or not.

--availability-zone <availability-zone>, --availability_zone <availability-zone>, --az <avai Availability zone in which share should be created.

--share-group <share-group>, --share_group <share-group>, --group <share-group> Optional share group name or ID in which to create the share (Experimental, Default=None).

manila credentials

```
usage: manila credentials
```

Show user credentials returned from auth.

manila delete

```
usage: manila delete [--share-group <share-group>] <share> [<share> ...]
```

Remove one or more shares.

Positional arguments:

<share> Name or ID of the share(s).

Optional arguments:

--share-group <share-group>, --share_group <share-group>, --group <share-group>
Optional share group name or ID which contains the share (Experimental, Default=None).

manila endpoints

```
usage: manila endpoints
```

Discover endpoints that get returned from the authenticate services.

manila extend

```
usage: manila extend <share> <new_size>
```

Increases the size of an existing share.

Positional arguments:

<share> Name or ID of share to extend.

<new_size> New size of share, in GiBs.

manila extra-specs-list

```
usage: manila extra-specs-list [--columns <columns>]
```

Print a list of current share types and extra specs (Admin Only).

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila force-delete

```
usage: manila force-delete <share> [<share> ...]
```

Attempt force-delete of share, regardless of state (Admin only).

Positional arguments:

<share> Name or ID of the share(s) to force delete.

manila list

```
usage: manila list [--all-tenants [<0|1>]] [--name <name>] [--status <status>]
                   [--share-server-id <share_server_id>]
                   [--metadata [<key=value> [<key=value> ...]]]
                   [--extra-specs [<key=value> [<key=value> ...]]]
                   [--share-type <share_type>] [--limit <limit>]
                   [--offset <offset>] [--sort-key <sort_key>]
                   [--sort-dir <sort_dir>] [--snapshot <snapshot>]
                   [--host <host>] [--share-network <share_network>]
                   [--project-id <project_id>] [--public]
                   [--share-group <share_group>] [--columns <columns>]
```

List NAS shares with filters.

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--name <name> Filter results by name.

--status <status> Filter results by status.

--share-server-id <share_server_id>, --share-server_id <share_server_id>, --share_server-id
Filter results by share server ID (Admin only).

--metadata [<key=value> [<key=value> ...]] Filters results by a metadata key and value. OPTIONAL: Default=None.

--extra-specs [<key=value> [<key=value> ...]], --extra_specs [<key=value> [<key=value> ...]]
Filters results by an extra specs key and value of share type that was used for share creation. OPTIONAL: Default=None.

--share-type <share_type>, --volume-type <share_type>, --share_type <share_type>, --share-ty
Filter results by a share type id or name that was used for share creation.

--limit <limit> Maximum number of shares to return. OPTIONAL: Default=None.

--offset <offset> Set offset to define start point of share listing. OPTIONAL: Default=None.

--sort-key <sort_key>, --sort_key <sort_key> Key to be sorted, available keys are (id, status, size, host, share_proto, availability_zone, user_id, project_id, created_at, updated_at, display_name, name, share_type_id, share_type, share_network_id, share_network, snapshot_id, snapshot). OPTIONAL: Default=None.

--sort-dir <sort_dir>, --sort_dir <sort_dir> Sort direction, available values are (asc, desc). OPTIONAL: Default=None.

--snapshot <snapshot> Filter results by snapshot name or id, that was used for share.

--host <host> Filter results by host.

--share-network <share_network>, --share_network <share_network> Filter results by share-network name or id.

--project-id <project_id>, --project_id <project_id> Filter results by project id. Useful with set key all-tenants.

--public Add public shares from all tenants to result.

--share-group <share_group>, --share_group <share_group>, --group <share_group> Filter results by share group name or ID (Experimental, Default=None).

--columns <columns> Comma separated list of columns to be displayed example columns export_location,is public.

manila list-extensions

```
usage: manila list-extensions
```

List all the os-api extensions that are available.

manila manage

```
usage: manila manage [--name <name>] [--description <description>]
                     [--share_type <share-type>]
                     [--driver_options [<key=value> [<key=value> ...]]]
                     [--public]
                     <service_host> <protocol> <export_path>
```

Manage share not handled by Manila (Admin only).

Positional arguments:

<service_host> manage-share service host: some.host@driver#pool.

<protocol> Protocol of the share to manage, such as NFS or CIFS.

<export_path> Share export path, NFS share such as: 10.0.0.1:/example_path, CIFS share such as: \\10.0.0.1\example_cifs_share.

Optional arguments:

--name <name> Optional share name. (Default=None)

--description <description> Optional share description. (Default=None)

--share_type <share-type>, --share-type <share-type> Optional share type assigned to share. (Default=None)

--driver_options [<key=value> [<key=value> ...]], --driver-options [<key=value> [<key=value> Driver option key=value pairs (Optional, Default=None).

--public Level of visibility for share. Defines whether other tenants are able to see it or not. Available only for microversion >= 2.8.

manila message-delete

```
usage: manila message-delete <message> [<message> ...]
```

Remove one or more messages.

Positional arguments:

<message> ID of the message(s).

manila message-list

```
usage: manila message-list [--resource_id <resource_id>]
                           [--resource_type <type>] [--action_id <id>]
                           [--detail_id <id>] [--request_id <request_id>]
                           [--level <level>] [--limit <limit>]
                           [--offset <offset>] [--sort-key <sort_key>]
                           [--sort-dir <sort_dir>] [--columns <columns>]
                           [--since <since>] [--before <before>]
```

Lists all messages.

Optional arguments:

--resource_id <resource_id>, --resource-id <resource_id>, --resource <resource_id> Filters results by a resource uuid. (Default=None).

--resource_type <type>, --resource-type <type> Filters results by a resource type. (Default=None). Example: manila message-list resource_type share

--action_id <id>, --action-id <id>, --action <id> Filters results by action id. (Default=None).

--detail_id <id>, --detail-id <id>, --detail <id> Filters results by detail id. (Default=None).

--request_id <request_id>, --request-id <request_id>, --request <request_id> Filters results by request id. (Default=None).

--level <level>, --message_level <level>, --message-level <level> Filters results by the message level. (Default=None). Example: manila message-list level ERROR.

--limit <limit> Maximum number of messages to return. (Default=None)

--offset <offset> Start position of message listing.

--sort-key <sort_key>, --sort_key <sort_key> Key to be sorted, available keys are (id, project_id, request_id, resource_type, action_id, detail_id, resource_id, message_level, expires_at, request_id, created_at). (Default=desc).

--sort-dir <sort_dir>, --sort_dir <sort_dir> Sort direction, available values are (asc, desc). OPTIONAL: Default=None.

--columns <columns> Comma separated list of columns to be displayed example columns resource_id, user_message.

--since <since> Return only user messages created since given date. The date format must be conforming to ISO8601. Available only for microversion >= 2.52.

--before <before> Return only user messages created before given date. The date format must be conforming to ISO8601. Available only for microversion >= 2.52.

manila message-show

```
usage: manila message-show <message>
```

Show details about a message.

Positional arguments:

<message> ID of the message.

manila metadata

```
usage: manila metadata <share> <action> <key=value> [<key=value> ...]
```

Set or delete metadata on a share.

Positional arguments:

<share> Name or ID of the share to update metadata on.

<action> Actions: set or unset.

<key=value> Metadata to set or unset (key is only necessary on unset).

manila metadata-show

```
usage: manila metadata-show <share>
```

Show metadata of given share.

Positional arguments:

<share> Name or ID of the share.

manila metadata-update-all

```
usage: manila metadata-update-all <share> <key=value> [<key=value> ...]
```

Update all metadata of a share.

Positional arguments:

<share> Name or ID of the share to update metadata on.

<key=value> Metadata entry or entries to update.

manila migration-cancel

```
usage: manila migration-cancel <share>
```

Cancels migration of a given share when copying (Admin only, Experimental).

Positional arguments:

<share> Name or ID of share to cancel migration.

manila migration-complete

```
usage: manila migration-complete <share>
```

Completes migration for a given share (Admin only, Experimental).

Positional arguments:

<share> Name or ID of share to complete migration.

manila migration-get-progress

```
usage: manila migration-get-progress <share>
```

Gets migration progress of a given share when copying (Admin only, Experimental).

Positional arguments:

<share> Name or ID of the share to get share migration progress information.

manila migration-start

```
usage: manila migration-start [--force_host_assisted_migration <True|False>]
                              --preserve-metadata <True|False>
                              --preserve-snapshots <True|False> --writable
                              <True|False> --nondisruptive <True|False>
                              [--new_share_network <new_share_network>]
                              [--new_share_type <new_share_type>]
                              <share> <host@backend#pool>
```

Migrates share to a new host (Admin only, Experimental).

Positional arguments:

<share> Name or ID of share to migrate.

<host@backend#pool> Destination host where share will be migrated to. Use the format host@backend#pool.

Optional arguments:

--force_host_assisted_migration <True|False>, --force-host-assisted-migration <True|False> Enforces the use of the host-assisted migration approach, which bypasses driver optimizations. Default=False.

--preserve-metadata <True|False>, --preserve_metadata <True|False> Enforces migration to preserve all file metadata when moving its contents. If set to True, host-assisted migration will not be attempted.

--preserve-snapshots <True|False>, --preserve_snapshots <True|False> Enforces migration of the share snapshots to the destination. If set to True, host-assisted migration will not be attempted.

--writable <True|False> Enforces migration to keep the share writable while contents are being moved. If set to True, host-assisted migration will not be attempted.

--nondisruptive <True|False> Enforces migration to be nondisruptive. If set to True, host-assisted migration will not be attempted.

--new_share_network <new_share_network>, --new-share-network <new_share_network> Specify the new share network for the share. Do not specify this parameter if the migrating share has to be retained within its current share network.

--new_share_type <new_share_type>, --new-share-type <new_share_type> Specify the new share type for the share. Do not specify this parameter if the migrating share has to be retained with its current share type.

manila pool-list

```
usage: manila pool-list [--host <host>] [--backend <backend>] [--pool <pool>]
                        [--columns <columns>] [--detail]
                        [--share-type <share_type>]
```

List all backend storage pools known to the scheduler (Admin only).

Optional arguments:

--host <host> Filter results by host name. Regular expressions are supported.

--backend <backend> Filter results by backend name. Regular expressions are supported.

--pool <pool> Filter results by pool name. Regular expressions are supported.

--columns <columns> Comma separated list of columns to be displayed example columns name,host.

--detail, --detailed Show detailed information about pools. (Default=False)

--share-type <share_type>, --share_type <share_type>, --share-type-id <share_type>, --share_ Filter results by share type name or ID. (Default=None)Available only for microversion >= 2.23.

manila quota-class-show

```
usage: manila quota-class-show <class>
```

List the quotas for a quota class.

Positional arguments:

<class> Name of quota class to list the quotas for.

manila quota-class-update

```
usage: manila quota-class-update [--shares <shares>] [--snapshots <snapshots>]
                                 [--gigabytes <gigabytes>]
                                 [--snapshot-gigabytes <snapshot_gigabytes>]
                                 [--share-networks <share-networks>]
                                 [--share-groups <share-groups>]
                                 [--share-group-snapshots <share-group-
↪snapshots>]
                                 <class-name>
```

Update the quotas for a quota class (Admin only).

Positional arguments:

<class-name> Name of quota class to set the quotas for.

Optional arguments:

--shares <shares> New value for the shares quota.

--snapshots <snapshots> New value for the snapshots quota.

--gigabytes <gigabytes> New value for the gigabytes quota.

--snapshot-gigabytes <snapshot_gigabytes>, --snapshot_gigabytes <snapshot_gigabytes> New value for the snapshot_gigabytes quota.

--share-networks <share-networks>, --share_networks <share-networks> New value for the share_networks quota.

--share-groups <share-groups>, --share_groups <share-groups> New value for the share_groups quota.

--share-group-snapshots <share-group-snapshots>, --share_group_snapshots <share-group-snapsh New value for the share_group_snapshots quota.

manila quota-defaults

```
usage: manila quota-defaults [--tenant <tenant-id>]
```

List the default quotas for a tenant.

Optional arguments:

--tenant <tenant-id> ID of tenant to list the default quotas for.

manila quota-delete

```
usage: manila quota-delete [--tenant <tenant-id>] [--user <user-id>]
                           [--share-type <share-type>]
```

Delete quota for a tenant/user. The quota will revert back to default (Admin only).

Optional arguments:

--tenant <tenant-id> ID of tenant to delete quota for.

--user <user-id> ID of user to delete quota for.

--share-type <share-type>, --share_type <share-type> UUID or name of a share type to set the quotas for. Optional. Mutually exclusive with user-id. Available only for microversion >= 2.39

manila quota-show

```
usage: manila quota-show [--tenant <tenant-id>] [--user <user-id>]
                         [--share-type <share-type>] [--detail]
```

List the quotas for a tenant/user.

Optional arguments:

--tenant <tenant-id> ID of tenant to list the quotas for.

--user <user-id> ID of user to list the quotas for.

--share-type <share-type>, --share_type <share-type> UUID or name of a share type to set the quotas for. Optional. Mutually exclusive with user-id. Available only for microversion >= 2.39

--detail Optional flag to indicate whether to show quota in detail. Default false, available only for microversion >= 2.25.

manila quota-update

```
usage: manila quota-update [--user <user-id>] [--shares <shares>]
                           [--snapshots <snapshots>] [--gigabytes <gigabytes>]
                           [--snapshot-gigabytes <snapshot_gigabytes>]
                           [--share-networks <share-networks>]
                           [--share-groups <share-groups>]
                           [--share-group-snapshots <share-group-snapshots>]
                           [--share-type <share-type>] [--force]
                           <tenant_id>
```

Update the quotas for a tenant/user (Admin only).

Positional arguments:

<tenant_id> UUID of tenant to set the quotas for.

Optional arguments:

--user <user-id> ID of user to set the quotas for.

--shares <shares> New value for the shares quota.

--snapshots <snapshots> New value for the snapshots quota.

--gigabytes <gigabytes> New value for the gigabytes quota.

--snapshot-gigabytes <snapshot_gigabytes>, --snapshot_gigabytes <snapshot_gigabytes> New value for the snapshot_gigabytes quota.

--share-networks <share-networks>, --share_networks <share-networks> New value for the share_networks quota.

--share-groups <share-groups>, --share_groups <share-groups> New value for the share_groups quota.

--share-group-snapshots <share-group-snapshots>, --share_group_snapshots <share-group-snapsh New value for the share_group_snapshots quota.

--share-type <share-type>, --share_type <share-type> UUID or name of a share type to set the quotas for. Optional. Mutually exclusive with user-id. Available only for microversion >= 2.39

--force Whether force update the quota even if the already used and reserved exceeds the new quota.

manila rate-limits

```
usage: manila rate-limits [--columns <columns>]
```

Print a list of rate limits for a user.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns verb,uri,value.

manila reset-state

```
usage: manila reset-state [--state <state>] <share>
```

Explicitly update the state of a share (Admin only).

Positional arguments:

<share> Name or ID of the share to modify.

Optional arguments:

--state <state> Indicate which state to assign the share. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila reset-task-state

```
usage: manila reset-task-state [--task-state <task_state>] <share>
```

Explicitly update the task state of a share (Admin only, Experimental).

Positional arguments:

<share> Name or ID of the share to modify.

Optional arguments:

--task-state <task_state>, --task_state <task_state>, --state <task_state>
Indicate which task state to assign the share. Options include migration_starting, migration_in_progress, migration_completing, migration_success, migration_error, migration_cancelled, migration_driver_in_progress, migration_driver_phase1_done, data_copying_starting, data_copying_in_progress, data_copying_completing, data_copying_completed, data_copying_cancelled, data_copying_error. If no value is provided, None will be used.

manila revert-to-snapshot

```
usage: manila revert-to-snapshot <snapshot>
```

Revert a share to the specified snapshot.

Positional arguments:

<snapshot> Name or ID of the snapshot to restore. The snapshot must be the most recent one known to manila.

manila security-service-create

```
usage: manila security-service-create [--dns-ip <dns_ip>] [--server <server>]
                                      [--domain <domain>] [--user <user>]
                                      [--password <password>] [--name <name>]
                                      [--description <description>]
                                      <type>
```

Create security service used by tenant.

Positional arguments:

<type> Security service type: ldap, kerberos or active_directory.

Optional arguments:

--dns-ip <dns_ip> DNS IP address used inside tenants network.

--server <server> Security service IP address or hostname.

--domain <domain> Security service domain.

--user <user> Security service user or group used by tenant.

--password <password> Password used by user.

--name <name> Security service name.

--description <description> Security service description.

manila security-service-delete

```
usage: manila security-service-delete <security-service>
                                      [<security-service> ...]
```

Delete one or more security services.

Positional arguments:

<security-service> Name or ID of the security service(s) to delete.

manila security-service-list

```
usage: manila security-service-list [--all-tenants [<0|1>]]
                                    [--share-network <share_network>]
                                    [--status <status>] [--name <name>]
                                    [--type <type>] [--user <user>]
                                    [--dns-ip <dns_ip>] [--server <server>]
                                    [--domain <domain>] [--detailed [<0|1>]]
                                    [--offset <offset>] [--limit <limit>]
                                    [--columns <columns>]
```

Get a list of security services.

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--share-network <share_network>, --share_network <share_network> Filter results by share network id or name.

--status <status> Filter results by status.

--name <name> Filter results by name.

--type <type> Filter results by type.

--user <user> Filter results by user or group used by tenant.

--dns-ip <dns_ip>, --dns_ip <dns_ip> Filter results by DNS IP address used inside tenants network.

--server <server> Filter results by security service IP address or hostname.

--domain <domain> Filter results by domain.

--detailed [<0|1>] Show detailed information about filtered security services.

--offset <offset> Start position of security services listing.

--limit <limit> Number of security services to return per request.

--columns <columns> Comma separated list of columns to be displayed example columns name,type.

manila security-service-show

```
usage: manila security-service-show <security-service>
```

Show security service.

Positional arguments:

<security-service> Security service name or ID to show.

manila security-service-update

```
usage: manila security-service-update [--dns-ip <dns-ip>] [--server <server>]
                                      [--domain <domain>] [--user <user>]
                                      [--password <password>] [--name <name>]
                                      [--description <description>]
                                      <security-service>
```

Update security service.

Positional arguments:

<security-service> Security service name or ID to update.

Optional arguments:

--dns-ip <dns-ip> DNS IP address used inside tenants network.

--server <server> Security service IP address or hostname.

--domain <domain> Security service domain.

--user <user> Security service user or group used by tenant.

--password <password> Password used by user.

--name <name> Security service name.

--description <description> Security service description.

manila service-disable

```
usage: manila service-disable <hostname> <binary>
```

Disables manila-share or manila-scheduler services (Admin only).

Positional arguments:

<hostname> Host name as example_host@example_backend.

<binary> Service binary, could be manila-share or manila-scheduler.

manila service-enable

```
usage: manila service-enable <hostname> <binary>
```

Enables manila-share or manila-scheduler services (Admin only).

Positional arguments:

<hostname> Host name as example_host@example_backend.

<binary> Service binary, could be manila-share or manila-scheduler.

manila service-list

```
usage: manila service-list [--host <hostname>] [--binary <binary>]
                           [--status <status>] [--state <state>]
                           [--zone <zone>] [--columns <columns>]
```

List all services (Admin only).

Optional arguments:

--host <hostname> Name of host.

--binary <binary> Service binary.

--status <status> Filter results by status.

--state <state> Filter results by state.

--zone <zone> Availability zone.

--columns <columns> Comma separated list of columns to be displayed example columns id,host.

manila share-export-location-list

```
usage: manila share-export-location-list [--columns <columns>] <share>
```

List export locations of a given share.

Positional arguments:

<share> Name or ID of the share.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,host,status.

manila share-export-location-show

```
usage: manila share-export-location-show <share> <export_location>
```

Show export location of the share.

Positional arguments:

<share> Name or ID of the share.

<export_location> ID of the share export location.

manila share-group-create

```
usage: manila share-group-create [--name <name>] [--description <description>]
                                 [--share-types <share_types>]
                                 [--share-group-type <share_group_type>]
                                 [--share-network <share_network>]
                                 [--source-share-group-snapshot <source_share_
↪group_snapshot>]
                                 [--availability-zone <availability-zone>]
```

Creates a new share group (Experimental).

Optional arguments:

--name <name> Optional share group name. (Default=None)

--description <description> Optional share group description. (Default=None)

--share-types <share_types>, --share_types <share_types> Comma-separated list of share types. (Default=None)

--share-group-type <share_group_type>, --share_group_type <share_group_type>, --type <share_
Share group type name or ID of the share group to be created. (Default=None)

--share-network <share_network>, --share_network <share_network> Specify share network name or id.

--source-share-group-snapshot <source_share_group_snapshot>, --source_share_group_snapshot <
Optional share group snapshot name or ID to create the share group from. (Default=None)

--availability-zone <availability-zone>, --availability_zone <availability-zone>, --az <ava
Optional availability zone in which group should be created. (Default=None)

manila share-group-delete

```
usage: manila share-group-delete [--force] <share_group> [<share_group> ...]
```

Remove one or more share groups (Experimental).

Positional arguments:

<share_group> Name or ID of the share_group(s).

Optional arguments:

--force Attempt to force delete the share group (Default=False) (Admin only).

manila share-group-list

```
usage: manila share-group-list [--all-tenants [<0|1>]] [--name <name>]
                               [--status <status>]
                               [--share-server-id <share_server_id>]
                               [--share-group-type <share_group_type>]
                               [--snapshot <snapshot>] [--host <host>]
                               [--share-network <share_network>]
                               [--project-id <project_id>] [--limit <limit>]
                               [--offset <offset>] [--sort-key <sort_key>]
                               [--sort-dir <sort_dir>] [--columns <columns>]
```

List share groups with filters (Experimental).

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--name <name> Filter results by name.

--status <status> Filter results by status.

--share-server-id <share_server_id>, --share-server_id <share_server_id>, --share_server-id
Filter results by share server ID (Admin only).

--share-group-type <share_group_type>, --share-group-type-id <share_group_type>, --share_gro
Filter results by a share group type ID or name that was used for share group creation.

--snapshot <snapshot> Filter results by share group snapshot name or ID that was used to create the share group.

--host <host> Filter results by host.

--share-network <share_network>, --share_network <share_network> Filter results by share-network name or ID.

--project-id <project_id>, --project_id <project_id> Filter results by project ID. Useful with set key all-tenants.

--limit <limit> Maximum number of share groups to return. (Default=None)

--offset <offset> Start position of share group listing.

--sort-key <sort_key>, --sort_key <sort_key> Key to be sorted, available keys are (id, name, status, host, user_id, project_id, created_at, availability_zone, share_network, share_network_id, share_group_type, share_group_type_id, source_share_group_snapshot_id). Default=None.

--sort-dir <sort_dir>, --sort_dir <sort_dir> Sort direction, available values are (asc, desc). OPTIONAL: Default=None.

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-group-reset-state

```
usage: manila share-group-reset-state [--state <state>] <share_group>
```

Explicitly update the state of a share group (Admin only, Experimental).

Positional arguments:

<share_group> Name or ID of the share group to modify.

Optional arguments:

--state <state> Indicate which state to assign the share group. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila share-group-show

```
usage: manila share-group-show <share_group>
```

Show details about a share group (Experimental).

Positional arguments:

<share_group> Name or ID of the share group.

manila share-group-snapshot-create

```
usage: manila share-group-snapshot-create [--name <name>]
                                          [--description <description>]
                                          <share_group>
```

Creates a new share group snapshot (Experimental).

Positional arguments:

<share_group> Name or ID of the share group.

Optional arguments:

--name <name> Optional share group snapshot name. (Default=None)

--description <description> Optional share group snapshot description. (Default=None)

manila share-group-snapshot-delete

```
usage: manila share-group-snapshot-delete [--force]
                                          <share_group_snapshot>
                                          [<share_group_snapshot> ...]
```

Remove one or more share group snapshots (Experimental).

Positional arguments:

<share_group_snapshot> Name or ID of the share group snapshot(s) to delete.

Optional arguments:

--force Attempt to force delete the share group snapshot(s) (Default=False) (Admin only).

manila share-group-snapshot-list

```
usage: manila share-group-snapshot-list [--all-tenants [<0|1>]]
                                        [--name <name>] [--status <status>]
                                        [--share-group-id <share_group_id>]
                                        [--limit <limit>] [--offset <offset>]
                                        [--sort-key <sort_key>]
                                        [--sort-dir <sort_dir>]
                                        [--detailed DETAILED]
                                        [--columns <columns>]
```

List share group snapshots with filters (Experimental).

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--name <name> Filter results by name.

--status <status> Filter results by status.

--share-group-id <share_group_id>, --share_group_id <share_group_id> Filter results by share group ID.

--limit <limit> Maximum number of share group snapshots to return. (Default=None)

--offset <offset> Start position of share group snapshot listing.

--sort-key <sort_key>, --sort_key <sort_key> Key to be sorted, available keys are (id, name, status, host, user_id, project_id, created_at, share_group_id). Default=None.

--sort-dir <sort_dir>, --sort_dir <sort_dir> Sort direction, available values are (asc, desc). OPTIONAL: Default=None.

--detailed DETAILED Show detailed information about share group snapshots.

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-group-snapshot-list-members

```
usage: manila share-group-snapshot-list-members [--columns <columns>]
                                                <share_group_snapshot>
```

List members of a share group snapshot (Experimental).

Positional arguments:

<share_group_snapshot> Name or ID of the share group snapshot.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-group-snapshot-reset-state

```
usage: manila share-group-snapshot-reset-state [--state <state>]
                                               <share_group_snapshot>
```

Explicitly update the state of a share group snapshot (Admin only, Experimental).

Positional arguments:

<share_group_snapshot> Name or ID of the share group snapshot.

Optional arguments:

--state <state> Indicate which state to assign the share group snapshot. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila share-group-snapshot-show

```
usage: manila share-group-snapshot-show <share_group_snapshot>
```

Show details about a share group snapshot (Experimental).

Positional arguments:

<share_group_snapshot> Name or ID of the share group snapshot.

manila share-group-snapshot-update

```
usage: manila share-group-snapshot-update [--name <name>]
                                          [--description <description>]
                                          <share_group_snapshot>
```

Update a share group snapshot (Experimental).

Positional arguments:

<share_group_snapshot> Name or ID of the share group snapshot to update.

Optional arguments:

--name <name> Optional new name for the share group snapshot. (Default=None)

--description <description> Optional share group snapshot description. (Default=None)

manila share-group-type-access-add

```
usage: manila share-group-type-access-add <share_group_type> <project_id>
```

Adds share group type access for the given project (Admin only).

Positional arguments:

<share_group_type> Share group type name or ID to add access for the given project.

<project_id> Project ID to add share group type access for.

manila share-group-type-access-list

```
usage: manila share-group-type-access-list <share_group_type>
```

Print access information about a share group type (Admin only).

Positional arguments:

<share_group_type> Filter results by share group type name or ID.

manila share-group-type-access-remove

```
usage: manila share-group-type-access-remove <share_group_type> <project_id>
```

Removes share group type access for the given project (Admin only).

Positional arguments:

<share_group_type> Share group type name or ID to remove access for the given project.

<project_id> Project ID to remove share group type access for.

manila share-group-type-create

```
usage: manila share-group-type-create [--is_public <is_public>]
                                      <name> <share_types>
```

Create a new share group type (Admin only).

Positional arguments:

<name> Name of the new share group type.

<share_types> Comma-separated list of share type names or IDs.

Optional arguments:

--is_public <is_public>, --is-public <is_public> Make type accessible to the public (default true).

manila share-group-type-delete

```
usage: manila share-group-type-delete <id>
```

Delete a specific share group type (Admin only).

Positional arguments:

<id> Name or ID of the share group type to delete.

manila share-group-type-key

```
usage: manila share-group-type-key <share_group_type> <action>
                                   [<key=value> [<key=value> ...]]
```

Set or unset group_spec for a share group type (Admin only).

Positional arguments:

<share_group_type> Name or ID of the share group type.

<action> Actions: set or unset.

<key=value> Group specs to set or unset (key is only necessary on unset).

manila share-group-type-list

```
usage: manila share-group-type-list [--all] [--columns <columns>]
```

Print a list of available share group types.

Optional arguments:

--all Display all share group types (Admin only).

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-group-type-specs-list

```
usage: manila share-group-type-specs-list [--columns <columns>]
```

Print a list of share group types specs (Admin Only).

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-group-update

```
usage: manila share-group-update [--name <name>] [--description <description>]
                                 <share_group>
```

Update a share group (Experimental).

Positional arguments:

<share_group> Name or ID of the share group to update.

Optional arguments:

--name <name> Optional new name for the share group. (Default=None)

--description <description> Optional share group description. (Default=None)

manila share-instance-export-location-list

```
usage: manila share-instance-export-location-list [--columns <columns>]
                                                  <instance>
```

List export locations of a given share instance.

Positional arguments:

<instance> Name or ID of the share instance.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,host,status.

manila share-instance-export-location-show

```
usage: manila share-instance-export-location-show <instance> <export_location>
```

Show export location for the share instance.

Positional arguments:

<instance> Name or ID of the share instance.

<export_location> ID of the share instance export location.

manila share-instance-force-delete

```
usage: manila share-instance-force-delete <instance> [<instance> ...]
```

Force-delete the share instance, regardless of state (Admin only).

Positional arguments:

<instance> Name or ID of the instance(s) to force delete.

manila share-instance-list

```
usage: manila share-instance-list [--share-id <share_id>]
                                  [--columns <columns>]
```

List share instances (Admin only).

Optional arguments:

--share-id <share_id>, --share_id <share_id> Filter results by share ID.

--columns <columns> Comma separated list of columns to be displayed example columns id,host,status.

manila share-instance-reset-state

```
usage: manila share-instance-reset-state [--state <state>] <instance>
```

Explicitly update the state of a share instance (Admin only).

Positional arguments:

<instance> Name or ID of the share instance to modify.

Optional arguments:

--state <state> Indicate which state to assign the instance. Options include available, error, creating, deleting, error_deleting, migrating,migrating_to. If no state is provided, available will be used.

manila share-instance-show

```
usage: manila share-instance-show <instance>
```

Show details about a share instance (Admin only).

Positional arguments:

<instance> Name or ID of the share instance.

manila share-network-create

```
usage: manila share-network-create [--neutron-net-id <neutron-net-id>]
                                   [--neutron-subnet-id <neutron-subnet-id>]
                                   [--name <name>]
                                   [--description <description>]
```

Create description for network used by the tenant.

Optional arguments:

--neutron-net-id <neutron-net-id>, --neutron-net_id <neutron-net-id>, --neutron_net_id <neut
Neutron network ID. Used to set up network for share servers.

--neutron-subnet-id <neutron-subnet-id>, --neutron-subnet_id <neutron-subnet-id>, --neutron_
Neutron subnet ID. Used to set up network for share servers. This subnet should belong to specified neutron network.

--name <name> Share network name.

--description <description> Share network description.

manila share-network-delete

```
usage: manila share-network-delete <share-network> [<share-network> ...]
```

Delete one or more share networks.

Positional arguments:

<share-network> Name or ID of share network(s) to be deleted.

manila share-network-list

```
usage: manila share-network-list [--all-tenants [<0|1>]]
                                 [--project-id <project_id>] [--name <name>]
                                 [--created-since <created_since>]
                                 [--created-before <created_before>]
                                 [--security-service <security_service>]
                                 [--neutron-net-id <neutron_net_id>]
                                 [--neutron-subnet-id <neutron_subnet_id>]
                                 [--network-type <network_type>]
                                 [--segmentation-id <segmentation_id>]
                                 [--cidr <cidr>] [--ip-version <ip_version>]
                                 [--offset <offset>] [--limit <limit>]
                                 [--columns <columns>]
```

Get a list of network info.

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--project-id <project_id>, --project_id <project_id> Filter results by project ID.

--name <name> Filter results by name.

--created-since <created_since>, --created_since <created_since> Return only share networks created since given date. The date is in the format yyyy-mm-dd.

--created-before <created_before>, --created_before <created_before> Return only share networks created until given date. The date is in the format yyyy-mm-dd.

--security-service <security_service>, --security_service <security_service> Filter results by attached security service.

--neutron-net-id <neutron_net_id>, --neutron_net_id <neutron_net_id>, --neutron_net-id <neut Filter results by neutron net ID.

--neutron-subnet-id <neutron_subnet_id>, --neutron_subnet_id <neutron_subnet_id>, --neutron- Filter results by neutron subnet ID.

--network-type <network_type>, --network_type <network_type> Filter results by network type.

--segmentation-id <segmentation_id>, --segmentation_id <segmentation_id> Filter results by segmentation ID.

--cidr <cidr> Filter results by CIDR.

--ip-version <ip_version>, --ip_version <ip_version> Filter results by IP version.

--offset <offset> Start position of share networks listing.

--limit <limit> Number of share networks to return per request.

--columns <columns> Comma separated list of columns to be displayed example columns id.

manila share-network-security-service-add

```
usage: manila share-network-security-service-add <share-network>
                                                 <security-service>
```

Associate security service with share network.

Positional arguments:

<share-network> Share network name or ID.

<security-service> Security service name or ID to associate with.

manila share-network-security-service-list

```
usage: manila share-network-security-service-list [--columns <columns>]
                                                  <share-network>
```

Get list of security services associated with a given share network.

Positional arguments:

<share-network> Share network name or ID.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila share-network-security-service-remove

```
usage: manila share-network-security-service-remove <share-network>
                                                    <security-service>
```

Dissociate security service from share network.

Positional arguments:

<share-network> Share network name or ID.

<security-service> Security service name or ID to dissociate.

manila share-network-show

```
usage: manila share-network-show <share-network>
```

Get a description for network used by the tenant.

Positional arguments:

<share-network> Name or ID of the share network to show.

manila share-network-update

```
usage: manila share-network-update [--neutron-net-id <neutron-net-id>]
                                   [--neutron-subnet-id <neutron-subnet-id>]
                                   [--name <name>]
                                   [--description <description>]
                                   <share-network>
```

Update share network data.

Positional arguments:

<share-network> Name or ID of share network to update.

Optional arguments:

--neutron-net-id <neutron-net-id>, --neutron-net_id <neutron-net-id>, --neutron_net_id <neut
Neutron network ID. Used to set up network for share servers. This option is deprecated and will be rejected in newer releases of OpenStack Manila.

--neutron-subnet-id <neutron-subnet-id>, --neutron-subnet_id <neutron-subnet-id>, --neutron_
Neutron subnet ID. Used to set up network for share servers. This subnet should belong to specified neutron network.

--name <name> Share network name.

--description <description> Share network description.

manila share-replica-create

```
usage: manila share-replica-create [--availability-zone <availability-zone>]
                                   [--share-network <network-info>]
                                   <share>
```

Create a share replica (Experimental).

Positional arguments:

<share> Name or ID of the share to replicate.

Optional arguments:

--availability-zone <availability-zone>, --availability_zone <availability-zone>, --az <ava Optional Availability zone in which replica should be created.

--share-network <network-info>, --share_network <network-info> Optional network info ID or name.

manila share-replica-delete

```
usage: manila share-replica-delete [--force] <replica> [<replica> ...]
```

Remove one or more share replicas (Experimental).

Positional arguments:

<replica> ID of the share replica.

Optional arguments:

--force Attempt to force deletion of a replica on its backend. Using this option will purge the replica from Manila even if it is not cleaned up on the backend. Defaults to False.

manila share-replica-list

```
usage: manila share-replica-list [--share-id <share_id>] [--columns <columns>]
```

List share replicas (Experimental).

Optional arguments:

--share-id <share_id>, --share_id <share_id>, --si <share_id> List replicas belonging to share.

--columns <columns> Comma separated list of columns to be displayed example columns replica_state,id.

manila share-replica-promote

```
usage: manila share-replica-promote <replica>
```

Promote specified replica to active replica_state (Experimental).

Positional arguments:

<replica> ID of the share replica.

manila share-replica-reset-replica-state

```
usage: manila share-replica-reset-replica-state
                                                [--replica-state <replica_
↪state>]
                                                <replica>
```

Explicitly update the replica_state of a share replica (Experimental).

Positional arguments:

<replica> ID of the share replica to modify.

Optional arguments:

--replica-state <replica_state>, --replica_state <replica_state>, --state <replica_state>
Indicate which replica_state to assign the replica. Options include in_sync, out_of_sync, active, error. If no state is provided, out_of_sync will be used.

manila share-replica-reset-state

```
usage: manila share-replica-reset-state [--state <state>] <replica>
```

Explicitly update the status of a share replica (Experimental).

Positional arguments:

<replica> ID of the share replica to modify.

Optional arguments:

--state <state> Indicate which state to assign the replica. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila share-replica-resync

```
usage: manila share-replica-resync <replica>
```

Attempt to update the share replica with its active mirror (Experimental).

Positional arguments:

<replica> ID of the share replica to resync.

manila share-replica-show

```
usage: manila share-replica-show <replica>
```

Show details about a replica (Experimental).

Positional arguments:

<replica> ID of the share replica.

manila share-server-delete

```
usage: manila share-server-delete <id> [<id> ...]
```

Delete one or more share servers (Admin only).

Positional arguments:

<id> ID of the share server(s) to delete.

manila share-server-details

```
usage: manila share-server-details <id>
```

Show share server details (Admin only).

Positional arguments:

<id> ID of share server.

manila share-server-list

```
usage: manila share-server-list [--host <hostname>] [--status <status>]
                                [--share-network <share_network>]
                                [--project-id <project_id>]
                                [--columns <columns>]
```

List all share servers (Admin only).

Optional arguments:

--host <hostname> Filter results by name of host.

--status <status> Filter results by status.

--share-network <share_network> Filter results by share network.

--project-id <project_id> Filter results by project ID.

--columns <columns> Comma separated list of columns to be displayed example columns id,host,status.

manila share-server-show

```
usage: manila share-server-show <id>
```

Show share server info (Admin only).

Positional arguments:

<id> ID of share server.

manila show

```
usage: manila show <share>
```

Show details about a NAS share.

Positional arguments:

<share> Name or ID of the NAS share.

manila shrink

```
usage: manila shrink <share> <new_size>
```

Decreases the size of an existing share.

Positional arguments:

<share> Name or ID of share to shrink.

<new_size> New size of share, in GiBs.

manila snapshot-access-allow

```
usage: manila snapshot-access-allow <snapshot> <access_type> <access_to>
```

Allow read only access to a snapshot.

Positional arguments:

<snapshot> Name or ID of the share snapshot to allow access to.

<access_type> Access rule type (only ip, user(user or group), cert or cephx are supported).

<access_to> Value that defines access.

manila snapshot-access-deny

```
usage: manila snapshot-access-deny <snapshot> <id> [<id> ...]
```

Deny access to a snapshot.

Positional arguments:

<snapshot> Name or ID of the share snapshot to deny access to.

<id> ID(s) of the access rule(s) to be deleted.

manila snapshot-access-list

```
usage: manila snapshot-access-list [--columns <columns>] <snapshot>
```

Show access list for a snapshot.

Positional arguments:

<snapshot> Name or ID of the share snapshot to list access of.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns access_type,access_to.

manila snapshot-create

```
usage: manila snapshot-create [--force <True|False>] [--name <name>]
                              [--description <description>]
                              <share>
```

Add a new snapshot.

Positional arguments:

<share> Name or ID of the share to snapshot.

Optional arguments:

--force <True|False> Optional flag to indicate whether to snapshot a share even if its busy. (Default=False)

--name <name> Optional snapshot name. (Default=None)

--description <description> Optional snapshot description. (Default=None)

manila snapshot-delete

```
usage: manila snapshot-delete <snapshot> [<snapshot> ...]
```

Remove one or more snapshots.

Positional arguments:

<snapshot> Name or ID of the snapshot(s) to delete.

manila snapshot-export-location-list

```
usage: manila snapshot-export-location-list [--columns <columns>] <snapshot>
```

List export locations of a given snapshot.

Positional arguments:

<snapshot> Name or ID of the snapshot.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,path.

manila snapshot-export-location-show

```
usage: manila snapshot-export-location-show <snapshot> <export_location>
```

Show export location of the share snapshot.

Positional arguments:

<snapshot> Name or ID of the snapshot.

<export_location> ID of the share snapshot export location.

manila snapshot-force-delete

```
usage: manila snapshot-force-delete <snapshot> [<snapshot> ...]
```

Attempt force-deletion of one or more snapshots. Regardless of the state (Admin only).

Positional arguments:

<snapshot> Name or ID of the snapshot(s) to force delete.

manila snapshot-instance-export-location-list

```
usage: manila snapshot-instance-export-location-list [--columns <columns>]
                                                     <instance>
```

List export locations of a given snapshot instance.

Positional arguments:

<instance> Name or ID of the snapshot instance.

Optional arguments:

--columns <columns> Comma separated list of columns to be displayed example columns id,path,is_admin_only.

manila snapshot-instance-export-location-show

```
usage: manila snapshot-instance-export-location-show <snapshot_instance>
                                                     <export_location>
```

Show export location of the share instance snapshot.

Positional arguments:

<snapshot_instance> ID of the share snapshot instance.

<export_location> ID of the share snapshot instance export location.

manila snapshot-instance-list

```
usage: manila snapshot-instance-list [--snapshot <snapshot>]
                                     [--columns <columns>]
                                     [--detailed <detailed>]
```

List share snapshot instances.

Optional arguments:

--snapshot <snapshot> Filter results by share snapshot ID.

--columns <columns> Comma separated list of columns to be displayed example columns id.

--detailed <detailed> Show detailed information about snapshot instances. (Default=False)

manila snapshot-instance-reset-state

```
usage: manila snapshot-instance-reset-state [--state <state>]
                                            <snapshot_instance>
```

Explicitly update the state of a share snapshot instance.

Positional arguments:

<snapshot_instance> ID of the snapshot instance to modify.

Optional arguments:

--state <state> Indicate which state to assign the snapshot instance. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila snapshot-instance-show

```
usage: manila snapshot-instance-show <snapshot_instance>
```

Show details about a share snapshot instance.

Positional arguments:

<snapshot_instance> ID of the share snapshot instance.

manila snapshot-list

```
usage: manila snapshot-list [--all-tenants [<0|1>]] [--name <name>]
                            [--status <status>] [--share-id <share_id>]
                            [--usage [any|used|unused]] [--limit <limit>]
                            [--offset <offset>] [--sort-key <sort_key>]
                            [--sort-dir <sort_dir>] [--columns <columns>]
```

List all the snapshots.

Optional arguments:

--all-tenants [<0|1>] Display information from all tenants (Admin only).

--name <name> Filter results by name.

--status <status> Filter results by status.

--share-id <share_id>, --share_id <share_id> Filter results by source share ID.

--usage [any|used|unused] Either filter or not snapshots by its usage. OPTIONAL: Default=any.

--limit <limit> Maximum number of share snapshots to return. OPTIONAL: Default=None.

--offset <offset> Set offset to define start point of share snapshots listing. OPTIONAL: Default=None.

--sort-key <sort_key>, --sort_key <sort_key> Key to be sorted, available keys are (id, status, size, share_id, user_id, project_id, progress, name, display_name). Default=None.

--sort-dir <sort_dir>, --sort_dir <sort_dir> Sort direction, available values are (asc, desc). OPTIONAL: Default=None.

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila snapshot-manage

```
usage: manila snapshot-manage [--name <name>] [--description <description>]
                              [--driver_options [<key=value> [<key=value> ...
↪]]]
                              <share> <provider_location>
```

Manage share snapshot not handled by Manila (Admin only).

Positional arguments:

<share> Name or ID of the share.

<provider_location> Provider location of the snapshot on the backend.

Optional arguments:

--name <name> Optional snapshot name (Default=None).

--description <description> Optional snapshot description (Default=None).

--driver_options [<key=value> [<key=value> ...]], --driver-options [<key=value> [<key=value> Optional driver options as key=value pairs (Default=None).

manila snapshot-rename

```
usage: manila snapshot-rename [--description <description>]
                              <snapshot> [<name>]
```

Rename a snapshot.

Positional arguments:

<snapshot> Name or ID of the snapshot to rename.

<name> New name for the snapshot.

Optional arguments:

--description <description> Optional snapshot description. (Default=None)

manila snapshot-reset-state

```
usage: manila snapshot-reset-state [--state <state>] <snapshot>
```

Explicitly update the state of a snapshot (Admin only).

Positional arguments:

<snapshot> Name or ID of the snapshot to modify.

Optional arguments:

--state <state> Indicate which state to assign the snapshot. Options include available, error, creating, deleting, error_deleting. If no state is provided, available will be used.

manila snapshot-show

```
usage: manila snapshot-show <snapshot>
```

Show details about a snapshot.

Positional arguments:

<snapshot> Name or ID of the snapshot.

manila snapshot-unmanage

```
usage: manila snapshot-unmanage <snapshot> [<snapshot> ...]
```

Unmanage one or more share snapshots (Admin only).

Positional arguments:

<snapshot> Name or ID of the snapshot(s).

manila type-access-add

```
usage: manila type-access-add <share_type> <project_id>
```

Adds share type access for the given project (Admin only).

Positional arguments:

<share_type> Share type name or ID to add access for the given project.

<project_id> Project ID to add share type access for.

manila type-access-list

```
usage: manila type-access-list <share_type>
```

Print access information about the given share type (Admin only).

Positional arguments:

<share_type> Filter results by share type name or ID.

manila type-access-remove

```
usage: manila type-access-remove <share_type> <project_id>
```

Removes share type access for the given project (Admin only).

Positional arguments:

<share_type> Share type name or ID to remove access for the given project.

<project_id> Project ID to remove share type access for.

manila type-create

```
usage: manila type-create [--snapshot_support <snapshot_support>]
                          [--create_share_from_snapshot_support <create_share_
↪from_snapshot_support>]
                          [--revert_to_snapshot_support <revert_to_snapshot_
↪support>]
                          [--mount_snapshot_support <mount_snapshot_support>]
                          [--extra-specs [<key=value> [<key=value> ...]]]
                          [--is_public <is_public>]
                          <name> <spec_driver_handles_share_servers>
```

Create a new share type (Admin only).

Positional arguments:

<name> Name of the new share type.

<spec_driver_handles_share_servers> Required extra specification. Valid values are true/1 and false/0.

Optional arguments:

--snapshot_support <snapshot_support>, --snapshot-support <snapshot_support>
Boolean extra spec used for filtering of back ends by their capability to create share snapshots.

--create_share_from_snapshot_support <create_share_from_snapshot_support>, --create-share-fr
Boolean extra spec used for filtering of back ends by their capability to create shares from snapshots.

--revert_to_snapshot_support <revert_to_snapshot_support>, --revert-to-snapshot-support <rev
Boolean extra spec used for filtering of back ends by their capability to revert shares to snapshots. (Default is False).

--mount_snapshot_support <mount_snapshot_support>, --mount-snapshot-support <mount_snapshot_
Boolean extra spec used for filtering of back ends by their capability to mount share snapshots. (Default is False).

--extra-specs [<key=value> [<key=value> ...]], --extra_specs [<key=value> [<key=value> ...]]
Extra specs key and value of share type that will be used for share type creation. OPTIONAL: Default=None. example extra-specs thin_provisioning=<is> True, replication_type=readable.

--is_public <is_public>, --is-public <is_public> Make type accessible to the public (default true).

manila type-delete

```
usage: manila type-delete <id> [<id> ...]
```

Delete one or more specific share types (Admin only).

Positional arguments:

<id> Name or ID of the share type(s) to delete.

manila type-key

```
usage: manila type-key <stype> <action> [<key=value> [<key=value> ...]]
```

Set or unset extra_spec for a share type (Admin only).

Positional arguments:

<stype> Name or ID of the share type.

<action> Actions: set or unset.

<key=value> Extra_specs to set or unset (key is only necessary on unset).

manila type-list

```
usage: manila type-list [--all] [--columns <columns>]
```

Print a list of available share types.

Optional arguments:

--all Display all share types (Admin only).

--columns <columns> Comma separated list of columns to be displayed example columns id,name.

manila unmanage

```
usage: manila unmanage <share>
```

Unmanage share (Admin only).

Positional arguments:

<share> Name or ID of the share(s).

manila update

```
usage: manila update [--name <name>] [--description <description>]
                     [--is-public <is_public>]
                     <share>
```

Rename a share.

Positional arguments:

<share> Name or ID of the share to rename.

Optional arguments:

--name <name> New name for the share.

--description <description> Optional share description. (Default=None)

--is-public <is_public>, --is_public <is_public> Public share is visible for all tenants.

manila-manage

control and manage shared filesystems

Author openstack@lists.launchpad.net

Date 2014-06-11

Copyright OpenStack LLC

Version 2014.2

Manual section 1

Manual group shared filesystems

SYNOPSIS

manila-manage <category> <action> [<args>]

DESCRIPTION

manila-manage controls shared filesystems service. More information about OpenStack Manila is at https://wiki.openstack.org/wiki/Manila

OPTIONS

The standard pattern for executing a manila-manage command is: `manila-manage <category> <command> [<args>]`

For example, to obtain a list of all hosts: `manila-manage host list`

Run without arguments to see a list of available command categories: `manila-manage`

Categories are shell, logs, service, db, host, version and config. Detailed descriptions are below.

These sections describe the available categories and arguments for manila-manage.

Manila Db

`manila-manage db version`

Print the current database version.

`manila-manage db sync`

Sync the database up to the most recent version. This is the standard way to create the db as well.

`manila-manage db downgrade <version>`

Downgrade database to given version.

`manila-manage db stamp <version>`

Stamp database with given version.

`manila-manage db revision <message> <autogenerate>`

Generate new migration.

`manila-manage db purge <age_in_days>`

Purge deleted rows older than a given age from manila database tables. If age_in_days is not given or is specified as 0 all available rows will be deleted.

Manila Logs

`manila-manage logs errors`

Displays manila errors from log files.

`manila-manage logs syslog <number>`

Displays manila alerts from syslog.

Manila Shell

`manila-manage shell bpython`

Starts a new bpython shell.

`manila-manage shell ipython`

Starts a new ipython shell.

`manila-manage shell python`

Starts a new python shell.

`manila-manage shell run`

Starts a new shell using python.

`manila-manage shell script <path/scriptname>`

Runs the named script from the specified path with flags set.

Manila Host

`manila-manage host list`

Returns list of running manila hosts.

Manila Config

`manila-manage config list`

Returns list of currently set config options and its values.

Manila Service

`manila-manage service list`

Returns list of manila services.

Manila Version

`manila-manage version list`

Returns list of versions.

FILES

The manila-manage.conf file contains configuration information in the form of python-gflags.

BUGS

- Manila is sourced in Launchpad so you can view current bugs at OpenStack Manila

manila-status

Synopsis

```
manila-status <category> <command> [<args>]
```

Description

`manila-status` is a tool that provides routines for checking the status of a Manila deployment.

Options

The standard pattern for executing a **`manila-status`** command is:

```
manila-status <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
manila-status
```

Categories are:

- `upgrade`

Detailed descriptions are below.

You can also run with a category argument such as `upgrade` to see a list of all commands in that category:

```
manila-status upgrade
```

These sections describe the available categories and arguments for **`manila-status`**.

Upgrade

manila-status upgrade check Performs a release-specific readiness check before restarting services with new code. This command expects to have complete configuration and access to databases and services.

Return Codes

Return code	Description
0	All upgrade readiness checks passed successfully and there is nothing to do.
1	At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.
2	There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.
255	An unexpected error occurred.

History of Checks

8.0.0 (Stein)

- Placeholder to be filled in with checks as they are added in Stein.

3.4 Additional resources

- Manila release notes

CHAPTER

FOUR

FOR CONTRIBUTORS

If you are a `new contributor` *start here*.

4.1 Contributor/Developer Guide

In this section you will find information helpful for contributing to manila.

4.1.1 Basic Information

So You Want to Contribute

For general information on contributing to OpenStack, check out the contributor guide to get started. It covers all the basics that are common to all OpenStack projects: the accounts you need, the basics of interacting with our Gerrit review system, how we communicate as a community, etc.

Below will cover the more project specific information you need to get started with Manila (Shared File System service).

Where is the code?

manila

The OpenStack Shared File System Service
code: https://opendev.org/openstack/manila
docs: https://docs.openstack.org/manila/
api-ref: https://docs.openstack.org/api-ref/shared-file-system
release model: https://releases.openstack.org/reference/release_models.html#cycle-with-rc
Launchpad: https://launchpad.net/manila

python-manilaclient

Python client library for the OpenStack Shared File System Service API; includes standalone CLI shells and OpenStack client plugin and shell
code: https://opendev.org/openstack/python-manilaclient
docs: https://docs.openstack.org/python-manilaclient
release model:
https://releases.openstack.org/reference/release_models.html#cycle-with-intermediary
Launchpad: https://launchpad.net/python-manilaclient

manila-ui

OpenStack dashboard plugin for the Shared File System Service
code: https://opendev.org/openstack/manila-ui
docs: https://docs.openstack.org/manila-ui
release model:
https://releases.openstack.org/reference/release_models.html#cycle-with-intermediary
Launchpad: https://launchpad.net/manila-ui

manila-tempest-plugin

An OpenStack test integration (tempest) plugin containing API and scenario tests for the Shared File System Service
code: https://opendev.org/openstack/manila-tempest-plugin
release model: https://releases.openstack.org/reference/release_models.html#cycle-automatic
Launchpad: https://launchpad.net/manila

manila-image-elements

A Disk Image Builder project with scripts to build a bootable Linux image for testing and use by some Shared File System Service storage drivers including the Generic Driver
code: https://opendev.org/openstack/manila-tempest-plugin
release model: no releases
Launchpad: https://launchpad.net/manila

manila-test-image

A project with scripts to create a Buildroot based image to create a small bootable Linux image, primarily for the purposes of testing Manila
code: https://opendev.org/openstack/manila-image-elements
images: https://tarballs.opendev.org/openstack/manila-image-elements/
release model: no releases
Launchpad: https://launchpad.net/manila-image-elements

manila-specs

Design Specifications for the Shared File System service
code: https://opendev.org/openstack/manila-specs
published specs: https://specs.openstack.org/openstack/manila-specs/
release model: no releases
Launchpad: https://launchpad.net/manila

See the `CONTRIBUTING.rst` file in each code repository for more information about contributing to that specific deliverable. Additionally, you should look over the docs links above; most components have helpful developer information specific to that deliverable.

Manila and its associated projects follow a coordinated release alongside other OpenStack projects. Development cycles are code named. See the OpenStack Releases website for names and schedules of the current, past and future development cycles.

Communication

IRC

The team uses IRC extensively for communication and coordination of project activities. The IRC channel is `#openstack-manila` on OFTC. Contributors work in various timezones across the world; so many of them run IRC Bouncers and appear to be always online. If you ping someone, or raise a question on the IRC channel, someone will get back to you when they are back on their computer. Additionally, the IRC channel is logged, so if you ask a question when no one is around, you can check the log to see if it has been answered.

Team Meetings

We host a one-hour IRC based community meeting every Thursday at 1500 UTC on `#openstack-meeting-alt` channel. See the OpenStack meetings page for the most up-to-date meeting information and for downloading the ICS file to integrate this slot with your calendar. The community meeting is a good opportunity to gather the attention of multiple contributors synchronously. If you wish to do so, add a meeting topic along with your IRC nick to the Meeting agenda.

Mailing List

In addition to IRC, the team uses the OpenStack Discuss Mailing List for development discussions. This list is meant for communication about all things developing OpenStack; so we also use this list to engage with contributors across projects, and make any release cycle announcements. Since it is a wide distribution list, the use of subject line tags is encouraged to make sure you reach the right people. Prefix the subject line with `[manila]` when sending email that concern Manila on this list.

Other Communication Avenues

Contributors gather at least once per release at the OpenDev Project Team Gathering to discuss plans for an upcoming development cycle. This is usually where developers pool ideas and brainstorm features and bug fixes. We have had both virtual, and in-person Project Technical Gathering events in the past. Before every such event, we gather opinions from the community via IRC Meetings and the Mailing list on planning these Project Technical Gatherings.

We make extensive use of Etherpads. You can find some of them that the team used in the past in the project Wiki. To share code snippets or logs, we use PasteBin.

Contacting the Core Team

When you contribute patches, your change will need to be approved by one or more maintainers (collectively known as the Core Team).

Were always looking for more maintainers! If youre looking to help maintain Manila, express your interest to the existing core team. We have mentored many individuals for one or more development cycles and added them to the core team.

Any new core reviewer needs to be nominated to the team by an existing core reviewer by making a proposal on OpenStack Discuss Mailing List. Other maintainers and contributors can then express their

approval or disapproval by responding to the proposal. If there is a decision, the project team lead will add the concerned individual to the core reviewers team. An example proposal is here.

New Feature Planning

If youd like to propose a new feature, do so by creating a blueprint on Launchpad. For significant changes we might require a design specification.

Feature changes that need a specification include:

- Adding new API methods
- Substantially modifying the behavior of existing API methods
- Adding a new database resource or modifying existing resources
- Modifying a share back end driver interface, thereby affecting all share back end drivers

What doesnt need a design specification:

- Making trivial (backwards compatible) changes to the behavior of an existing API method. Examples include adding a new field to the response schema of an existing method, or introducing a new query parameter. See *API Microversions* on how Manila APIs are versioned.
- Adding new share back end drivers or modifying share drivers, without affecting the share back end driver interface
- Adding or changing tests

After filing a blueprint, if youre in doubt whether to create a design specification, contact the maintainers.

Design specifications are tracked in the Manila Specifications repository and are published on the OpenStack Project Specifications website. Refer to the specification template to structure your design spec.

Specifications and new features have deadlines. Usually, specifications for an upcoming release are frozen midway into the release development cycle. To determine the exact deadlines, see the published release calendars by navigating to the specific release from the OpenStack releases website.

Task Tracking

- We track our bugs in Launchpad:

  https://bugs.launchpad.net/manila

  If youre looking for some smaller, easier work item to pick up and get started on, search for the low-hanging-fruit tag

- We track future features as blueprints on Launchpad:

  https://blueprints.launchpad.net/manila

- Unimplemented specifications are tracked here:

  https://specs.openstack.org/openstack/manila-specs/#unimplemented-specs

These specifications need a new owner. If youre interested to pick them up and drive them to completion, you can update the corresponding blueprint and get in touch with the project maintainers for help

Reporting a Bug

You found an issue and want to make sure we are aware of it? You can do so on Launchpad.

Getting Your Patch Merged

When you submit your change through Gerrit, a number of automated Continuous Integration tests are run on your change. A change must receive a +1 vote from the OpenStack CI system in order for it to be merge-worthy. If these tests are failing and you cant determine why, contact the maintainers.

See the *Manila team code review policy* to understand our code review conventions. Generally, reviewers look at new code submissions pro-actively; if you do not have sufficient attention to your change, or are looking for help, do not hesitate to jump into the teams IRC channel, or bring our attention to your issue during a community meeting. The core team would prefer to have an open discussion instead of a one-on-one/private chat.

Project Team Lead Duties

A project team lead is elected from the project contributors each cycle. Manila Project specific responsibilities for a lead are listed in the *Manila Project Team Lead guide*.

4.1.2 Programming HowTos and Tutorials

Setting Up a Development Environment

This page describes how to setup a working Python development environment that can be used in developing manila on Ubuntu, Fedora or Mac OS X. These instructions assume youre already familiar with git. Refer to Getting the code for additional information.

Following these instructions will allow you to run the manila unit tests. If you want to be able to run manila (i.e., create NFS/CIFS shares), you will also need to install dependent projects: nova, neutron, cinder and glance. For this purpose devstack project can be used (A documented shell script to build complete OpenStack development environments). You can check out Setting up a development environment with devstack for instructions on how to enable manila on devstack.

Virtual environments

Manila development uses virtualenv to track and manage Python dependencies while in development and testing. This allows you to install all of the Python package dependencies in a virtual environment or virtualenv (a special subdirectory of your manila directory), instead of installing the packages at the system level.

Note: Virtualenv is useful for running the unit tests, but is not typically used for full integration testing or production usage.

Linux Systems

Note: This section is tested for manila on Ubuntu and Fedora-based distributions. Feel free to add notes and change according to your experiences or operating system.

Install the prerequisite packages.

- On Ubuntu/Debian:

```
sudo apt-get install python-dev libssl-dev python-pip \
libmysqlclient-dev libxml2-dev libxslt-dev libpq-dev git \
git-review libffi-dev gettext graphviz libjpeg-dev
```

- On RHEL8/Centos8:

```
sudo dnf install openssl-devel python3-pip mysql-devel \
libxml2-devel libxslt-devel postgresql-devel git git-review \
libffi-devel gettext graphviz gcc libjpeg-turbo-devel \
python3-tox python3-devel python3
```

Note: If using RHEL and yum reports No package python3-pip available and No package git-review available, use the EPEL software repository. Instructions can be found at http://fedoraproject.org/wiki/EPEL/FAQ#howtouse.

- On Fedora 22 and higher:

```
sudo dnf install python-devel openssl-devel python-pip mysql-devel \
libxml2-devel libxslt-devel postgresql-devel git git-review \
libffi-devel gettext graphviz gcc libjpeg-turbo-devel \
python-tox python3-devel python3
```

Note: Additionally, if using Fedora 23, `redhat-rpm-config` package should be installed so that development virtualenv can be built successfully.

Mac OS X Systems

Install virtualenv:

```
sudo easy_install virtualenv
```

Check the version of OpenSSL you have installed:

```
openssl version
```

If you have installed OpenSSL 1.0.0a, which can happen when installing a MacPorts package for OpenSSL, you will see an error when running `manila.tests.auth_unittest.AuthTestCase.test_209_can_generate_x509`.

The stock version of OpenSSL that ships with Mac OS X 10.6 (OpenSSL 0.9.8l) or Mac OS X 10.7 (OpenSSL 0.9.8r) works fine with manila.

Getting the code

Grab the code:

```
git clone https://opendev.org/openstack/manila
cd manila
```

Running unit tests

The preferred way to run the unit tests is using `tox`. Tox executes tests in isolated environment, by creating separate virtualenv and installing dependencies from the `requirements.txt` and `test-requirements.txt` files, so the only package you install is `tox` itself:

```
sudo pip install tox
```

Run the unit tests with:

```
tox -e py{python-version}
```

Example:

```
tox -epy36
```

See *Unit Tests* for more details.

Manually installing and using the virtualenv

You can also manually install the virtual environment:

```
tox -epy36 --notest
```

This will install all of the Python packages listed in the `requirements.txt` file into your virtualenv.

To activate the Manila virtualenv you can run:

```
$ source .tox/py36/bin/activate
```

To exit your virtualenv, just type:

```
$ deactivate
```

Or, if you prefer, you can run commands in the virtualenv on a case by case basis by running:

```
$ tox -e venv -- <your command>
```

Contributing Your Work

Once your work is complete you may wish to contribute it to the project. Manila uses the Gerrit code review system. For information on how to submit your branch to Gerrit, see GerritWorkflow.

Setting up a development environment with devstack

This page describes how to setup a working development environment that can be used in deploying `manila` and `manila-ui` on latest releases of Ubuntu, Fedora or CentOS. These instructions assume you are already familiar with git.

We recommend using devstack to develop and test code changes to `manila` and/or `manila-ui`, in order to simply evaluate the manila and/or project. Devstack is a shell script to build complete OpenStack development environments on a virtual machine. If you are not familiar with devstack, these pages can give you context:

- Testing Changes with DevStack
- Devstack project documentation

Be aware that `manila` and `manila-ui` are not enabled in devstack by default; you will need to add a few lines to the devstack `local.conf` file to let devstack deploy and configure `manila` and `manila-ui` on your virtual machine.

Note: If you do not intend to deploy with the OpenStack Dashboard (horizon) service, you can ignore instructions about enabling `manila-ui`.

Getting devstack

Start by cloning the devstack repository:

```
git clone https://opendev.org/openstack/devstack
```

Change to devstack directory:

```
cd devstack/
```

Youre now on `master` branch of devstack, switch to the branch you want to test or develop against.

Sample local.conf files that get you started

Now that you have cloned the devstack repository, you need to configure devstack before deploying it. This is done with a `local.conf` file. For manila, the local.conf file can also determine which back end(s) are set up.

Caution: When using devstack with the below configurations, be aware that you will be setting up fake storage. The *LVM*, *Generic*, *ZFSOnLinux* drivers have not been developed for production use. They exist to provide a vanilla development and testing environment for manila contributors.

DHSS=False (*driver_handles_share_servers=False*) mode:

This is the easier mode for new contributors. Manila share back-end drivers that operate in `driver_handles_share_servers=False` mode do not allow creating shares on private project networks. On the resulting stack, all manila shares created by you are exported on the host network and hence are accessible to any compute resource (e.g.: virtual machine, baremetal, container) that is able to reach the devstack host.

- `LVM driver`
- `ZFSOnLinux driver`
- `CEPHFS driver`

DHSS=True (*driver_handles_share_servers=True*) mode:

You may use the following setups if you are familiar with manila, and would like to test with the project (tenant) isolation that manila provides on the network and data path. Manila share back-end drivers that operate in `driver_handles_share_servers=True` mode create shares on isolated project networks if told to do so. On the resulting stack, when creating a share, you must specify a share network to export the share to, and the share will be accessible to any compute resource (e.g.: Virtual machine, baremetal, containers) that is able to reach the share network you indicated.

Typically, new contributors take a while to understand OpenStack networking, and we recommend that you familiarize yourself with the `DHSS=False` mode setup before attempting `DHSS=True`.

- `Generic driver`

- `Container driver`

Building your devstack

- Copy the appropriate sample local.conf file into the devstack folder on your virtual machine, make sure to name it `local.conf`
- Make sure to read inline comments and customize values where necessary
- If you would like to run minimal services in your stack, or allow devstack to bootstrap tempest testing framework for you, see *More devstack customizations*
- Finally, run the `stack.sh` script from within the devstack directory. We recommend that your run this inside a screen or tmux session because it could take a while:

```
./stack.sh
```

- After the script completes, you should have manila services running. You can verify that the services are running with the following commands:

```
$ systemctl status devstack@m-sch
$ systemctl status devstack@m-shr
$ systemctl status devstack@m-dat
```

- By default, devstack sets up manila-api behind apache. The service name is `httpd` on Red Hat based systems and `apache2` on Debian based systems.
- You may also use your demo credentials to invoke the command line clients:

```
$ source DEVSTACK_DIR/openrc admin demo
$ manila service-list
```

- The logs are accessible through `journalctl`. The following commands let you query logs. You may use the `-f` option to tail these logs:

```
$ journalctl -a -o short-precise --unit devstack@m-sch
$ journalctl -a -o short-precise --unit devstack@m-shr
$ journalctl -a -o short-precise --unit devstack@m-dat
```

- If running behind apache, the manila-api logs will be in `/var/log/httpd/manila_api.log` (Red Hat) or in `/var/log/apache2/manila_api.log` (Debian).
- Manila UI will now be available through OpenStack Horizon; look for the Shares tab under Project > Share.

More devstack customizations

Testing branches and changes submitted for review

To test a patch in review:

```
enable_plugin manila https://opendev.org/openstack/manila <ref>
```

If the ref is from review.opendev.org, it is structured as:

```
refs/changes/<last two digits of review number>/<review number>/<patchset␣
↪number>
```

For example, if you want to test patchset 4 of https://review.opendev.org/#/c/614170/, you can provide this in your `local.conf`:

```
enable_plugin manila https://opendev.org/openstack/manila refs/changes/70/
↪614170/4
```

ref can also simply be a stable branch name, for example:

```
enable_plugin manila https://opendev.org/openstack/manila stable/train
```

Limiting the services enabled in your stack

Manila needs only a message queue (rabbitmq) and a database (mysql, postgresql) to operate. Additionally, keystone service provides project administration if necessary, all other OpenStack services are not necessary to set up a basic test system.[1][2]

You can add the following to your `local.conf` to deploy your stack in a minimal fashion. This saves you a lot of time and resources, but could limit your testing:

```
ENABLED_SERVICES=key,mysql,rabbit,tempest,manila,m-api,m-sch,m-shr,m-dat
```

Optionally, you can deploy with Manila, Nova, Neutron, Glance and Tempest:

```
ENABLED_SERVICES=key,mysql,rabbit,tempest,g-api
ENABLED_SERVICES+=n-api,n-cpu,n-cond,n-sch,n-crt,n-cauth,n-obj,placement-api,
↪placement-client
ENABLED_SERVICES+=q-svc,q-dhcp,q-meta,q-l3,q-agt
ENABLED_SERVICES+=tempest
```

You can also enable `tls-proxy` with `ENABLED_SERVICES` to allow devstack to use Apache and setup a TLS proxy to terminate TLS connections. Using tls-proxy secures all OpenStack service API endpoints and inter-service communication on your devstack.

[1] The Generic driver cannot be run without deploying Cinder, Nova, Glance and Neutron.

[2] You must enable Horizon to use manila-ui. Horizon will not work well when Nova, Cinder, Glance and Neutron are not enabled.

Bootstrapping Tempest

Add the following options in your `local.conf` to set up tempest:

```
ENABLE_ISOLATED_METADATA=True
TEMPEST_USE_TEST_ACCOUNTS=True
TEMPEST_ALLOW_TENANT_ISOLATION=False
TEMPEST_CONCURRENCY=8
```

Running manila API with a web server

As part of the community goals for Pike, manila has packaged a wsgi script entrypoint that allows you to run it with a real web server like Apache HTTPD or NGINX.

This doc shows a sample of deploying manila with uwsgi

Installing the API via uwsgi

For this deployment we use uwsgi as a web server bound to a random local port. Then we configure apache using mod_proxy to forward all incoming requests on the specified endpoint to that local webserver. This has the advantage of letting apache manage all inbound http connections, but allowing uwsgi run the python code. This also means that when we make changes to manila code or configuration we dont need to restart all of apache (which may be running other services as well) and just need to restart the local uwsgi daemon.

The httpd/ directory contains sample files for configuring HTTPD to run manila under uwsgi. To use sample configs, simply copy *httpd/uwsgi-manila.conf* to the appropiate location for your apache server.

On RHEL/CentOS/Fedora it is:

```
/etc/httpd/conf.d/uwsgi-manila.conf
```

On SLES/OpenSUSE it is:

```
/etc/apache2/vhosts.d/uwsgi-manila.conf
```

On Debian/Ubuntu it is:

```
/etc/apache2/sites-available/uwsgi-manila.conf
```

Enable mod_proxy by running `sudo a2enmod proxy`

On Ubuntu/Debian systems enable the site using the a2ensite tool:

```
sudo a2ensite /etc/apache2/sites-available/uwsgi-manila.conf
```

This is not required on RHEL/CentOS/Fedora systems.

Start or restart HTTPD/Apache2 to pick up the new configuration.

Now we have to configure and start the uwsgi service. Copy the *httpd/manila-uwsgi.ini* file to */etc/manila*. Update the file to match your system configuration (i.e. tweak the number of processes and threads)

Install uwsgi.

On RHEL/CentOS:

```
sudo yum install uwsgi-plugin-python3
```

On Fedora:

```
sudo dnf install uwsgi-plugin-python3
```

On SLES/OpenSUSE:

```
sudo zypper install uwsgi-python3
```

On Ubuntu/Debian:

```
sudo apt-get install uwsgi-plugin-python3
```

And start the manila server using uwsgi:

```
uwsgi --ini /etc/manila/manila-uwsgi.ini
```

Note: In the sample configs port 51999 is used, this is a randomly selected number.

Installing the API via mod_wsgi

The httpd/ directory contains sample files for configuring HTTPD to run manila API via mod_wsgi. To use sample configs, simply copy *httpd/mod_wsgi-manila.conf* to the appropiate location for your apache server.

On RHEL/CentOS/Fedora it is:

```
/etc/httpd/conf.d/mod_wsgi-manila.conf
```

On SLES/OpenSUSE it is:

```
/etc/apache2/vhosts.d/mod_wsgi-manila.conf
```

On Debian/Ubuntu it is:

```
/etc/apache2/sites-available/mod_wsgi-manila.conf
```

On Ubuntu/Debian systems enable the site using the a2ensite tool:

```
sudo a2ensite /etc/apache2/sites-available/mod_wsgi-manila.conf
```

This is not required on RHEL/CentOS/Fedora systems.

Start or restart HTTPD/Apache2 to pick up the new configuration.

Note: manilas primary configuration file (etc/manila.conf) and the PasteDeploy configuration file

(etc/manila-paste.ini) must be readable to httpd in one of the default locations described in Configuring Manila.

Access Control

If you are running with Linux kernel security module enabled (for example SELinux or AppArmor), make sure that the configuration file has the appropriate context to access the linked file.

Unit Tests

Manila contains a suite of unit tests, in the manila/tests directory.

Any proposed code change will be automatically rejected by the OpenStack Jenkins server if the change causes unit test failures.

Running the tests

To run all unit tests simply run:

```
tox
```

This will create a virtual environment, load all the packages from test-requirements.txt and run all unit tests as well as run flake8 and hacking checks against the code.

You may run individual test targets, for example only unit tests, by running:

```
tox -e py3
```

Note that you can inspect the tox.ini file to get more details on the available options and what the test run does by default.

Running a subset of tests

Instead of running all tests, you can specify an individual directory, file, class, or method that contains test code.

To run the tests in the `manila/tests/scheduler` directory:

```
tox -epy3 -- manila.tests.scheduler
```

To run the tests in the *ShareManagerTestCase* class in `manila/tests/share/test_manager.py`:

```
tox -epy3 -- manila.tests.share.test_manager.ShareManagerTestCase
```

To run the *ShareManagerTestCase::test_share_manager_instance* test method in `manila/tests/share/test_manager.py`:

```
tox -epy3 -- manila.tests.share.test_manager.ShareManagerTestCase.test_share_
↪manager_instance
```

For more information on these options and details about stestr, please see the stestr documentation.

Database Setup

Some unit tests will use a local database. You can use `tools/test-setup.sh` to set up your local system the same way as its setup in the CI environment.

Gotchas

Running Tests from Shared Folders

If you are running the unit tests from a shared folder, you may see tests start to fail or stop completely as a result of Python lockfile issues[3]. You can get around this by manually setting or updating the following line in `manila/tests/conf_fixture.py`:

```
FLAGS['lock_path'].SetDefault('/tmp')
```

Note that you may use any location (not just `/tmp`!) as long as it is not a shared folder.

Tempest Tests

Manilas functional API and scenario tests are in the manila tempest plugin repository.

Installation of plugin to tempest

Tempest plugin installation is common for all its plugins and detailed information can be found in its docs. In simple words: if you have installed manila project on the same machine as tempest, then tempest will find it.

In case the plugin is not installed (see the verification steps below), you can clone and install it yourself.

```
$ git clone https://opendev.org/openstack/manila-tempest-plugin
$ pip install -e manila-tempest-plugin
```

Verifying installation

To verify that the plugin is installed on your system, run the following command and find manila_tests in its output.

```
$ tempest list-plugins
```

Alternatively, or to double-check, list all the tests available on the system and find manila tests in it.

```
$ tempest run -l
```

[3] See Vishs comment in this bug report: https://bugs.launchpad.net/manila/+bug/882933

Configuration of manila-related tests in tempest.conf

All config options for manila are defined in manila_tempest_tests/config.py module. They can be set/redefined in `tempest.conf` file.

Here is a configuration example:

```
[service_available]
manila = True

[share]
# Capabilities
capability_storage_protocol = NFS
capability_snapshot_support = True
capability_create_share_from_snapshot_support = True
backend_names = Backendname1,BackendName2
backend_replication_type = readable

# Enable/Disable test groups
multi_backend = True
multitenancy_enabled = True
enable_protocols = nfs,cifs,glusterfs,cephfs
enable_ip_rules_for_protocols = nfs
enable_user_rules_for_protocols = cifs
enable_cert_rules_for_protocols = glusterfs
enable_cephx_rules_for_protocols = cephfs
username_for_user_rules = foouser
enable_ro_access_level_for_protocols = nfs
run_quota_tests = True
run_extend_tests = True
run_shrink_tests = True
run_snapshot_tests = True
run_replication_tests = True
run_migration_tests = True
run_manage_unmanage_tests = True
run_manage_unmanage_snapshot_tests = True
```

Note: None of existing share drivers support all features. So, make sure that share backends really support features you enable in config. See the *Manila share features support mapping* to see what features are supported by the back end that you are testing.

Running tests

To run tests, it is required to install pip, tox and virtualenv packages on host machine. Then run following command from tempest root directory:

```
$ tempest run -r manila_tempest_tests.tests.api
```

or to run only scenario tests:

```
$ tempest run -r manila_tempest_tests.tests.scenario
```

Running a subset of tests based on test location

Instead of running all tests, you can specify an individual directory, file, class, or method that contains test code.

To run the tests in the `manila_tempest_tests/tests/api/admin` directory:

```
$ tempest run -r manila_tempest_tests.tests.api.admin
```

To run the tests in the `manila_tempest_tests/tests/api/admin/test_admin_actions.py` module:

```
$ tempest run -r manila_tempest_tests.tests.api.admin.test_admin_actions
```

To run the tests in the *AdminActionsTest* class in `manila_tempest_tests/tests/api/admin/test_admin_actions.py` module:

```
$ tempest run -r manila_tempest_tests.tests.api.admin.test_admin_actions.
↪AdminActionsTest
```

To run the *AdminActionsTest.test_reset_share_state* test method in `manila_tempest_tests/tests/api/admin/test_admin_actions.py` module:

```
$ tempest run -r manila_tempest_tests.tests.api.admin.test_admin_actions.
↪AdminActionsTest.test_reset_share_state
```

Running a subset of tests based on service involvement

To run the tests that require only *manila-api* service running:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\bapi\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

To run the tests that require all manila services running, but intended to test API behaviour:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\b\(api\|api_with_backend\)\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

To run the tests that require all manila services running, but intended to test back-end (manila-share) behaviour:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\bbackend\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

Running a subset of positive or negative tests

To run only positive tests, use following command:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\bpositive\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

To run only negative tests, use following command:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\bnegative\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

To run only positive API tests, use following command:

```
$ tempest run -r \
  \(\?\=\.\*\\\[\.\*\\bpositive\\b\.\*\\\]\) \
  \(\?\=\.\*\\\[\.\*\\bapi\\b\.\*\\\]\) \
  \(\^manila_tempest_tests.tests.api\)
```

Adding a Method to the OpenStack Manila API

The interface to manila is a RESTful API. REST stands for Representational State Transfer and provides an architecture style for distributed systems using HTTP for transport. Figure out a way to express your request and response in terms of resources that are being created, modified, read, or destroyed. Manilas API aims to conform to the guidelines set by OpenStack API SIG.

Routing

To map URLs to controllers+actions, manila uses the Routes package. See the routes package documentation for more information.

URLs are mapped to action methods on controller classes in `manila/api/<VERSION>/router.py`.

These are two methods of the routes package that are used to perform the mapping and the routing:

- mapper.connect() lets you map a single URL to a single action on a controller.
- mapper.resource() connects many standard URLs to actions on a controller.

Controllers and actions

Controllers live in `manila/api/v1` and `manila/api/v2`.

See `manila/api/v1/shares.py` for an example.

Action methods take parameters that are sucked out of the URL by mapper.connect() or .resource(). The first two parameters are self and the WebOb request, from which you can get the req.environ, req.body, req.headers, etc.

Actions return a dictionary, and wsgi.Controller serializes that to JSON.

Faults

If you need to return a non-200, you should return faults.Fault(webob.exc .HTTPNotFound()) replacing the exception as appropriate.

Evolving the API

The `v1` version of the manila API has been deprecated. The `v2` version of the API supports micro versions. So all changes to the v2 API strive to maintain stability at any given API micro version, so consumers can safely rely on a specific micro version of the API never to change the request and response semantics. Read more about *API Microversions* to understand how stability and backwards compatibility are maintained.

Documenting your work

As with most OpenStack services and libraries, manila suffers from appearing very complicated to understand, develop, deploy, administer and use. As OpenStack developers working on manila, our responsibility goes beyond introducing new features and maintaining existing features. We ought to provide adequate documentation for the benefit of all kinds of audiences. The guidelines below will explain how you can document (or maintain documentation for) new (or existing) features and bug fixes in the core manila project and other projects that are part of the manila suite.

Where to add documentation?

OpenStack User Guide

- Any documentation targeted at end users of manila in OpenStack needs to go here. This contains high level information about any feature as long as it is available on `python-manilaclient` and/or `manila-ui`.
- If you develop an end user facing feature, you need to provide an overview, use cases and example work-flows as part of this documentation.
- The source files for the user guide live in manilas code tree.
- **Link**: User guide

OpenStack Administrator Guide

- Documentation for administrators of manila deployments in OpenStack clouds needs to go here.
- Document instructions for administrators to perform necessary set up for utilizing a feature, along with managing and troubleshooting manila when the feature is used.
- Relevant configuration options may be mentioned here briefly.
- The source files for the administrator guide live in manilas code tree.
- **Link**: Administrator guide

OpenStack Configuration Reference

- Instructions regarding configuration of different manila back ends need to be added in this document.
- The configuration reference also contains sections where manilas configuration options are auto-documented.
- It contains sample configuration files for using manila with various configuration options.
- If you are a driver maintainer, please ensure that your driver and all of its relevant configuration is documented here.
- The source files for the configuration guide live in manilas code tree.
- **Link**: Manila release configuration reference

OpenStack Installation Tutorial

- Instructions regarding setting up manila on OpenStack need to be documented here.
- This tutorial covers step-by-step deployment of OpenStack services using a functional example architecture suitable for new users of OpenStack with sufficient Linux experience.
- The instructions are written with reference to different distributions.
- The source files for this tutorial live in manilas code tree.
- **Link**: Draft installation tutorial

OpenStack API Reference

- When you add or change a REST API in manila, you will need to add or edit descriptions of the API, request and response parameters, microversions and expected HTTP response codes as part of the API reference.
- For releases prior to Newton, the API reference was maintained in Web Application Description Language (WADL) in the api-site project.
- Since the Newton release, manilas API reference is maintained in-tree in custom YAML/JSON format files.
- **Link**: REST API reference of the Shared File Systems Project v2.0

Manila Developer Reference

- When working on a feature in manila, provide judicious inline documentation in the form of comments and docstrings. Code is our best developer reference.
- Driver entry-points must be documented with docstrings explaining the expected behavior from a driver routine.
- Apart from inline documentation, further developer facing documentation will be necessary when you are introducing changes that will affect vendor drivers, consumers of the manila database and when building a utility in manila that can be consumed by other developers.
- The developer reference for manila is maintained in-tree.
- Feel free to use it as a sandbox for other documentation that does not live in manilas code-tree.
- **Link**: Manila developer reference

OpenStack Security Guide

- Any feature that has a security impact needs to be documented here.
- In general, administrators will follow the guidelines regarding best practices of setting up their manila deployments with this guide.
- Any changes to `policy.yaml` based authorization, share network related security, `access` to manila resources, tenant and user related information needs to be documented here.
- **Link**: Security guide
- **Repository**: The security guide is maintained within the OpenStack Security-doc project

OpenStack Command Line Reference

- Help text provided in the `python-manilaclient` is extracted into this document automatically.
- No manual corrections are allowed on this repository; make necessary corrections in the `python-manilaclient` repository.
- **Link**: Manila CLI reference.

Important things to note

- When implementing a new feature, use appropriate Commit Message Tags (*Using Commit Message Tags in Manila*).
- Using the `DocImpact` flag in particular will create a `[doc]` bug under the manila project in launchpad. When your code patch merges, assign this bug to yourself and track your documentation changes with it.
- When writing documentation outside of manila, use either a commit message header that includes the word `Manila` or set the topic of the change-set to `manila-docs`. This will make it easy for manila reviewers to find your patches to aid with a technical content review.

- When writing documentation in user/admin/config/api/install guides, *always* refer to the project with its service name: `Shared File Systems service` and not the service type (`share`) or the project name (`manila`).
- Follow documentation styles prescribed in the OpenStack Documentation Contributor Guide. Pay heed to the RST formatting conventions and Writing style.
- Use CamelCase to spell out *OpenStack* and sentence casing to spell out service types, ex: *Shared File Systems service* and lower case to spell out project names, ex: *manila* (except when the project name is in the beginning of a sentence or a title).
- **ALWAYS** use a first party driver when documenting a feature in the user or administrator guides. Provide cross-references to configuration reference sections to lead readers to detailed setup instructions for these drivers.
- The manila developer reference, the OpenStack user guide, administrator reference, API reference and security guide are always *current*, i.e, get built with every commit in the respective codebase. Therefore, documentation added here need not be backported to previous releases.
- You may backport changes to some documentation such as the configuration reference and the installation guide.
- **Important documentation that isnt really documentation** - `specs` and `release notes` are *NOT* documentation. A specification document is written to initiate a dialogue and gather feedback regarding the design of a feature. Neither developers nor users will regard a specification document as official documentation after a feature has been implemented. Release notes (*Release Notes*) allow for gathering release summaries and they are not used to understand, configure, use or troubleshoot any manila feature.
- **Less is not more, more is more** - Always add detail when possible. The health and maturity of our community is reflected in our documentation.

Release Notes

What are release notes?

Release notes are important for change management within manila. Since manila follows a release cycle with milestones, release notes provide a way for the community and users to quickly grasp what changes occurred within a development milestone. To the OpenStack release management and documentation teams, release notes are a way to compile changes per milestone. These notes are published on the OpenStack Releases website. Automated tooling is built around `releasenotes` and they get appropriately handled per release milestone, including any back-ports to stable releases.

What needs a release note?

- Changes that impact an upgrade, most importantly, those that require a deployer to take some action while upgrading
- API changes
  - New APIs
  - Changes to the response schema of existing APIs
  - Changes to request/response headers

  - Non-trivial API changes such as response code changes from 2xx to 4xx
  - Deprecation of APIs or response fields
  - Removal of APIs
- A new feature is implemented, such as a new core feature in manila, driver support for an existing manila feature or a new driver
- An existing feature is deprecated
- An existing feature is removed
- Behavior of an existing feature has changed in a discernible way to an end user or administrator
- Backend driver interface changes
- A security bug is fixed
- New configuration option is added

What does not need a release note?

- A code change that doesnt change the general behavior of any feature such as code refactor or logging changes. One case of this could be the exercise that all drivers went through by removing `allow_access` and `deny_access` interfaces in favor of an `update_access` interface as suggested in the Mitaka release.
- Tempest or unit test coverage enhancement
- Changes to response message with API failure codes 4xx and 5xx
- Any change submitted with a justified TrivialFix flag added in the commit message
- Adding or changing documentation within in-tree documentation guides

How do I add a release note?

We use Reno to create and manage release notes. The new subcommand combines a random suffix with a slug value to make the new file with a unique name that is easy to identify again later. To create a release note for your change, use:

```
$ reno new slug-goes-here
```

If reno is not installed globally on your system, you can use it from venv of your manilas tox. Prior to running the above command, run:

```
$ source .tox/py3/bin/activate
```

Or directly as a one-liner, with:

```
$ tox -e venv -- reno new slug-goes-here
```

Note: When you are adding a bug-fix reno, name your file using the template: bug-<launchpad-bug-id>-slug-goes-here.

Then add the notes in `yaml` format in the file created. Pay attention to the type of section. The following are general sections to use:

prelude

> General comments about the change. The prelude from all notes in a release are combined, in note order, to produce a single prelude introducing the release.

features

> New features introduced

issues

> A list of known issues with respect to the change being introduced. For example, if the new feature in the change is experimental or known to not work in some cases, it should be mentioned here.

upgrade

> A list of upgrade notes in the release. Any removals that affect upgrades are to be noted here.

deprecations

> Any features, APIs, configuration options that the change has deprecated. Deprecations are not removals. Deprecations suggest that there will be support for a certain timeline. Deprecation should allow time for users to make necessary changes for the removal to happen in a future release. It is important to note the timeline of deprecation in this section.

critical

> A list of *fixed* critical bugs (descriptions only).

security

> A list of *fixed* security issues (descriptions only).

fixes

> A list of other *fixed* bugs (descriptions only).

other

> Other notes that are important but do not fall into any of the given categories.

```
---
prelude: >
    Replace this text with content to appear at the
    top of the section for this change.
features:
  - List new features here, or remove this section.
issues:
  - List known issues here, or remove this section.
upgrade:
  - List upgrade notes here, or remove this section.
deprecations:
  - List deprecation notes here, or remove this section
critical:
  - Add critical notes here, or remove this section.
```

(continues on next page)

(continued from previous page)

```
security:
  - Add security notes here, or remove this section.
fixes:
  - Add normal bug fixes here, or remove this section.
other:
  - Add other notes here, or remove this section.
```

Dos and Donts

- Release notes need to be succinct. Short and unambiguous descriptions are preferred
- Write in past tense, unless you are writing an imperative statement
- Do not have blank sections in the file
- Do not include code or links
- Avoid special rst formatting unless absolutely necessary
- Always prefer including a release note in the same patch
- Release notes are not a replacement for developer/user/admin documentation
- Release notes are not a way of conveying behavior of any features or usage of any APIs
- Limit a release note to fewer than 2-3 lines per change per section
- OpenStack prefers atomic changes. So remember that your change may need the fewest sections possible
- General writing guidelines can be found here
- Proofread your note. Pretend you are a user or a deployer who is reading the note after a milestone or a release has been cut

Examples

The following need only be considered as directions for formatting. They are **not** fixes or features in manila.

- *fix-failing-automount-23aef89a7e98c8.yaml*

```
---
deprecations:
 - displaying mount options via the array listing API is deprecated.
fixes:
 - users can mount shares on debian systems with kernel version 32.2.41.*
   with share-mount API
```

- *add-librsync-backup-plugin-for-m-bkup-41cad17c1498a3.yaml*

```
---
features:
```

(continues on next page)

(continued from previous page)

```
  - librsync support added for NFS incremental backup
upgrade:
  - Copy new rootwrap.d/librsync.filters file into /etc/manila/rootwrap.d
    directory.
issues:
  - librsync has not been tested thoroughly in all operating systems that
    manila is qualified for. m-bkup is an experimental feature.
```

Using Commit Message Tags in Manila

When writing git commit messages for code submissions into manila, it can be useful to provide tags in the message for both human consumption as well as linking to other external resources, such as Launchpad. Each tag should be placed on a separate line. The following tags are used in manila.

- **APIImpact** - Use this tag when the code change modifies a public HTTP API interface. This tag indicates that the patch creates, changes, or deletes a public API interface or changes its behavior. The tag may be followed by a reason beginning on the next line. If you are touching manilas API layer and you are unsure if your change has an impact on the API, use this tag anyway.
- **Change-id** - This tag is automatically generated by a Gerrit hook and is a unique hash that describes the change. This hash should not be changed when rebasing as it is used by Gerrit to keep track of the change.
- **Closes-Bug: | Partial-Bug: | Related-Bug:** *<#launchpad_bug_id>* - These tags are used when the change closes, partially closes, or relates to the bug referenced by the Launchpad bug ID respectively. This will automatically generate a link to the bug in Launchpad for easy access for reviewers.
- **DocImpact** - Use this tag when the code change requires changes or updates to documentation in order to be understood. This tag can also be used if the documentation is provided along with the patch itself. This will also generate a Launchpad bug in manila for triaging and tracking. Refer to the section on *Documenting your work* to understand where to add documentation.
- **Implements: | Partially Implements:** *blueprint <name_of_blueprint>* - Use this tag when a change implements or partially implements the given blueprint in Launchpad. This will automatically generate a link to the blueprint in Gerrit for easy access for reviewers.
- **TrivialFix** - This tag is used for a trivial issue, such as a typo, an unclear log message, or a simple code refactor that does not change existing behavior which does not require the creation of a separate bug or blueprint in Launchpad.

Make sure that the **Closes-Bug**, **Partial-Bug**, **Related-Bug**, **blueprint**, and **Change-id** tags are at the very end of the commit message. The Gerrit hooks will automatically put the hash at the end of the commit message. For more information on tags and some examples of good commit messages, refer to the GitCommitMessages documentation.

Guru Meditation Reports

Manila contains a mechanism whereby developers and system administrators can generate a report about the state of a running Manila executable. This report is called a *Guru Meditation Report* (*GMR* for short).

Generating a GMR

A *GMR* can be generated by sending the *SIGUSR1/SIGUSR2* signal to any Manila process with support (see below). The *GMR* will then output to standard error for that particular process.

For example, suppose that `manila-api` has process id `8675`, and was run with `2>/var/log/manila/manila-api-err.log`. Then, `kill -SIGUSR1 8675` will trigger the Guru Meditation report to be printed to `/var/log/manila/manila-api-err.log`.

It could save these reports to a well known directory for later analysis by the sysadmin or automated bug analysis tools. To configure *GMR* you have to add the following section to manila.conf:

[oslo_reports] log_dir = /path/to/logs/dir

There is other way to trigger a generation of report, user should add a configuration in Manilas conf file:

```
[oslo_reports]
file_event_handler=['The path to a file to watch for changes to trigger '
                    'the reports, instead of signals. Setting this option '
                    'disables the signal trigger for the reports.']
file_event_handler_interval=['How many seconds to wait between polls when '
                             'file_event_handler is set, default value '
                             'is 1']
```

a *GMR* can be generated by touching the file which was specified in file_event_handler. The *GMR* will then output to standard error for that particular process.

For example, suppose that `manila-api` was run with `2>/var/log/manila/manila-api-err.log`, and the file path is `/tmp/guru_report`. Then, `touch /tmp/guru_report` will trigger the Guru Meditation report to be printed to `/var/log/manila/manila-api-err.log`.

Structure of a GMR

The *GMR* is designed to be extensible; any particular executable may add its own sections. However, the base *GMR* consists of several sections:

Package Shows information about the package to which this process belongs, including version information

Threads Shows stack traces and thread ids for each of the threads within this process

Green Threads Shows stack traces for each of the green threads within this process (green threads dont have thread ids)

Configuration Lists all the configuration options currently accessible via the CONF object for the current process

Adding Support for GMRs to New Executables

Adding support for a *GMR* to a given executable is fairly easy.

First import the module (currently residing in oslo.reports), as well as the Manila version module:

```
from oslo_reports import guru_meditation_report as gmr
from manila import version
```

Then, register any additional sections (optional):

```
TextGuruMeditation.register_section('Some Special Section',
                                    some_section_generator)
```

Finally (under main), before running the main loop of the executable (usually `service.server(server)` or something similar), register the *GMR* hook:

```
TextGuruMeditation.setup_autorun(version)
```

Extending the GMR

As mentioned above, additional sections can be added to the GMR for a particular executable. For more information, see the inline documentation about oslo.reports: oslo.reports

User Messages

User messages are a way to inform users about the state of asynchronous operations. One example would be notifying the user of why a share provisioning request failed. These messages can be requested via the */messages* API. All user visible messages must be defined in the permitted messages module in order to prevent sharing sensitive information with users.

Example message generation:

```
from manila import context
from manila.message import api as message_api
from manila.message import message_field

self.message_api = message_api.API()

context = context.RequestContext()
project_id = '6c430ede-9476-4128-8838-8d3929ced223'
share_id = 'f292cc0c-54a7-4b3b-8174-d2ff82d87008'

self.message_api.create(
    context,
    message_field.Actions.CREATE,
    project_id,
    resource_type=message_field.Resource.SHARE,
    resource_id=SHARE_id,
    detail=message_field.Detail.NO_VALID_HOST)
```

Will produce the following:

```
GET /v2/6c430ede-9476-4128-8838-8d3929ced223/messages
{
  "messages": [
    {
     "id": "5429fffa-5c76-4d68-a671-37a8e24f37cf",
     "action_id": "001",
     "detail_id": "002",
     "user_message": "create: No storage could be allocated for this share "
                     "request. Trying again with a different size "
                     "or share type may succeed.",
     "message_level": "ERROR",
     "resource_type": "SHARE",
     "resource_id": "f292cc0c-54a7-4b3b-8174-d2ff82d87008",
     "created_at": 2015-08-27T09:49:58-05:00,
     "expires_at": 2015-09-26T09:49:58-05:00,
     "request_id": "req-936666d2-4c8f-4e41-9ac9-237b43f8b848",
    }
  ]
}
```

The Message API Module

Handles all requests related to user facing messages.

class API(*db_driver=None*)

API for handling user messages.

cleanup_expired_messages(*context*)

create(*context*, *action*, *project_id*, *resource_type=None*, *resource_id=None*, *exception=None*, *detail=None*, *level='ERROR'*)

Create a message with the specified information.

delete(*context*, *id*)

Delete message with the specified message id.

get(*context*, *id*)

Return message with the specified message id.

get_all(*context*, *search_opts=None*, *limit=None*, *offset=None*, *sort_key=None*, *sort_dir=None*)

Return messages for the given context.

The Permitted Messages Module

class Action
Bases: object

ADD_UPDATE_SECURITY_SERVICE = ('011', 'add or update security service')

ALL = (('001', 'allocate host'), ('002', 'create'), ('003', 'delete access rules'), ('004', 'promote'), ('005', 'update'), ('006', 'revert to snapshot'), ('007', 'delete'), ('008', 'extend'), ('009', 'shrink'), ('010', 'update access rules'), ('011', 'add or update security service'))

ALLOCATE_HOST = ('001', 'allocate host')

CREATE = ('002', 'create')

DELETE = ('007', 'delete')

DELETE_ACCESS_RULES = ('003', 'delete access rules')

EXTEND = ('008', 'extend')

PROMOTE = ('004', 'promote')

REVERT_TO_SNAPSHOT = ('006', 'revert to snapshot')

SHRINK = ('009', 'shrink')

UPDATE = ('005', 'update')

UPDATE_ACCESS_RULES = ('010', 'update access rules')

class Detail
Bases: object

```
ALL = (('001', 'An unknown error occurred.'), ('002', 'No storage could be
allocated for this share request. Trying again with a different size or
share type may succeed.'), ('003', 'Driver does not expect share-network
to be provided with current configuration.'), ('004', "Could not find an
existing share server or allocate one on the share network provided. You
may use a different share network, or verify the network details in the
share network and retry your request. If this doesn't work, contact your
administrator to troubleshoot issues with your network."), ('005', "An
'active' replica must exist in 'available' state to create a new replica
for share."), ('006', "Share has no replica with 'replica_state' set to
'active'."), ('007', "No storage could be allocated for this share
request, AvailabilityZone filter didn't succeed."), ('008', "No storage
could be allocated for this share request, Capabilities filter didn't
succeed."), ('009', "No storage could be allocated for this share request,
Capacity filter didn't succeed."), ('010', "No storage could be allocated
for this share request, Driver filter didn't succeed."), ('011', "No
storage could be allocated for this share request, IgnoreAttemptedHosts
filter didn't succeed."), ('012', "No storage could be allocated for this
share request, Json filter didn't succeed."), ('013', "No storage could be
allocated for this share request, Retry filter didn't succeed."), ('014',
"No storage could be allocated for this share request, ShareReplication
filter didn't succeed."), ('015', 'Share Driver failed to extend share,
The share status has been set to extending_error. This action cannot be
re-attempted until the status has been rectified. Contact your
administrator to determine the cause of this failure.'), ('016', "No
storage could be allocated for this share request, CreateFromSnapshot
filter didn't succeed."), ('017', 'Share Driver has failed to create the
share from snapshot. This operation can be re-attempted by creating a new
share. Contact your administrator to determine the cause of this
failure.'), ('018', 'Share Driver refused to shrink the share. The size to
be shrunk is smaller than the current used space. The share status has
been set to available. Please select a size greater than the current used
space.'), ('019', 'Share Driver does not support shrinking shares.
Shrinking share operation failed.'), ('020', 'Failed to grant access to
client. The client ID used may be forbidden. You may try again with a
different client identifier.'), ('021', 'Failed to grant access to client.
The access level or type may be unsupported. You may try again with a
different access level or access type.'), ('022', 'Share driver has failed
to setup one or more security services that are associated with the used
share network. The security service may be unsupported or the provided
parameters are invalid. You may try again with a different set of
configurations.'), ('023', 'Share Driver failed to create share due to a
security service authentication issue. The security service user has
either insufficient privileges or wrong credentials. Please check your
user, password, ou and domain.'), ('024', 'No default share type has been
made available. You must specify a share type for creating shares.'),
('025', 'Share Driver failed to create share because a security service
has not been added to the share network used. Please add a security
service to the share network.'))
```

DRIVER_FAILED_CREATING_FROM_SNAP = ('017', 'Share Driver has failed to create the share from snapshot. This operation can be re-attempted by creating a new share. Contact your administrator to determine the cause of this failure.')

DRIVER_FAILED_EXTEND = ('015', 'Share Driver failed to extend share, The share status has been set to extending_error. This action cannot be re-attempted until the status has been rectified. Contact your administrator to determine the cause of this failure.')

DRIVER_FAILED_SHRINK = ('019', 'Share Driver does not support shrinking shares. Shrinking share operation failed.')

DRIVER_REFUSED_SHRINK = ('018', 'Share Driver refused to shrink the share. The size to be shrunk is smaller than the current used space. The share status has been set to available. Please select a size greater than the current used space.')

EXCEPTION_DETAIL_MAPPINGS = {('002', 'No storage could be allocated for this share request. Trying again with a different size or share type may succeed.'): ['NoValidHost']}

FILTER_AVAILABILITY = ('007', "No storage could be allocated for this share request, AvailabilityZone filter didn't succeed.")

FILTER_CAPABILITIES = ('008', "No storage could be allocated for this share request, Capabilities filter didn't succeed.")

FILTER_CAPACITY = ('009', "No storage could be allocated for this share request, Capacity filter didn't succeed.")

FILTER_CREATE_FROM_SNAPSHOT = ('016', "No storage could be allocated for this share request, CreateFromSnapshot filter didn't succeed.")

FILTER_DETAIL_MAPPINGS = {'AvailabilityZoneFilter': ('007', "No storage could be allocated for this share request, AvailabilityZone filter didn't succeed."), 'CapabilitiesFilter': ('008', "No storage could be allocated for this share request, Capabilities filter didn't succeed."), 'CapacityFilter': ('009', "No storage could be allocated for this share request, Capacity filter didn't succeed."), 'CreateFromSnapshotFilter': ('016', "No storage could be allocated for this share request, CreateFromSnapshot filter didn't succeed."), 'DriverFilter': ('010', "No storage could be allocated for this share request, Driver filter didn't succeed."), 'IgnoreAttemptedHostsFilter': ('011', "No storage could be allocated for this share request, IgnoreAttemptedHosts filter didn't succeed."), 'JsonFilter': ('012', "No storage could be allocated for this share request, Json filter didn't succeed."), 'RetryFilter': ('013', "No storage could be allocated for this share request, Retry filter didn't succeed."), 'ShareReplicationFilter': ('014', "No storage could be allocated for this share request, ShareReplication filter didn't succeed.")}

FILTER_DRIVER = ('010', "No storage could be allocated for this share request, Driver filter didn't succeed.")

FILTER_IGNORE = ('011', "No storage could be allocated for this share request, IgnoreAttemptedHosts filter didn't succeed.")

FILTER_JSON = ('012', "No storage could be allocated for this share request, Json filter didn't succeed.")

FILTER_MSG = "No storage could be allocated for this share request, %s filter didn't succeed."

FILTER_REPLICATION = ('014', "No storage could be allocated for this share request, ShareReplication filter didn't succeed.")

FILTER_RETRY = ('013', "No storage could be allocated for this share request, Retry filter didn't succeed.")

FORBIDDEN_CLIENT_ACCESS = ('020', 'Failed to grant access to client. The client ID used may be forbidden. You may try again with a different client identifier.')

MISSING_SECURITY_SERVICE = ('025', 'Share Driver failed to create share because a security service has not been added to the share network used. Please add a security service to the share network.')

NO_ACTIVE_AVAILABLE_REPLICA = ('005', "An 'active' replica must exist in 'available' state to create a new replica for share.")

NO_ACTIVE_REPLICA = ('006', "Share has no replica with 'replica_state' set to 'active'.")

NO_DEFAULT_SHARE_TYPE = ('024', 'No default share type has been made available. You must specify a share type for creating shares.')

NO_SHARE_SERVER = ('004', "Could not find an existing share server or allocate one on the share network provided. You may use a different share network, or verify the network details in the share network and retry your request. If this doesn't work, contact your administrator to troubleshoot issues with your network.")

NO_VALID_HOST = ('002', 'No storage could be allocated for this share request. Trying again with a different size or share type may succeed.')

SECURITY_SERVICE_FAILED_AUTH = ('023', 'Share Driver failed to create share due to a security service authentication issue. The security service user has either insufficient privileges or wrong credentials. Please check your user, password, ou and domain.')

UNEXPECTED_NETWORK = ('003', 'Driver does not expect share-network to be provided with current configuration.')

UNKNOWN_ERROR = ('001', 'An unknown error occurred.')

UNSUPPORTED_ADD_UDPATE_SECURITY_SERVICE = ('022', 'Share driver has failed to setup one or more security services that are associated with the used share network. The security service may be unsupported or the provided parameters are invalid. You may try again with a different set of configurations.')

UNSUPPORTED_CLIENT_ACCESS = ('021', 'Failed to grant access to client. The access level or type may be unsupported. You may try again with a different access level or access type.')

class Resource

Bases: object

SECURITY_SERVICE = 'SECURITY_SERVICE'

SHARE = 'SHARE'

SHARE_GROUP = 'SHARE_GROUP'

SHARE_REPLICA = 'SHARE_REPLICA'

SHARE_SNAPSHOT = 'SHARE_SNAPSHOT'

translate_action(*action_id*)

translate_detail(*detail_id*)

translate_detail_id(*excep*, *detail*)

Ganesha Library

The Ganesha Library provides base classes that can be used by drivers to provision shares via NFS (NFSv3 and NFSv4), utilizing the NFS-Ganesha NFS server.

Supported operations

- Allow NFS Share access
  - Only IP access type is supported.
- Deny NFS Share access

Supported manila drivers

- CephFS driver uses `ganesha.GaneshaNASHelper2` library class
- GlusterFS driver uses `ganesha.GaneshaNASHelper` library class

Requirements

- Preferred:

  NFS-Ganesha v2.4 or later, which allows dynamic update of access rules. Use with manilas `ganesha.GaneshaNASHelper2` class as described later in *Using Ganesha Library in drivers*.

  (or)

  NFS-Ganesha v2.5.4 or later that allows dynamic update of access rules, and can make use of highly available Ceph RADOS (distributed object storage) as its shared storage for NFS client recovery data, and exports. Use with Ceph v12.2.2 or later, and `ganesha.GaneshaNASHelper2` library class in manila Queens release or later.

- For use with limitations documented in *Known Issues*:

  NFS-Ganesha v2.1 to v2.3. Use with manilas `ganesha.GaneshaNASHelper` class as described later in *Using Ganesha Library in drivers*.

NFS-Ganesha configuration

The library has just modest requirements against general NFS-Ganesha (in the following: Ganesha) configuration; a best effort was made to remain agnostic towards it as much as possible. This section describes the few requirements.

Note that Ganeshas concept of storage backend modules is called FSAL (File System Abstraction Layer). The FSAL the driver intends to leverage needs to be enabled in Ganesha config.

Beyond that (with default manila config) the following line is needed to be present in the Ganesha config file (that defaults to /etc/ganesha/ganesha.conf):

```
%include /etc/ganesha/export.d/INDEX.conf
```

The above paths can be customized through manila configuration as follows:

- ***ganesha_config_dir* = toplevel directory for Ganesha configuration,** defaults to /etc/ganesha
- ***ganesha_config_path* = location of the Ganesha config file, defaults** to ganesha.conf in *ganesha_config_dir*
- ***ganesha_export_dir* = directory where manila generated config bits are** stored, defaults to *export.d* in *ganesha_config_dir*. The following line is required to be included (with value expanded) in the Ganesha config file (at *ganesha_config_path*):

  ```
  %include <ganesha_export_dir>/INDEX.conf
  ```

In versions 2.5.4 or later, Ganesha can store NFS client recovery data in Ceph RADOS, and also read exports stored in Ceph RADOS. These features are useful to make Ganesha server that has access to a Ceph (luminous or later) storage backend, highly available. The Ganesha library class *GaneshaNASHelper2* (in manila Queens or later) allows you to store Ganesha exports directly in a shared storage, RADOS objects, by setting the following manila config options in the driver section:

- *ganesha_rados_store_enable* = True to persist Ganesha exports and export counter in Ceph RADOS objects
- *ganesha_rados_store_pool_name* = name of the Ceph RADOS pool to store Ganesha exports and export counter objects
- *ganesha_rados_export_index* = name of the Ceph RADOS object used to store a list of export RADOS object URLs (defaults to ganesha-export-index)

Check out the *cephfs_driver* documentation for an example driver section that uses these options.

To allow Ganesha to read from RADOS objects add the below code block in ganeshas configuration file, substituting values per your setup.

```
# To read exports from RADOS objects
RADOS_URLS {
    ceph_conf = "/etc/ceph/ceph.conf";
    userid = "admin";
}
# Replace with actual pool name, and export index object
%url rados://<ganesha_rados_store_pool_name>/<ganesha_rados_export_index>
# To store client recovery data in the same RADOS pool
NFSv4 {
    RecoveryBackend = "rados_kv";
```

(continues on next page)

(continued from previous page)

```
}
RADOS_KV {
    ceph_conf = "/etc/ceph/ceph.conf";
    userid = "admin";
    # Replace with actual pool name
    pool = <ganesha_rados_store_pool_name>;
}
```

For a fresh setup, make sure to create the Ganesha export index object as an empty object before starting the Ganesha server.

```
echo | sudo rados -p ${GANESHA_RADOS_STORE_POOL_NAME} put ganesha-export-
↪index -
```

Further Ganesha related manila configuration

There are further Ganesha related options in manila (which affect the behavior of Ganesha, but do not affect how to set up the Ganesha service itself).

These are:

- *ganesha_service_name* = name of the system service representing Ganesha, defaults to ganesha.nfsd
- *ganesha_db_path* = location of on-disk database storing permanent Ganesha state, e.g. an export ID counter to generate export IDs for shares

  (or)

  When *ganesha_rados_store_enabled* is set to True, the ganesha export counter is stored in a Ceph RADOS object instead of in a SQLite database local to the manila driver. The counter can be optionally configured with, *ganesha_rados_export_counter* = name of the Ceph RADOS object used as the Ganesha export counter (defaults to ganesha-export-counter)
- ***ganesha_export_template_dir* = directory from where Ganesha loads** export customizations (cf. Customizing Ganesha exports).

Using Ganesha Library in drivers

A driver that wants to use the Ganesha Library has to inherit from `driver.GaneshaMixin`.

The driver has to contain a subclass of `ganesha.GaneshaNASHelper2`, instantiate it along with the driver instance and delegate `update_access` method to it (when appropriate, i.e., when `access_proto` is NFS).

Note: You can also subclass `ganesha.GaneshaNASHelper`. It works with NFS-Ganesha v2.1 to v2.3 that doesnt support dynamic update of exports. To update access rules without having to restart NFS-Ganesha server, the class manipulates exports created per share access rule (rather than per share) introducing limitations documented in *Known Issues*.

In the following we explain what has to be implemented by the `ganesha.GaneshaNASHelper2` subclass (to which we refer as helper class).

Ganesha exports are described by so-called *Ganesha export blocks* (introduced in the 2.* release series), that is, snippets of Ganesha config specifying key-pair values.

The Ganesha Library generates sane default export blocks for the exports it manages, with one thing left blank, the so-called *FSAL subblock*. The helper class has to implement the `_fsal_hook` method which returns the FSAL subblock (in Python represented as a dict with string keys and values). It has one mandatory key, `Name`, to which the value should be the name of the FSAL (eg.: `{"Name": "CEPH"}`). Further content of it is optional and FSAL specific.

Customizing Ganesha exports

As noted, the Ganesha Library provides sane general defaults.

However, the driver is allowed to:

- customize defaults
- allow users to customize exports

The config format for Ganesha Library is called *export block template*. They are syntactically either Ganesha export blocks, (please consult the Ganesha documentation about the format), or isomorphic JSON (as Ganesha export blocks are by-and-large equivalent to arrayless JSON), with two special placeholders for values: `@config` and `@runtime`. `@config` means a value that shall be filled from manila config, and `@runtime` means a value thats filled at runtime with dynamic data.

As an example, we show the librarys defaults in JSON format (also valid Python literal):

```
{
  "EXPORT": {
    "Export_Id": "@runtime",
    "Path": "@runtime",
    "FSAL": {
      "Name": "@config"
    },
    "Pseudo": "@runtime",
    "SecType": "sys",
    "Tag": "@runtime",
    "CLIENT": {
      "Clients": "@runtime",
      "Access_Type": "RW"
    },
    "Squash": "None"
  }
}
```

The Ganesha Library takes these values from

manila/share/drivers/ganesha/conf/00-base-export-template.conf

where the same data is stored in Ganesha conf format (also supplied with comments).

For customization, the driver has to extend the `_default_config_hook` method as follows:

- take the result of the super method (a dict representing an export block template)
- set up another export block dict that include your custom values, either by
  - using a predefined export block dict stored in code
  - loading a predefined export block from the manila source tree
  - loading an export block from an user exposed location (to allow user configuration)
- merge the two export block dict using the `ganesha_utils.patch` method
- return the result

With respect to *loading export blocks*, that can be done through the utility method `_load_conf_dir`.

Known Restrictions

- The library does not support network segmented multi-tenancy model but instead works over a flat network, where the tenants share a network.

Known Issues

Following issues concern only users of *ganesha.GaneshaNASHelper* class that works with NFS-Ganesha v2.1 to v2.3.

- The export location for shares of a driver that uses the Ganesha Library will be of the format `<ganesha-server>:/share-<share-id>`. However, this is incomplete information, because it pertains only to NFSv3 access, which is partially broken. NFSv4 mounts work well but the actual NFSv4 export paths differ from the above. In detail:
  - The export location is usable only for NFSv3 mounts.
  - The export location works only for the first access rule thats added for the given share. Tenants that should be allowed to access according to a further access rule will be refused (cf. https://bugs.launchpad.net/manila/+bug/1513061).
  - The share is, however, exported through NFSv4, just on paths that differ from the one indicated by the export location, namely at: `<ganesha-server>:/share-<share-id>--<access-id>`, where `<access-id>` ranges over the ID-s of access rules of the share (and the export with `<access-id>` is accessible according to the access rule of that ID).
  - NFSv4 access also works with pseudofs. That is, the tenant can do a v4 mount of"<ganesha-server>:/" and access the shares allowed for her at the respective `share-<share-id>--<access-id>` subdirectories.

Deployment considerations

When using NFS-Ganesha v2.4 or later and manilas `ganesha.GaneshaNASHelper2` class, dynamic export of access rules is implemented by using the dbus-send command to signal NFS-Ganesha to update its exports. The dbus-send command is executed on the host where NFS-Ganesha runs. This may be the same host where the *manila-share* service runs, or it may be remote to *manila-share* depending on how the relevant driver has been configured. Either way, the dbus-send command and NFS-Ganesha must be able to communicate over an *abstract socket* and *must be in the same namespace*. Consequently, if you deploy NFS-Ganesha in a container you likely should run the container in the host namespace (e.g. docker run net=host) rather than in its own network namespace. For details, see this article.

The `manila.share.drivers.ganesha` Module

class GaneshaNASHelper(*execute*, *config*, *tag='<no name>'*, ***kwargs*)
Bases: `manila.share.drivers.ganesha.NASHelperBase`

Perform share access changes using Ganesha version < 2.4.

init_helper()
Initializes protocol-specific NAS drivers.

supported_access_levels = ('rw', 'ro')

supported_access_types = ('ip',)

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules of share.

class GaneshaNASHelper2(*execute*, *config*, *tag='<no name>'*, ***kwargs*)
Bases: `manila.share.drivers.ganesha.GaneshaNASHelper`

Perform share access changes using Ganesha version >= 2.4.

init_helper()
Initializes protocol-specific NAS drivers.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules of share.

Creates an export per share. Modifies access rules of shares by dynamically updating exports via DBUS.

class NASHelperBase(*execute*, *config*, ***kwargs*)
Bases: `object`

Interface to work with share.

init_helper()
Initializes protocol-specific NAS drivers.

supported_access_levels = ()

supported_access_types = ()

abstract update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules of share.

4.1.3 Background Concepts for manila

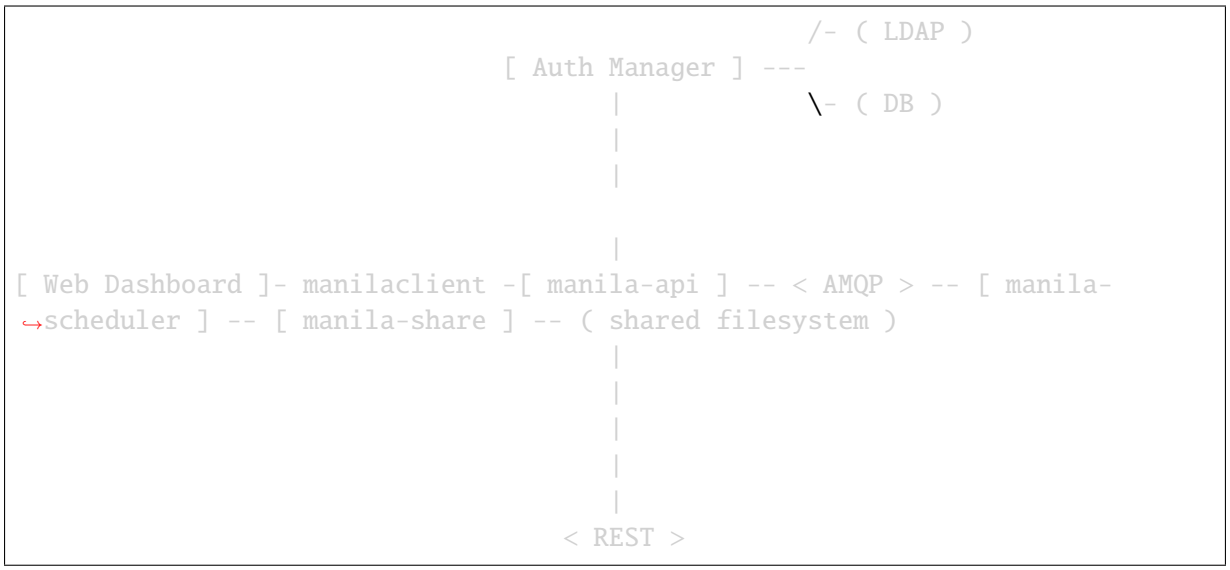
Manila System Architecture

The Shared File Systems service is intended to be ran on one or more nodes.

Manila uses a sql-based central database that is shared by all manila services in the system. The amount and depth of the data fits into a sql database quite well. For small deployments this seems like an optimal solution. For larger deployments, and especially if security is a concern, manila will be moving towards multiple data stores with some kind of aggregation system.

Components

Below you will a brief explanation of the different components.



- DB: sql database for data storage. Used by all components (LINKS NOT SHOWN)
- Web Dashboard: external component that talks to the api, implemented as a plugin to the OpenStack Dashboard (Horizon) project, source is here.
- *manila-api*
- Auth Manager: component responsible for users/projects/and roles. Can backend to DB or LDAP. This is not a separate binary, but rather a python class that is used by most components in the system.
- *manila-scheduler*
- *manila-share*

Further Challenges

- More efficient share/snapshot size calculation
- Create a notion of attached shares with automation of mount operations
- Allow admin-created share-servers and share-networks to be used by multiple tenants
- Support creation of new subnets for share servers (to connect VLANs with VXLAN/GRE/etc)
- Gateway mediated networking model with NFS-Ganesha
- Add support for more backends

Threading model

All OpenStack services use *green thread* model of threading, implemented through using the Python eventlet and greenlet libraries.

Green threads use a cooperative model of threading: thread context switches can only occur when specific eventlet or greenlet library calls are made (e.g., sleep, certain I/O calls). From the operating systems point of view, each OpenStack service runs in a single thread.

The use of green threads reduces the likelihood of race conditions, but does not completely eliminate them. In some cases, you may need to use the `@utils.synchronized(...)` decorator to avoid races.

In addition, since there is only one operating system thread, a call that blocks that main thread will block the entire process.

Yielding the thread in long-running tasks

If a code path takes a long time to execute and does not contain any methods that trigger an eventlet context switch, the long-running thread will block any pending threads.

This scenario can be avoided by adding calls to the eventlet sleep method in the long-running code path. The sleep call will trigger a context switch if there are pending threads, and using an argument of 0 will avoid introducing delays in the case that there is only a single green thread:

```
from eventlet import greenthread
...
greenthread.sleep(0)
```

In current code, time.sleep(0) does the same thing as greenthread.sleep(0) if time module is patched through eventlet.monkey_patch(). To be explicit, we recommend contributors to use `greenthread.sleep()` instead of `time.sleep()`.

MySQL access and eventlet

There are some MySQL DB API drivers for oslo.db, like PyMySQL, MySQL-python, etc. PyMySQL is the default MySQL DB API driver for oslo.db, and it works well with eventlet. MySQL-python uses an external C library for accessing the MySQL database. Since eventlet cannot use monkey-patching to intercept blocking calls in a C library, queries to the MySQL database will block the main thread of a service.

The Diablo release contained a thread-pooling implementation that did not block, but this implementation resulted in a bug and was removed.

See this mailing list thread for a discussion of this issue, including a discussion of the impact on performance.

Internationalization

Manila uses gettext so that user-facing strings appear in the appropriate language in different locales.

Beginning with the Pike series, OpenStack no longer supports log translation. It is not useful to add translation instructions to new code, and the instructions can be removed from old code.

Other user-facing strings, e.g. in exception messages, should be translated.

To use gettext, make sure that the strings passed to the logger are wrapped in a `_()` function call. For example:

```
msg = _("Share group %s not found.") % share_group_id
raise exc.HTTPNotFound(explanation=msg)
```

Do not use `locals()` for formatting messages because: 1. It is not as clear as using explicit dicts. 2. It could produce hidden errors during refactoring. 3. Changing the name of a variable causes a change in the message. 4. It creates a lot of otherwise unused variables.

If you do not follow the project conventions, your code may cause the LocalizationTestCase.test_multiple_positional_format_placeholders test to fail in manila/tests/test_localization.py.

The `_()` function is brought into the global scope by doing:

```
from manila.openstack.common import gettextutils
gettextutils.install("manila")
```

These lines are needed in any toplevel script before any manila modules are imported. If this code is missing, it may result in an error that looks like:

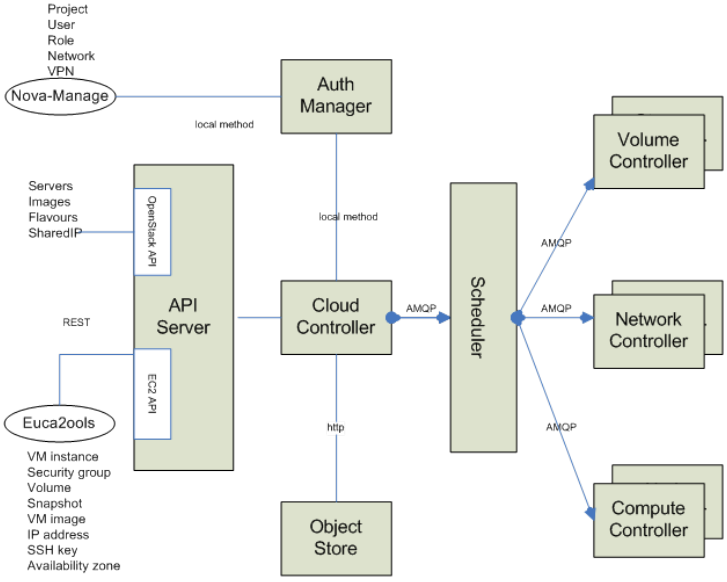
```
NameError: name '_' is not defined
```

AMQP and manila

AMQP is the messaging technology chosen by the OpenStack cloud. The AMQP broker, either RabbitMQ or Qpid, sits between any two manila components and allows them to communicate in a loosely coupled fashion. More precisely, manila components (the compute fabric of OpenStack) use Remote Procedure Calls (RPC hereinafter) to communicate to one another; however such a paradigm is built atop the publish/subscribe paradigm so that the following benefits can be achieved:

- Decoupling between client and servant (such as the client does not need to know where the servants reference is).
- Full a-synchronism between client and servant (such as the client does not need the servant to run at the same time of the remote call).
- Random balancing of remote calls (such as if more servants are up and running, one-way calls are transparently dispatched to the first available servant).

Manila uses direct, fanout, and topic-based exchanges. The architecture looks like the one depicted in the figure below:



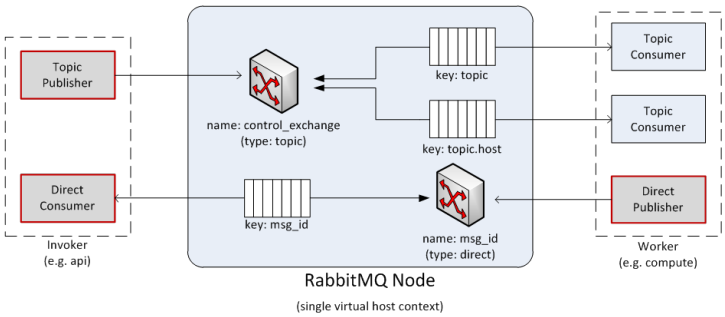
Manila implements RPC (both request+response, and one-way, respectively nicknamed rpc.call and rpc.cast) over AMQP by providing an adapter class which take cares of marshaling and unmarshaling of messages into function calls. Each manila service (for example Compute, Volume, etc.) create two queues at the initialization time, one which accepts messages with routing keys NODE-TYPE.NODE-ID (for example compute.hostname) and another, which accepts messages with routing keys as generic NODE-TYPE (for example compute). The former is used specifically when Manila-API needs to redirect commands to a specific node like euca-terminate instance. In this case, only the compute node whose hosts hypervisor is running the virtual machine can kill the instance. The API acts as a consumer when RPC calls are request/response, otherwise is acts as publisher only.

Manila RPC Mappings

The figure below shows the internals of a message broker node (referred to as a RabbitMQ node in the diagrams) when a single instance is deployed and shared in an OpenStack cloud. Every manila component connects to the message broker and, depending on its personality (for example a compute node or a network node), may use the queue either as an Invoker (such as API or Scheduler) or a Worker (such as Compute, Volume or Network). Invokers and Workers do not actually exist in the manila object model, but we are going to use them as an abstraction for sake of clarity. An Invoker is a component that sends messages in the queuing system via two operations: 1) rpc.call and ii) rpc.cast; a Worker is a component that receives messages from the queuing system and reply accordingly to rcp.call operations.

Figure 2 shows the following internal elements:

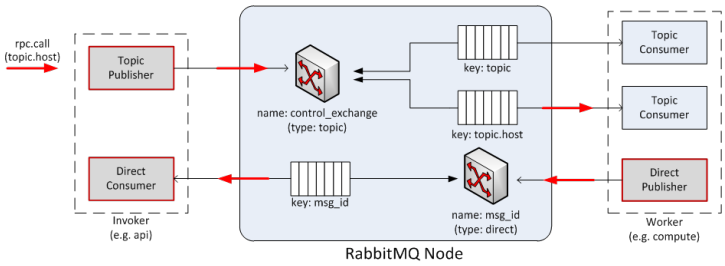
- Topic Publisher: A Topic Publisher comes to life when an rpc.call or an rpc.cast operation is executed; this object is instantiated and used to push a message to the queuing system. Every publisher connects always to the same topic-based exchange; its life-cycle is limited to the message delivery.
- Direct Consumer: A Direct Consumer comes to life if (an only if) a rpc.call operation is executed; this object is instantiated and used to receive a response message from the queuing system; Every consumer connects to a unique direct-based exchange via a unique exclusive queue; its life-cycle is limited to the message delivery; the exchange and queue identifiers are determined by a UUID generator, and are marshaled in the message sent by the Topic Publisher (only rpc.call operations).
- Topic Consumer: A Topic Consumer comes to life as soon as a Worker is instantiated and exists throughout its life-cycle; this object is used to receive messages from the queue and it invokes the appropriate action as defined by the Worker role. A Topic Consumer connects to the same topic-based exchange either via a shared queue or via a unique exclusive queue. Every Worker has two topic consumers, one that is addressed only during rpc.cast operations (and it connects to a shared queue whose exchange key is topic) and the other that is addressed only during rpc.call operations (and it connects to a unique queue whose exchange key is topic.host).
- Direct Publisher: A Direct Publisher comes to life only during rpc.call operations and it is instantiated to return the message required by the request/response operation. The object connects to a direct-based exchange whose identity is dictated by the incoming message.
- Topic Exchange: The Exchange is a routing table that exists in the context of a virtual host (the multi-tenancy mechanism provided by Qpid or RabbitMQ); its type (such as topic vs. direct) determines the routing policy; a message broker node will have only one topic-based exchange for every topic in manila.
- Direct Exchange: This is a routing table that is created during rpc.call operations; there are many instances of this kind of exchange throughout the life-cycle of a message broker node, one for each rpc.call invoked.
- Queue Element: A Queue is a message bucket. Messages are kept in the queue until a Consumer (either Topic or Direct Consumer) connects to the queue and fetch it. Queues can be shared or can be exclusive. Queues whose routing key is topic are shared amongst Workers of the same personality.



RPC Calls

The diagram below shows the message flow during an rpc.call operation:

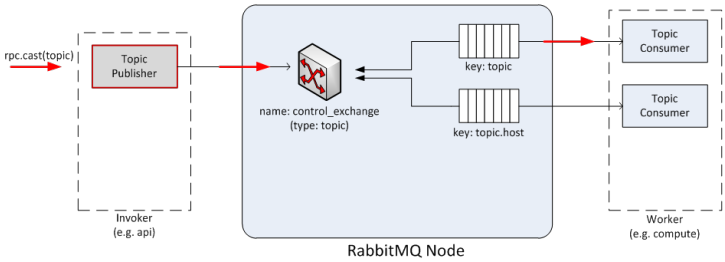
1. A Topic Publisher is instantiated to send the message request to the queuing system; immediately before the publishing operation, a Direct Consumer is instantiated to wait for the response message.
2. Once the message is dispatched by the exchange, it is fetched by the Topic Consumer dictated by the routing key (such as topic.host) and passed to the Worker in charge of the task.
3. Once the task is completed, a Direct Publisher is allocated to send the response message to the queuing system.
4. Once the message is dispatched by the exchange, it is fetched by the Direct Consumer dictated by the routing key (such as msg_id) and passed to the Invoker.



RPC Casts

The diagram below the message flow during an rp.cast operation:

1. A Topic Publisher is instantiated to send the message request to the queuing system.
2. Once the message is dispatched by the exchange, it is fetched by the Topic Consumer dictated by the routing key (such as topic) and passed to the Worker in charge of the task.



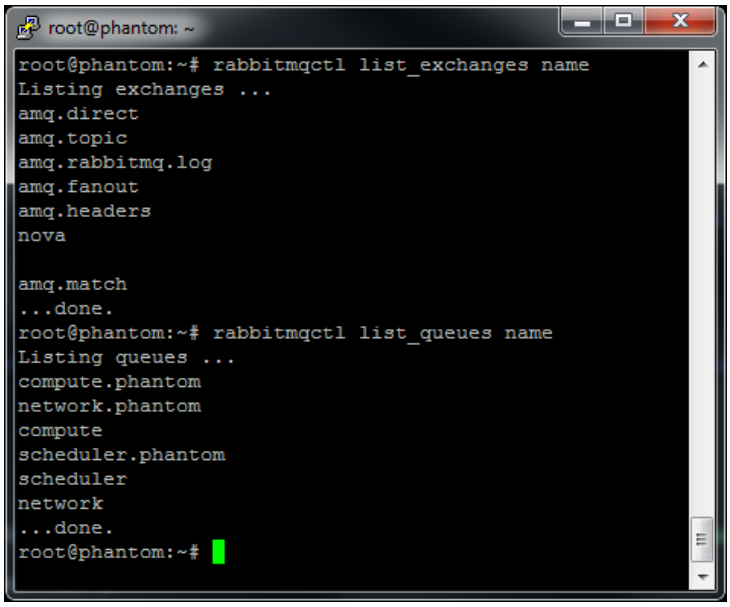
AMQP Broker Load

At any given time the load of a message broker node running either Qpid or RabbitMQ is function of the following parameters:

- Throughput of API calls: The number of API calls (more precisely rpc.call ops) being served by the OpenStack cloud dictates the number of direct-based exchanges, related queues and direct consumers connected to them.
- Number of Workers: There is one queue shared amongst workers with the same personality; however there are as many exclusive queues as the number of workers; the number of workers dictates also the number of routing keys within the topic-based exchange, which is shared amongst all workers.

The figure below shows the status of a RabbitMQ node after manila components bootstrap in a test environment. Exchanges and queues being created by manila components are:

- **Exchanges**
    1. manila (topic exchange)
- **Queues**
    1. compute.phantom (phantom is hostname)
    2. compute
    3. network.phantom (phantom is hostname)
    4. network
    5. share.phantom (phantom is hostname)
    6. share
    7. scheduler.phantom (phantom is hostname)
    8. scheduler



RabbitMQ Gotchas

Manila uses Kombu to connect to the RabbitMQ environment. Kombu is a Python library that in turn uses AMQPLib, a library that implements the standard AMQP 0.8 at the time of writing. When using Kombu, Invokers and Workers need the following parameters in order to instantiate a Connection object that connects to the RabbitMQ server (please note that most of the following material can be also found in the Kombu documentation; it has been summarized and revised here for sake of clarity):

- Hostname: The hostname to the AMQP server.
- Userid: A valid username used to authenticate to the server.
- Password: The password used to authenticate to the server.
- Virtual_host: The name of the virtual host to work with. This virtual host must exist on the server, and the user must have access to it. Default is /.
- Port: The port of the AMQP server. Default is 5672 (amqp).

The following parameters are default:

- Insist: Insist on connecting to a server. In a configuration with multiple load-sharing servers, the Insist option tells the server that the client is insisting on a connection to the specified server. Default is False.
- Connect_timeout: The timeout in seconds before the client gives up connecting to the server. The default is no timeout.
- SSL: use SSL to connect to the server. The default is False.

More precisely Consumers need the following parameters:

- Connection: The above mentioned Connection object.
- Queue: Name of the queue.
- Exchange: Name of the exchange the queue binds to.
- Routing_key: The interpretation of the routing key depends on the value of the exchange_type attribute.
  - Direct exchange: If the routing key property of the message and the routing_key attribute of the queue are identical, then the message is forwarded to the queue.
  - Fanout exchange: messages are forwarded to the queues bound the exchange, even if the binding does not have a key.
  - Topic exchange: If the routing key property of the message matches the routing key of the key according to a primitive pattern matching scheme, then the message is forwarded to the queue. The message routing key then consists of words separated by dots (., like domain names), and two special characters are available; star () and hash (#). The star matches any word, and the hash matches zero or more words. For example .stock.# matches the routing keys usd.stock and eur.stock.db but not stock.nasdaq.
- Durable: This flag determines the durability of both exchanges and queues; durable exchanges and queues remain active when a RabbitMQ server restarts. Non-durable exchanges/queues (transient exchanges/queues) are purged when a server restarts. It is worth noting that AMQP specifies that durable queues cannot bind to transient exchanges. Default is True.
- Auto_delete: If set, the exchange is deleted when all queues have finished using it. Default is False.

- Exclusive: exclusive queues (such as non-shared) may only be consumed from by the current connection. When exclusive is on, this also implies auto_delete. Default is False.
- Exchange_type: AMQP defines several default exchange types (routing algorithms) that covers most of the common messaging use cases.
- Auto_ack: acknowledgment is handled automatically once messages are received. By default auto_ack is set to False, and the receiver is required to manually handle acknowledgment.
- No_ack: It disable acknowledgment on the server-side. This is different from auto_ack in that acknowledgment is turned off altogether. This functionality increases performance but at the cost of reliability. Messages can get lost if a client dies before it can deliver them to the application.
- Auto_declare: If this is True and the exchange name is set, the exchange will be automatically declared at instantiation. Auto declare is on by default. Publishers specify most the parameters of Consumers (such as they do not specify a queue name), but they can also specify the following:
- Delivery_mode: The default delivery mode used for messages. The value is an integer. The following delivery modes are supported by RabbitMQ:
  - 1 or transient: The message is transient. Which means it is stored in memory only, and is lost if the server dies or restarts.
  - 2 or persistent: The message is persistent. Which means the message is stored both in-memory, and on disk, and therefore preserved if the server dies or restarts.

The default value is 2 (persistent). During a send operation, Publishers can override the delivery mode of messages so that, for example, transient messages can be sent over a durable queue.

Manila minimum requirements and features

In order for a driver to be accepted into manila code base, there are certain minimum requirements and features that must be met, in order to ensure interoperability and standardized manila functionality among cloud providers.

At least one driver mode (DHSS true/false)

Driver modes determine if the driver is managing network resources (*DHSS* = true) in an automated way, in order to segregate tenants and private networks by making use of manila Share Networks, or if it is up to the administrator to manually configure all networks (*DHSS* = false) and be responsible for segregation, if that is desired. At least one driver mode must be supported. In *DHSS* = true mode, Share Server entities are used, so the driver must implement functions that setup and teardown such servers.

At least one file system sharing protocol

In order to serve shares as a shared file system service, the driver must support at least one file system sharing protocol, which can be a new protocol or one of the currently supported protocols. The current list of supported protocols is as follows:

- NFS
- CIFS
- GlusterFS

- HDFS
- MapRFS
- CephFS

Access rules

Access rules control how shares are accessible, by whom, and what the level of access is. Access rule operations include allowing access and denying access to a given share. The authentication type should be based on IP, User and/or Certificate. Drivers must support read-write and read-only access levels for each supported protocol, either through individual access rules or separate export locations.

Shares

Share servicing is the core functionality of a shared file system service, so a driver must be able to create and delete shares.

Share extending

In order to best satisfy cloud service requirements, shares must be elastic, so drivers must implement a share extend function that allows shares size to be increased.

Capabilities

In order for manila to function accordingly to the driver being used, the driver must provide a set of information to manila, known as capabilities. Share driver can use Share type extra-specs (scoped and un-scoped) to serve new shares. See *Capabilities and Extra-Specs* for more information. At a minimum your driver must report:

- share_backend_name: a name for the backend;
- driver_handles_share_servers: driver mode, whether this driver instance handles share servers, possible values are true or false;
- vendor_name: driver vendor name;
- driver_version: current driver instance version;
- storage_protocol: list of shared file system protocols supported by this driver instance;
- total_capacity_gb: total amount of storage space provided, in GB;
- free_capacity_gb: amount of storage space available for use, in GB;
- reserved_percentage: percentage of total storage space to be kept from being used.

Certain features, if supported by drivers, need to be reported in order to function correctly in manila, such as:

- dedupe: whether the backend supports deduplication;
- compression: whether the backend supports compressed shares;
- thin_provisioning: whether the backend is overprovisioning shares;

- pools: list of storage pools managed by this driver instance;
- qos: whether the backend supports quality of service for shares;
- replication_domain: string specifying a common group name for all backends that can replicate between each other;
- replication_type: string specifying the type of replication supported by the driver. Can be one of (readable, writable or dr).
- security_service_update_support: boolean specifying whether the driver supports updating or adding security services in an already deployed share server. It defaults to `False`.

Below is an example of drivers with multiple pools. my is used as an example vendor prefix:

```
{
    'driver_handles_share_servers': 'False',   #\
    'share_backend_name': 'My Backend',        # backend level
    'vendor_name': 'MY',                       # mandatory/fixed
    'driver_version': '1.0',                   # stats & capabilities
    'storage_protocol': 'NFS_CIFS',            #/
                                               #\
    'my_capability_1': 'custom_val',           # "my" optional vendor
    'my_capability_2': True,                   # stats & capabilities
                                               #/
    'pools': [
        {'pool_name':
           'thin-dedupe-compression pool',     #\
         'total_capacity_gb': 500,             #  mandatory stats for
         'free_capacity_gb': 230,              #  pools
         'reserved_percentage': 0,             #/
                                               #\
         'dedupe': True,                       # common capabilities
         'compression': True,                  #
         'snapshot_support': True,             #
         'create_share_from_snapshot_support': True,
         'revert_to_snapshot_support': True,
         'qos': True,                          # this backend supports QoS
         'thin_provisioning': True,            #
         'max_over_subscription_ratio': 10,    # (mandatory for thin)
         'provisioned_capacity_gb': 270,       # (mandatory for thin)
                                               #
                                               #
         'replication_type': 'dr',             # this backend supports
                                               # replication_type 'dr'
                                               #/
         'my_dying_disks': 100,                #\
         'my_super_hero_1': 'Hulk',            #  "my" optional vendor
         'my_super_hero_2': 'Spider-Man',      #  stats & capabilities
                                               #/
                                               #\
                                               # can replicate to other
         'replication_domain': 'asgard',       # backends in
```

(continues on next page)

(continued from previous page)

```
                                            # replication_domain 'asgard'
                                            #/
            'ipv4_support': True,
            'ipv6_support': True,
            'security_service_update_support': False,

        },
        {'pool_name': 'thick pool',
         'total_capacity_gb': 1024,
         'free_capacity_gb': 1024,
         'qos': False,
         'snapshot_support': True,
         'create_share_from_snapshot_support': False, # this pool does not
                                                      # allow creating
                                                      # shares from
                                                      # snapshots
         'revert_to_snapshot_support': True,
         'reserved_percentage': 0,
         'dedupe': False,
         'compression': False,
         'thin_provisioning': False,
         'replication_type': None,
         'my_dying_disks': 200,
         'my_super_hero_1': 'Batman',
         'my_super_hero_2': 'Robin',
         'ipv4_support': True,
         'ipv6_support': True,
         'security_service_update_support': False,
        },
    ]
}
```

Continuous Integration systems

Every driver vendor must supply a CI system that tests its drivers continuously for each patch submitted to OpenStack gerrit. This allows for better QA and quicker response and notification for driver vendors when a patch submitted affects an existing driver. The CI system must run all applicable tempest tests, test all patches Jenkins has posted +1 and post its test results.

Note: for more information please see http://docs.openstack.org/infra/system-config/third_party.html

Unit tests

All drivers submitted must be contemplated with unit tests covering at least 90% of the code, preferably 100% if possible. Unit tests must use mock framework and be located in-tree using a structure that mirrors the functional code, such as directory names and filenames. See template below:

```
manila/[tests/]path/to/brand/new/[test_]driver.py
```

Documentation

Drivers submitted must provide and maintain related documentation on openstack-manuals, containing instructions on how to properly install and configure. The intended audience for this manual is cloud operators and administrators. Also, driver maintainers must update the manila share features support mapping documentation found at https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_mapping.html

Manila optional requirements and features since Mitaka

Additional to the minimum required features supported by manila, other optional features can be supported by drivers as they are already supported in manila and can be accessed through the API.

Snapshots

Share Snapshots allow for data respective to a particular point in time to be saved in order to be used later. In manila API, share snapshots taken can only be restored by creating new shares from them, thus the original share remains unaffected. If Snapshots are supported by drivers, they must be crash-consistent.

Managing/Unmanaging shares

If *DHSS* = false mode is used, then drivers may implement a function that supports reading existing shares in the backend that were not created by manila. After the previously existing share is registered in manila, it is completely controlled by manila and should not be handled externally anymore. Additionally, a function that de-registers such shares from manila but do not delete from backend may also be supported.

Share shrinking

Manila API supports share shrinking, thus a share can be shrunk in a similar way it can be extended, but the driver is responsible for making sure no data is compromised.

Share ensuring

In some situations, such as when the driver is restarted, manila attempts to perform maintenance on created shares, on the purpose of ensuring previously created shares are available and being serviced correctly. The driver can implement this function by checking shares status and performing maintenance operations if needed, such as re-exporting.

Manila experimental features since Mitaka

Some features are initially released as experimental and can be accessed by including specific additional HTTP Request headers. Those features are not recommended for production cloud environments while in experimental stage.

Share Migration

Shares can be migrated between different backends and pools. Manila implements migration using an approach that works for any manufacturer, but driver vendors can implement a better optimized migration function for when migration involves backends or pools related to the same vendor.

Share Groups (since Ocata)

The share groups provides the ability to manage a group of shares together. This feature is implemented at the manager level, every driver gets this feature by default. If a driver wants to override the default behavior to support additional functionalities such as consistent group snapshot, the driver vendors may report this capability as a group capability, such as: Ordered writes, Consistent snapshots, Group replication.

Drivers need to report group capabilities as part of the updated stats (e.g. capacity) and filled in share_group_stats node for their back end. Share group type group-specs (scoped and un-scoped) are available for the driver implementation to use as-needed. Below is an example of the share stats payload from the driver having multiple pools and group capabilities. my is used as an example vendor prefix:

```
{
    'driver_handles_share_servers': 'False',          #\
    'share_backend_name': 'My Backend',               # backend level
    'vendor_name': 'MY',                              # mandatory/fixed
    'driver_version': '1.0',                          # stats & capabilities
    'storage_protocol': 'NFS_CIFS',                   #/
                                                      #\
    'my_capability_1': 'custom_val',                  # "my" optional vendor
    'my_capability_2': True,                          # stats & capabilities
                                                      #/
    'share_group_stats': {
                                                      #\
            'my_group_capability_1': 'custom_val',    # "my" optional vendor
            'my_group_capability_2': True,            # stats & group␣
↪capabilities
                                                      #/
            'consistent_snapshot_support': 'host',    #\
```

(continues on next page)

(continued from previous page)

```
                                                    # common group␣
↪capabilities
                                                    #/
        },
    ]
}
```

Note: for more information please see *Group Capabilities and group-specs*

Share Replication

Replicas of shares can be created for either data protection (for disaster recovery) or for load sharing. In order to utilize this feature, drivers must report the `replication_type` they support as a capability and implement necessary methods.

More details can be found at: *Share replication*

Update used_size of shares

Drivers can update, for all the shares created on a particular backend, the consumed space in GiB. While the polling interval for drivers to update this information is configurable, drivers can choose to submit cached information as necessary, but specify a time at which this information was gathered_at.

Share Server Migration (Since Victoria)

Shares servers can be migrated between different backends. Driver vendors need to implement the share server migration functions in order to migrate share servers in an efficient way.

Pool-Aware Scheduler Support

https://blueprints.launchpad.net/manila/+spec/dynamic-storage-pools

Manila currently sees each share backend as a whole, even if the backend consists of several smaller pools with totally different capabilities and capacities.

Extending manila to support storage pools within share backends will make manila scheduling decisions smarter as it now knows the full set of capabilities of a backend.

Problem Description

The provisioning decisions in manila are based on the statistics reported by backends. Any backend is assumed to be a single discrete unit with a set of capabilities and single capacity. In reality this assumption is not true for many storage providers, as their storage can be further divided or partitioned into pools to offer completely different sets of capabilities and capacities. That is, there are storage backends which are a combination of storage pools rather than a single homogeneous entity. Usually shares/snapshots cant be placed across pools on such backends.

In the current implementation, an attempt is made to map a single backend to a single storage controller, and the following problems may arise:

- After the scheduler selects a backend on which to place a new share, the backend may have to make a second decision about where to place the share within that backend. This logic is driver-specific and hard for admins to deal with.
- The capabilities that the backend reports back to the scheduler may not apply universally. A single backend may support both SATA and SSD-based storage, but perhaps not at the same time. Backends need a way to express exactly what they support and how much space is consumed out of each type of storage.

Therefore, it is important to extend manila so that it is aware of storage pools within each backend and can use them as the finest granularity for resource placement.

Proposed change

A pool-aware scheduler will address the need for supporting multiple pools from one storage backend.

Terminology

Pool A logical concept to describe a set of storage resources that can be used to serve core manila requests, e.g. shares/snapshots. This notion is almost identical to manila Share Backend, for it has similar attributes (capacity, capability). The difference is that a Pool may not exist on its own; it must reside in a Share Backend. One Share Backend can have multiple Pools but Pools do not have sub-Pools (meaning even if they have them, sub-Pools do not get to exposed to manila, yet). Each Pool has a unique name in the Share Backend namespace, which means a Share Backend cannot have two pools using same name.

Design

The workflow in this change is simple:

1) Share Backends report how many pools and what those pools look like and are capable of to scheduler;
2) When request comes in, scheduler picks a pool that fits the need best to serve the request, it passes the request to the backend where the target pool resides;
3) Share driver gets the message and lets the target pool serve the request as scheduler instructed.

To support placing resources (share/snapshot) onto a pool, these changes will be made to specific components of manila:

1. Share Backends reporting capacity/capabilities at pool level;
2. Scheduler filtering/weighing based on pool capacity/capability and placing shares/snapshots to a pool of a certain backend;
3. Record which backend and pool a resource is located on.

Data model impact

No DB schema change involved, however, the host field of Shares table will now include pool information but no DB migration is needed.

Original host field of Shares: `HostX@BackendY`

With this change: `HostX@BackendY#pool0`

REST API impact

With pool support added to manila, there is an awkward situation where we require admin to input the exact location for shares to be imported, which must have pool info. But there is no way to find out what pools are there for backends except looking at the scheduler log. That causes a poor user experience and thus is a problem from the Users Point of View. This change simply adds a new admin-api extension to allow admin to fetch all the pool information from scheduler cache (memory), which closes the gap for end users. This extension provides two level of pool information: names only or detailed information:

Pool name only: GET http://MANILA_API_ENDPOINT/v1/TENANT_ID/scheduler-stats/pools

Detailed Pool info: GET http://MANILA_API_ENDPOINT/v1/TENANT_ID/scheduler-stats/pools/detail

Security impact

N/A

Notifications impact

Host attribute of shares now includes pool information in it, consumer of notification can now extend to extract pool information if needed.

Admin impact

Administrators now need to suffix commands with `#pool` to manage shares.

Other end user impact

No impact visible to the end user directly, but administrators now need to prefix commands that refer to the backend host with the concatenation of the hashtag (#) sign and the name of the pool (e.g. `#poolName`) to manage shares. Other impacts might include scenarios where if a backend does not expose pools, the backend name is used as the pool name. For instance, `HostX@BackendY#BackendY` would be used when the driver does not expose pools.

Performance Impact

The size of RPC message for each share stats report will be bigger than before (linear to the number of pools a backend has). It should not really impact the RPC facility in terms of performance and even if it did, pure text compression should easily mitigate this problem.

Developer impact

For those share backends that would like to expose internal pools to manila for more flexibility, developers should update their drivers to include all pool capacities and capabilities in the share stats it reports to scheduler. Share backends without multiple pools do not need to change their implementation. Below is an example of new stats message having multiple pools:

```
{
    'share_backend_name': 'My Backend',    #\
    'vendor_name': 'OpenStack',            #  backend level
    'driver_version': '1.0',               #  mandatory/fixed
    'storage_protocol': 'NFS/CIFS',        #- stats&capabilities

    'active_shares': 10,                   #\
    'IOPS_provisioned': 30000,             #  optional custom
    'fancy_capability_1': 'eat',           #  stats & capabilities
    'fancy_capability_2': 'drink',         #/

    'pools': [
        {'pool_name': '1st pool',          #\
         'total_capacity_gb': 500,         #  mandatory stats for
         'free_capacity_gb': 230,          #  pools
         'allocated_capacity_gb': 270,     # |
         'qos': True,                      # |
         'reserved_percentage': 0,         #/

         'dying_disks': 100,               #\
         'super_hero_1': 'spider-man',     #  optional custom
         'super_hero_2': 'flash',          #  stats & capabilities
         'super_hero_3': 'neoncat'         #/
         },
        {'pool_name': '2nd pool',
         'total_capacity_gb': 1024,
         'free_capacity_gb': 1024,
         'allocated_capacity_gb': 0,
```

(continues on next page)

(continued from previous page)

```
            'qos': False,
            'reserved_percentage': 0,

            'dying_disks': 200,
            'super_hero_1': 'superman',
            'super_hero_2': ' ',
            'super_hero_2': 'Hulk',
            }
        ]
}
```

Documentation Impact

Documentation impact for changes in manila are introduced by the API changes. Also, doc changes are needed to append pool names to host names. Driver changes may also introduce new configuration options which would lead to Doc changes.

4.1.4 Other Resources

Project hosting with Launchpad

Launchpad hosts the manila project. The manila project homepage on Launchpad is http://launchpad.net/manila.

Launchpad credentials

Creating a login on Launchpad is important even if you dont use the Launchpad site itself, since Launchpad credentials are used for logging in on several OpenStack-related sites. These sites include:

- Wiki
- Gerrit (see *Code Reviews with Gerrit*)

Mailing list

The mailing list email is `openstack@lists.launchpad.net`. This is a common mailing list across the OpenStack projects. To participate in the mailing list:

1. Join the Manila Team on Launchpad.
2. Subscribe to the list on the OpenStack Team page on Launchpad.

The mailing list archives are at https://lists.launchpad.net/openstack.

Bug tracking

Report manila bugs at https://bugs.launchpad.net/manila

Feature requests (Blueprints)

Manila uses Launchpad Blueprints to track feature requests. Blueprints are at https://blueprints.launchpad.net/manila.

Technical support (Answers)

Manila uses Launchpad Answers to track manila technical support questions. The manila Answers page is at https://answers.launchpad.net/manila.

Note that the OpenStack Forums (which are not hosted on Launchpad) can also be used for technical support requests.

Code Reviews with Gerrit

Manila uses the Gerrit tool to review proposed code changes. The review site is https://review.opendev.org.

Gerrit is a complete replacement for Github pull requests. *All Github pull requests to the manila repository will be ignored.*

See the Development Workflow for more detailed documentation on how to work with Gerrit.

Manila team code review policy

Peer code review and the OpenStack Way

Manila adheres to the OpenStack code review policy and guidelines. Similar to other projects hosted on opendev.org, all of manilas code is curated and maintained by a small group of individuals called the core team. The primary core team consists of members from diverse affiliations. There are special core teams such as the manila release core team and the manila stable maintenance core team that have specific roles as the names suggest.

To make a code change in openstack/manila or any of the associated code repositories (openstack/manila-image-elements, openstack/manila-specs, openstack/manila-tempest-plugin, openstack/manila-test-image, openstack/manila-ui and openstack/python-manilaclient), contributors need to follow the *Code Submission Process* and upload their code on the OpenStack Gerrit website. They can then seek reviews by adding individual members of the manila core team or alert the entire core team by inviting the Gerrit group manila-core to the review. Anyone with a membership to the OpenStack Gerrit system may review the code change. However, only the core team can accept and merge the code change. Reviews from contributors outside the core team are encouraged. Reviewing code meticulously and often is a pre-requisite for contributors aspiring to join the core reviewer team.

One or more core reviewers will take cognizance of the contribution and provide feedback, or accept the code. For the submission to be accepted, it will need a minimum of one Code-Review:+2 and one Workflow:+1 votes, along with getting a Verified:+1 vote from the CI system. If no core reviewer pays

attention to a code submission, feel free to remind the team on the #openstack-manila IRC channel on irc.oftc.net.[1][2]

Core code review guidelines

By convention rather than rule, we require that a minimum of two code reviewers provide a Code-Review:+2 vote on each code submission before it is given a Workflow:+1 vote. Having two core reviewers approve a change adds diverse perspective, and is extremely valuable in case of:

- Feature changes in the manila service stack
- Changes to configuration options
- Addition of new tests or significant test bug-fixes in manila-tempest-plugin
- New features to manila-ui, manila-test-image, manila-image-elements
- Bug fixes

Trivial changes

Trivial changes are:

- Continuous Integration (CI) system break-fixes that are simple, i.e.:
  - No job or test is being deleted
  - Change does not break third-party CI
- Documentation changes, especially typographical fixes and grammar corrections.
- Automated changes generated by tooling - translations, lower-requirements changes, etc.

We do not need two core reviewers to approve trivial changes.

Affiliation of core reviewers

Previously, the manila core team informally enforced a code review convention that each code change be reviewed and merged by reviewers of different affiliations. This was followed because the OpenStack Technical Committee used the diversity of affiliation of the core reviewer team as a metric for maturity of the project. However, since the Rocky release cycle, the TC has changed its view on the subject[3][4]. We believe this is a step in the right direction.

While there is no strict requirement that two core reviewers accepting a code change have different affiliations. Other things being equal, we will continue to informally encourage organizational diversity by having core reviewers from different organizations. Core reviewers have the professional responsibility of avoiding conflicts of interest.

[1] Getting started with IRC: https://docs.openstack.org/contributors/common/irc.html
[2] IRC guidelines: https://docs.openstack.org/infra/manual/irc.html
[3] TC Report 18-28: https://anticdent.org/tc-report-18-28.html
[4] TC vote to remove team diversity tags: https://review.opendev.org/#/c/579870/

Vendor code and review

All code in the manila repositories is open-source and anyone can submit changes to these repositories as long as they seek to improve the code base. Manila supports over 30 vendor storage systems, and many of these vendors participate in the development and maintenance of their drivers. To the extent possible, core reviewers will seek out driver maintainer feedback on code changes pertaining to vendor integrations.

References

Manila Project Team Lead guide

A project team lead for Manila is elected from the project contributors. A candidate for PTL neednt be a core reviewer on the team, but, must be a contributor, and be familiar with the project to lead the project through its release process. If you would like to be a core reviewer begin by *Contacting the Core Team*. All the responsibilities below help us in maintaining the project. A project team lead can perform any of these or delegate tasks to other contributors.

General Responsibilities

- Ensure manila meetings have a chair
  - https://opendev.org/opendev/irc-meetings/src/branch/master/meetings/manila-team-meeting.yaml
- Update the team people wiki
  - https://wiki.openstack.org/wiki/Manila#People

Release cycle activities

- Get acquainted with the release schedule and set Project specific milestones in the OpenStack Releases repository
  - Example: https://releases.openstack.org/victoria/schedule.html
- Ensure the Manila Cross Project Liaisons are aware of their duties and are plugged into the respective areas
- Acknowledge community wide cycle goals and find leaders and coordinate with the goal liaisons
- Plan team activities such as:
  - `Documentation day/s` to groom documentation bugs and re-write release cycle docs
  - `Bug Triage day/s` to ensure the bug backlog is well groomed
  - `Bug Squash day/s` to close bugs
  - `Collaborative Review meeting/s` to perform a high-touch review of a code submission over a synchronous call
- Milestone driven work:
  - `Milestone-1`:

    - Request a release for the python-manilaclient and manila-ui
    - Retarget any bugs whose fixes missed Milestone-1
- `Milestone-2`:
    - Retarget any bugs whose fixes missed Milestone-2
    - Create a review priority etherpad and share it with the community and have reviewers sign up
- `Milestone-3`:
    - Groom the release notes for python-manilaclient and add a prelude section describing the most important changes in the release
    - Request a final cycle release for python-manilaclient
    - Retarget any bugs whose fixes missed Milestone-3
    - Grant/Deny any Feature Freeze Exception Requests
    - Update task trackers for Community Wide Goals
    - Write the cycle-highlights in marketing-friendly sentences and propose to the openstack/releases repo. Usually based on reno prelude but made more readable and friendly
        - Example: https://review.opendev.org/717801/
    - Create the launchpad series and milestones for the next cycle in manila, python-manilaclient and manila-ui. Examples:
        - manila: https://launchpad.net/manila/ussuri
        - python-manilaclient: https://launchpad.net/python-manilaclient/ussuri
        - manila-ui: https://launchpad.net/manila-ui/ussuri
- `Before RC-1`:
    - Groom the release notes for manila-ui and add a prelude section describing the most important changes in the release
    - Request a final cycle release for manila-ui
    - Groom the release notes for manila, add a prelude section describing the most important changes in the release
    - Mark bugs as {release}-rc-potential bugs in launchpad, ensure they are targeted and addressed by RC
- `RC-1`:
    - Request a RC-1 release for manila
    - Request a final cycle tagged release for manila-tempest-plugin
    - Ensure all blueprints for the release have been marked Implemented or are re-targeted
- `After RC-1`:
    - Close the currently active series on Launchpad for manila, python-manilaclient and manila-ui and set the Development Focus to the next release. Alternatively, you can switch this on the series page by setting the next release to active development

        * Set the last series status in each of these projects to current stable branch release
        * Set the previous releases series status to supported
        * Move any Unimplemented specs in the specs repo to Unimplemented
        * Create a new specs directory in the specs repo for the next cycle so people can start proposing new specs
- You should NOT plan to have more than one RC. RC2 should only happen if there was a mistake and something was missed for RC-1, or a new regression was discovered
- Periodically during the release:
    - `Every Week`:
        * Coordinate the weekly Community Meeting agenda
        * Coordinate with the Bug Czar and ensure bugs are properly triaged
        * Check whether any bug-fixes must be back-ported to older stable releases
    - `Every 3 weeks`:
        * Ensure stable branch releases are proposed in case there are any release worthy changes. If there are only documentation or CI/test related fixes, no release for that branch is necessary
- To request a release of any manila deliverable:
    - `git checkout {branch-to-release-from}`
    - `git log --no-merges {last tag}..`
        * Examine commits that will go into the release and use it to decide whether the release is a major, minor, or revision bump according to semver
    - Then, propose the release with version according to semver x.y.z
        * X - backward-incompatible changes
        * Y - features
        * Z - bug fixes
    - Use the `new-release` command to generate the release
        * https://releases.openstack.org/reference/using.html#using-new-release-command

Project Team Gathering

- Create etherpads for PTG planning, cycle retrospective and PTG discussions and announce the Planning etherpad to the community members via the Manila community meeting as well as the *OpenStack Discuss Mailing List*
    - Example PTG Planning Etherpad
    - Example Retrospective Etherpad
    - Example PTG Discussions Etherpad

- If the PTG is a physical event, gather an estimate of attendees and request the OpenDev Foundation staff for appropriate meeting space. Ensure the sessions are remote attendee friendly. Coordinate A/V logistics
- Set discussion schedule and find an owner to run each proposed discussion at the PTG
- All sessions must be recorded, nominate note takers for each discussion
- Sign up for group photo at the PTG (if applicable)
- After the event, send PTG session summaries and the meeting recording to the *OpenStack Discuss Mailing List*

Summit

- Prepare the project update presentation. Enlist help of others
- Prepare the on-boarding session materials. Enlist help of others

4.1.5 API Reference

API Microversions

Background

Manila uses a framework we called API Microversions for allowing changes to the API while preserving backward compatibility. The basic idea is that a user has to explicitly ask for their request to be treated with a particular version of the API. So breaking changes can be added to the API without breaking users who dont specifically ask for it. This is done with an HTTP header `X-OpenStack-Manila-API-Version` which is a monotonically increasing semantic version number starting from `1.0`.

If a user makes a request without specifying a version, they will get the `DEFAULT_API_VERSION` as defined in `manila/api/openstack/api_version_request.py`. This value is currently `2.0` and is expected to remain so for quite a long time.

The Nova project was the first to implement microversions. For full details please read Novas Kilo spec for microversions

When do I need a new Microversion?

A microversion is needed when the contract to the user is changed. The user contract covers many kinds of information such as:

- the Request
  - the list of resource urls which exist on the server

    Example: adding a new shares/{ID}/foo which didnt exist in a previous version of the code
  - the list of query parameters that are valid on urls

    Example: adding a new parameter `is_yellow` servers/{ID}?is_yellow=True

  - the list of query parameter values for non free form fields

    Example: parameter filter_by takes a small set of constants/enums A, B, C. Adding support for new enum D.

  - new headers accepted on a request

- the Response
  - the list of attributes and data structures returned

    Example: adding a new attribute locked: True/False to the output of shares/{ID}

  - the allowed values of non free form fields

    Example: adding a new allowed `status` to shares/{ID}

  - the list of status codes allowed for a particular request

    Example: an API previously could return 200, 400, 403, 404 and the change would make the API now also be allowed to return 409.

  - changing a status code on a particular response

    Example: changing the return code of an API from 501 to 400.

  - new headers returned on a response

The following flow chart attempts to walk through the process of do we need a microversion.

Footnotes

[1] - When fixing 500 errors that previously caused stack traces, try to map the new error into the existing set of errors that API call could previously return (400 if nothing else is appropriate). Changing the set of allowed status codes from a request is changing the contract, and should be part of a microversion.

The reason why we are so strict on contract is that wed like application writers to be able to know, for sure, what the contract is at every microversion in manila. If they do not, they will need to write conditional code in their application to handle ambiguities.

When in doubt, consider application authors. If it would work with no client side changes on both manila versions, you probably dont need a microversion. If, on the other hand, there is any ambiguity, a microversion is probably needed.

In Code

In `manila/api/openstack/wsgi.py` we define an `@api_version` decorator which is intended to be used on top-level Controller methods. It is not appropriate for lower-level methods. Some examples:

Adding a new API method

In the controller class:

```
@wsgi.Controller.api_version("2.4")
def my_api_method(self, req, id):
    ....
```

This method would only be available if the caller had specified an `X-OpenStack-Manila-API-Version` of `>= 2.4`. If they had specified a lower version (or not specified it and received the default of `2.1`) the server would respond with `HTTP/404`.

Removing an API method

In the controller class:

```
@wsgi.Controller.api_version("2.1", "2.4")
def my_api_method(self, req, id):
    ....
```

This method would only be available if the caller had specified an `X-OpenStack-Manila-API-Version` of `<= 2.4`. If `2.5` or later is specified the server will respond with `HTTP/404`.

Changing a methods behaviour

In the controller class:

```
@wsgi.Controller.api_version("2.1", "2.3")
def my_api_method(self, req, id):
    .... method_1 ...

@wsgi.Controller.api_version("2.4")  # noqa
def my_api_method(self, req, id):
    .... method_2 ...
```

If a caller specified `2.1`, `2.2` or `2.3` (or received the default of `2.1`) they would see the result from `method_1`, `2.4` or later `method_2`.

It is vital that the two methods have the same name, so the second of them will need `# noqa` to avoid failing flake8s `F811` rule. The two methods may be different in any kind of semantics (schema validation, return values, response codes, etc)

A method with only small changes between versions

A method may have only small changes between microversions, in which case you can decorate a private method:

```
@api_version("2.1", "2.4")
def _version_specific_func(self, req, arg1):
    pass

@api_version(min_version="2.5")  # noqa
def _version_specific_func(self, req, arg1):
    pass

def show(self, req, id):
    .... common stuff ....
    self._version_specific_func(req, "foo")
    .... common stuff ....
```

A change in schema only

If there is no change to the method, only to the schema that is used for validation, you can add a version range to the `validation.schema` decorator:

```
@wsgi.Controller.api_version("2.1")
@validation.schema(dummy_schema.dummy, "2.3", "2.8")
@validation.schema(dummy_schema.dummy2, "2.9")
def update(self, req, id, body):
    ....
```

This method will be available from version `2.1`, validated according to `dummy_schema.dummy` from `2.3` to `2.8`, and validated according to `dummy_schema.dummy2` from `2.9` onward.

When not using decorators

When you dont want to use the `@api_version` decorator on a method or you want to change behaviour within a method (say it leads to simpler or simply a lot less code) you can directly test for the requested version with a method as long as you have access to the api request object (commonly called `req`). Every API method has an api_version_request object attached to the req object and that can be used to modify behaviour based on its value:

```
def index(self, req):
    <common code>

    req_version = req.api_version_request
    if req_version.matches("2.1", "2.5"):
        ....stuff....
    elif req_version.matches("2.6", "2.10"):
        ....other stuff....
    elif req_version > api_version_request.APIVersionRequest("2.10"):
```

(continues on next page)

(continued from previous page)

```
        ....more stuff.....

    <common code>
```

The first argument to the matches method is the minimum acceptable version and the second is maximum acceptable version. A specified version can be null:

```
null_version = APIVersionRequest()
```

If the minimum version specified is null then there is no restriction on the minimum version, and likewise if the maximum version is null there is no restriction the maximum version. Alternatively a one sided comparison can be used as in the example above.

Other necessary changes

If you are adding a patch which adds a new microversion, it is necessary to add changes to other places which describe your change:

- Update `REST_API_VERSION_HISTORY` in `manila/api/openstack/api_version_request.py`
- Update `_MAX_API_VERSION` in `manila/api/openstack/api_version_request.py`
- Add a verbose description to `manila/api/openstack/rest_api_version_history.rst`. There should be enough information that it could be used by the docs team for release notes.
- Update the expected versions in affected tests.

Allocating a microversion

If you are adding a patch which adds a new microversion, it is necessary to allocate the next microversion number. Except under extremely unusual circumstances and this would have been mentioned in the blueprint for the change, the minor number of `_MAX_API_VERSION` will be incremented. This will also be the new microversion number for the API change.

It is possible that multiple microversion patches would be proposed in parallel and the microversions would conflict between patches. This will cause a merge conflict. We dont reserve a microversion for each patch in advance as we dont know the final merge order. Developers may need over time to rebase their patch calculating a new version number as above based on the updated value of `_MAX_API_VERSION`.

Testing Microversioned API Methods

Testing a microversioned API method is very similar to a normal controller method test, you just need to add the `X-OpenStack-Manila-API-Version` header, for example:

```
req = fakes.HTTPRequest.blank('/testable/url/endpoint')
req.headers = {'X-OpenStack-Manila-API-Version': '2.2'}
req.api_version_request = api_version.APIVersionRequest('2.6')
```

(continues on next page)

(continued from previous page)

```
controller = controller.TestableController()

res = controller.index(req)
... assertions about the response ...
```

REST API Version History

This documents the changes made to the REST API with every microversion change. The description for each version should be a verbose one which has enough information to be suitable for use in user documentation.

1.0 (Maximum in Kilo)

The 1.0 Manila API includes all v1 core APIs existing prior to the introduction of microversions. The /v1 URL is used to call 1.0 APIs, and microversions headers sent to this endpoint are ignored.

2.0

This is the initial version of the Manila API which supports microversions. The /v2 URL is used to call 2.x APIs.

A user can specify a header in the API request:

```
X-OpenStack-Manila-API-Version: <version>
```

where `<version>` is any valid api version for this API.

If no version is specified then the API will behave as if version 2.0 was requested.

The only API change in version 2.0 is versions, i.e. GET http://localhost:8786/, which now returns information about both 1.0 and 2.x versions and their respective /v1 and /v2 endpoints.

All other 2.0 APIs are functionally identical to version 1.0.

2.1

Share create() method doesnt ignore availability_zone field of provided share.

2.2

Snapshots become optional and share payload now has boolean attr snapshot_support.

2.3

Share instances admin API and update of Admin Actions extension.

2.4

Consistency groups support. /consistency-groups and /cgsnapshots are implemented. AdminActions os-force_delete and os-reset_status have been updated for both new resources.

2.5

Share Migration admin API.

2.6 (Maximum in Liberty)

Return share_type UUID instead of name in Share API and add share_type_name field.

2.7

Rename old extension-like API URLs to core-API-like.

2.8

Allow to set share visibility explicitly using manage API.

2.9

Add export locations API. Remove export locations from shares and share instances APIs.

2.10

Field access_rules_status was added to shares and share instances.

2.11

Share Replication support added. All Share replication APIs are tagged Experimental. Share APIs return two new attributes: has_replicas and replication_type. Share instance APIs return a new attribute, replica_state.

2.12

Share snapshot manage and unmanage API.

2.13

Add cephx authentication type for the CephFS Native driver.

2.14

Added attribute preferred to export locations. Drivers may use this field to identify which export locations are most efficient and should be used preferentially by clients. Also, change uuid field to id, move timestamps to detail view, and return all non-admin fields to users.

2.15 (Maximum in Mitaka)

Added Share migration migration_cancel, migration_get_progress, migration_complete APIs, renamed migrate_share to migration_start and added notify parameter to migration_start.

2.16

Add user_id in share show/create/manage API.

2.17

Added user_id and project_id in snapshot show/create/manage APIs.

2.18

Add gateway in share network show API.

2.19

Add admin APIs(list/show/detail/reset-status) of snapshot instances.

2.20

Add MTU in share network show API.

2.21

Add access_key in access_list API.

2.22 (Maximum in Newton)

Updated migration_start API with preserve_metadata, writable, nondisruptive and new_share_network_id parameters, renamed force_host_copy to force_host_assisted_migration, removed notify parameter and removed previous migrate_share API support. Updated reset_task_state API to accept None value.

2.23

Added share_type to filter results of scheduler-stats/pools API.

2.24

Added optional create_share_from_snapshot_support extra spec. Made snapshot_support extra spec optional.

2.25

Added quota-show detail API.

2.26

Removed nova-net plugin support and removed nova_net_id parameter from share_network API.

2.27

Added share revert to snapshot. This API reverts a share to the specified snapshot. The share is reverted in place, and the snapshot must be the most recent one known to manila. The feature is controlled by a new standard optional extra spec, revert_to_snapshot_support.

2.28

Added transitional states (queued_to_apply - was previously new, queued_to_deny, applying and denying) to access rules. updating, updating_multiple and out_of_sync are no longer valid values for the access_rules_status field of shares, they have been collapsed into the transitional state syncing. Access rule changes can be made independent of a shares access_rules_status.

2.29

Updated migration_start API adding mandatory parameter preserve_snapshots and changed preserve_metadata, writable, nondisruptive to be mandatory as well. All previous migration_start APIs prior to this microversion are now unsupported.

2.30

Added cast_rules_to_readonly field to share_instances.

2.31

Convert consistency groups to share groups.

2.32 (Maximum in Ocata)

Added mountable snapshots APIs.

2.33

Added created_at and updated_at in access_list API.

2.34

Added availability_zone_id and consistent_snapshot_support fields to share_group object.

2.35

Added support to retrieve shares filtered by export_location_id and export_location_path.

2.36

Added like filter support in `shares`, `snapshots`, `share-networks`, `share-groups` list APIs.

2.37

Added /messages APIs.

2.38

Support IPv6 format validation in allow_access API to enable IPv6.

2.39

Added share-type quotas.

2.40 (Maximum in Pike)

Added share group and share group snapshot quotas.

2.41

Added description in share type create/list APIs.

2.42 (Maximum in Queens)

Added `with_count` in share list API to get total count info.

2.43

Added filter search by extra spec for share type list.

2.44

Added ou field to security_service object.

2.45

Added access metadata for share access and also introduced the GET /share-access-rules API. The prior API to retrieve access rules will not work with API version >=2.45.

2.46 (Maximum in Rocky)

Added is_default field to share_type and share_group_type objects.

2.47

Export locations for non-active share replicas are no longer retrievable through the export locations APIs: `GET /v2/{tenant_id}/shares/{share_id}/export_locations` and `GET /v2/{tenant_id}/shares/{share_id}/export_locations/{export_location_id}`. A new API is introduced at this version: `GET /v2/{tenant_id}/share-replicas/{replica_id}/export-locations` to allow retrieving export locations of share replicas if available.

2.48

Administrators can now use the common, user-visible extra-spec availability_zones within share types to allow provisioning of shares only within specific availability zones. The extra-spec allows using comma separated names of one or more availability zones.

2.49 (Maximum in Stein)

Added Manage/Unmanage Share Server APIs. Updated Manage/Unmanage Shares and Snapshots APIs to work in `driver_handles_shares_servers` enabled mode.

2.50

Added update share type API to Share Type APIs. We can update the `name`, `description` and/or `share_type_access:is_public` fields of the share type by the update share type API.

2.51 (Maximum in Train)

Added to the service the possibility to have multiple subnets per share network, each of them associated to a different AZ. It is also possible to configure a default subnet that spans all availability zones.

2.52

Added created_before and created_since field to list messages api, support querying user messages within the specified time period.

2.53

Added quota control for share replicas and replica gigabytes.

2.54

Share and share instance objects include a new field called progress which indicates the completion of a share creation operation as a percentage.

2.55 (Maximum in Ussuri)

Share groups feature is no longer considered experimental.

2.56

Share replication feature is no longer considered experimental.

2.57 (Maximum in Victoria)

Added share server migration feature. A two-phase approach that migrates a share server and all its resources to a new host.

2.58

Added share_groups and share_group_snapshots to the limits view.

2.59

Added details field to migration get progress api, which optionally may hold additional driver data related to the progress of share migration.

2.60

API URLs no longer need a project_id argument in them. For example, the API route: https://\protect\T1\textdollar(controller)s/share/v2/\protect\T1\textdollar(project_id)s/shares is equivalent to https://\protect\T1\textdollar(controller)s/share/v2/shares. When interacting with the manila service as system or domain scoped users, project_id should not be specified in the API path.

2.61

Ability to add minimum and maximum share size restrictions which can be set on a per share-type granularity. Added new extra specs provisioning:max_share_size and provisioning:min_share_size.

2.62

Added quota control to per share size.

2.63 (Maximum in Wallaby)

Added the possibility to attach security services to share networks in use. Also, an attached security service can be replaced for another one of the same type. In order to support those operations a status field was added in the share networks as well as, a new property called security_service_update_support was included in the share networks and share servers. Also new action APIs have been added to the share-networks endpoint: update_security_service, update_security_service_check and add_security_service_check.

2.64

Added force field to extend share api, which can extend share directly without go through share scheduler.

2.65 (Maximum in Xena)

Added ability to specify scheduler_hints in the request body of the POST /shares request. These hints will invoke Affinity/Anti-Affinity scheduler filters during share creation and share migration.

Experimental APIs

Background

Manila uses API microversions to allow natural evolution of its REST APIs over time. But microversions alone cannot solve the question of how to ship APIs that are experimental in nature, are expected to change at any time, and could even be removed entirely without a typical deprecation period.

In conjunction with microversions, manila has added a facility for marking individual REST APIs as experimental. To call an experimental API, clients must include a specific HTTP header, `X-OpenStack-Manila-API-Experimental`, with a value of `True`. If a user calls an experimental API without including the experimental header, the server would respond with `HTTP/404`. This forces the client to acknowledge the experimental status of the API and prevents anyone from building an application around a manila feature without realizing the feature could change significantly or even disappear.

On the other hand, if a request is made to a non-experimental manila API with `X-OpenStack-Manila-API-Experimental: True`, the server would respond as if the header had not been included. This is a convenience mechanism, as it allows the client to specify both the requested API version as well as the experimental header (if desired) in one place instead of having to set the headers separately for each API call (although that would be fine, too).

When do I need to set an API experimental?

An API should be marked as experimental if any of the following is true:

- the API is not yet considered a stable, core API
- the API is expected to change in later releases
- the API could be removed altogether if a feature is redesigned
- the API controls a feature that could change or be removed

When do I need to remove the experimental annotation from an API?

When the community is satisfied that an experimental feature and its APIs have had sufficient time to gather and incorporate user feedback to consider it stable, which could be one or more OpenStack release cycles, any relevant APIs must be re-released with a microversion bump and without the experimental flag. The maturation period can vary between features, but experimental is NOT a stable state, and an experimental feature should not be left in that state any longer than necessary.

Because experimental APIs have no conventional deprecation period, the manila core team may optionally choose to remove any experimental versions of an API at the same time that a microversioned stable version is added.

In Code

The `@api_version` decorator defined in `manila/api/openstack/wsgi.py`, which is used for specifying API versions on top-level Controller methods, also allows for tagging an API as experimental. For example:

In the controller class:

```
@wsgi.Controller.api_version("2.4", experimental=True)
def my_api_method(self, req, id):
    ....
```

This method would only be available if the caller had specified an `X-OpenStack-Manila-API-Version` of `>= 2.4`. and had also included `X-OpenStack-Manila-API-Experimental: True`. If they had specified a lower version (or not specified it and received a lower default version), or if they had failed to include the experimental header, the server would respond with `HTTP/404`.

4.1.6 Module Reference

Introduction to the Shared File Systems service

Manila is the file share service project for OpenStack. Manila provides the management of file shares for example, NFS and CIFS as a core service to OpenStack. Manila works with a variety of proprietary backend storage arrays and appliances, with open source distributed filesystems, as well as with a base Linux NFS or Samba server. There are a number of concepts that will help in better understanding of the solutions provided by manila. One aspect can be to explore the different service possibilities provided by manila.

Manila, depending on the driver, requires the user by default to create a share network using neutron-net-id and neutron-subnet-id (GlusterFS native driver does not require it). After creation of the share network, the user can proceed to create the shares. Users in manila can configure multiple back-ends just like Cinder. Manila has a share server assigned to every tenant. This is the solution for all back-ends except for GlusterFS. The customer in this scenario is prompted to create a share server using neutron net-id and subnet-id before even trying to create a share.

The current low-level services available in manila are:

- *manila-api*
- *manila-scheduler*
- *manila-share*

Services, Managers and Drivers

The responsibilities of Services, Managers, and Drivers, can be a bit confusing to people that are new to manila. This document attempts to outline the division of responsibilities to make understanding the system a little bit easier.

Currently, Managers and Drivers are specified by flags and loaded using utils.load_object(). This method allows for them to be implemented as singletons, classes, modules or objects. As long as the path specified by the flag leads to an object (or a callable that returns an object) that responds to getattr, it should work as a manager or driver.

The manila.service Module

Generic Node base class for all workers that run on hosts.

class Service(*host*, *binary*, *topic*, *manager*, *report_interval=None*, *periodic_interval=None*, *periodic_fuzzy_delay=None*, *service_name=None*, *coordination=False*, **args*, ***kwargs*)

Bases: oslo_service.service.Service

Service object for binaries running on hosts.

A service takes a manager and enables rpc by listening to queues based on topic. It also periodically runs tasks on the manager and reports it state to the database services table.

classmethod create(*host=None*, *binary=None*, *topic=None*, *manager=None*, *report_interval=None*, *periodic_interval=None*, *periodic_fuzzy_delay=None*, *service_name=None*, *coordination=False*)

Instantiates class and passes back application object.

Parameters

- **host** defaults to CONF.host
- **binary** defaults to basename of executable
- **topic** defaults to bin_name - manila- part
- **manager** defaults to CONF.<topic>_manager
- **report_interval** defaults to CONF.report_interval
- **periodic_interval** defaults to CONF.periodic_interval
- **periodic_fuzzy_delay** defaults to CONF.periodic_fuzzy_delay

kill()

Destroy the service object in the datastore.

periodic_tasks(*raise_on_error=False*)

Tasks to be run at a periodic interval.

report_state()

Update the state of this service in the datastore.

start()

Start a service.

stop()

Stop a service.

Parameters **graceful** indicates whether to wait for all threads to finish or terminate them instantly

wait()

Wait for a service to shut down.

class WSGIService(*name*, *loader=None*)

Bases: oslo_service.service.ServiceBase

Provides ability to launch API from a paste configuration.

reset()
Reset server greenpool size to default.

Returns None

start()
Start serving this service using loaded configuration.

Also, retrieve updated port number in case 0 was passed in, which indicates a random port should be used.

Returns None

stop()
Stop serving this API.

Returns None

wait()
Wait for the service to stop serving this API.

Returns None

process_launcher()

serve(*server*, *workers=None*)

setup_profiler(*binary*, *host*)

wait()

The manila.manager Module

Base Manager class.

Managers are responsible for a certain aspect of the system. It is a logical grouping of code relating to a portion of the system. In general other components should be using the manager to make changes to the components that it is responsible for.

For example, other components that need to deal with volumes in some way, should do so by calling methods on the VolumeManager instead of directly changing fields in the database. This allows us to keep all of the code relating to volumes in the same place.

We have adopted a basic strategy of Smart managers and dumb data, which means rather than attaching methods to data objects, components should call manager methods that act on the data.

Methods on managers that can be executed locally should be called directly. If a particular method must execute on a remote host, this should be done via rpc to the service that wraps the manager

Managers should be responsible for most of the db access, and non-implementation specific data. Anything implementation specific that cant be generalized should be done by the Driver.

In general, we prefer to have one manager with multiple drivers for different implementations, but sometimes it makes sense to have multiple managers. You can think of it this way: Abstract different overall strategies at the manager level(FlatNetwork vs VlanNetwork), and different implementations at the driver level(LinuxNetDriver vs CiscoNetDriver).

Managers will often provide methods for initial setup of a host or periodic tasks to a wrapping service.

This module provides Manager, a base class for managers.

class Manager(*host=None*, *db_driver=None*)
Bases: manila.db.base.Base, manila.manager.PeriodicTasks

property RPC_API_VERSION
Redefine this in child classes.

init_host()
Handle initialization if this is a standalone service.

A hook point for services to execute tasks before the services are made available (i.e. showing up on RPC and starting to accept RPC calls) to other components. Child classes should override this method.

init_host_with_rpc()
A hook for service to do jobs after RPC is ready.

Like init_host(), this method is a hook where services get a chance to execute tasks that *need* RPC. Child classes should override this method.

is_service_ready()
Method indicating if service is ready.

This method should be overridden by subclasses which will return False when the back end is not ready yet.

periodic_tasks(*context*, *raise_on_error=False*)
Tasks to be run at a periodic interval.

service_config(*context*)

service_version(*context*)

property target
This property is used by oslo_messaging.

https://wiki.openstack.org/wiki/Oslo/Messaging#API_Version_Negotiation

class PeriodicTasks
Bases: oslo_service.periodic_task.PeriodicTasks

class SchedulerDependentManager(*host=None*, *db_driver=None*, *service_name='undefined'*)
Bases: manila.manager.Manager

Periodically send capability updates to the Scheduler services.

Services that need to update the Scheduler of their capabilities should derive from this class. Otherwise they can derive from manager.Manager directly. Updates are only sent after update_service_capabilities is called with non-None values.

update_service_capabilities(*capabilities*)
Remember these capabilities to send on next periodic update.

Implementation-Specific Drivers

A manager will generally load a driver for some of its tasks. The driver is responsible for specific implementation details. Anything running shell commands on a host, or dealing with other non-python code should probably be happening in a driver.

Drivers should minimize touching the database, although it is currently acceptable for implementation specific data. This may be reconsidered at some point.

It usually makes sense to define an Abstract Base Class for the specific driver (i.e. VolumeDriver), to define the methods that a different driver would need to implement.

The Database Layer

The `manila.db.api` Module

Defines interface for DB access.

The underlying driver is loaded as a `LazyPluggable`.

Functions in this module are imported into the manila.db namespace. Call these functions from manila.db namespace, not the manila.db.api namespace.

All functions in this module return objects that implement a dictionary-like interface. Currently, many of these objects are sqlalchemy objects that implement a dictionary interface. However, a future goal is to have all of these objects be simple dictionaries.

Related Flags

backend string to lookup in the list of LazyPluggable backends. *sqlalchemy* is the only supported backend right now.

connection string specifying the sqlalchemy connection to use, like: *sqlite:///var/lib/manila/manila.sqlite*.

enable_new_services when adding a new service to the database, is it in the pool of available hardware (Default: True)

`async_operation_data_delete`(*context*, *entity_id*, *key=None*)
Remove one, list or all key-value pairs for given entity_id.

`async_operation_data_get`(*context*, *entity_id*, *key=None*, *default=None*)
Get one, list or all key-value pairs for given entity_id.

`async_operation_data_update`(*context*, *entity_id*, *details*, *delete_existing=False*)
Update key-value pairs for given entity_id.

`authorize_project_context`(*context*, *project_id*)
Ensures a request has permission to access the given project.

`authorize_quota_class_context`(*context*, *class_name*)
Ensures a request has permission to access the given quota class.

`availability_zone_get`(*context*, *id_or_name*)
Get availability zone by name or id.

`availability_zone_get_all`(*context*)
Get all active availability zones.

backend_info_get(*context*, *host*)
Get hash info for given host.

backend_info_update(*context*, *host*, *value=None*, *delete_existing=False*)
Update hash info for host.

cleanup_expired_messages(*context*)
Soft delete expired messages

count_share_group_snapshot_members_in_share(*context*, *share_id*, *session=None*)
Returns the number of group snapshot members linked to the share.

count_share_group_snapshots_in_share_group(*context*, *share_group_id*)
Returns the number of sg snapshots with the specified share group.

count_share_groups_in_share_network(*context*, *share_network_id*, *session=None*)
Return the number of groups with the specified share network.

count_share_networks(*context*, *project_id*, *user_id=None*, *share_type_id=None*, *session=None*)

count_shares_in_share_group(*context*, *share_group_id*)
Returns the number of undeleted shares with the specified group.

driver_private_data_delete(*context*, *entity_id*, *key=None*)
Remove one, list or all key-value pairs for given entity_id.

driver_private_data_get(*context*, *entity_id*, *key=None*, *default=None*)
Get one, list or all key-value pairs for given entity_id.

driver_private_data_update(*context*, *entity_id*, *details*, *delete_existing=False*)
Update key-value pairs for given entity_id.

export_location_metadata_delete(*context*, *export_location_uuid*, *keys*, *session=None*)
Delete metadata of an export location.

export_location_metadata_get(*context*, *export_location_uuid*, *session=None*)
Get all metadata of an export location.

export_location_metadata_update(*context*, *export_location_uuid*, *metadata*, *delete*, *session=None*)
Update metadata of an export location.

get_all_shares_by_share_group(*context*, *share_group_id*)

message_create(*context*, *values*)
Creates a new message with the specified values.

message_destroy(*context*, *message_id*)
Deletes message with the specified ID.

message_get(*context*, *message_id*)
Return a message with the specified ID.

message_get_all(*context*, *filters=None*, *limit=None*, *offset=None*, *sort_key=None*, *sort_dir=None*)
Returns all messages with the project of the specified context.

network_allocation_create(*context*, *values*)
Create a network allocation DB record.

network_allocation_delete(*context*, *id*)
Delete a network allocation DB record.

network_allocation_get(*context*, *id*, *session=None*, *read_deleted=None*)
Get a network allocation DB record.

network_allocation_update(*context*, *id*, *values*, *read_deleted=None*)
Update a network allocation DB record.

network_allocations_get_by_ip_address(*context*, *ip_address*)
Get network allocations by IP address.

network_allocations_get_for_share_server(*context*, *share_server_id*, *session=None*, *label=None*)
Get network allocations for share server.

purge_deleted_records(*context*, *age_in_days*)
Purge deleted rows older than given age from all tables

Raises InvalidParameterValue if age_in_days is incorrect.

quota_class_create(*context*, *class_name*, *resource*, *limit*)
Create a quota class for the given name and resource.

quota_class_get(*context*, *class_name*, *resource*)
Retrieve a quota class or raise if it does not exist.

quota_class_get_all_by_name(*context*, *class_name*)
Retrieve all quotas associated with a given quota class.

quota_class_get_default(*context*)
Retrieve all default quotas.

quota_class_update(*context*, *class_name*, *resource*, *limit*)
Update a quota class or raise if it does not exist.

quota_create(*context*, *project_id*, *resource*, *limit*, *user_id=None*, *share_type_id=None*)
Create a quota for the given project and resource.

quota_destroy_all_by_project(*context*, *project_id*)
Destroy all quotas associated with a given project.

quota_destroy_all_by_project_and_user(*context*, *project_id*, *user_id*)
Destroy all quotas associated with a given project and user.

quota_destroy_all_by_share_type(*context*, *share_type_id*, *project_id=None*)
Destroy all quotas associated with a given share type and project.

quota_get_all(*context*, *project_id*)
Retrieve all user quotas associated with a given project.

quota_get_all_by_project(*context*, *project_id*)
Retrieve all quotas associated with a given project.

quota_get_all_by_project_and_share_type(*context*, *project_id*, *share_type_id*)
Retrieve all quotas associated with a given project and user.

quota_get_all_by_project_and_user(*context*, *project_id*, *user_id*)
Retrieve all quotas associated with a given project and user.

quota_reserve(*context*, *resources*, *quotas*, *user_quotas*, *share_type_quotas*, *deltas*, *expire*, *until_refresh*, *max_age*, *project_id=None*, *user_id=None*, *share_type_id=None*, *overquota_allowed=False*)
Check quotas and create appropriate reservations.

quota_update(*context*, *project_id*, *resource*, *limit*, *user_id=None*, *share_type_id=None*)
Update a quota or raise if it does not exist.

quota_usage_create(*context*, *project_id*, *user_id*, *resource*, *in_use*, *reserved=0*, *until_refresh=None*, *share_type_id=None*)
Create a quota usage.

quota_usage_get(*context*, *project_id*, *resource*, *user_id=None*, *share_type_id=None*)
Retrieve a quota usage or raise if it does not exist.

quota_usage_get_all_by_project(*context*, *project_id*)
Retrieve all usage associated with a given resource.

quota_usage_get_all_by_project_and_share_type(*context*, *project_id*, *share_type_id*)
Retrieve all usage associated with a given resource.

quota_usage_get_all_by_project_and_user(*context*, *project_id*, *user_id*)
Retrieve all usage associated with a given resource.

quota_usage_update(*context*, *project_id*, *user_id*, *resource*, *share_type_id=None*, ***kwargs*)
Update a quota usage or raise if it does not exist.

reservation_commit(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)
Commit quota reservations.

reservation_expire(*context*)
Roll back any expired reservations.

reservation_rollback(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)
Roll back quota reservations.

security_service_create(*context*, *values*)
Create security service DB record.

security_service_delete(*context*, *id*)
Delete security service DB record.

security_service_get(*context*, *id*, ***kwargs*)
Get security service DB record.

security_service_get_all(*context*)
Get all security service DB records.

security_service_get_all_by_project(*context*, *project_id*)
Get all security service DB records for the given project.

security_service_update(*context*, *id*, *values*)
Update security service DB record.

service_create(*context*, *values*)
Create a service from the values dictionary.

service_destroy(*context*, *service_id*)
Destroy the service or raise if it does not exist.

service_get(*context*, *service_id*)
Get a service or raise if it does not exist.

service_get_all(*context*, *disabled=None*)
Get all services.

service_get_all_by_topic(*context*, *topic*)
Get all services for a given topic.

service_get_all_share_sorted(*context*)
Get all share services sorted by share count.

Returns a list of (Service, share_count) tuples.

service_get_by_args(*context*, *host*, *binary*)
Get the state of an service by node name and binary.

service_get_by_host_and_topic(*context*, *host*, *topic*)
Get a service by host its on and topic it listens to.

service_update(*context*, *service_id*, *values*)
Set the given properties on an service and update it.

Raises NotFound if service does not exist.

share_access_check_for_existing_access(*context*, *share_id*, *access_type*, *access_to*)
Returns True if rule corresponding to the type and client exists.

share_access_create(*context*, *values*)
Allow access to share.

share_access_get(*context*, *access_id*)
Get share access rule.

share_access_get_all_by_type_and_access(*context*, *share_id*, *access_type*, *access*)
Returns share access by given type and access.

share_access_get_all_for_instance(*context*, *instance_id*, *filters=None*, *with_share_access_data=True*)
Get all access rules related to a certain share instance.

share_access_get_all_for_share(*context*, *share_id*, *filters=None*)
Get all access rules for given share.

share_access_metadata_delete(*context*, *access_id*, *key*)
Delete metadata of share access rule.

share_access_metadata_update(*context*, *access_id*, *metadata*)
Update metadata of share access rule.

share_and_snapshot_instances_status_update(*context*, *values*, *share_instance_ids=None*, *snapshot_instance_ids=None*, *current_expected_status=None*)

share_create(*context*, *share_values*, *create_share_instance=True*)
Create new share.

share_delete(*context*, *share_id*)
Delete share.

share_export_location_get_by_uuid(*context*, *export_location_uuid*, *ignore_secondary_replicas=False*)
Get specific export location of a share.

share_export_locations_get(*context*, *share_id*)
Get all export locations of a share.

share_export_locations_get_by_share_id(*context*, *share_id*, *include_admin_only=True*, *ignore_migration_destination=False*, *ignore_secondary_replicas=False*)
Get all export locations of a share by its ID.

share_export_locations_get_by_share_instance_id(*context*, *share_instance_id*, *include_admin_only=True*)
Get all export locations of a share instance by its ID.

share_export_locations_update(*context*, *share_instance_id*, *export_locations*, *delete=True*)
Update export locations of a share instance.

share_get(*context*, *share_id*)
Get share by id.

share_get_all(*context*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all shares.

share_get_all_by_project(*context*, *project_id*, *filters=None*, *is_public=False*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given project ID.

share_get_all_by_project_with_count(*context*, *project_id*, *filters=None*, *is_public=False*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given project ID.

share_get_all_by_share_group_id(*context*, *share_group_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given project ID and share group id.

share_get_all_by_share_group_id_with_count(*context*, *share_group_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given project ID and share group id.

share_get_all_by_share_server(*context*, *share_server_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given share server ID.

share_get_all_by_share_server_with_count(*context*, *share_server_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Returns all shares with given share server ID.

share_get_all_with_count(*context*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all shares.

share_group_create(*context*, *values*)
Create a share group from the values dictionary.

share_group_destroy(*context*, *share_group_id*)
Destroy the share group or raise if it does not exist.

share_group_get(*context*, *share_group_id*)
Get a share group or raise if it does not exist.

share_group_get_all(*context*, *detailed=True*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share groups.

share_group_get_all_by_host(*context*, *host*, *detailed=True*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share groups belonging to a host.

share_group_get_all_by_project(*context*, *project_id*, *detailed=True*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share groups belonging to a project.

share_group_get_all_by_share_server(*context*, *share_server_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share groups associated with a share server.

share_group_snapshot_create(*context*, *values*)
Create a share group snapshot from the values dictionary.

share_group_snapshot_destroy(*context*, *share_group_snapshot_id*)
Destroy the share_group_snapshot or raise if it does not exist.

share_group_snapshot_get(*context*, *share_group_snapshot_id*)
Get a share group snapshot.

share_group_snapshot_get_all(*context*, *detailed=True*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share group snapshots.

share_group_snapshot_get_all_by_project(*context*, *project_id*, *detailed=True*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all share group snapshots belonging to a project.

share_group_snapshot_member_create(*context*, *values*)
Create a share group snapshot member from the values dictionary.

share_group_snapshot_member_update(*context*, *member_id*, *values*)
Set the given properties on a share group snapshot member and update it.

Raises NotFound if share_group_snapshot member does not exist.

share_group_snapshot_members_get_all(*context*, *share_group_snapshot_id*)
Return the members of a share group snapshot.

share_group_snapshot_update(*context*, *share_group_snapshot_id*, *values*)
Set the given properties on a share group snapshot and update it.

Raises NotFound if share group snapshot does not exist.

share_group_type_access_add(*context*, *type_id*, *project_id*)
Add share group type access for project.

share_group_type_access_get_all(*context*, *type_id*)
Get all share group type access of a share group type.

share_group_type_access_remove(*context*, *type_id*, *project_id*)
Remove share group type access for project.

share_group_type_create(*context*, *values*, *projects=None*)
Create a new share group type.

share_group_type_destroy(*context*, *type_id*)
Delete a share group type.

share_group_type_get(*context*, *type_id*, *inactive=False*, *expected_fields=None*)
Get share_group type by id.

Parameters

- **context** context to query under
- **type_id** group type id to get.
- **inactive** Consider inactive group types when searching
- **expected_fields** Return those additional fields. Supported fields are: projects.

Returns share group type

share_group_type_get_all(*context*, *inactive=False*, *filters=None*)
Get all share group types.

Parameters

- **context** context to query under
- **inactive** Include inactive share group types to the result set
- **filters** Filters for the query in the form of key/value. :is_public: Filter share group types based on visibility:
  - **True**: List public group types only
  - **False**: List private group types only
  - **None**: List both public and private group types

Returns list of matching share group types

share_group_type_get_by_name(*context*, *name*)
Get share group type by name.

share_group_type_specs_delete(*context*, *type_id*, *key*)
Delete the given group specs item.

share_group_type_specs_get(*context*, *type_id*)
Get all group specs for a share group type.

share_group_type_specs_update_or_create(*context*, *type_id*, *group_specs*)
Create or update share group type specs.

This adds or modifies the key/value pairs specified in the group specs dict argument.

share_group_update(*context*, *share_group_id*, *values*)
Set the given properties on a share group and update it.

Raises NotFound if share group does not exist.

share_instance_access_copy(*context*, *share_id*, *instance_id*)
Maps the existing access rules for the share to the instance in the DB.

Adds the instance mapping to the shares access rules and returns the shares access rules.

share_instance_access_create(*context*, *values*, *share_instance_id*)
Allow access to share instance.

share_instance_access_delete(*context*, *mapping_id*)
Deny access to share instance.

share_instance_access_get(*context*, *access_id*, *instance_id*, *with_share_access_data=True*)
Get access rule mapping for share instance.

share_instance_access_update(*context*, *access_id*, *instance_id*, *updates*)
Update the access mapping row for a given share instance and access.

share_instance_create(*context*, *share_id*, *values*)
Create new share instance.

share_instance_delete(*context*, *instance_id*, *session=None*, *need_to_update_usages=False*)
Delete share instance.

share_instance_get(*context*, *instance_id*, *with_share_data=False*)
Get share instance by id.

share_instance_update(*context*, *instance_id*, *values*, *with_share_data=False*)
Update share instance fields.

share_instances_get_all(*context*, *filters=None*)
Returns all share instances.

share_instances_get_all_by_host(*context*, *host*, *with_share_data=False*, *status=None*)
Returns all share instances with given host.

share_instances_get_all_by_share(*context*, *share_id*)
Returns list of shares that belong to given share.

share_instances_get_all_by_share_group_id(*context*, *share_group_id*)
Returns list of share instances that belong to given share group.

share_instances_get_all_by_share_network(*context*, *share_network_id*)
Returns list of shares that belong to given share network.

share_instances_get_all_by_share_server(*context*, *share_server_id*, *with_share_data=False*)
Returns all share instances with given share_server_id.

share_instances_status_update(*context*, *share_instance_ids*, *values*)
Updates the status of a bunch of share instances at once.

share_metadata_delete(*context*, *share_id*, *key*)
Delete the given metadata item.

share_metadata_get(*context*, *share_id*)
Get all metadata for a share.

share_metadata_get_item(*context*, *share_id*, *key*)
Get metadata item for given key and for a given share..

share_metadata_update(*context*, *share*, *metadata*, *delete*)
Update metadata if it exists, otherwise create it.

share_metadata_update_item(*context*, *share_id*, *item*)
update meta item containing key and value for given share.

share_network_add_security_service(*context*, *id*, *security_service_id*)
Associate a security service with a share network.

share_network_create(*context*, *values*)
Create a share network DB record.

share_network_delete(*context*, *id*)
Delete a share network DB record.

share_network_get(*context*, *id*)
Get requested share network DB record.

share_network_get_all(*context*)
Get all share network DB records.

share_network_get_all_by_filter(*context*, *filters=None*)
Get all share network DB records for the given filter.

share_network_get_all_by_project(*context*, *project_id*)
Get all share network DB records for the given project.

share_network_get_all_by_security_service(*context*, *security_service_id*)
Get all share network DB records for the given project.

share_network_remove_security_service(*context*, *id*, *security_service_id*)
Dissociate a security service from a share network.

share_network_security_service_association_get(*context*, *share_network_id*, *security_service_id*)
Get given share network and security service association.

share_network_subnet_create(*context*, *values*)
Create a share network subnet DB record.

share_network_subnet_delete(*context*, *network_subnet_id*)
Delete a share network subnet DB record.

share_network_subnet_get(*context*, *network_subnet_id*, *session=None*)
Get requested share network subnet DB record.

share_network_subnet_get_all(*context*)
Get all share network subnet DB record.

share_network_subnet_get_by_availability_zone_id(*context*, *share_network_id*, *availability_zone_id*)
Get a share network subnet DB record.

This method returns a subnet DB record for a given share network id and an availability zone. If the availability_zone_id is None, a record may be returned and it will represent the default share network subnet. Be aware that if there is no subnet for a specific availability zone id, this method will return the default share network subnet, if it exists.

share_network_subnet_get_default_subnet(*context*, *share_network_id*)
Get the default share network subnet DB record.

share_network_subnet_update(*context*, *network_subnet_id*, *values*)
Update a share network subnet DB record.

share_network_update(*context*, *id*, *values*)
Update a share network DB record.

share_network_update_security_service(*context*, *id*, *current_security_service_id*, *new_security_service_id*)
Update a security service association with a share network.

share_replica_delete(*context*, *share_replica_id*, *need_to_update_usages=True*)
Deletes a share replica.

share_replica_get(*context*, *replica_id*, *with_share_server=False*, *with_share_data=False*)
Get share replica by id.

share_replica_update(*context*, *share_replica_id*, *values*, *with_share_data=False*)
Updates a share replica with given values.

share_replicas_get_all(*context*, *with_share_server=False*, *with_share_data=False*)
Returns all share replicas regardless of share.

share_replicas_get_all_by_share(*context*, *share_id*, *with_share_server=False*, *with_share_data=False*)
Returns all share replicas for a given share.

share_replicas_get_available_active_replica(*context*, *share_id*, *with_share_server=False*, *with_share_data=False*)
Returns an active replica for a given share.

share_resources_host_update(*context*, *current_host*, *new_host*)
Update the host attr of all share resources that are on current_host.

share_server_backend_details_delete(*context*, *share_server_id*)
Delete backend details DB records for a share server.

share_server_backend_details_set(*context*, *share_server_id*, *server_details*)
Create DB record with backend details.

share_server_create(*context*, *values*)
Create share server DB record.

share_server_delete(*context*, *id*)
Delete share server DB record.

share_server_get(*context*, *id*, *session=None*)
Get share server DB record by ID.

share_server_get_all(*context*)
Get all share server DB records.

share_server_get_all_by_host(*context*, *host*, *filters=None*)
Get all share servers related to particular host.

share_server_get_all_by_host_and_share_subnet_valid(*context*, *host*, *share_subnet_id*, *session=None*)
Get share server DB records by host and share net not error.

share_server_get_all_unused_deletable(*context*, *host*, *updated_before*)
Get all free share servers DB records.

share_server_get_all_with_filters(*context*, *filters*)
Get all share servers that match with the specified filters.

share_server_search_by_identifier(*context*, *identifier*, *session=None*)
Search for share servers based on given identifier.

share_server_update(*context*, *id*, *values*)
Update share server DB record.

share_servers_update(*context*, *share_server_ids*, *values*)
Updates values of a bunch of share servers at once.

share_snapshot_access_create(*context*, *values*)
Create a share snapshot access from the values dictionary.

share_snapshot_access_get(*context*, *access_id*)
Get share snapshot access rule from given access_id.

share_snapshot_access_get_all_for_share_snapshot(*context*, *share_snapshot_id*, *filters*)
Get all access rules for a given share snapshot according to filters.

share_snapshot_access_get_all_for_snapshot_instance(*context*, *snapshot_instance_id*, *session=None*)
Get all access rules related to a certain snapshot instance.

share_snapshot_check_for_existing_access(*context*, *share_snapshot_id*, *access_type*, *access_to*)
Returns True if rule corresponding to the type and client exists.

share_snapshot_create(*context*, *values*)
Create a snapshot from the values dictionary.

share_snapshot_export_locations_get(*context*, *snapshot_id*)
Get all export locations for a given share snapshot.

share_snapshot_get(*context*, *snapshot_id*)
Get a snapshot or raise if it does not exist.

share_snapshot_get_all(*context*, *filters=None*, *limit=None*, *offset=None*, *sort_key=None*, *sort_dir=None*)
Get all snapshots.

share_snapshot_get_all_by_project(*context*, *project_id*, *filters=None*, *limit=None*, *offset=None*, *sort_key=None*, *sort_dir=None*)
Get all snapshots belonging to a project.

share_snapshot_get_all_for_share(*context*, *share_id*, *filters=None*, *sort_key=None*, *sort_dir=None*)
Get all snapshots for a share.

share_snapshot_get_latest_for_share(*context*, *share_id*)
Get the most recent snapshot for a share.

share_snapshot_instance_access_delete(*context*, *access_id*, *snapshot_instance_id*)
Delete share snapshot instance access given its id.

share_snapshot_instance_access_get(*context*, *share_snapshot_instance_id*, *access_id*)
Get the share snapshot instance access related to given ids.

share_snapshot_instance_access_update(*context*, *access_id*, *instance_id*, *updates*)
Update the state of the share snapshot instance access.

share_snapshot_instance_create(*context*, *snapshot_id*, *values*)
Create a share snapshot instance for an existing snapshot.

share_snapshot_instance_delete(*context*, *snapshot_instance_id*)
Delete a share snapshot instance.

share_snapshot_instance_export_location_create(*context*, *values*)
Create a share snapshot instance export location.

share_snapshot_instance_export_location_delete(*context*, *el_id*)
Delete share snapshot instance export location given its id.

share_snapshot_instance_export_location_get(*context*, *el_id*)
Get the share snapshot instance export location for given id.

share_snapshot_instance_export_locations_get_all(*context*, *share_snapshot_instance_id*)
Get the share snapshot instance export locations for given id.

share_snapshot_instance_export_locations_update(*context*, *share_snapshot_instance_id*, *export_locations*, *delete=True*)
Update export locations of a share instance.

share_snapshot_instance_get(*context*, *instance_id*, *with_share_data=False*)
Get a snapshot instance or raise a NotFound exception.

share_snapshot_instance_get_all_with_filters(*context*, *filters*, *with_share_data=False*)
Get all snapshot instances satisfying provided filters.

share_snapshot_instance_update(*context*, *instance_id*, *values*)
Set the given properties on a share snapshot instance and update it.

Raises NotFound if snapshot instance does not exist.

share_snapshot_instances_status_update(*context*, *snapshot_instance_ids*, *values*)
Updates the status of a bunch of share snapshot instances at once.

share_snapshot_update(*context*, *snapshot_id*, *values*)
Set the given properties on an snapshot and update it.

Raises NotFound if snapshot does not exist.

share_type_access_add(*context*, *type_id*, *project_id*)
Add share type access for project.

share_type_access_get_all(*context*, *type_id*)
Get all share type access of a share type.

share_type_access_remove(*context*, *type_id*, *project_id*)
Remove share type access for project.

share_type_create(*context*, *values*, *projects=None*)
Create a new share type.

share_type_destroy(*context*, *id*)
Delete a share type.

share_type_extra_specs_delete(*context*, *share_type_id*, *key*)
Delete the given extra specs item.

share_type_extra_specs_get(*context*, *share_type_id*)
Get all extra specs for a share type.

share_type_extra_specs_update_or_create(*context*, *share_type_id*, *extra_specs*)
Create or update share type extra specs.

This adds or modifies the key/value pairs specified in the extra specs dict argument.

share_type_get(*context*, *type_id*, *inactive=False*, *expected_fields=None*)
Get share type by id.

Parameters

- **context** context to query under
- **type_id** share type id to get.
- **inactive** Consider inactive share types when searching
- **expected_fields** Return those additional fields. Supported fields are: projects.

Returns share type

share_type_get_all(*context*, *inactive=False*, *filters=None*)
Get all share types.

Parameters

- **context** context to query under
- **inactive** Include inactive share types to the result set
- **filters** Filters for the query in the form of key/value. :is_public: Filter share types based on visibility:
  - **True**: List public share types only
  - **False**: List private share types only
  - **None**: List both public and private share types

Returns list of matching share types

share_type_get_by_name(*context*, *name*)
Get share type by name.

share_type_get_by_name_or_id(*context*, *name_or_id*)
Get share type by name or ID and return None if not found.

share_type_update(*context*, *share_type_id*, *values*)
Update an exist share type.

share_update(*context*, *share_id*, *values*)
Update share fields.

The Sqlalchemy Driver

The `manila.db.sqlalchemy.api` Module

Implementation of SQLAlchemy backend.

The manila.db.sqlalchemy.models Module

SQLAlchemy models for Manila data.

class AsynchronousOperationData(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents data as key-value pairs for asynchronous operations.

created_at

deleted

deleted_at

entity_uuid

key

updated_at

value

class AvailabilityZone(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a private data as key-value pairs for a driver.

created_at

deleted

deleted_at

id

name

updated_at

class BackendInfo(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represent Backend Info.

created_at

deleted

deleted_at

host

info_hash

updated_at

class DriverPrivateData(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a private data as key-value pairs for a driver.

created_at

deleted

deleted_at

entity_uuid

key

updated_at

value

class ManilaBase

Bases: oslo_db.sqlalchemy.models.ModelBase, oslo_db.sqlalchemy.models.TimestampMixin, oslo_db.sqlalchemy.models.SoftDeleteMixin

Base class for Manila Models.

metadata = None

soft_delete(*session*, *update_status=False*, *status_field_name='status'*)

Mark this object as deleted.

to_dict()

class Message(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a user message.

User messages show information about API operations to the API end-user.

action_id

created_at

deleted

deleted_at

detail_id

expires_at

id

message_level

project_id

request_id

resource_id

resource_type

updated_at

class NetworkAllocation(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents network allocation data.

cidr

created_at

deleted

deleted_at

gateway

id

ip_address

ip_version

label

mac_address

mtu

network_type

segmentation_id

share_server_id

updated_at

class ProjectShareTypeQuota(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a single quota override for a share type within a project.

created_at

deleted

deleted_at

hard_limit

id

project_id

resource

share_type_id

updated_at

class ProjectUserQuota(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a single quota override for a user with in a project.

created_at

deleted

deleted_at

hard_limit

id

project_id

resource

updated_at

user_id

class Quota(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a single quota override for a project.

If there is no row for a given project id and resource, then the default for the quota class is used. If there is no row for a given quota class and resource, then the default for the deployment is used. If the row is present but the hard limit is Null, then the resource is unlimited.

created_at

deleted

deleted_at

hard_limit

id

project_id

resource

updated_at

class QuotaClass(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a single quota override for a quota class.

If there is no row for a given quota class and resource, then the default for the deployment is used. If the row is present but the hard limit is Null, then the resource is unlimited.

class_name

created_at

deleted

deleted_at

hard_limit

id

resource

updated_at

class QuotaUsage(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents the current usage for a given resource.

created_at

deleted

deleted_at

id

in_use

project_id

reserved

resource

share_type_id

property total

until_refresh

updated_at

user_id

class Reservation(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a resource reservation for quotas.

created_at

deleted

deleted_at

delta

expire

id

project_id

resource

share_type_id

updated_at

usage_id

user_id

uuid

class SecurityService(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Security service information for manila shares.

created_at

deleted

deleted_at

description

dns_ip

domain

id

name

ou

password

project_id

server

type

updated_at

user

class Service(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a running service on a host.

availability_zone

availability_zone_id

binary

created_at

deleted

deleted_at

disabled

host

id

report_count

topic

updated_at

class Share(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents an NFS and CIFS shares.

property access_rules_status

create_share_from_snapshot_support

created_at

deleted

deleted_at

display_description

display_name

property export_location

property export_locations

property has_replicas

id

property instance

instances

property is_busy

is_public

mount_snapshot_support

property name

property progress

project_id

replication_type

revert_to_snapshot_support

share_group_id

share_proto

property share_server_id

size

snapshot_id

snapshot_support

source_share_group_snapshot_member_id

task_state

updated_at

user_id

class ShareAccessMapping(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents access to share.

access_key

access_level

access_to

access_type

created_at

deleted

deleted_at

id

instance_mappings

share_id

property state
Get the aggregated state from all the instance mapping states.

An access rule is supposed to be truly active when it has been applied across all of the share instances of the parent share object.

updated_at

class ShareAccessRulesMetadata(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a metadata key/value pair for a share access rule.

access

access_id

created_at

deleted

deleted_at

id

key

updated_at

value

class ShareGroup(**kwargs)
Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a share group.

property availability_zone

availability_zone_id

consistent_snapshot_support

created_at

deleted

deleted_at

description

host

id

name

project_id

share_group_type

share_group_type_id

share_network_id

share_server_id

source_share_group_snapshot_id

status

updated_at

user_id

class ShareGroupShareTypeMapping(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents the share types in a share group.

created_at

deleted

deleted_at

id

share_group

share_group_id

share_type_id

updated_at

class ShareGroupSnapshot(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a share group snapshot.

created_at

deleted

deleted_at

description

id

name

project_id

share_group

share_group_id

status

updated_at

user_id

class ShareGroupTypeProjects(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represent projects associated share group types.

created_at

deleted

deleted_at

id

project_id

share_group_type

share_group_type_id

updated_at

class ShareGroupTypeShareTypeMapping(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents the share types supported by a share group type.

created_at

deleted

deleted_at

id

share_group_type

share_group_type_id

share_type_id

updated_at

class ShareGroupTypeSpecs(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents additional specs for a share group type.

created_at

deleted

deleted_at

id

key

share_group_type

share_group_type_id

updated_at

value

class ShareGroupTypes(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represent possible share group types of shares offered.

created_at

deleted

deleted_at

id

is_public

name

updated_at

class ShareInstance(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

ACCESS_STATUS_PRIORITIES = {'active': 0, 'error': 2, 'syncing': 1}

access_rules_status

property availability_zone

availability_zone_id

cast_rules_to_readonly

created_at

deleted

deleted_at

property export_location

export_locations

host

id

launched_at

property name

progress

replica_state

scheduled_at

set_share_data(*share*)

share_id

share_network_id

share_server_id

share_type

share_type_id

status

terminated_at

updated_at

class ShareInstanceAccessMapping(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents access to individual share instances.

access_id

created_at

deleted

deleted_at

id

instance

set_share_access_data(*share_access*)

share_instance_id

state

updated_at

class ShareInstanceExportLocations(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents export locations of share instances.

created_at

deleted

deleted_at

property el_metadata

id

is_admin_only

path

property replica_state

share_instance_id

updated_at

uuid

class ShareInstanceExportLocationsMetadata(***kwargs*)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents export location metadata of share instances.

created_at

deleted

deleted_at

export_location

export_location_id

property export_location_uuid

id

key

updated_at

value

class ShareMetadata(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a metadata key/value pair for a share.

created_at

deleted

deleted_at

id

key

share

share_id

updated_at

value

class ShareNetwork(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents network data used by share.

created_at

deleted

deleted_at

description

id

name

project_id

property security_service_update_support

security_services

share_instances

share_network_subnets

status

updated_at

user_id

class ShareNetworkSecurityServiceAssociation(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Association table between compute_zones and compute_nodes tables.

created_at

deleted

deleted_at

id

security_service_id

share_network_id

updated_at

class ShareNetworkSubnet(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a share network subnet used by some resources.

property availability_zone

availability_zone_id

cidr

created_at

deleted

deleted_at

gateway

id

ip_version

property is_default

mtu

network_type

neutron_net_id

neutron_subnet_id

segmentation_id

share_network_id

property share_network_name

share_servers

updated_at

class ShareServer(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents share server used by share.

property backend_details

created_at

deleted

deleted_at

host

id

identifier

is_auto_deletable

network_allocations

security_service_update_support

share_groups

share_instances

share_network_subnet_id

source_share_server_id

status

task_state

updated_at

class ShareServerBackendDetails(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a metadata key/value pair for a share server.

created_at

deleted

deleted_at

id

key

share_server_id

updated_at

value

class ShareSnapshot(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a snapshot of a share.

property aggregate_status

Get the aggregated status of all instances.

A snapshot is supposed to be truly available when it is available across all of the share instances of the parent share object. In case of replication, we only consider replicas (share instances) that are in in_sync replica_state.

created_at

deleted

deleted_at

display_description

display_name

property export_locations

id

property instance

property name

project_id

share

share_id

property share_name

share_proto

share_size

size

updated_at

user_id

class ShareSnapshotAccessMapping(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents access to share snapshot.

access_to

access_type

created_at

deleted

deleted_at

id

instance_mappings

share_snapshot_id

property state

Get the aggregated state from all the instance mapping states.

An access rule is supposed to be truly active when it has been applied across all of the share snapshot instances of the parent share snapshot object.

updated_at

class ShareSnapshotInstance(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents a snapshot of a share.

created_at

deleted

deleted_at

export_locations

id

instance_size

property name

progress

project_id

provider_location

share_group_snapshot

share_group_snapshot_id

property share_id

share_instance

share_instance_id

property share_name

share_proto

property size

snapshot

snapshot_id

status

updated_at

user_id

class ShareSnapshotInstanceAccessMapping(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents access to individual share snapshot instances.

access_id

created_at

deleted

deleted_at

id

instance

set_snapshot_access_data(*snapshot_access*)

share_snapshot_instance_id

state

updated_at

class ShareSnapshotInstanceExportLocation(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents export locations of share snapshot instances.

created_at

deleted

deleted_at

id

is_admin_only

path

share_snapshot_instance_id

updated_at

class ShareTypeExtraSpecs(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represents additional specs as key/value pairs for a share_type.

created_at

deleted

deleted_at

id

key

share_type

share_type_id

updated_at

value

class ShareTypeProjects(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represent projects associated share_types.

created_at

deleted

deleted_at

id

project_id

share_type

share_type_id

updated_at

class ShareTypes(**kwargs)

Bases: sqlalchemy.orm.decl_api.Base, manila.db.sqlalchemy.models.ManilaBase

Represent possible share_types of volumes offered.

created_at

deleted

deleted_at

description

id

is_public

name

updated_at

get_access_rules_status(*instances*)

get_aggregated_access_rules_state(*instance_mappings*)

register_models()

Register Models and create metadata.

Called from manila.db.sqlalchemy.__init__ as part of loading the driver, it will never need to be called explicitly elsewhere unless the connection is lost and needs to be reestablished.

Tests

Tests are lacking for the db api layer and for the sqlalchemy driver. Failures in the drivers would be detected in other test cases, though.

DB migration revisions

If a DB schema needs to be updated, a new DB migration file needs to be added in `manila/db/migrations/alembic/versions`. To create such a file its possible to use `manila-manage db revision` or the corresponding tox command:

```
tox -e dbrevision "change_foo_table"
```

In addition every migration script must be tested. See examples in `manila/tests/db/migrations/alembic/migrations_data_checks.py`.

Shared Filesystems

The `manila.share.manager` Module

NAS share manager managers creating shares and access rights.

Related Flags

share_driver Used by `ShareManager`.

class ShareManager(*share_driver=None*, *service_name=None*, **args*, ***kwargs*)

Bases: `manila.manager.SchedulerDependentManager`

Manages NAS storages.

RPC_API_VERSION = '1.22'

check_update_share_network_security_service(*context*, *share_network_id*, *new_security_service_id*, *current_security_service_id=None*)

connection_get_info(*context*, *share_instance_id*)

create_replicated_snapshot(***kwargs*)

create_share_group(*context*, *share_group_id*)

create_share_group_snapshot(*context*, *share_group_snapshot_id*)

create_share_instance(*context*, *share_instance_id*, *request_spec=None*, *filter_properties=None*, *snapshot_id=None*)

Creates a share instance.

create_share_replica(***kwargs*)

create_share_server(*context*, *share_server_id*, *share_instance_id*)

Invoked to create a share server in this backend.

This method is invoked to create the share server defined in the model obtained by the supplied id.

Parameters

- **context** The context.RequestContext object for the request.
- **share_server_id** The id of the server to be created.
- **share_instance_id** The id of the share instance

create_snapshot(*context*, *share_id*, *snapshot_id*)

Create snapshot for share.

delete_free_share_servers(*ctxt*)

delete_replicated_snapshot(***kwargs*)

delete_share_group(*context*, *share_group_id*)

delete_share_group_snapshot(*context*, *share_group_snapshot_id*)

delete_share_instance(*context*, *share_instance_id*, *force=False*)

Delete a share instance.

delete_share_replica(***kwargs*)

delete_share_server(*context*, *share_server*)

delete_snapshot(*context*, *snapshot_id*, *force=False*)

Delete share snapshot.

ensure_driver_resources(*ctxt*)

extend_share(*context*, *share_id*, *new_size*, *reservations*)

init_host()

Initialization for a standalone service.

is_service_ready()

Return if Manager is ready to accept requests.

This is to inform Service class that in case of manila driver initialization failure the manager is actually down and not ready to accept any requests.

manage_share(*context*, *share_id*, *driver_options*)

manage_share_server(*context*, *share_server_id*, *identifier*, *driver_opts*)

manage_snapshot(*context*, *snapshot_id*, *driver_options*)

migration_cancel(*context*, *src_instance_id*, *dest_instance_id*)

migration_complete(*context*, *src_instance_id*, *dest_instance_id*)

migration_driver_continue(*context*)
Invokes driver to continue migration of shares.

migration_get_progress(*context*, *src_instance_id*, *dest_instance_id*)

migration_start(*context*, *share_id*, *dest_host*, *force_host_assisted_migration*, *preserve_metadata*, *writable*, *nondisruptive*, *preserve_snapshots*, *new_share_network_id=None*, *new_share_type_id=None*)
Migrates a share from current host to another host.

periodic_share_replica_snapshot_update(*context*)

periodic_share_replica_update(*context*)

periodic_share_status_update(*context*)
Invokes share driver to update shares status.

promote_share_replica(***kwargs*)

provide_share_server(*context*, *share_instance_id*, *share_network_id*, *snapshot_id=None*)
Invoked to provide a compatible share server.

This method is invoked to find a compatible share server among the existing ones or create a share server database instance with the share server properties that will be used to create the share server later.

Parameters

- **context** The context.RequestContext object for the request.
- **share_instance_id** The id of the share instance whose model attributes will be used to provide the share server.
- **share_network_id** The id of the share network the share server to be provided has to be related to.
- **snapshot_id** The id of the snapshot to be used to obtain the share server if applicable.

Returns The id of the share server that is being provided.

publish_service_capabilities(*context*)
Collect driver status and then publish it.

revert_to_snapshot(*context*, *snapshot_id*, *reservations*)

share_server_migration_cancel(*context*, *src_share_server_id*, *dest_share_server_id*)

share_server_migration_check(*context*, *share_server_id*, *dest_host*, *writable*, *nondisruptive*, *preserve_snapshots*, *new_share_network_id*)

share_server_migration_complete(*context*, *src_share_server_id*, *dest_share_server_id*)
Invokes driver to complete the migration of share server.

share_server_migration_driver_continue(*context*)
Invokes driver to continue migration of share server.

share_server_migration_get_progress(*context*, *src_share_server_id*, *dest_share_server_id*)

share_server_migration_start(*context*, *share_server_id*, *dest_host*, *writable*, *nondisruptive*, *preserve_snapshots*, *new_share_network_id=None*)
Migrates a share server from current host to another host.

shrink_share(*context*, *share_id*, *new_size*)

snapshot_update_access(*context*, *snapshot_instance_id*)

unmanage_share(*context*, *share_id*)

unmanage_share_server(*context*, *share_server_id*, *force=False*)

unmanage_snapshot(*context*, *snapshot_id*)

update_access(*context*, *share_instance_id*)
Allow/Deny access to some share.

update_access_for_instances(*context*, *share_instance_ids*, *share_server_id=None*)
Allow/Deny access to shares that belong to the same share server.

update_share_network_security_service(*context*, *share_network_id*, *new_security_service_id*, *current_security_service_id=None*)

update_share_replica(*context*, *share_replica_id*, *share_id=None*)
Initiated by the force_update API.

update_share_usage_size(*context*)
Invokes driver to gather usage size of shares.

add_hooks(*f*)
Hook decorator to perform action before and after a share method call

The hook decorator can perform actions before some share driver methods calls and after a call with results of driver call and preceding hook call.

locked_share_network_operation(*operation*)
Lock decorator for share network operations.

Takes a named lock prior to executing the operation. The lock is named with the id of the share network.

locked_share_replica_operation(*operation*)
Lock decorator for share replica operations.

Takes a named lock prior to executing the operation. The lock is named with the id of the share to which the replica belongs.

Intended use: If a replica operation uses this decorator, it will block actions on all share replicas of the share until the named lock is free. This is used to protect concurrent operations on replicas of the same share e.g. promote ReplicaA while deleting ReplicaB, both belonging to the same share.

The manila.share.driver Module

Drivers for shares.

class ExecuteMixin
Bases: object

Provides an executable functionality to a driver class.

init_execute_mixin(**args*, ***kwargs*)

set_execute(*execute*)

class GaneshaMixin
Bases: object

Augment derived classes with Ganesha configuration.

init_ganesha_mixin(**args*, ***kwargs*)

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)
Bases: object

Class defines interface of NAS driver.

add_ip_version_capability(*data*)
Add IP version support capabilities.

When DHSS is true, the capabilities are determined by driver and configured network plugin. When DHSS is false, the capabilities are determined by driver only. :param data: the capability dictionary :returns: capability data

property admin_network_api

allocate_admin_network(*context*, *share_server*, *count=None*, ***kwargs*)
Allocate admin network resources using given network information.

allocate_network(*context*, *share_server*, *share_network*, *share_network_subnet*, *count=None*, ***kwargs*)
Allocate network resources using given network information.

allow_access(*context*, *share*, *access*, *share_server=None*)
Allow access to the share.

check_for_setup_error()
Check for setup error.

check_update_share_server_security_service(*context*, *share_server*, *network_info*, *share_instances*, *share_instance_rules*, *new_security_service*, *current_security_service=None*)
Check if the current share server security service is supported.

If the driver supports different security services, the user can request the addition of a new security service, with a different type. If the user wants to update the current security ser-

vice configuration, the driver will receive both current and new security services, which will always be of the same type.

Parameters

- **context** The context.RequestContext object for the request.
- **share_server** Reference to the share server object that will be updated.
- **network_info** All network allocation associated with the share server that will be updated.
- **share_instances** A list of share instances that belong to the share server that is affected by the update.
- **share_instance_rules** A list of access rules, grouped by share instance, in the following format.

Example:

```
[
   {
    'share_instance_id': '3bc10d67-2598-4122-bb62-0bdeaa8c6db3',
    'access_rules':
    [
        {
        'access_id':'906d0094-3e34-4d6c-a184-d08a908033e3',
        'access_type':'ip',
        'access_key':None,
        'access_to':'10.0.0.1',
        'access_level':'rw'
         ...
        },
    ],
    },
]
```

Parameters

- **new_security_service** New security service object to be configured in the share server.
- **current_security_service** When provided, represents the current security service that will be replaced by the new_security_service.

Returns True if the driver support the requested update, False otherwise.

choose_share_server_compatible_with_share(*context*, *share_servers*, *share*, *snapshot=None*, *share_group=None*)

Method that allows driver to choose share server for provided share.

If compatible share-server is not found, method should return None.

Parameters

- **context** Current context
- **share_servers** list with share-server models

- **share** share model
- **snapshot** snapshot model
- **share_group** ShareGroup model with shares

Returns share-server or None

choose_share_server_compatible_with_share_group(*context*, *share_servers*, *share_group_ref*, *share_group_snapshot=None*)

connection_get_info(*context*, *share*, *share_server=None*)

Is called to provide necessary generic migration logic.

Parameters

- **context** The context.RequestContext object for the request.
- **share** Reference to the share being migrated.
- **share_server** Share server model or None.

Returns A dictionary with migration information.

create_replica(*context*, *replica_list*, *new_replica*, *access_rules*, *replica_snapshots*, *share_server=None*)

Replicate the active replica to a new replica on this backend.

Note: This call is made on the host that the new replica is being created upon.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share. This list also contains the replica to be created. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
```

(continues on next page)

(continued from previous page)

```
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **new_replica** The share replica dictionary.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'creating',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'out_of_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': 'e6155221-ea00-49ef-abf9-9f89b7dd900a',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules. These are rules that other instances of the share already obey. Drivers are expected to apply access rules to the new replica or disregard access rules that dont apply.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
```

(continues on next page)

(continued from previous page)

```
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica that will need to exist on the new share replica that is being created. The driver needs to ensure that this snapshot instance is truly available before transitioning the replica from out_of_sync to in_sync. Snapshots instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the `update_replicated_snapshot` method.

Example:

```
[
 {
 'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
    },
 'share_replica_snapshot': {
    'id': '',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'status': 'available',
    'provider_location': None,
        ...
    },
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica being created.

Returns None or a dictionary. The dictionary can contain export_locations replica_state and access_rules_status. export_locations is a list of paths and replica_state is one of active, in_sync, out_of_sync or error.

Important: A backend supporting writable type replication should return active as the replica_state.

Export locations should be in the same format as returned during the `create_share` call.

Example:

```
{
    'export_locations': [
        {
            'path': '172.16.20.22/sample/export/path',
            'is_admin_only': False,
            'metadata': {'some_key': 'some_value'},
        },
    ],
    'replica_state': 'in_sync',
    'access_rules_status': 'in_sync',
}
```

create_replicated_snapshot(*context*, *replica_list*, *replica_snapshots*, *share_server=None*)

Create a snapshot on active instance and update across the replicas.

Note: This call is made on the active replicas host. Drivers are expected to transfer the snapshot created to the respective replicas.

The driver is expected to return model updates to the share manager. If it was able to confirm the creation of any number of the snapshot instances passed in this interface, it can set their status to available as a cue for the share manager to set the progress attr to 100%.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
```

(continues on next page)

(continued from previous page)

```
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to creating.

Example:

```
[
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being created.

Raises Exception. Any exception in this method will set all instances to error.

create_share(*context*, *share*, *share_server=None*)

Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Is called to create share from snapshot.

Creating a share from snapshot can take longer than a simple clone operation if data copy is required from one host to another. For this reason driver will be able complete this creation asynchronously, by providing a creating_from_snapshot status in the model update.

When answering asynchronously, drivers must implement the call get_share_status in order to provide updates for shares with creating_from_snapshot status.

It is expected that the driver returns a model update to the share manager that contains: share status and a list of export_locations. A list of export_locations is mandatory only for share in available status. The current supported status are available and creating_from_snapshot.

Parameters

- **context** Current context
- **share** Share instance model with share data.
- **snapshot** Snapshot instance model .
- **share_server** Share server model or None.
- **parent_share** Share model from parent snapshot with share data and share server model.

Returns

a dictionary of updates containing current share status and its export_location (if available).

Example:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance to error and the operation will end.

create_share_group(*context*, *share_group_dict*, *share_server=None*)

Create a share group.

Parameters

- **context**
- **share_group_dict** The share group details EXAMPLE: { status: creating, project_id: 13c0be6290934bd98596cfa004650049, user_id: a0314a441ca842019b0952224aa39192, description: None, deleted: False, created_at: datetime.datetime(2015, 8, 10, 15, 14, 6), updated_at: None, source_share_group_snapshot_id: some_fake_uuid, share_group_type_id: some_fake_uuid, host: hostname@backend_name, share_network_id: None, share_server_id: None, deleted_at: None, share_types: [<models.ShareGroupShareTypeMapping>], id: some_fake_uuid, name: None }

Returns (share_group_model_update, share_update_list) share_group_model_update - a dict containing any values to be updated for the SG in the database. This value may be None.

create_share_group_from_share_group_snapshot(*context*, *share_group_dict*, *share_group_snapshot_dict*, *share_server=None*)

Create a share group from a share group snapshot.

When creating a share from snapshot operation takes longer than a simple clone operation, drivers will be able to complete this creation asynchronously, by providing a creating_from_snapshot status in the returned model update. The current supported status are available and creating_from_snapshot.

In order to provide updates for shares with creating_from_snapshot status, drivers must implement the call get_share_status.

Parameters

- **context**
- **share_group_dict** The share group details EXAMPLE: .. code:

```
{
'status': 'creating',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': 'False',
'created_at': datetime.datetime(2015, 8, 10, 15, 14, 6),
'updated_at': None,
'source_share_group_snapshot_id':
    'f6aa3b59-57eb-421e-965c-4e182538e36a',
'host': 'hostname@backend_name',
'deleted_at': None,
'shares': [<models.Share>], # The new shares being␣
↪created
'share_types': [<models.ShareGroupShareTypeMapping>],
'id': 'some_fake_uuid',
'name': None
}
```

- **share_group_snapshot_dict** The share group snapshot details EXAMPLE: .. code:

```
{
'status': 'available',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': '0',
'created_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'updated_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'share_group_id': 'some_fake_uuid',
'share_share_group_snapshot_members': [
    {
     'status': 'available',
     'user_id': 'a0314a441ca842019b0952224aa39192',
     'deleted': 'False',
     'created_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share': <models.Share>,
     'updated_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share_proto': 'NFS',
     'project_id': '13c0be6290934bd98596cfa004650049',
```

(continues on next page)

(continued from previous page)

```
    'share_group_snapshot_id': 'some_fake_uuid',
    'deleted_at': None,
    'id': 'some_fake_uuid',
    'size': 1
   }
],
'deleted_at': None,
'id': 'f6aa3b59-57eb-421e-965c-4e182538e36a',
'name': None
}
```

Returns

(share_group_model_update, share_update_list) share_group_model_update - a dict containing any values to be updated for the share group in the database. This value may be None

share_update_list - a list of dictionaries containing dicts for every share created in the share group. Any share dicts should at a minimum contain the id key and, for synchronous creation, the export_locations. For asynchronous share creation this dict must also contain the key status with the value set to creating_from_snapshot. The current supported status are available and creating_from_snapshot. Export locations should be in the same format as returned by a share_create. This list may be empty or None. EXAMPLE: .. code:

```
[
    {
    'id': 'uuid',
    'export_locations': [{...}, {...}],
    },
    {
    'id': 'uuid',
    'export_locations': [],
    'status': 'creating_from_snapshot',
    },
]
```

create_share_group_snapshot(*context*, *snap_dict*, *share_server=None*)

Create a share group snapshot.

Parameters

- **context**
- **snap_dict** The share group snapshot details EXAMPLE: .. code:

```
{
'status': 'available',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': '0',
```

(continues on next page)

(continued from previous page)

```
'created_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'updated_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'share_group_id': 'some_fake_uuid',
'share_group_snapshot_members': [
    {
     'status': 'available',
     'share_type_id': 'some_fake_uuid',
     'user_id': 'a0314a441ca842019b0952224aa39192',
     'deleted': 'False',
     'created_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share': <models.Share>,
     'updated_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share_proto': 'NFS',
     'share_name': 'share_some_fake_uuid',
     'name': 'share-snapshot-some_fake_uuid',
     'project_id': '13c0be6290934bd98596cfa004650049',
     'share_group_snapshot_id': 'some_fake_uuid',
     'deleted_at': None,
     'share_id': 'some_fake_uuid',
     'id': 'some_fake_uuid',
     'size': 1,
     'provider_location': None,
    }
],
'deleted_at': None,
'id': 'some_fake_uuid',
'name': None
}
```

Returns

(share_group_snapshot_update, member_update_list) share_group_snapshot_update - a dict containing any values to be updated for the CGSnapshot in the database. This value may be None.

member_update_list - a list of dictionaries containing for every member of the share group snapshot. Each dict should contains values to be updated for the ShareGroupSnapshotMember in the database. This list may be empty or None.

create_snapshot(*context*, *snapshot*, *share_server=None*)

Is called to create snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

Returns None or a dictionary with key export_locations containing a list of export

locations, if snapshots can be mounted.

property creating_shares_from_snapshots_is_supported
Calculate default value for create_share_from_snapshot_support.

deallocate_network(*context*, *share_server_id*)
Deallocate network resources for the given share server.

delete_replica(*context*, *replica_list*, *replica_snapshots*, *replica*, *share_server=None*)
Delete a replica.

Note: This call is made on the host that hosts the replica being deleted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be deleted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the share replica being deleted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '53099868-65f1-11e5-9d70-feff819cdc9f',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. The dict contains snapshot instances that are associated with the share replica being deleted. No model updates to snapshot instances are possible in this method. The driver should return when the cleanup is completed on the backend for both, the snapshots and the replica itself. Drivers must handle situations where the snapshot may not yet have finished creating on this replica.

Example:

```
[
    {
    'id': '89dafd00-0999-4d23-8614-13eaa6b02a3b',
    'snapshot_id': '3ce1caf7-0945-45fd-a320-714973e949d3',
    'status: 'available',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be deleted.

Returns None.

Raises Exception. Any exception raised will set the share replicas status and replica_state attributes to error_deleting. It will not affect snapshots belonging to this replica.

delete_replicated_snapshot(*context*, *replica_list*, *replica_snapshots*, *share_server=None*)

Delete a snapshot by deleting its instances across the replicas.

Note: This call is made on the active replicas host, since drivers may not be able to delete the snapshot from an individual replica.

The driver is expected to return model updates to the share manager. If it was able to confirm the removal of any number of the snapshot instances passed in this interface, it can set their status to deleted as a cue for the share manager to clean up that instance from the database.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to deleting.

Example:

```
[
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status': 'deleting',
    'progress': '100%',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'deleting',
    'progress': '100%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being deleted. To confirm the deletion of the snapshot instance, set the status attribute of the instance to deleted (constants.STATUS_DELETED)

Raises Exception. Any exception in this method will set the status attribute of all snapshot instances to error_deleting.

delete_share(*context*, *share*, *share_server=None*)
Is called to remove share.

delete_share_group(*context*, *share_group_dict*, *share_server=None*)
Delete a share group

Parameters

- **context** The request context
- **share_group_dict** The share group details EXAMPLE: .. code:

```
{
'status': 'creating',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': 'False',
'created_at': datetime.datetime(2015, 8, 10, 15, 14, 6),
'updated_at': None,
'source_share_group_snapshot_id': 'some_fake_uuid',
'share_share_group_type_id': 'some_fake_uuid',
'host': 'hostname@backend_name',
'deleted_at': None,
'shares': [<models.Share>], # The new shares being␣
↪created
```

(continues on next page)

(continued from previous page)

```
'share_types': [<models.ShareGroupShareTypeMapping>],
'id': 'some_fake_uuid',
'name': None
}
```

Returns share_group_model_update share_group_model_update - a dict containing any values to be updated for the group in the database. This value may be None.

delete_share_group_snapshot(*context*, *snap_dict*, *share_server=None*)

Delete a share group snapshot

Parameters

- **context**
- **snap_dict** The share group snapshot details EXAMPLE: .. code:

```
{
'status': 'available',
'project_id': '13c0be6290934bd98596cfa004650049',
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': '0',
'created_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'updated_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
'share_group_id': 'some_fake_uuid',
'share_group_snapshot_members': [
    {
     'status': 'available',
     'share_type_id': 'some_fake_uuid',
     'share_id': 'some_fake_uuid',
     'user_id': 'a0314a441ca842019b0952224aa39192',
     'deleted': 'False',
     'created_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share': <models.Share>,
     'updated_at': datetime.datetime(2015, 8, 10, 0, 5,␣
↪58),
     'share_proto': 'NFS',
     'share_name':'share_some_fake_uuid',
     'name': 'share-snapshot-some_fake_uuid',
     'project_id': '13c0be6290934bd98596cfa004650049',
     'share_group_snapshot_id': 'some_fake_uuid',
     'deleted_at': None,
     'id': 'some_fake_uuid',
     'size': 1,
     'provider_location': 'fake_provider_location_value',
    }
],
'deleted_at': None,
```

(continues on next page)

(continued from previous page)

```
'id': 'f6aa3b59-57eb-421e-965c-4e182538e36a',
'name': None
}
```

Returns (share_group_snapshot_update, member_update_list) share_group_snapshot_update - a dict containing any values to be updated for the ShareGroupSnapshot in the database. This value may be None.

delete_snapshot(*context*, *snapshot*, *share_server=None*)

Is called to remove snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

deny_access(*context*, *share*, *access*, *share_server=None*)

Deny access to the share.

do_setup(*context*)

Any initialization the share driver does while starting.

property driver_handles_share_servers

ensure_share(*context*, *share*, *share_server=None*)

Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

ensure_shares(*context*, *shares*)

Invoked to ensure that shares are exported.

Driver can use this method to update the list of export locations of the shares if it changes. To do that, a dictionary of shares should be returned. :shares: A list of all shares for updates. :returns: None or a dictionary of updates in the format.

Example:

```
{
    '09960614-8574-4e03-89cf-7cf267b0bd08': {
        'export_locations': [{...}, {...}],
        'status': 'error',
    },

    '28f6eabb-4342-486a-a7f4-45688f0c0295': {
        'export_locations': [{...}, {...}],
        'status': 'available',
    },
```

(continues on next page)

(continued from previous page)

```
}
```

extend_share(*share*, *new_size*, *share_server=None*)
Extends size of existing share.

Parameters

- **share** Share model
- **new_size** New size of share (new_size > share[size])
- **share_server** Optional Share server model

get_admin_network_allocations_number()

get_backend_info(*context*)
Get driver and array configuration parameters.

Driver can use this method to get the special configuration info and return for assessment.

Returns

A dictionary containing driver-specific info.

Example:

```
{
    'version': '2.23'
    'port': '80',
    'logicalportip': '1.1.1.1',
     ...
}
```

get_configured_ip_versions()
Get allowed IP versions.

The supported versions are returned with list, possible values are: [4], [6], or [4, 6]

Drivers that assert ipv6_implemented = True must override this method. If the returned list includes 4, then shares created by this driver must have an IPv4 export location. If the list includes 6, then shares created by the driver must have an IPv6 export location.

Drivers should check that their storage controller actually has IPv4/IPv6 enabled and configured properly.

get_default_filter_function(*pool=None*)
Get the default filter_function string.

Each driver could overwrite the method to return a well-known default string if it is available.

Parameters **pool** pool name to get the filter or None

Returns None

get_default_goodness_function()
Get the default goodness_function string.

Each driver could overwrite the method to return a well-known default string if it is available.

Returns None

get_filter_function(*pool=None*)
Get filter_function string.

Returns either the string from the driver instance or global section in manila.conf. If nothing is specified in manila.conf, then try to find the default filter_function. When None is returned the scheduler will always pass the driver instance.

Parameters **pool** pool name to get the filter or None

Returns a filter_function string or None

get_goodness_function()
Get good_function string.

Returns either the string from the driver instance or global section in manila.conf. If nothing is specified in manila.conf, then try to find the default goodness_function. When None is returned the scheduler will give the lowest score to the driver instance.

Returns a goodness_function string or None

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

get_periodic_hook_data(*context*, *share_instances*)
Dedicated for update/extend of data for existing share instances.

Redefine this method in share driver to be able to update/change/extend share instances data that will be used by periodic hook action. One of possible updates is add-on of automount CLI commands for each share instance for case of notification is enabled using hook approach.

Parameters

- **context** Current context
- **share_instances** share instances list provided by share manager

Returns list of share instances.

get_pool(*share*)
Return pool name where the share resides on.

Parameters **share** The share hosted by the driver.

get_share_server_network_info(*context*, *share_server*, *identifier*, *driver_options*)
Obtain network allocations used by share server.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier
- **driver_options** Dictionary of driver options to assist managing the share server

Returns A list containing IP addresses allocated in the backend.

Example:

```
['10.10.10.10', 'fd11::2000', '192.168.10.10']
```

get_share_server_pools(*share_server*)
Return list of pools related to a particular share server.

Parameters **share_server** ShareServer class instance.

get_share_stats(*refresh=False*)
Get share status.

If refresh is True, run update the stats first.

get_share_status(*share*, *share_server=None*)
Invoked periodically to get the current status of a given share.

Driver can use this method to update the status of a share that is still pending from other operations. This method is expected to be called in a periodic interval set by the periodic_interval configuration in seconds.

Parameters

- **share** share to get updated status from.
- **share_server** share server model or None.

Returns

a dictionary of updates with the current share status, that must be available, creating_from_snapshot or error, a list of export locations, if available, and a progress field which indicates the completion of the share creation operation. EXAMPLE:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
    'progress': '50%'
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance status to error.

manage_existing(*share*, *driver_options*)
Brings an existing share under Manila management.

If the provided share is not valid, then raise a ManageInvalidShare exception, specifying a reason for the failure.

If the provided share is not in a state that can be managed, such as being replicated on the backend, the driver *MUST* raise ManageInvalidShare exception with an appropriate message.

The share has a share_type, and the driver can inspect that and compare against the properties of the referenced backend share. If they are incompatible, raise a ManageExistingShareTypeMismatch, specifying a reason for the failure.

This method is invoked when the share is being managed with a share type that has `driver_handles_share_servers` extra-spec set to False.

Parameters

- **share** Share model
- **driver_options** Driver-specific options provided by admin.

Returns share_update dictionary with required key size, which should contain size of the share.

manage_existing_snapshot(*snapshot*, *driver_options*)

Brings an existing snapshot under Manila management.

If provided snapshot is not valid, then raise a ManageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being managed belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to False.

Parameters **snapshot** ShareSnapshotInstance model with ShareSnapshot data.

Example:: { id: <instance id>, snapshot_id: < snapshot id>, provider_location: <location>, }

Parameters **driver_options** Optional driver-specific options provided by admin.

Example:

```
{
'key': 'value',
...
}
```

Returns model_update dictionary with required key size, which should contain size of the share snapshot, and key export_locations containing a list of export locations, if snapshots can be mounted.

manage_existing_snapshot_with_server(*snapshot*, *driver_options*, *share_server=None*)

Brings an existing snapshot under Manila management.

If provided snapshot is not valid, then raise a ManageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being managed belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to True.

Parameters **snapshot** ShareSnapshotInstance model with ShareSnapshot data.

Example:: { id: <instance id>, snapshot_id: < snapshot id>, provider_location: <location>, }

Parameters **driver_options** Optional driver-specific options provided by admin.

Example:

```
{
'key': 'value',
...
}
```

Parameters **share_server** Share server model or None.

Returns model_update dictionary with required key size, which should contain size of the share snapshot, and key export_locations containing a list of export locations, if snapshots can be mounted.

manage_existing_with_server(*share*, *driver_options*, *share_server=None*)

Brings an existing share under Manila management.

If the provided share is not valid, then raise a ManageInvalidShare exception, specifying a reason for the failure.

If the provided share is not in a state that can be managed, such as being replicated on the backend, the driver *MUST* raise ManageInvalidShare exception with an appropriate message.

The share has a share_type, and the driver can inspect that and compare against the properties of the referenced backend share. If they are incompatible, raise a ManageExistingShareTypeMismatch, specifying a reason for the failure.

This method is invoked when the share is being managed with a share type that has `driver_handles_share_servers` extra-spec set to True.

Parameters

- **share** Share model
- **driver_options** Driver-specific options provided by admin.
- **share_server** Share server model or None.

Returns share_update dictionary with required key size, which should contain size of the share.

manage_server(*context*, *share_server*, *identifier*, *driver_options*)

Manage the share server and return compiled back end details.

Parameters

- **context** Current context.
- **share_server** Share server model.
- **identifier** A driver-specific share server identifier
- **driver_options** Dictionary of driver options to assist managing the share server

Returns Identifier and dictionary with back end details to be saved in the database.

Example:

```
'my_new_server_identifier',{'server_name': 'my_old_server'}
```

property max_share_server_size

property max_shares_per_share_server

migration_cancel(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Cancels migration of a given share to another host.

Note: Is called in source shares backend to cancel migration.

If possible, driver can implement a way to cancel an in-progress migration.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

migration_check_compatibility(*context*, *source_share*, *destination_share*, *share_server=None*, *destination_share_server=None*)

Checks destination compatibility for migration of a given share.

Note: Is called to test compatibility with destination backend.

Driver should check if it is compatible with destination backend so driver-assisted migration can proceed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the share to be migrated.
- **destination_share** Reference to the share model to be used by migrated share.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

A dictionary containing values indicating if destination backend is compatible, if share can remain writable during migration, if it can preserve all file metadata and if it can perform migration of given share non-disruptively.

Example:

```
{
    'compatible': True,
    'writable': True,
    'preserve_metadata': True,
    'nondisruptive': True,
    'preserve_snapshots': True,
}
```

migration_complete(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Completes migration of a given share to another host.

Note: Is called in source shares backend to complete migration.

If driver is implementing 2-phase migration, this method should perform the disruptive tasks related to the 2nd phase of migration, thus completing it. Driver should also delete all original share data from source backend.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

If the migration changes the share export locations, snapshot provider locations or snapshot export locations, this method should return a dictionary with the relevant info. In such case, a dictionary containing a list of export locations and a list of model updates for each snapshot indexed by their IDs.

Example:

```
{
    'export_locations':
    [
        {
        'path': '1.2.3.4:/foo',
        'metadata': {},
        'is_admin_only': False
        },
        {
```

(continues on next page)

(continued from previous page)

```
            'path': '5.6.7.8:/foo',
            'metadata': {},
            'is_admin_only': True
            },
        ],
        'snapshot_updates':
        {
            'bc4e3b28-0832-4168-b688-67fdc3e9d408':
            {
            'provider_location': '/snapshots/foo/bar_1',
            'export_locations':
            [
                {
                'path': '1.2.3.4:/snapshots/foo/bar_1',
                'is_admin_only': False,
                },
                {
                'path': '5.6.7.8:/snapshots/foo/bar_1',
                'is_admin_only': True,
                },
            ],
            },
            '2e62b7ea-4e30-445f-bc05-fd523ca62941':
            {
            'provider_location': '/snapshots/foo/bar_2',
            'export_locations':
            [
                {
                'path': '1.2.3.4:/snapshots/foo/bar_2',
                'is_admin_only': False,
                },
                {
                'path': '5.6.7.8:/snapshots/foo/bar_2',
                'is_admin_only': True,
                },
            ],
            },
        },
}
```

migration_continue(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Continues migration of a given share to another host.

Note: Is called in source shares backend to continue migration.

Driver should implement this method to continue monitor the migration progress in storage and perform following steps until 1st phase is completed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns Boolean value to indicate if 1st phase is finished.

migration_get_progress(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Obtains progress of migration of a given share to another host.

Note: Is called in source shares backend to obtain migration progress.

If possible, driver can implement a way to return migration progress information.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns A dictionary with at least total_progress field containing the percentage value.

migration_start(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Starts migration of a given share to another host.

Note: Is called in source shares backend to start migration.

Driver should implement this method if willing to perform migration in a driver-assisted way, useful for when source shares backend driver is compatible with destination backend driver.

This method should start the migration procedure in the backend and end. Following steps should be done in migration_continue.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

promote_replica(*context*, *replica_list*, *replica*, *access_rules*, *share_server=None*)

Promote a replica to active replica state.

Note: This call is made on the host that hosts the replica being promoted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be promoted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
```

(continues on next page)

(continued from previous page)

```
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica to be promoted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be promoted.

Returns updated_replica_list or None. The driver can return the updated list as in the request parameter. Changes that will be updated to the Database are: export_locations, access_rules_status and replica_state.

Raises Exception. This can be any exception derived from BaseException. This is re-raised by the manager after some necessary cleanup. If the driver raises an exception during promotion, it is assumed that all of the replicas of the share are in an inconsistent state. Recovery is only possible through the periodic update call and/or administrator intervention to correct the status of the affected replicas if they become healthy again.

property replication_domain

revert_to_replicated_snapshot(*context*, *active_replica*, *replica_list*, *active_replica_snapshot*, *replica_snapshots*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a replicated share (in place) to the specified snapshot.

Note: This call is made on the active replicas host, since drivers may not be able to revert snapshots on individual replicas.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **active_replica** The current active replica
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active and its status set to reverting.
- **active_replica_snapshot** snapshot to be restored
- **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. The snapshot of the active replica to be restored with have its status attribute set to restoring.
- **share_access_rules** List of access rules for the affected share.
- **snapshot_access_rules** List of access rules for the affected snapshot.
- **share_server** Optional Share server model

revert_to_snapshot(*context*, *snapshot*, *share_access_rules*, *snapshot_access_rules*, *share_server=None*)

Reverts a share (in place) to the specified snapshot.

Does not delete the share snapshot. The share and snapshot must both be available for the restore to be attempted. The snapshot must be the most recent one taken by Manila; the API layer performs this check so the driver doesnt have to.

The share must be reverted in place to the contents of the snapshot. Application admins should quiesce or otherwise prepare the application for the shared file system contents to change suddenly.

Parameters

- **context** Current context
- **snapshot** The snapshot to be restored
- **share_access_rules** List of all access rules for the affected share
- **snapshot_access_rules** List of all access rules for the affected snapshot
- **share_server** Optional Share server model or None

setup_server(**args*, ***kwargs*)

share_server_migration_cancel(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Cancels migration of a given share server to another host.

Note: Is called in destination share servers backend to continue migration.

If possible, driver can implement a way to cancel an in-progress migration.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

share_server_migration_check_compatibility(*context*, *share_server*, *dest_host*, *old_share_network*, *new_share_network*, *shares_request_spec*)

Checks destination compatibility for migration of a share server.

Note: Is called in destination share servers backend to continue migration. Can be called by an admin to check if a given host is compatible or by the share manager to test compatibility with destination backend.

Driver should check if it is compatible with destination backend so driver-assisted migration can proceed.

Parameters

- **context** The context.RequestContext object for the request.
- **share_server** Share server model.
- **dest_host** Reference to the hos to be used by the migrated share server.
- **old_share_network** Share network model where the source share server is placed.
- **new_share_network** Share network model where the share server is going to be migrated to.
- **shares_request_spec** Dict. Contains information about all shares and share types that belong to the source share server. The drivers can use this information to check if the capabilities match with the destination backend and if there is available space to hold the new share server and all its resource.

Example:

```
{
'shares_size': 100,
'snapshots_size': 100,
'shares_req_spec':
[
    {
    'share_properties':
        {
        'size': 10
        'user_id': '2f5c1df4-5203-444e-b68e-1e60f3f26fc3'
        'project_id': '0b82b278-51d6-4357-b273-0d7263982c31'
        'snapshot_support': True
        'create_share_from_snapshot_support': True
        'revert_to_snapshot_support': False
        'mount_snapshot_support': False
        'share_proto': NFS
        'share_type_id': '360e01c1-a4f7-4782-9676-dc013f1a2f21'
        'is_public': False
        'share_group_id': None
        'source_share_group_snapshot_member_id': None
        'snapshot_id': None
        },
    'share_instance_properties':
        {
        'availability_zone_id':
            '02377ad7-381c-4b25-a04c-6fd218f22a91',
        'share_network_id': '691544aa-da83-4669-8522-22719f236e16',
        'share_server_id': 'cd658413-d02c-4d1b-ac8a-b6b972e76bac',
        'share_id': 'e42fec45-781e-4dcc-a4d2-44354ad5ae91',
        'host': 'hostA@backend1#pool0',
        'status': 'available',
        },
    'share_type':
        {
        'id': '360e01c1-a4f7-4782-9676-dc013f1a2f21',
```

(continues on next page)

(continued from previous page)

```
        'name': 'dhss_false',
        'is_public': False,
        'extra_specs':
            {
            'driver_handles_share_servers': False,
            }
        },
    'share_id': e42fec45-781e-4dcc-a4d2-44354ad5ae91,
    },
],
}
```

Returns

A dictionary containing values indicating if destination backend is compatible, if share can remain writable during migration, if it can preserve all file metadata and if it can perform migration of given share non-disruptively.

Example:

```
{
    'compatible': True,
    'writable': True,
    'nondisruptive': True,
    'preserve_snapshots': True,
    'migration_cancel': True,
    'migration_get_progress': False,
}
```

share_server_migration_complete(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*, *new_network_info*)

Completes migration of a given share server to another host.

Note: Is called in destination share servers backend to complete migration.

If driver is implementing 2-phase migration, this method should perform the disruptive tasks related to the 2nd phase of migration, thus completing it. Driver should also delete all original data from source backend.

It expected that all shares and snapshots will be available at the destination share server in the end of the migration complete and all updates provided in the returned model update.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.

- **snapshots** All snapshots in the source share server that should be migrated.
- **new_network_info** Network allocation associated to the destination share server.

Returns

If the migration changes the shares export locations, snapshots provider locations or snapshots export locations, this method should return a dictionary containing a list of share instances and snapshot instances indexed by their ids, where each instance should provide a dict with the relevant information that need to be updated.

Example:

```
{
    'share_updates':
    {
        '4363eb92-23ca-4888-9e24-502387816e2a':
        {
        'export_locations':
        [
            {
            'path': '1.2.3.4:/foo',
            'metadata': {},
            'is_admin_only': False
            },
            {
            'path': '5.6.7.8:/foo',
            'metadata': {},
            'is_admin_only': True
            },
        ],
        'pool_name': 'poolA',
        },
    },
    'snapshot_updates':
    {
        'bc4e3b28-0832-4168-b688-67fdc3e9d408':
        {
        'provider_location': '/snapshots/foo/bar_1',
        'export_locations':
        [
            {
            'path': '1.2.3.4:/snapshots/foo/bar_1',
            'is_admin_only': False,
            },
            {
            'path': '5.6.7.8:/snapshots/foo/bar_1',
            'is_admin_only': True,
            },
        ],
        },
```

(continues on next page)

(continued from previous page)

```
        '2e62b7ea-4e30-445f-bc05-fd523ca62941':
        {
        'provider_location': '/snapshots/foo/bar_2',
        'export_locations':
        [
            {
            'path': '1.2.3.4:/snapshots/foo/bar_2',
            'is_admin_only': False,
            },
            {
            'path': '5.6.7.8:/snapshots/foo/bar_2',
            'is_admin_only': True,
            },
        ],
        },
    }
    'backend_details':
    {
        'new_share_server_info_key':
            'new_share_server_info_value',
    },
}
```

share_server_migration_continue(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Continues migration of a given share server to another host.

Note: Is called in destination share servers backend to continue migration.

Driver should implement this method to continue monitor the migration progress in storage and perform following steps until 1st phase is completed.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns Boolean value to indicate if 1st phase is finished.

share_server_migration_get_progress(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Obtains progress of migration of a share server to another host.

Note: Is called in destination shares backend to obtain migration progress.

If possible, driver can implement a way to return migration progress information.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns A dictionary with at least total_progress field containing the percentage value.

share_server_migration_start(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Starts migration of a given share server to another host.

Note: Is called in destination share servers backend to start migration.

Driver should implement this method if willing to perform a server migration in driver-assisted way, useful when source share servers backend driver is compatible with destination backend driver. This method should start the migration procedure in the backend and return immediately. Following steps should be done in share_server_migration_continue.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used by as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns

Dict with migration information to be set in the destination share server.

Example:

```
{
    'backend_details': {
        'migration_info_key':  'migration_info_value',
    }
}
```

shrink_share(*share*, *new_size*, *share_server=None*)
Shrinks size of existing share.

If consumed space on share larger than new_size driver should raise ShareShrinkingPossibleDataLoss exception: raise ShareShrinkingPossibleDataLoss(share_id=share[id])

Parameters

- **share** Share model
- **new_size** New size of share (new_size < share[size])
- **share_server** Optional Share server model

:raises ShareShrinkingPossibleDataLoss, NotImplementedError

snapshot_update_access(*context*, *snapshot*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)
Update access rules for given snapshot.

`access_rules` contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the `add_rules` and `delete_rules` parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in `add_rules` and rules that need to be removed to be present in `delete_rules`.

When a rule in `add_rules` already exists in the back end, drivers must not raise an exception. When a rule in `delete_rules` was never applied, drivers must not raise an exception, or attempt to set the rule to `error` state.

`add_rules` and `delete_rules` can be empty lists, in this situation, drivers should ensure that the rules present in `access_rules` are the same as those on the back end.

Parameters

- **context** Current context
- **snapshot** Snapshot model with snapshot data.
- **access_rules** All access rules for given snapshot
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

property snapshots_are_supported

teardown_server(**args*, ***kwargs*)

unmanage(*share*)
Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to False.

unmanage_server(*server_details*, *security_services=None*)
Unmanages the share server.

If a driver supports unmanaging of share servers, the driver must override this method and return successfully.

Parameters

- **server_details** share server backend details.
- **security_services** list of security services configured with this share server.

unmanage_snapshot(*snapshot*)
Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to False.

unmanage_snapshot_with_server(*snapshot*, *share_server=None*)
Removes the specified snapshot from Manila management.

Does not delete the underlying backend share snapshot.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share snapshot.

If provided share snapshot cannot be unmanaged, then raise an UnmanageInvalidShareSnapshot exception, specifying a reason for the failure.

This method is invoked when the snapshot that is being unmanaged belongs to a share that has its share type with `driver_handles_share_servers` extra-spec set to True.

unmanage_with_server(*share*, *share_server=None*)
Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to True.

update_access(*context*, *share*, *access_rules*, *add_rules*, *delete_rules*, *share_server=None*)

Update access rules for given share.

`access_rules` contains all access_rules that need to be on the share. If the driver can make bulk access rule updates, it can safely ignore the `add_rules` and `delete_rules` parameters.

If the driver cannot make bulk access rule changes, it can rely on new rules to be present in `add_rules` and rules that need to be removed to be present in `delete_rules`.

When a rule in `delete_rules` was never applied, drivers must not raise an exception, or attempt to set the rule to `error` state.

`add_rules` and `delete_rules` can be empty lists, in this situation, drivers should ensure that the rules present in `access_rules` are the same as those on the back end. One scenario where this situation is forced is when the access_level is changed for all existing rules (share migration and for readable replicas).

Drivers must be mindful of this call for share replicas. When update_access is called on one of the replicas, the call is likely propagated to all replicas belonging to the share, especially when individual rules are added or removed. If a particular access rule does not make sense to the driver in the context of a given replica, the driver should be careful to report a correct behavior, and take meaningful action. For example, if R/W access is requested on a replica that is part of a readable type replication; R/O access may be added by the driver instead of R/W. Note that raising an exception *will* result in the access_rules_status on the replica, and the share itself being out_of_sync. Drivers can sync on the valid access rules that are provided on the `create_replica` and `promote_replica` calls.

Parameters

- **context** Current context
- **share** Share model with share data.
- **access_rules** A list of access rules for given share
- **add_rules** Empty List or List of access rules which should be added. access_rules already contains these rules.
- **delete_rules** Empty List or List of access rules which should be removed. access_rules doesnt contain these rules.
- **share_server** None or Share server model

Returns

None, or a dictionary of updates in the format:

```
{
  09960614-8574-4e03-89cf-7cf267b0bd08: {
    access_key: alice31493e5441b8171d2310d80e37e, state: error,
  },
  28f6eabb-4342-486a-a7f4-45688f0c0295: {
```

```
            access_key: bob0078aa042d5a7325480fd13228b, state: active,
        },

    }
```

The top level keys are access_id fields of the access rules that need to be updated. `access_key``s are credentials (str) of the entities granted access. Any rule in the ``access_rules` parameter can be updated.

Important: Raising an exception in this method will force *all* rules in applying and denying states to error.

An access rule can be set to error state, either explicitly via this return parameter or because of an exception raised in this method. Such an access rule will no longer be sent to the driver on subsequent access rule updates. When users deny that rule however, the driver will be asked to deny access to the client/s represented by the rule. We expect that a rule that was error-ed at the driver should never exist on the back end. So, do not fail the deletion request.

Also, it is possible that the driver may receive a request to add a rule that is already present on the back end. This can happen if the share manager service goes down while the driver is committing access rule changes. Since we cannot determine if the rule was applied successfully by the driver before the disruption, we will treat all applying transitional rules as new rules and repeat the request.

---

update_admin_network_allocation(*context*, *share_server*)
Update admin network allocation after share server creation.

update_network_allocation(*context*, *share_server*)
Update network allocation after share server creation.

update_replica_state(*context*, *replica_list*, *replica*, *access_rules*, *replica_snapshots*, *share_server=None*)
Update the replica_state of a replica.

Note: This call is made on the host which hosts the replica being updated.

Drivers should fix replication relationships that were broken if possible inside this method.

This method is called periodically by the share manager; and whenever requested by the administrator through the resync API.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be updated. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
```

(continues on next page)

(continued from previous page)

```
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica being updated Replica state will always be in_sync, out_of_sync, or error. Replicas in active state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
```

(continues on next page)

(continued from previous page)

```
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share. The driver could attempt to sync on any unapplied access_rules.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica being updated. The driver needs to ensure that this snapshot instance is truly available before transitioning from out_of_sync to in_sync. Snapshots instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the update_replicated_snapshot method.

Example:

```
 [
  {
'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
   },
 'share_replica_snapshot': {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'status': 'creating',
    'provider_location': None,
       ...
   },
```

(continues on next page)

(continued from previous page)

```
  }
 ]
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_state: a str value denoting the replica_state. Valid values are in_sync and out_of_sync or None (to leave the current replica_state unchanged).

update_replicated_snapshot(*context*, *replica_list*, *share_replica*, *replica_snapshots*, *replica_snapshot*, *share_server=None*)

Update the status of a snapshot instance that lives on a replica.

Note: For DR and Readable styles of replication, this call is made on the replicas host and not the active replicas host.

This method is called periodically by the share manager. It will query for snapshot instances that track the parent snapshot across non-active replicas. Drivers can expect the status of the instance to be creating or deleting. If the driver sees that a snapshot instance has been removed from the replicas backend and the instance status was set to deleting, it is expected to raise a SnapshotResourceNotFound exception. All other exceptions will set the snapshot instance status to error. If the instance was not in deleting state, raising a SnapshotResourceNotFound will set the instance status to error.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
     'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'in_sync',
     ...
     'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
     'share_server': <models.ShareServer> or None,
    },
    {
     'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'active',
     ...
     'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
     'share_server': <models.ShareServer> or None,
    },
```

(continues on next page)

(continued from previous page)

```
    ...
]
```

Parameters **share_replica** Share replica dictionary. This replica is associated with the snapshot instance whose status is being updated. Replicas in active replica_state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. This will include the snapshot instance being updated as well.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-feffff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    ...
]
```

Parameters **replica_snapshot** Dictionary of the snapshot instance. This is the

instance to be updated. It will be in creating or deleting state when sent via this parameter.

Example:

```
{
    'name': 'share-snapshot-18825630-574f-4912-93bb-af4611ef35a2',
    'share_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_name': 'share-d487b88d-e428-4230-a465-a800c2cce5f8',
    'status': 'creating',
    'id': '18825630-574f-4912-93bb-af4611ef35a2',
    'deleted': False,
    'created_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share': <models.ShareInstance>,
    'updated_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'progress': '0%',
    'deleted_at': None,
    'provider_location': None,
}
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_snapshot_model_update: a dictionary. The dictionary must contain values that need to be updated on the database for the snapshot instance that represents the snapshot on the replica.

Raises exception.SnapshotResourceNotFound Raise this exception for snapshots that are not found on the backend and their status was deleting.

update_share_server_security_service(*context*, *share_server*, *network_info*, *share_instances*, *share_instance_rules*, *new_security_service*, *current_security_service=None*)

Updates share server security service configuration.

If the driver supports different security services, the user can request the addition of a new security service, with a different type. If the user wants to update the current security service configuration, the driver will receive both current and new security services, which will always be of the same type.

Parameters

- **context** The context.RequestContext object for the request.
- **share_server** Reference to the share server object that will be updated.
- **network_info** All network allocation associated with the share server that will be updated.
- **share_instances** A list of share instances that belong to the share server that is being updated.
- **share_instance_rules** A list of access rules, grouped by share instance, in the following format.

Example:

```
[
   {
    'share_instance_id': '3bc10d67-2598-4122-bb62-0bdeaa8c6db3',
    'access_rules':
    [
        {
        'access_id':'906d0094-3e34-4d6c-a184-d08a908033e3',
        'access_type':'ip',
        'access_key':None,
        'access_to':'10.0.0.1',
        'access_level':'rw'
         ...
        },
    ],
    },
]
```

Parameters

- **new_security_service** New security service object to be configured in the share server.
- **current_security_service** When provided, represents the current security service that will be replaced by the new_security_service.

Raises ShareBackendException. A ShareBackendException should only be raised if the share server failed to update the security service, compromising all its access rules. By raising an exception, the share server and all its share instances will be set to error.

Returns None, or a dictionary of updates in the following format.

Example:

```
{
    '3bc10d67-2598-4122-bb62-0bdeaa8c6db3':
    {
        '09960614-8574-4e03-89cf-7cf267b0bd08':
        {
            'access_key': 'alice31493e5441b8171d2310d80e37e',
            'state': 'error',
        },
        '28f6eabb-4342-486a-a7f4-45688f0c0295':
        {
            'access_key': 'bob0078aa042d5a7325480fd13228b',
            'state': 'active',
        },
    },
}
```

The top level keys are share_instance_ids which should provide another dictionary of access

rules to be updated, indexed by their access_id. The inner access rules dictionary should only contain the access rules that need to be updated.

update_share_usage_size(*context*, *shares*)
Invoked to get the usage size of given shares.

Driver can use this method to update the share usage size of the shares. To do that, a dictionary of shares should be returned. :param shares: None or a list of all shares for updates. :returns: An empty list or a list of dictionary of updates in the following format. The value of used_size can be specified in GiB units, as a floating point number:

```
[
    {
        'id': '09960614-8574-4e03-89cf-7cf267b0bd08',
        'used_size': '200',
        'gathered_at': datetime.datetime(2017, 8, 10, 15, 14, 6),
    },
]
```

Manila share driver hooks

Manila share driver hooks are designed to provide additional possibilities for each *manila-share* service; such as any kind of notification and additional actions before and after share driver calls.

Possibilities

- Perform actions before some share driver method calls.
- **Perform actions after some share driver method calls with results of** driver call and preceding hook call.
- Call additional periodic hook each N ticks.
- Possibility to update results of drivers action by post-running hook.

Features

- Errors in hook execution can be suppressed.
- Any hook can be disabled.
- **Any amount of hook instances can be run at once for each manila-share** service.

Limitations

- **Hooks approach is not asynchronous. That is, if we run hooks, and** especially, more than one hook instance, then all of them will be executed in one thread.

Implementation in share drivers

Share drivers can [re]define method *get_periodic_hook_data* that runs with each execution of periodic hook and receives list of shares (as parameter) with existing access rules. So, each share driver, for each of its shares can add/update some information that will be used then in the periodic hook.

What is required for writing new hook implementation?

All implementations of hook interface are expected to be in manila/share/hooks. Each implementation should inherit class manila.share.hook:HookBase and redefine its abstract methods.

How to use hook implementations?

Just set config option hook_drivers in drivers config group. For example:

```
[MY_DRIVER]
hook_drivers=path.to:FooClass,path.to:BarClass
```

Then all classes defined above will be initialized. In the same config group, any config option of hook modules can be redefined too.

Note: More info about common config options for hooks can be found in module *manila.share.hook*

Driver methods that are wrapped with hooks

- allow_access
- create_share_instance
- create_snapshot
- delete_share_instance
- delete_share_server
- delete_snapshot
- deny_access
- extend_share
- init_host
- manage_share
- publish_service_capabilities

- shrink_share
- unmanage_share
- create_share_replica
- promote_share_replica
- delete_share_replica
- update_share_replica
- create_replicated_snapshot
- delete_replicated_snapshot
- update_replicated_snapshot

Above list with wrapped methods can be extended in future.

The `manila.share.hook.py` Module

Module with hook interface for actions performed by share driver.

All available hooks are placed in manila/share/hooks dir.

Hooks are used by share services and can serve several use cases such as any kind of notification and performing additional backend-specific actions.

class HookBase(*configuration*, *host*)

Bases: `object`

execute_periodic_hook(*context*, *periodic_hook_data*, **args*, ***kwargs*)

Hook called on periodic basis.

execute_post_hook(*context=None*, *func_name=None*, *pre_hook_data=None*, *driver_action_results=None*, **args*, ***kwargs*)

Hook called after drivers action.

execute_pre_hook(*context=None*, *func_name=None*, **args*, ***kwargs*)

Hook called before drivers action.

get_config_option(*key*)

Authentication and Authorization

The `manila.quota` Module

Quotas for shares.

class AbsoluteResource(*name*, *flag=None*)

Bases: `manila.quota.BaseResource`

Describe a non-reservable resource.

class BaseResource(*name*, *flag=None*)

Bases: `object`

Describe a single resource for quota checking.

property default
Return the default value of the quota.

class CountableResource(*name*, *count*, *flag=None*)
Bases: manila.quota.AbsoluteResource

Describe a countable resource.

Describe a resource where the counts arent based solely on the project ID.

class DbQuotaDriver
Bases: object

Database Quota driver.

Driver to perform necessary checks to enforce quotas and obtain quota information. The default driver utilizes the local database.

commit(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)
Commit reservations.

Parameters

- **context** The request context, for access checks.
- **reservations** A list of the reservation UUIDs, as returned by the reserve() method.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.
- **user_id** Specify the user_id if current context is admin and admin wants to impact on common user. (Special case: user operates on resource, owned/created by different user)

destroy_all_by_project(*context*, *project_id*)
Destroy metadata associated with a project.

Destroy all quotas, usages, and reservations associated with a project.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project being deleted.

destroy_all_by_project_and_share_type(*context*, *project_id*, *share_type_id*)
Destroy metadata associated with a project and share_type.

Destroy all quotas, usages, and reservations associated with a project and share_type.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project.
- **share_type_id** The UUID of the share type.

destroy_all_by_project_and_user(*context*, *project_id*, *user_id*)
Destroy metadata associated with a project and user.

Destroy all quotas, usages, and reservations associated with a project and user.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project being deleted.
- **user_id** The ID of the user being deleted.

expire(*context*)
Expire reservations.

Explores all currently existing reservations and rolls back any that have expired.

Parameters **context** The request context, for access checks.

get_by_class(*context*, *quota_class*, *resource*)
Get a specific quota by quota class.

get_class_quotas(*context*, *resources*, *quota_class*, *defaults=True*)
Retrieve quotas for a quota class.

Given a list of resources, retrieve the quotas for the given quota class.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **quota_class** The name of the quota class to return quotas for.
- **defaults** If True, the default value will be reported if there is no specific value for the resource.

get_defaults(*context*, *resources*)
Given a list of resources, retrieve the default quotas.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.

get_project_quotas(*context*, *resources*, *project_id*, *quota_class=None*, *defaults=True*, *usages=True*, *remains=False*)
Retrieve quotas for project.

Given a list of resources, retrieve the quotas for the given project.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **project_id** The ID of the project to return quotas for.
- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified. It will be ignored if project_id == context.project_id.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.

- **usages** If True, the current in_use and reserved counts will also be returned.
- **remains** If True, the current remains of the project will will be returned.

get_settable_quotas(*context*, *resources*, *project_id*, *user_id=None*, *share_type_id=None*)
Retrieve range of settable quotas.

Given a list of resources, retrieve the range of settable quotas for the given user or project.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **project_id** The ID of the project to return quotas for.
- **user_id** The ID of the user to return quotas for.
- **share_type_id** The UUID of the share_type to return quotas for.

get_share_type_quotas(*context*, *resources*, *project_id*, *share_type_id*, *quota_class=None*, *defaults=True*, *usages=True*)
Retrieve quotas for share_type and project.

Given a list of resources, retrieve the quotas for the given share_type and project.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **project_id** The UUID of the project to return quotas for.
- **share_type** UUID/name of a share type to return quotas for.
- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified. It will be ignored if project_id == context.project_id.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.
- **usages** If True, the current in_use and reserved counts will also be returned.

get_user_quotas(*context*, *resources*, *project_id*, *user_id*, *quota_class=None*, *defaults=True*, *usages=True*)
Retrieve quotas for user and project.

Given a list of resources, retrieve the quotas for the given user and project.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **project_id** The ID of the project to return quotas for.
- **user_id** The ID of the user to return quotas for.

- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified. It will be ignored if project_id == context.project_id.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.
- **usages** If True, the current in_use and reserved counts will also be returned.

limit_check(*context*, *resources*, *values*, *project_id=None*)

Check simple quota limits.

For limitsthose quotas for which there is no usage synchronization functionthis method checks that a set of proposed values are permitted by the limit restriction.

This method will raise a QuotaResourceUnknown exception if a given resource is unknown or if it is not a simple limit resource.

If any of the proposed values is over the defined quota, an OverQuota exception will be raised with the sorted list of the resources which are too high. Otherwise, the method returns nothing.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **values** A dictionary of the values to check against the quota.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.

reserve(*context*, *resources*, *deltas*, *expire=None*, *project_id=None*, *user_id=None*, *share_type_id=None*, *overquota_allowed=False*)

Check quotas and reserve resources.

For counting quotasthose quotas for which there is a usage synchronization functionthis method checks quotas against current usage and the desired deltas.

This method will raise a QuotaResourceUnknown exception if a given resource is unknown or if it does not have a usage synchronization function.

If any of the proposed values is over the defined quota, an OverQuota exception will be raised with the sorted list of the resources which are too high. Otherwise, the method returns a list of reservation UUIDs which were created.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **deltas** A dictionary of the proposed delta changes.
- **expire** An optional parameter specifying an expiration time for the reservations. If it is a simple number, it is interpreted as a number of seconds and added to the current time; if it is a datetime.timedelta object, it will also be added to the current time. A datetime.datetime object will be interpreted as the absolute expiration time. If None is specified, the default expiration time

set by default-reservation-expire will be used (this value will be treated as a number of seconds).

- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.
- **user_id** Specify the user_id if current context is admin and admin wants to impact on common user. (Special case: user operates on resource, owned/created by different user)

rollback(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)
Roll back reservations.

Parameters

- **context** The request context, for access checks.
- **reservations** A list of the reservation UUIDs, as returned by the reserve() method.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.
- **user_id** Specify the user_id if current context is admin and admin wants to impact on common user. (Special case: user operates on resource, owned/created by different user)

usage_reset(*context*, *resources*)
Reset usage records.

Reset the usage records for a particular user on a list of resources. This will force that users usage records to be refreshed the next time a reservation is made.

Note: this does not affect the currently outstanding reservations the user has; those reservations must be committed or rolled back (or expired).

Parameters

- **context** The request context, for access checks.
- **resources** A list of the resource names for which the usage must be reset.

class QuotaEngine(*quota_driver_class=None*)
Bases: `object`

Represent the set of recognized quotas.

commit(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)
Commit reservations.

Parameters

- **context** The request context, for access checks.
- **reservations** A list of the reservation UUIDs, as returned by the reserve() method.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.

count(*context*, *resource*, **args*, ***kwargs*)
Count a resource.

For countable resources, invokes the count() function and returns its result. Arguments following the context and resource are passed directly to the count function declared by the resource.

Parameters

- **context** The request context, for access checks.
- **resource** The name of the resource, as a string.

destroy_all_by_project(*context*, *project_id*)
Destroy metadata associated with a project.

Destroy all quotas, usages, and reservations associated with a project.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project being deleted.

destroy_all_by_project_and_share_type(*context*, *project_id*, *share_type_id*)
Destroy metadata associated with a project and share_type.

Destroy all quotas, usages, and reservations associated with a project and share_type.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project.
- **share_type_id** The UUID of the share_type.

destroy_all_by_project_and_user(*context*, *project_id*, *user_id*)
Destroy metadata associated with a project and user.

Destroy all quotas, usages, and reservations associated with a project and user.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project being deleted.
- **user_id** The ID of the user being deleted.

expire(*context*)
Expire reservations.

Explores all currently existing reservations and rolls back any that have expired.

Parameters **context** The request context, for access checks.

get_by_class(*context*, *quota_class*, *resource*)
Get a specific quota by quota class.

get_class_quotas(*context*, *quota_class*, *defaults=True*)
Retrieve the quotas for the given quota class.

Parameters

- **context** The request context, for access checks.
- **quota_class** The name of the quota class to return quotas for.

- **defaults** If True, the default value will be reported if there is no specific value for the resource.

get_defaults(*context*)
Retrieve the default quotas.

Parameters **context** The request context, for access checks.

get_project_quotas(*context*, *project_id*, *quota_class=None*, *defaults=True*, *usages=True*, *remains=False*)
Retrieve the quotas for the given project.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project to return quotas for.
- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.
- **usages** If True, the current in_use and reserved counts will also be returned.
- **remains** If True, the current remains of the project will will be returned.

get_settable_quotas(*context*, *project_id*, *user_id=None*, *share_type_id=None*)
Get settable quotas.

Given a list of resources, retrieve the range of settable quotas for the given user or project.

Parameters

- **context** The request context, for access checks.
- **resources** A dictionary of the registered resources.
- **project_id** The ID of the project to return quotas for.
- **user_id** The ID of the user to return quotas for.
- **share_type_id** The UUID of the share_type to return quotas for.

get_share_type_quotas(*context*, *project_id*, *share_type_id*, *quota_class=None*, *defaults=True*, *usages=True*)
Retrieve the quotas for the given user and project.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project to return quotas for.
- **share_type_id** The UUID of the user to return quotas for.
- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.

- **usages** If True, the current in_use and reserved counts will also be returned.

get_user_quotas(*context*, *project_id*, *user_id*, *quota_class=None*, *defaults=True*, *usages=True*)

Retrieve the quotas for the given user and project.

Parameters

- **context** The request context, for access checks.
- **project_id** The ID of the project to return quotas for.
- **user_id** The ID of the user to return quotas for.
- **quota_class** If project_id != context.project_id, the quota class cannot be determined. This parameter allows it to be specified.
- **defaults** If True, the quota class value (or the default value, if there is no value from the quota class) will be reported if there is no specific value for the resource.
- **usages** If True, the current in_use and reserved counts will also be returned.

limit_check(*context*, *project_id=None*, ***values*)

Check simple quota limits.

For limitsthose quotas for which there is no usage synchronization functionthis method checks that a set of proposed values are permitted by the limit restriction. The values to check are given as keyword arguments, where the key identifies the specific quota limit to check, and the value is the proposed value.

This method will raise a QuotaResourceUnknown exception if a given resource is unknown or if it is not a simple limit resource.

If any of the proposed values is over the defined quota, an OverQuota exception will be raised with the sorted list of the resources which are too high. Otherwise, the method returns nothing.

Parameters

- **context** The request context, for access checks.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.

register_resource(*resource*)

Register a resource.

register_resources(*resources*)

Register a list of resources.

reserve(*context*, *expire=None*, *project_id=None*, *user_id=None*, *share_type_id=None*, *overquota_allowed=False*, ***deltas*)

Check quotas and reserve resources.

For counting quotasthose quotas for which there is a usage synchronization functionthis method checks quotas against current usage and the desired deltas. The deltas are given as keyword arguments, and current usage and other reservations are factored into the quota check.

This method will raise a QuotaResourceUnknown exception if a given resource is unknown or if it does not have a usage synchronization function.

If any of the proposed values is over the defined quota, an OverQuota exception will be raised with the sorted list of the resources which are too high. Otherwise, the method returns a list of reservation UUIDs which were created.

Parameters

- **context** The request context, for access checks.
- **expire** An optional parameter specifying an expiration time for the reservations. If it is a simple number, it is interpreted as a number of seconds and added to the current time; if it is a datetime.timedelta object, it will also be added to the current time. A datetime.datetime object will be interpreted as the absolute expiration time. If None is specified, the default expiration time set by default-reservation-expire will be used (this value will be treated as a number of seconds).
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.

property resources

rollback(*context*, *reservations*, *project_id=None*, *user_id=None*, *share_type_id=None*)

Roll back reservations.

Parameters

- **context** The request context, for access checks.
- **reservations** A list of the reservation UUIDs, as returned by the reserve() method.
- **project_id** Specify the project_id if current context is admin and admin wants to impact on common users tenant.

usage_reset(*context*, *resources*)

Reset usage records.

Reset the usage records for a particular user on a list of resources. This will force that users usage records to be refreshed the next time a reservation is made.

Note: this does not affect the currently outstanding reservations the user has; those reservations must be committed or rolled back (or expired).

Parameters

- **context** The request context, for access checks.
- **resources** A list of the resource names for which the usage must be reset.

class ReservableResource(*name*, *sync*, *flag=None*)

Bases: manila.quota.BaseResource

Describe a reservable resource.

The manila.policy Module

Policy Engine For Manila

authorize(*context*, *action*, *target*, *do_raise=True*, *exc=None*)
Verifies that the action is valid on the target in this context.

Parameters

- **context** manila context
- **action** string representing the action to be checked this should be colon separated for clarity. i.e. `share:create`,
- **target** dictionary representing the object of the action for object creation this should be a dictionary representing the location of the object e.g. `{'project_id': context.project_id}`
- **do_raise** if True (the default), raises PolicyNotAuthorized; if False, returns False
- **exc** Class of the exception to raise if the check fails. Any remaining arguments passed to `authorize()` (both positional and keyword arguments) will be passed to the exception class. If not specified, `PolicyNotAuthorized` will be used.

Raises **manila.exception.PolicyNotAuthorized** if verification fails and do_raise is True. Or if exc is specified it will raise an exception of that type.

Returns returns a non-False value (not necessarily True) if authorized, and the exact value False if not authorized and do_raise is False.

check_is_admin(*context*)
Whether or not user is admin according to policy setting.

check_policy(*context*, *resource*, *action*, *target_obj=None*, *do_raise=True*)

default_target(*context*)

enforce(*context*, *action*, *target*, *do_raise=True*)
Verifies that the action is valid on the target in this context.

IMPORTANT ONLY for use in API extensions. This method ignores unregistered rules and applies a default rule on them; there should be no unregistered rules in first party manila APIs.

Parameters

- **context** manila context
- **action** string representing the action to be checked, this should be colon separated for clarity. i.e. `share:create`,
- **target** dictionary representing the object of the action for object creation, this should be a dictionary representing the location of the object e.g. `{'project_id': context.project_id}`
- **do_raise** Whether to raise an exception if check fails.

Returns When `do_raise` is `False`, returns a value that evaluates as `True` or `False` depending on whether the policy allows action on the target.

Raises manila.exception.PolicyNotAuthorized if verification fails and `do_raise` is `True`.

`get_enforcer()`

`get_rules()`

`init(rules=None, use_conf=True, suppress_deprecation_warnings=False)`
Init an Enforcer class.

Parameters

- **policy_file** Custom policy file to use, if none is specified, *CONF.policy_file* will be used.
- **rules** Default dictionary / Rules to use. It will be considered just in the first instantiation.
- **use_conf** Whether to load rules from config file.
- **suppress_deprecation_warnings** Whether to suppress policy deprecation warnings.

`register_rules(enforcer)`

`reset()`

`set_rules(rules, overwrite=True, use_conf=False)`
Set rules based on the provided dict of rules.

Parameters

- **rules** New rules to use. It should be an instance of dict.
- **overwrite** Whether to overwrite current rules or update them with the new rules.
- **use_conf** Whether to reload rules from config file.

`wrap_check_policy(resource)`
Check policy corresponding to the wrapped methods prior to execution.

System limits

The following limits need to be defined and enforced:

- Maximum cumulative size of shares and snapshots (GB)
- Total number of shares
- Total number of snapshots
- Total number of share networks

Scheduler

The manila.scheduler.manager Module

Scheduler Service

class SchedulerManager(*scheduler_driver=None*, *service_name=None*, **args*, ***kwargs*)

Bases: manila.manager.Manager

Chooses a host to create shares.

RPC_API_VERSION = '1.11'

create_share_group(*context*, *share_group_id*, *request_spec=None*, *filter_properties=None*)

create_share_instance(*context*, *request_spec=None*, *filter_properties=None*)

create_share_replica(*context*, *request_spec=None*, *filter_properties=None*)

extend_share(*context*, *share_id*, *new_size*, *reservations*, *request_spec=None*, *filter_properties=None*)

get_host_list(*context*)

Get a list of hosts from the HostManager.

get_pools(*context*, *filters=None*, *cached=False*)

Get active pools from the schedulers cache.

get_service_capabilities(*context*)

Get the normalized set of capabilities for this zone.

init_host_with_rpc()

A hook for service to do jobs after RPC is ready.

Like init_host(), this method is a hook where services get a chance to execute tasks that *need* RPC. Child classes should override this method.

manage_share(*context*, *share_id*, *driver_options*, *request_spec*, *filter_properties=None*)

Ensure that the host exists and can accept the share.

migrate_share_to_host(*context*, *share_id*, *host*, *force_host_assisted_migration*, *preserve_metadata*, *writable*, *nondisruptive*, *preserve_snapshots*, *new_share_network_id*, *new_share_type_id*, *request_spec*, *filter_properties=None*)

Ensure that the host exists and can accept the share.

request_service_capabilities(*context*)

update_service_capabilities(*context*, *service_name=None*, *host=None*, *capabilities=None*, *timestamp=None*, ***kwargs*)

Process a capability update from a service node.

The manila.scheduler.base_handler Module

A common base for handling extension classes.

Used by BaseFilterHandler and BaseWeightHandler

class BaseHandler(*modifier_class_type*, *modifier_namespace*)
Bases: object

Base class to handle loading filter and weight classes.

get_all_classes()

The manila.scheduler.host_manager Module

Manage hosts in the current zone.

class HostManager
Bases: object

Base HostManager class.

get_all_host_states_share(*context*)
Returns a dict of all the hosts the HostManager knows about.

Each of the consumable resources in HostState are populated with capabilities scheduler received from RPC.

For example: {192.168.1.100: HostState(), }

get_filtered_hosts(*hosts*, *filter_properties*, *filter_class_names=None*)
Filter hosts and return only ones passing all filters.

get_pools(*context*, *filters=None*, *cached=False*)
Returns a dict of all pools on all hosts HostManager knows about.

get_weighed_hosts(*hosts*, *weight_properties*, *weigher_class_names=None*)
Weigh the hosts.

host_state_cls
alias of manila.scheduler.host_manager.HostState

update_service_capabilities(*service_name*, *host*, *capabilities*, *timestamp*)
Update the per-service capabilities based on this notification.

class HostState(*host*, *capabilities=None*, *service=None*)
Bases: object

Mutable and immutable information tracked for a host.

consume_from_share(*share*)
Incrementally update host state from an share.

update_backend(*capability*)

update_capabilities(*capabilities=None*, *service=None*)

update_from_share_capability(*capability*, *service=None*, *context=None*)
Update information about a host from its share_node info.

capability is the status info reported by share backend, a typical capability looks like this:

```
capability = {
    'share_backend_name': 'Local NFS',    #                'vendor_
↪name': 'OpenStack',           #  backend level
    'driver_version': '1.0',              #  mandatory/fixed
    'storage_protocol': 'NFS',            #/ stats&capabilities

    'active_shares': 10,                  #                'IOPS_
↪provisioned': 30000,            #  optional custom
    'fancy_capability_1': 'eat',          #  stats & capabilities
    'fancy_capability_2': 'drink',        #/

    'pools':[
      {
         'pool_name': '1st pool',           #
↪'total_capacity_gb': 500,          #  mandatory stats
         'free_capacity_gb': 230,           #   for pools
         'allocated_capacity_gb': 270,      # |
         'qos': 'False',                    # |
         'reserved_percentage': 0,          # |
         'reserved_snapshot_percentage': 0, #/

         'dying_disks': 100,                #
↪'super_hero_1': 'spider-man',    #  optional custom
         'super_hero_2': 'flash',           #  stats &
         'super_hero_3': 'neoncat',         #  capabilities
         'super_hero_4': 'green lantern', #/
       },
      {
         'pool_name': '2nd pool',
         'total_capacity_gb': 1024,
         'free_capacity_gb': 1024,
         'allocated_capacity_gb': 0,
         'qos': 'False',
         'reserved_percentage': 0,
         'reserved_snapshot_percentage': 0,

         'dying_disks': 200,
         'super_hero_1': 'superman',
         'super_hero_2': 'Hulk',
      }]
}
```

update_pools(*capability*, *service*, *context=None*)
Update storage pools information from backend reported info.

class PoolState(*host*, *capabilities*, *pool_name*)
Bases: `manila.scheduler.host_manager.HostState`

update_from_share_capability(*capability*, *service=None*, *context=None*)
Update information about a pool from its share_node info.

update_pools(*capability*)
Update storage pools information from backend reported info.

class ReadOnlyDict(*source=None*)
Bases: collections.UserDict

A read-only dict.

clear() → None. Remove all items from D.

pop(*k*[, *d*]) → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise KeyError is raised.

popitem() → (k, v), remove and return some (key, value) pair
as a 2-tuple; but raise KeyError if D is empty.

update([*E*], ***F*) → None. Update D from mapping/iterable E and F.
If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

The manila.scheduler.rpcapi Module

Client side of the scheduler manager RPC API.

class SchedulerAPI
Bases: object

Client side of the scheduler rpc API.

API version history:

1.0 - Initial version. 1.1 - Add get_pools method 1.2 - Introduce Share Instances. Replace create_share() with create_share_instance() 1.3 - Add create_consistency_group method (renamed in 1.7) 1.4 - Add migrate_share_to_host method 1.5 - Add create_share_replica 1.6 - Add manage_share 1.7 - Updated migrate_share_to_host method with new parameters 1.8 - Rename create_consistency_group -> create_share_group method 1.9 - Add cached parameter to get_pools method 1.10 - Add timestamp to update_service_capabilities 1.11 - Add extend_share

RPC_API_VERSION = '1.11'

create_share_group(*context*, *share_group_id*, *request_spec=None*, *filter_properties=None*)
Casts an rpc to the scheduler to create a share group.

Example of request_spec argument value:

```
{

    'share_group_type_id': 'fake_share_group_type_id',
    'share_group_id': 'some_fake_uuid',
    'availability_zone_id': 'some_fake_az_uuid',
    'share_types': [models.ShareType],
    'resource_type': models.ShareGroup,

}
```

create_share_instance(*context*, *request_spec=None*, *filter_properties=None*)

create_share_replica(*context*, *request_spec=None*, *filter_properties=None*)

extend_share(*context*, *share_id*, *new_size*, *reservations*, *request_spec*, *filter_properties=None*)

get_pools(*context*, *filters=None*, *cached=False*)

manage_share(*context*, *share_id*, *driver_options*, *request_spec=None*, *filter_properties=None*)

migrate_share_to_host(*context*, *share_id*, *host*, *force_host_assisted_migration*, *preserve_metadata*, *writable*, *nondisruptive*, *preserve_snapshots*, *new_share_network_id*, *new_share_type_id*, *request_spec=None*, *filter_properties=None*)

update_service_capabilities(*context*, *service_name*, *host*, *capabilities*)

The manila.scheduler.scheduler_options Module

SchedulerOptions monitors a local .json file for changes and loads it if needed. This file is converted to a data structure and passed into the filtering and weighing functions which can use it for dynamic configuration.

class SchedulerOptions

Bases: object

Monitor and load local .json file for filtering and weighing.

SchedulerOptions monitors a local .json file for changes and loads it if needed. This file is converted to a data structure and passed into the filtering and weighing functions which can use it for dynamic configuration.

get_configuration(*filename=None*)

Check the json file for changes and load it if needed.

The manila.scheduler.drivers.filter Module

The FilterScheduler is for scheduling of share and share group creation. You can customize this scheduler by specifying your own share/share group filters and weighing functions.

class FilterScheduler(**args*, ***kwargs*)

Bases: manila.scheduler.drivers.base.Scheduler

Scheduler that can be used for filtering and weighing.

get_pools(*context*, *filters*, *cached*)

Must override schedule method for scheduler to work.

host_passes_filters(*context*, *host*, *request_spec*, *filter_properties*)

Must override schedule method for migration to work.

populate_filter_properties_share(*context*, *request_spec*, *filter_properties*)

Stuff things into filter_properties.

Can be overridden in a subclass to add more data.

populate_filter_properties_share_scheduler_hint(*context*, *share_id*, *hints*, *key*, *hint*)

schedule_create_replica(*context*, *request_spec*, *filter_properties*)
Must override schedule method for create replica to work.

schedule_create_share(*context*, *request_spec*, *filter_properties*)
Must override schedule method for scheduler to work.

schedule_create_share_group(*context*, *share_group_id*, *request_spec*, *filter_properties*)
Must override schedule method for scheduler to work.

The manila.scheduler.drivers.base Module

Scheduler base class that all Schedulers should inherit from

class Scheduler
Bases: object

The base class that all Scheduler classes should inherit from.

get_host_list()
Get a list of hosts from the HostManager.

get_pools(*context*, *filters*)
Must override schedule method for scheduler to work.

get_service_capabilities()
Get the normalized set of capabilities for the services.

host_passes_filters(*context*, *host*, *request_spec*, *filter_properties*)
Must override schedule method for migration to work.

hosts_up(*context*, *topic*)
Return the list of hosts that have a running service for topic.

schedule(*context*, *topic*, *method*, **_args*, ***_kwargs*)
Must override schedule method for scheduler to work.

schedule_create_replica(*context*, *request_spec*, *filter_properties*)
Must override schedule method for create replica to work.

schedule_create_share(*context*, *request_spec*, *filter_properties*)
Must override schedule method for scheduler to work.

schedule_create_share_group(*context*, *share_group_id*, *request_spec*, *filter_properties*)
Must override schedule method for scheduler to work.

update_service_capabilities(*service_name*, *host*, *capabilities*, *timestamp*)
Process a capability update from a service node.

share_group_update_db(*context*, *share_group_id*, *host*)
Set the host and set the updated_at field of a share group.

Returns A share group with the updated fields set properly.

share_replica_update_db(*context*, *share_replica_id*, *host*)
Set the host and the scheduled_at field of a share replica.

Returns A Share Replica with the updated fields set.

share_update_db(*context*, *share_id*, *host*)
Set the host and set the scheduled_at field of a share.

Returns A Share with the updated fields set properly.

The manila.scheduler.drivers.chance Module

Chance (Random) Scheduler implementation

class ChanceScheduler

Bases: manila.scheduler.drivers.base.Scheduler

Implements Scheduler as a random node selector.

schedule_create_share(*context*, *request_spec*, *filter_properties*)

Picks a host that is up at random.

The manila.scheduler.drivers.simple Module

Simple Scheduler

class SimpleScheduler

Bases: manila.scheduler.drivers.chance.ChanceScheduler

Implements Naive Scheduler that tries to find least loaded host.

schedule_create_share(*context*, *request_spec*, *filter_properties*)

Picks a host that is up and has the fewest shares.

Scheduler Filters

The manila.scheduler.filters.availability_zone Filter

class AvailabilityZoneFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

Filters Hosts by availability zone.

host_passes(*host_state*, *filter_properties*)

Return True if the HostState passes the filter, otherwise False.

Override this in a subclass.

run_filter_once_per_request = True

The manila.scheduler.filters.base Filter

Filter support

class BaseFilter

Bases: object

Base class for all filter classes.

filter_all(*filter_obj_list*, *filter_properties*)

Yield objects that pass the filter.

Can be overridden in a subclass, if you need to base filtering decisions on all objects. Otherwise, one can just override _filter_one() to filter a single object.

run_filter_for_index(*index*)
Check if filter needs to be run for the index-th instance.

Return True if the filter needs to be run for the index-th instance in a request. Only need to override this if a filter needs anything other than first only or all behaviour.

run_filter_once_per_request = False

class BaseFilterHandler(*modifier_class_type*, *modifier_namespace*)
Bases: manila.scheduler.base_handler.BaseHandler

Base class to handle loading filter classes.

This class should be subclassed where one needs to use filters.

get_filtered_objects(*filter_classes*, *objs*, *filter_properties*, *index=0*)
Get objects after filter

Parameters

- **filter_classes** filters that will be used to filter the objects
- **objs** objects that will be filtered
- **filter_properties** client filter properties
- **index** This value needs to be increased in the caller function of get_filtered_objects when handling each resource.

The manila.scheduler.filters.base_host Filter

Scheduler host filters

class BaseHostFilter
Bases: manila.scheduler.filters.base.BaseFilter

Base class for host filters.

host_passes(*host_state*, *filter_properties*)
Return True if the HostState passes the filter, otherwise False.

Override this in a subclass.

class HostFilterHandler(*namespace*)
Bases: manila.scheduler.filters.base.BaseFilterHandler

The manila.scheduler.filters.capabilities Filter

class CapabilitiesFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

HostFilter to work with resource (instance & volume) type records.

host_passes(*host_state*, *filter_properties*)

Return a list of hosts that can create resource_type.

The manila.scheduler.filters.capacity Filter

class CapacityFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

CapacityFilter filters based on share hosts capacity utilization.

host_passes(*host_state*, *filter_properties*)

Return True if host has sufficient capacity.

The manila.scheduler.filters.extra_specs_ops Filter

match(*value*, *req*)

The manila.scheduler.filters.ignore_attempted_hosts Filter

class IgnoreAttemptedHostsFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

Filter out previously attempted hosts

A host passes this filter if it has not already been attempted for scheduling. The scheduler needs to add previously attempted hosts to the retry key of filter_properties in order for this to work correctly. For example:

```
{
 'retry': {
           'hosts': ['host1', 'host2'],
           'num_attempts': 3,
          }
}
```

host_passes(*host_state*, *filter_properties*)

Skip nodes that have already been attempted.

The manila.scheduler.filters.json Filter

class JsonFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

Host Filter to allow simple JSON-based grammar for selecting hosts.

commands = {'<': <function JsonFilter._less_than>, '<=': <function JsonFilter._less_than_equal>, '=': <function JsonFilter._equals>, '>': <function JsonFilter._greater_than>, '>=': <function JsonFilter._greater_than_equal>, 'and': <function JsonFilter._and>, 'in': <function JsonFilter._in>, 'not': <function JsonFilter._not>, 'or': <function JsonFilter._or>}

host_passes(*host_state*, *filter_properties*)

Filters hosts.

Return a list of hosts that can fulfill the requirements specified in the query.

The manila.scheduler.filters.retry Filter

class RetryFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

Filter out already tried nodes for scheduling purposes.

host_passes(*host_state*, *filter_properties*)

Skip nodes that have already been attempted.

The manila.scheduler.filters.share_replication Filter

class ShareReplicationFilter

Bases: manila.scheduler.filters.base_host.BaseHostFilter

ShareReplicationFilter filters hosts based on replication support.

host_passes(*host_state*, *filter_properties*)

Return True if active replicas host can replicate with host.

Design of this filter:

- Share replication is symmetric. All backends that can replicate between each other must share the same replication_domain.
- For scheduling a share that can be replicated in the future, this filter checks for replication_domain capability.
- For scheduling a replica, it checks for the replication_domain compatibility.

Scheduler Weighers

The manila.scheduler.weighers.base Weigher

Pluggable Weighing support

class BaseWeigher
Bases: object

Base class for pluggable weighers.

The attributes maxval and minval can be specified to set up the maximum and minimum values for the weighed objects. These values will then be taken into account in the normalization step, instead of taking the values from the calculated weighers.

maxval = None

minval = None

weigh_objects(*weighed_obj_list*, *weight_properties*)
Weigh multiple objects.

Override in a subclass if you need access to all objects in order to calculate weighers. Do not modify the weight of an object here, just return a list of weighers.

weight_multiplier()
How weighted this weigher should be.

Override this method in a subclass, so that the returned value is read from a configuration option to permit operators specify a multiplier for the weigher.

class BaseWeightHandler(*modifier_class_type*, *modifier_namespace*)
Bases: manila.scheduler.base_handler.BaseHandler

get_weighed_objects(*weigher_classes*, *obj_list*, *weighing_properties*)
Return a sorted (descending), normalized list of WeighedObjects.

object_class
alias of manila.scheduler.weighers.base.WeighedObject

class WeighedObject(*obj*, *weight*)
Bases: object

Object with weight information.

normalize(*weight_list*, *minval=None*, *maxval=None*)
Normalize the values in a list between 0 and 1.0.

The normalization is made regarding the lower and upper values present in weight_list. If the minval and/or maxval parameters are set, these values will be used instead of the minimum and maximum from the list.

If all the values are equal, they are normalized to 0.

The manila.scheduler.weighers.base_host Weigher

Scheduler host weighers

class BaseHostWeigher
Bases: manila.scheduler.weighers.base.BaseWeigher

Base class for host weighers.

class HostWeightHandler(*namespace*)
Bases: manila.scheduler.weighers.base.BaseWeightHandler

object_class
alias of manila.scheduler.weighers.base_host.WeighedHost

class WeighedHost(*obj*, *weight*)
Bases: manila.scheduler.weighers.base.WeighedObject

to_dict()

The manila.scheduler.weighers.capacity Weigher

Capacity Weigher. Weigh hosts by their virtual or actual free capacity.

For thin provisioning, weigh hosts by their virtual free capacity calculated by the total capacity multiplied by the max over subscription ratio and subtracting the provisioned capacity; Otherwise, weigh hosts by their actual free capacity, taking into account the reserved space.

The default is to spread shares across all hosts evenly. If you prefer stacking, you can set the capacity_weight_multiplier option to a negative number and the weighing has the opposite effect of the default.

class CapacityWeigher
Bases: manila.scheduler.weighers.base_host.BaseHostWeigher

weigh_objects(*weighed_obj_list*, *weight_properties*)
Weigh multiple objects.

Override in a subclass if you need access to all objects in order to calculate weighers. Do not modify the weight of an object here, just return a list of weighers.

weight_multiplier()
Override the weight multiplier.

The manila.scheduler.weighers.pool Weigher

class PoolWeigher
Bases: manila.scheduler.weighers.base_host.BaseHostWeigher

weight_multiplier()
Override the weight multiplier.

Fake Drivers

When the real thing isnt available and you have some development to do these fake implementations of various drivers let you get on with your day.

The fake_compute Module

class API
Bases: object

Fake Compute API.

add_security_group_to_server(**args*, ***kwargs*)

image_get(**args*, ***kwargs*)

image_list(**args*, ***kwargs*)

instance_volume_attach(*ctx*, *server_id*, *volume_id*, *mount_path*)

instance_volume_detach(*ctx*, *server_id*, *volume_id*)

instance_volumes_list(*ctx*, *server_id*)

keypair_delete(**args*, ***kwargs*)

keypair_import(**args*, ***kwargs*)

keypair_list(**args*, ***kwargs*)

security_group_create(**args*, ***kwargs*)

security_group_list(**args*, ***kwargs*)

security_group_rule_create(**args*, ***kwargs*)

server_create(**args*, ***kwargs*)

server_delete(**args*, ***kwargs*)

server_get(**args*, ***kwargs*)

server_get_by_name_or_id(**args*, ***kwargs*)

server_reboot(**args*, ***kwargs*)

class FakeImage(***kwargs*)
Bases: object

class FakeKeypair(***kwargs*)
Bases: object

class FakeSecurityGroup(***kwargs*)
Bases: object

class FakeServer(***kwargs*)
Bases: object

get(*attr*, *default*)

update(**args*, ***kwargs*)

The fake_driver Module

class FakeShareDriver(**args*, ***kwargs*)

Bases: *manila.share.driver.ShareDriver*

Fake share driver.

This fake driver can be also used as a test driver within a real running manila-share instance. To activate it use this in manila.conf:

```
enabled_share_backends = fake

[fake]
driver_handles_share_servers = True
share_backend_name = fake
share_driver = manila.tests.fake_driver.FakeShareDriver
```

With it you basically mocked all backend driver calls but e.g. networking will still be activated.

allow_access(*context*, *share*, *access*, *share_server=None*)

Allow access to the share.

create_share(*context*, *share*, *share_server=None*)

Is called to create share.

create_share_from_snapshot(*context*, *share*, *snapshot*, *share_server=None*, *parent_share=None*)

Is called to create share from snapshot.

Creating a share from snapshot can take longer than a simple clone operation if data copy is required from one host to another. For this reason driver will be able complete this creation asynchronously, by providing a creating_from_snapshot status in the model update.

When answering asynchronously, drivers must implement the call get_share_status in order to provide updates for shares with creating_from_snapshot status.

It is expected that the driver returns a model update to the share manager that contains: share status and a list of export_locations. A list of export_locations is mandatory only for share in available status. The current supported status are available and creating_from_snapshot.

Parameters

- **context** Current context
- **share** Share instance model with share data.
- **snapshot** Snapshot instance model .
- **share_server** Share server model or None.
- **parent_share** Share model from parent snapshot with share data and share server model.

Returns

a dictionary of updates containing current share status and its export_location (if available).

Example:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance to error and the operation will end.

create_share_group(*context*, *group_id*, *share_server=None*)
Create a share group.

Parameters

- **context**
- **share_group_dict** The share group details EXAMPLE: { status: creating, project_id: 13c0be6290934bd98596cfa004650049, user_id: a0314a441ca842019b0952224aa39192, description: None, deleted: False, created_at: datetime.datetime(2015, 8, 10, 15, 14, 6), updated_at: None, source_share_group_snapshot_id: some_fake_uuid, share_group_type_id: some_fake_uuid, host: hostname@backend_name, share_network_id: None, share_server_id: None, deleted_at: None, share_types: [<models.ShareGroupShareTypeMapping>], id: some_fake_uuid, name: None }

Returns (share_group_model_update, share_update_list) share_group_model_update - a dict containing any values to be updated for the SG in the database. This value may be None.

create_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to create snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

Returns None or a dictionary with key export_locations containing a list of export locations, if snapshots can be mounted.

delete_share(*context*, *share*, *share_server=None*)
Is called to remove share.

delete_share_group(*context*, *group_id*, *share_server=None*)
Delete a share group

Parameters

- **context** The request context
- **share_group_dict** The share group details EXAMPLE: .. code:

```
{
'status': 'creating',
'project_id': '13c0be6290934bd98596cfa004650049',
```

(continues on next page)

(continued from previous page)

```
'user_id': 'a0314a441ca842019b0952224aa39192',
'description': None,
'deleted': 'False',
'created_at': datetime.datetime(2015, 8, 10, 15, 14, 6),
'updated_at': None,
'source_share_group_snapshot_id': 'some_fake_uuid',
'share_share_group_type_id': 'some_fake_uuid',
'host': 'hostname@backend_name',
'deleted_at': None,
'shares': [<models.Share>], # The new shares being␣
↪created
'share_types': [<models.ShareGroupShareTypeMapping>],
'id': 'some_fake_uuid',
'name': None
}
```

Returns share_group_model_update share_group_model_update - a dict containing any values to be updated for the group in the database. This value may be None.

delete_snapshot(*context*, *snapshot*, *share_server=None*)
Is called to remove snapshot.

Parameters

- **context** Current context
- **snapshot** Snapshot model. Share model could be retrieved through snapshot[share].
- **share_server** Share server model or None.

deny_access(*context*, *share*, *access*, *share_server=None*)
Deny access to the share.

do_setup(*context*)
Any initialization the share driver does while starting.

property driver_handles_share_servers

ensure_share(*context*, *share*, *share_server=None*)
Invoked to ensure that share is exported.

Driver can use this method to update the list of export locations of the share if it changes. To do that, you should return list with export locations.

Returns None or list with export locations

get_network_allocations_number()
Returns number of network allocations for creating VIFs.

Drivers that use Nova for share servers should return zero (0) here same as Generic driver does. Because Nova will handle network resources allocation. Drivers that handle networking itself should calculate it according to their own requirements. It can have 1+ network interfaces.

get_share_stats(*refresh=False*)
Get share status.

If refresh is True, run update the stats first.

get_share_status(*share*, *share_server=None*)
Invoked periodically to get the current status of a given share.

Driver can use this method to update the status of a share that is still pending from other operations. This method is expected to be called in a periodic interval set by the periodic_interval configuration in seconds.

Parameters

- **share** share to get updated status from.
- **share_server** share server model or None.

Returns

a dictionary of updates with the current share status, that must be available, creating_from_snapshot or error, a list of export locations, if available, and a progress field which indicates the completion of the share creation operation. EXAMPLE:

```
{
    'status': 'available',
    'export_locations': [{...}, {...}],
    'progress': '50%'
}
```

Raises ShareBackendException. A ShareBackendException in this method will set the instance status to error.

manage_existing(*share*, *driver_options*, *share_server=None*)
Brings an existing share under Manila management.

If the provided share is not valid, then raise a ManageInvalidShare exception, specifying a reason for the failure.

If the provided share is not in a state that can be managed, such as being replicated on the backend, the driver *MUST* raise ManageInvalidShare exception with an appropriate message.

The share has a share_type, and the driver can inspect that and compare against the properties of the referenced backend share. If they are incompatible, raise a ManageExistingShareTypeMismatch, specifying a reason for the failure.

This method is invoked when the share is being managed with a share type that has `driver_handles_share_servers` extra-spec set to False.

Parameters

- **share** Share model
- **driver_options** Driver-specific options provided by admin.

Returns share_update dictionary with required key size, which should contain size of the share.

setup_server(**args*, ***kwargs*)

teardown_server(**args*, ***kwargs*)

unmanage(*share*, *share_server=None*)
Removes the specified share from Manila management.

Does not delete the underlying backend share.

For most drivers, this will not need to do anything. However, some drivers might use this call as an opportunity to clean up any Manila-specific configuration that they have associated with the backend share.

If provided share cannot be unmanaged, then raise an UnmanageInvalidShare exception, specifying a reason for the failure.

This method is invoked when the share is being unmanaged with a share type that has `driver_handles_share_servers` extra-spec set to False.

The `fake_network` Module

class FakeNeutronNetworkHelper
Bases: `object`

setup_connectivity_with_service_instances()

class FakeServiceInstanceManager(**args*, ***kwargs*)
Bases: `object`

get_service_instance(*context*, *share_network_id*, *create=True*)

property network_helper

The `fake_utils` Module

This modules stubs out functions in manila.utils.

fake_execute(**cmd_parts*, ***kwargs*)
This function stubs out execute.

It optionally executes a preconfigued function to return expected data.

fake_execute_clear_log()

fake_execute_default_reply_handler(**ignore_args*, ***ignore_kwargs*)
A reply handler for commands that havent been added to the reply list.

Returns empty strings for stdout and stderr.

fake_execute_get_log()

fake_execute_set_repliers(*repliers*)
Allows the client to configure replies to commands.

get_fake_lock_context()

stub_out_utils_execute(*testcase*)

The fake_volume Module

class API
Bases: object

Fake Volume API.

create(**args*, ***kwargs*)

create_snapshot_force(**args*, ***kwargs*)

delete(*volume_id*)

delete_snapshot(**args*, ***kwargs*)

extend(**args*, ***kwargs*)

get(**args*, ***kwargs*)

get_all(*search_opts*)

get_all_snapshots(*search_opts*)

get_snapshot(**args*, ***kwargs*)

class FakeVolume(***kwargs*)
Bases: object

class FakeVolumeSnapshot(***kwargs*)
Bases: object

Common and Misc Libraries

Libraries common throughout manila or just ones that havent yet been categorized in depth.

The manila.context Module

RequestContext: context for requests that persist through all of manila.

class RequestContext(*user_id=None*, *project_id=None*, *is_admin=None*, *read_deleted='no'*, *project_name=None*, *remote_address=None*, *timestamp=None*, *quota_class=None*, *service_catalog=None*, ***kwargs*)
Bases: oslo_context.context.RequestContext

Security context and request information.

Represents the user taking a given action within the system.

elevated(*read_deleted=None*, *overwrite=False*)
Return a version of this context with admin flag set.

classmethod from_dict(*values*)
Construct a context object from a provided dictionary.

property read_deleted

to_dict()
Return a dictionary of context attributes.

to_policy_values()
A dictionary of context attributes to enforce policy with.

oslo.policy enforcement requires a dictionary of attributes representing the current logged in user on which it applies policy enforcement. This dictionary defines a standard list of attributes that should be available for enforcement across services.

It is expected that services will often have to override this method with either deprecated values or additional attributes used by that service specific policy.

get_admin_context(*read_deleted='no'*)

The manila.exception Module

Manila base exception handling.

Includes decorator for re-raising Manila-type exceptions.

SHOULD include dedicated exception logging.

exception AdminIPNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Admin port IP for service instance not found: %(reason)s'

exception AdminRequired(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotAuthorized

message = 'User does not have admin privileges.'

exception AvailabilityZoneNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Availability zone %(id)s could not be found.'

exception BadConfigurationException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Bad configuration: %(reason)s.'

exception BridgeDoesNotExist(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Bridge %(bridge)s does not exist.'

exception ConfigNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Could not find config at %(path)s.'

exception Conflict(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 409

message = '%(err)s'

exception ConvertedException(*code=400*, *title=''*, *explanation=''*)
Bases: webob.exc.WSGIHTTPException

exception DefaultShareTypeNotConfigured(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'No default share type is configured. Either configure a default share type or explicitly specify a share type.'

exception DriverNotInitialized(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = "Share driver '%(driver)s' not initialized."

exception EMCPowerMaxInvalidMoverID(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Invalid mover or vdm %(id)s.'

exception EMCPowerMaxLockRequiredException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unable to acquire lock(s).'

exception EMCPowerMaxXMLAPIError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(err)s'

exception EMCUnityError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = '%(err)s'

exception EMCVnxInvalidMoverID(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Invalid mover or vdm %(id)s.'

exception EMCVnxLockRequiredException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unable to acquire lock(s).'

exception EMCVnxXMLAPIError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(err)s'

exception Error
Bases: Exception

exception EvaluatorParseException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Error during evaluator parsing: %(reason)s'

exception ExportLocationNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Export location %(uuid)s could not be found.'

exception FileNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'File %(file_path)s could not be found.'

exception Found(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 302

message = 'Resource was found.'

safe = True

exception GPFSException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'GPFS exception occurred.'

exception GPFSGaneshaException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'GPFS Ganesha exception occurred.'

exception GaneshaCommandFailure(***kw*)
Bases: oslo_concurrency.processutils.ProcessExecutionError

exception GaneshaException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unknown NFS-Ganesha library exception.'

exception GlusterfsException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unknown Gluster exception.'

exception HDFSException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HDFS exception occurred!'

exception HNASBackendException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS Backend Exception: %(msg)s'

exception HNASConnException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS Connection Exception: %(msg)s'

exception HNASDirectoryNotEmpty(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS Directory is not empty: %(msg)s'

exception HNASItemNotFoundException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.StorageResourceNotFound

message = 'HNAS Item Not Found Exception: %(msg)s'

exception HNASNothingToCloneException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS Nothing To Clone Exception: %(msg)s'

exception HNASSSCContextChange(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS SSC Context has been changed unexpectedly: %(msg)s'

exception HNASSSCIsBusy(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'HNAS SSC is busy and cannot execute the command: %(msg)s'

exception HPE3ParInvalid(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(err)s'

exception HPE3ParInvalidClient(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(err)s'

exception HPE3ParUnexpectedError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = '%(err)s'

exception HSPBackendException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'HSP Backend Exception: %(msg)s'

exception HSPItemNotFoundException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'HSP Item Not Found Exception: %(msg)s'

exception HSPTimeoutException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'HSP Timeout Exception: %(msg)s'

exception HostBinaryNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Could not find binary %(binary)s on host %(host)s.'

exception HostNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Host %(host)s could not be found.'

exception IPAddressInUse(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InUse

message = 'IP address %(ip)s is already used.'

exception InUse(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Resource is in use.'

exception InfortrendCLIException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Infortrend CLI exception: %(err)s Return Code: %(rc)s,
Output: %(out)s'

exception InfortrendNASException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Infortrend NAS exception: %(err)s'

exception InstanceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Instance %(instance_id)s could not be found.'

exception Invalid(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 400

message = 'Unacceptable parameters.'

exception InvalidAPIVersionString(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'API Version String %(version)s is of invalid format. Must be of format MajorNum.MinorNum.'

exception InvalidCapacity(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid capacity: %(name)s = %(value)s.'

exception InvalidContentType(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid content type %(content_type)s.'

exception InvalidDriverMode(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid driver mode: %(driver_mode)s.'

exception InvalidExtraSpec(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid extra_spec: %(reason)s.'

exception InvalidGlobalAPIVersion(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Version %(req_ver)s is not supported by the API. Minimum is %(min_ver)s and maximum is %(max_ver)s.'

exception InvalidHost(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid host: %(reason)s'

exception InvalidInput(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid input received: %(reason)s.'

exception InvalidMetadata(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid metadata.'

exception InvalidMetadataSize(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid metadata size.'

exception InvalidParameterValue(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(err)s'

exception InvalidQuotaValue(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Change would make usage less than 0 for the following resources: %(unders)s.'

exception InvalidRequest(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'The request is invalid.'

exception InvalidReservationExpiration(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid reservation expiration %(expire)s.'

exception InvalidResults(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'The results are invalid.'

exception InvalidSecurityService(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid security service: %(reason)s'

exception InvalidShare(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share: %(reason)s.'

exception InvalidShareAccess(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid access rule: %(reason)s'

exception InvalidShareAccessLevel(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid or unsupported share access level: %(level)s.'

exception InvalidShareAccessType(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid or unsupported share access type: %(type)s.'

exception InvalidShareGroup(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share group: %(reason)s'

exception InvalidShareGroupSnapshot(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share group snapshot: %(reason)s'

exception InvalidShareGroupType(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share group type: %(reason)s.'

exception InvalidShareInstance(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share instance: %(reason)s.'

exception InvalidShareNetwork(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share network: %(reason)s'

exception InvalidShareServer(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share server: %(reason)s'

exception InvalidShareSnapshot(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share snapshot: %(reason)s.'

exception InvalidShareSnapshotInstance(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share snapshot instance: %(reason)s.'

exception InvalidShareType(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid share type: %(reason)s.'

exception InvalidSnapshotAccess(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid access rule: %(reason)s'

exception InvalidSqliteDB(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid Sqlite database.'

exception InvalidUUID(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = '%(uuid)s is not a valid uuid.'

exception InvalidVolume(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Invalid volume.'

exception LockCreationFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unable to create lock. Coordination backend not started.'

exception LockingFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Lock acquisition failed.'

exception MalformedRequestBody(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Malformed message body: %(reason)s.'

exception ManageExistingShareTypeMismatch(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Manage existing share failed due to share type mismatch: %(reason)s'

exception ManageInvalidShare(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InvalidShare

message = 'Manage existing share failed due to invalid share: %(reason)s'

exception ManageInvalidShareSnapshot(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InvalidShareSnapshot

message = 'Manage existing share snapshot failed due to invalid share snapshot: %(reason)s.'

exception ManageShareServerError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Manage existing share server failed due to: %(reason)s'

exception ManilaException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: Exception

Base Manila Exception

To correctly use this class, inherit from it and define a message property. That message will get printfd with the keyword arguments provided to the constructor.

code = 500

headers = {}

message = 'An unknown exception occurred.'

safe = False

exception MapRFSException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'MapRFS exception occurred: %(msg)s'

exception MessageNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Message %(message_id)s could not be found.'

exception MigrationError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Migration error: %(reason)s.'

exception MigrationNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Migration %(migration_id)s could not be found.'

exception MigrationNotFoundByStatus(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.MigrationNotFound

message = 'Migration not found for instance %(instance_id)s with status %(status)s.'

exception NetAppBusyAggregateForFlexGroupException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception due to an aggregate being busy while trying to provision the FlexGroup.'

exception NetAppException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception due to NetApp failure.'

exception NetworkBadConfigurationException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NetworkException

message = 'Bad network configuration: %(reason)s.'

exception NetworkBindException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception due to failed port status in binding.'

exception NetworkException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception due to network failure.'

exception NexentaException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Exception due to Nexenta failure. %(reason)s'

exception NoValidHost(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'No valid host was found. %(reason)s.'

exception NotAuthorized(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 403

message = 'Not authorized.'

exception NotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 404

message = 'Resource could not be found.'

safe = True

exception OverQuota(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Quota exceeded for resources: %(overs)s.'

exception PasteAppNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = "Could not load paste app '%(name)s' from %(path)s."

exception PolicyNotAuthorized(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotAuthorized

message = "Policy doesn't allow %(action)s to be performed."

exception PortLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of ports exceeded.'

exception ProjectQuotaNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota for project %(project_id)s could not be found.'

exception ProjectShareTypeQuotaNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota for share_type %(share_type)s in project %(project_id)s could not be found.'

exception ProjectUserQuotaNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota for user %(user_id)s in project %(project_id)s could not be found.'

exception QBException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Quobyte exception occurred: %(msg)s'

exception QBRpcException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

Quobyte backend specific exception.

message = 'Quobyte JsonRpc call to backend raised an exception: %(result)s, Quobyte error code %(qbcode)s'

exception QuotaClassNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota class %(class_name)s could not be found.'

exception QuotaError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

code = 413

headers = {'Retry-After': '0'}

message = 'Quota exceeded: code=%(code)s.'

safe = True

exception QuotaExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Quota exists for project %(project_id)s, resource %(resource)s.'

exception QuotaNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Quota could not be found.'

exception QuotaResourceUnknown(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Unknown quota resources %(unknown)s.'

exception QuotaUsageNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota usage for project %(project_id)s could not be found.'

exception ReplicationException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Unable to perform a replication action: %(reason)s.'

exception ReservationNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaNotFound

message = 'Quota reservation %(uuid)s could not be found.'

exception SSHException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception in SSH protocol negotiation or logic.'

exception SSHInjectionThreat(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'SSH command injection detected: %(command)s'

exception SchedulerHostFilterNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Scheduler host filter %(filter_name)s could not be found.'

exception SchedulerHostWeigherNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Scheduler host weigher %(weigher_name)s could not be found.'

exception SecurityServiceFailedAuth(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to authenticate user against security service.'

exception SecurityServiceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Security service %(security_service_id)s could not be found.'

exception ServiceIPNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Service IP for instance not found: %(reason)s'

exception ServiceInstanceException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Exception in service instance manager occurred.'

exception ServiceInstanceUnavailable(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ServiceInstanceException

message = 'Service instance is not available.'

exception ServiceIsDown(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Service %(service)s is down.'

exception ServiceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Service %(service_id)s could not be found.'

exception ShareAccessExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share access %(access_type)s:%(access)s exists.'

exception ShareAccessMetadataNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share access rule metadata does not exist.'

exception ShareBackendException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share backend error: %(msg)s.'

exception ShareBusyException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'Share is busy with an active task: %(reason)s.'

exception ShareCopyDataException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to copy data: %(reason)s'

exception ShareDataCopyCancelled(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Copy of contents from share instance %(src_instance)s to share instance %(dest_instance)s was cancelled.'

exception ShareDataCopyFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share Data copy failed: %(reason)s'

exception ShareExtendingError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share %(share_id)s could not be extended due to error in the driver: %(reason)s'

exception ShareGroupNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group %(share_group_id)s could not be found.'

exception ShareGroupSnapshotMemberNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group snapshot member %(member_id)s could not be found.'

exception ShareGroupSnapshotNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group snapshot %(share_group_snapshot_id)s could not be found.'

exception ShareGroupSnapshotNotSupported(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share group %(share_group)s does not support snapshots.'

exception ShareGroupSnapshotsLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of allowed share-group-snapshots is exceeded.'

exception ShareGroupTypeAccessExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share group type access for %(type_id)s / %(project_id)s combination already exists.'

exception ShareGroupTypeAccessNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group type access not found for %(type_id)s / %(project_id)s combination.'

exception ShareGroupTypeCreateFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Cannot create share group type with name %(name)s and specs %(group_specs)s.'

exception ShareGroupTypeExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share group type %(type_id)s already exists.'

exception ShareGroupTypeInUse(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share group Type %(type_id)s deletion is not allowed with groups present with the type.'

exception ShareGroupTypeNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group type %(type_id)s could not be found.'

exception ShareGroupTypeNotFoundByName(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareTypeNotFound

message = 'Share group type with name %(type_name)s could not be found.'

exception ShareGroupTypeSpecsNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share group type %(type_id)s has no group specs with key %(specs_key)s.'

exception ShareGroupsLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of allowed share-groups is exceeded.'

exception ShareInstanceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share instance %(share_instance_id)s could not be found.'

exception ShareLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of shares allowed (%(allowed)d) either per project/user or share type quota is exceeded.'

exception ShareMetadataNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Metadata item is not found.'

exception ShareMigrationError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Error in share migration: %(reason)s'

exception ShareMigrationFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share migration failed: %(reason)s'

exception ShareMountException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to mount share: %(reason)s'

exception ShareNetworkNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share network %(share_network_id)s could not be found.'

exception ShareNetworkSecurityServiceAssociationError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to associate share network %(share_network_id)s and security service %(security_service_id)s: %(reason)s.'

exception ShareNetworkSecurityServiceDissociationError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to dissociate share network %(share_network_id)s and security service %(security_service_id)s: %(reason)s.'

exception ShareNetworkSubnetNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share network subnet %(share_network_subnet_id)s could not be found.'

exception ShareNetworksLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of share-networks allowed (%(allowed)d) exceeded.'

exception ShareNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share %(share_id)s could not be found.'

exception ShareReplicaNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share Replica %(replica_id)s could not be found.'

exception ShareReplicaSizeExceedsAvailableQuota(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Requested share replica exceeds allowed project/user or share type gigabytes quota.'

exception ShareReplicasLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of allowed share-replicas is exceeded.'

exception ShareResourceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.StorageResourceNotFound

message = 'Share id %(share_id)s could not be found in storage backend.'

exception ShareServerBackendDetailsNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share server backend details does not exist.'

exception ShareServerInUse(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InUse

message = 'Share server %(share_server_id)s is in use.'

exception ShareServerMigrationError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Error in share server migration: %(reason)s'

exception ShareServerMigrationFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share server migration failed: %(reason)s'

exception ShareServerNotCreated(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share server %(share_server_id)s failed on creation.'

exception ShareServerNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share server %(share_server_id)s could not be found.'

exception ShareServerNotFoundByFilters(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareServerNotFound

message = 'Share server could not be found by filters: %(filters_description)s.'

exception ShareServerNotReady(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = "Share server %(share_server_id)s failed to reach '%(state)s' within %(time)s seconds."

exception ShareShrinkingError(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share %(share_id)s could not be shrunk due to error in the driver: %(reason)s'

exception ShareShrinkingPossibleDataLoss(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share %(share_id)s could not be shrunk due to possible data loss'

exception ShareSizeExceedsAvailableQuota(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Requested share exceeds allowed project/user or share type gigabytes quota.'

exception ShareSizeExceedsLimit(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Requested share size %(size)d is larger than maximum allowed limit %(limit)d.'

exception ShareSnapshotAccessExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InvalidInput

message = 'Share snapshot access %(access_type)s:%(access)s exists.'

exception ShareSnapshotInstanceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Snapshot instance %(instance_id)s could not be found.'

exception ShareSnapshotIsBusy(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Deleting snapshot %(snapshot_name)s that has dependent shares.'

exception ShareSnapshotNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Snapshot %(snapshot_id)s could not be found.'

exception ShareSnapshotNotSupported(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share %(share_name)s does not support snapshots.'

exception ShareTypeAccessExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share type access for %(share_type_id)s / %(project_id)s combination already exists.'

exception ShareTypeAccessNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share type access not found for %(share_type_id)s / %(project_id)s combination.'

exception ShareTypeCreateFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Cannot create share_type with name %(name)s and specs %(extra_specs)s.'

exception ShareTypeDoesNotExist(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share Type %(share_type)s does not exist.'

exception ShareTypeExists(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share Type %(id)s already exists.'

exception ShareTypeExtraSpecsNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share Type %(share_type_id)s has no extra specs with key %(extra_specs_key)s.'

exception ShareTypeInUse(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Share Type %(share_type_id)s deletion is not allowed while shares or share group types are associated with the type.'

exception ShareTypeNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Share type %(share_type_id)s could not be found.'

exception ShareTypeNotFoundByName(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareTypeNotFound

message = 'Share type with name %(share_type_name)s could not be found.'

exception ShareTypeUpdateFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Cannot update share_type %(id)s.'

exception ShareUmountException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Failed to unmount share: %(reason)s'

exception SnapshotLimitExceeded(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Maximum number of snapshots allowed (%(allowed)d) either per project/user or share type quota is exceeded.'

exception SnapshotResourceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.StorageResourceNotFound

message = 'Snapshot %(name)s not found.'

exception SnapshotSizeExceedsAvailableQuota(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.QuotaError

message = 'Requested snapshot exceeds allowed project/user or share type gigabytes quota.'

exception SnapshotUnavailable(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.StorageResourceException

message = 'Snapshot %(name)s info not available.'

exception StorageCommunicationException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Could not communicate with storage array.'

exception StorageResourceException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'Storage resource exception.'

exception StorageResourceNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.StorageResourceException

code = 404

message = 'Storage resource %(name)s not found.'

exception TegileAPIException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Unexpected response from Tegile IntelliFlash API: %(response)s'

exception UnmanageInvalidShare(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InvalidShare

message = 'Unmanage existing share failed due to invalid share: %(reason)s'

exception UnmanageInvalidShareSnapshot(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.InvalidShareSnapshot

message = 'Unmanage existing share snapshot failed due to invalid share snapshot: %(reason)s.'

exception VersionNotFoundForAPIMethod(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.Invalid

message = 'API version %(version)s is not supported on this method.'

exception VolumeNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Volume %(volume_id)s could not be found.'

exception VolumeSnapshotNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Snapshot %(snapshot_id)s could not be found.'

exception VserverNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NetAppException

message = 'Vserver %(vserver)s not found.'

exception VserverNotReady(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NetAppException

message = 'Vserver %(vserver)s is not ready yet.'

exception VserverNotSpecified(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NetAppException

message = 'Vserver not specified.'

exception WillNotSchedule(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = "Host %(host)s is not up or doesn't exist."

exception ZFSonLinuxException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ManilaException

message = 'ZFSonLinux exception occurred: %(msg)s'

exception ZadaraAttachmentsNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Failed to retrieve attachments for volume %(name)s'

exception ZadaraBadHTTPResponseStatus(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Bad HTTP response status %(status)s'

exception ZadaraExtendShareFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Failed to extend VPSA backend share. Error: %(error)s'

exception ZadaraFailedCmdWithDump(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Operation failed with status=%(status)s. Full dump: %(data)s'

exception ZadaraInvalidProtocol(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'The type of protocol %(protocol_type)s for Zadara manila driver is not supported. Only NFS or CIFS protocol is supported.'

exception ZadaraInvalidShareAccessType(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Only ip access type allowed for the Zadara manila share.'

exception ZadaraManilaInvalidAccessKey(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Invalid VPSA access key'

exception ZadaraServerCreateFailure(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Unable to create server object for initiator %(name)s'

exception ZadaraServerNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.NotFound

message = 'Unable to find server object for initiator %(name)s'

exception ZadaraSessionRequestException(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = '%(msg)s'

exception ZadaraShareNotFound(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Share %(name)s could not be found.'

exception ZadaraShareNotValid(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Share %(name)s is not valid.'

exception ZadaraUnknownCmd(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Unknown or unsupported command %(cmd)s'

exception ZadaraVPSANoActiveController(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Unable to find any active VPSA controller'

exception ZadaraVPSASnapshotCreateFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Failed to create VPSA share %(name)s snapshot. Error:
%(error)s'

exception ZadaraVPSASnapshotManageFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Failed to manage VPSA share snapshot with id %(snap_id)s.
Error: %(error)s'

exception ZadaraVPSAVolumeShareFailed(*message=None*, *detail_data={}*, ***kwargs*)
Bases: manila.exception.ShareBackendException

message = 'Failed to create VPSA backend share. Error: %(error)s'

The manila.test Module

Base classes for our unit tests.

Allows overriding of flags for use of fakes, and some black magic for inline callbacks.

class Database(*db_session*, *db_migrate*, *sql_connection*, *sqlite_db*, *sqlite_clean_db*)
Bases: fixtures.fixture.Fixture

setUp()
Prepare the Fixture for use.

This should not be overridden. Concrete fixtures should implement _setUp. Overriding of setUp is still supported, just not recommended.

After setUp has completed, the fixture will have one or more attributes which can be used (these depend totally on the concrete subclass).

Raises MultipleExceptions if _setUp fails. The last exception captured within the MultipleExceptions will be a SetupError exception.

Returns None.

Changed in 1.3 The recommendation to override setUp has been reversed - before 1.3, setUp() should be overridden, now it should not be.

Changed in 1.3.1 BaseException is now caught, and only subclasses of Exception are wrapped in MultipleExceptions.

setup_sqlite(*db_migrate*)

class TestCase(**args*, ***kwds*)
Bases: oslotest.base.BaseTestCase

Test case base class for all unit tests.

assertDictListMatch(*L1*, *L2*)
Assert a list of dicts are equivalent.

assertIn(*a*, *b*, **args*, ***kwargs*)
Python < v2.7 compatibility. Assert a in b.

assertIsInstance(*a*, *b*, **args*, ***kwargs*)
Python < v2.7 compatibility.

assertIsNone(*a*, **args*, ***kwargs*)
Python < v2.7 compatibility.

assertNotIn(*a*, *b*, **args*, ***kwargs*)
Python < v2.7 compatibility. Assert a NOT in b.

assertSubDictMatch(*sub_dict*, *super_dict*)
Assert a sub_dict is subset of super_dict.

assert_notify_called(*mock_notify*, *calls*)

flags(***kw*)
Override flag variables for a test.

is_microversion_ge(*left*, *right*)

is_microversion_lt(*left*, *right*)

mock_class(*class_name*, *new_val=None*, ***kwargs*)
Use python mock to mock a class

Mocks the specified objects attribute with the given value. Automatically performs addCleanup for the mock.

mock_object(*obj*, *attr_name*, *new_attr=None*, ***kwargs*)
Use python mock to mock an object attribute

Mocks the specified objects attribute with the given value. Automatically performs addCleanup for the mock.

override_config(*name*, *override*, *group=None*)
Cleanly override CONF variables.

setUp()
Run before each test method to initialize test environment.

start_service(*name*, *host=None*, ***kwargs*)

tearDown()
Runs after each test method to tear down test environment.

The `manila.utils` Module

Utilities and helper functions.

class ComparableMixin
Bases: `object`

DO_NOTHING
Class that literrally does nothing.

We inherit from str in case its called with json.dumps.

class DoNothing
Bases: `str`

Class that literrally does nothing.

We inherit from str in case its called with json.dumps.

class IsAMatcher(*expected_value=None*)
Bases: `object`

class LazyPluggable(*pivot*, ***backends*)
Bases: `object`

A pluggable backend loaded lazily based on some value.

class SSHPool(*ip*, *port*, *conn_timeout*, *login*, *password=None*, *privatekey=None*, **args*, ***kwargs*)
Bases: `eventlet.pools.Pool`

A simple eventlet pool to hold ssh connections.

create()
Generate a new pool item. In order for the pool to function, either this method must be overriden in a subclass or the pool must be constructed with the *create* argument. It accepts no arguments and returns a single instance of whatever thing the pool is supposed to contain.

In general, `create()` is called whenever the pool exceeds its previous high-water mark of concurrently-checked-out-items. In other words, in a new pool with *min_size* of 0, the very first call to `get()` will result in a call to `create()`. If the first caller calls `put()` before some other caller calls `get()`, then the first item will be returned, and `create()` will not be called a second time.

get()
Return an item from the pool, when one is available.

This may cause the calling greenthread to block. Check if a connection is active before returning it. For dead connections create and return a new connection.

remove(*ssh*)
Close an ssh client and remove it from free_items.

check_params_are_boolean(*keys*, *params*, *default=False*)
Validates if keys in params are boolean.

Parameters

- **keys** List of keys to check
- **params** Parameters received from REST API
- **default** default value when it does not exist

Returns a dictionary with keys and respective retrieved value

check_params_exist(*keys*, *params*)
Validates if keys exist in params.

Parameters

- **keys** List of keys to check
- **params** Parameters received from REST API

check_ssh_injection(*cmd_list*)

cidr_to_netmask(*cidr*)
Convert cidr to netmask.

cidr_to_network(*cidr*)
Convert cidr to network.

cidr_to_prefixlen(*cidr*)
Convert cidr to prefix length.

convert_str(*text*)
Convert to native string.

Convert bytes and Unicode strings to native strings:

- convert to bytes on Python 2: encode Unicode using encodeutils.safe_encode()
- convert to Unicode on Python 3: decode bytes from UTF-8

execute(**cmd*, ***kwargs*)
Convenience wrapper around oslos execute() function.

file_open(**args*, ***kwargs*)
Open file

see built-in open() documentation for more details

Note: The reason this is kept in a separate module is to easily be able to provide a stub module that doesnt alter system state at all (for unit tests)

get_bool_from_api_params(*key*, *params*, *default=False*, *strict=True*)
Parse bool value from request params.

HTTPBadRequest will be directly raised either of the cases below: 1. invalid bool string was found by key(with strict on). 2. key not found while default value is invalid(with strict on).

get_bool_param(*param_string*, *params*, *default=False*)

if_notifications_enabled(*function*)
Calls decorated method only if notifications are enabled.

is_all_tenants(*search_opts*)
Checks to see if the all_tenants flag is in search_opts

Parameters **search_opts** (*dict*) The search options for a request

Returns boolean indicating if all_tenants are being requested or not

is_valid_ip_address(*ip_address*, *ip_version*)

isotime(*at=None*, *subsecond=False*)
Stringify time in ISO 8601 format.

monkey_patch()
Patch decorator.

If the Flags.monkey_patch set as True, this function patches a decorator for all functions in specified modules. You can set decorators for each modules using CONF.monkey_patch_modules. The format is Module path:Decorator function. Example: manila.api.ec2.cloud: manila.openstack.common.notifier.api.notify_decorator

Parameters of the decorator is as follows. (See manila.openstack.common.notifier.api.notify_decorator)

name - name of the function function - object of the function

notifications_enabled(*conf*)
Check if oslo notifications are enabled.

require_driver_initialized(*func*)

retry(*retry_param=<class 'Exception'>*, *interval=1*, *retries=10*, *backoff_rate=2*, *backoff_sleep_max=None*, *wait_random=False*, *infinite=False*, *retry=<class 'tenacity.retry.retry_if_exception_type'>*)

class retry_if_exit_code(*codes*)
Bases: `tenacity.retry.retry_if_exception`

Retry on ProcessExecutionError specific exit codes.

service_is_up(*service*)
Check whether a service is up based on last heartbeat.

tempdir(***kwargs*)

translate_string_size_to_float(*string*, *multiplier='G'*)
Translates human-readable storage size to float value.

Supported values for multiplier are following: K - kilo | 1 M - mega | 1024 G - giga | 1024 * 1024 T - tera | 1024 * 1024 * 1024 P = peta | 1024 * 1024 * 1024 * 1024

returns:

- float if correct input data provided
- None if incorrect

validate_service_host(*context*, *host*)

wait_for_access_update(*context*, *db*, *share_instance*, *migration_wait_access_rules_timeout*)

walk_class_hierarchy(*clazz*, *encountered=None*)
Walk class hierarchy, yielding most derived classes first.

write_local_file(*filename*, *contents*, *as_root=False*)

write_remote_file(*ssh*, *filename*, *contents*, *as_root=False*)

The manila.wsgi Module

Tests

The test_exception Module

class FakeNotifier
Bases: object

Acts like the manila.openstack.common.notifier.api module.

ERROR = 88

notify(*context*, *publisher*, *event*, *priority*, *payload*)

class ManilaExceptionResponseCode400(**args*, ***kwds*)
Bases: manila.test.TestCase

test_invalid()

test_invalid_content_type()

test_invalid_input()

test_invalid_parameter_value()

test_invalid_quota_value()

test_invalid_request()

test_invalid_reservation_expiration()

test_invalid_results()

test_invalid_share()

test_invalid_share_access()

test_invalid_share_metadata()

test_invalid_share_metadata_size()

test_invalid_share_snapshot()

test_invalid_share_snapshot_instance()

test_invalid_share_type()

test_invalid_uuid()

test_invalid_volume()

test_manage_invalid_share_snapshot()

test_unmanage_invalid_share_snapshot()

class ManilaExceptionResponseCode403(*args, **kwds)
Bases: manila.test.TestCase

test_admin_required()

test_not_authorized()

test_policy_not_authorized()

class ManilaExceptionResponseCode404(*args, **kwds)
Bases: manila.test.TestCase

test_config_not_found()

test_default_share_type_not_configured()

test_export_location_not_found()

test_file_not_found()

test_host_binary_not_found()

test_host_not_found()

test_instance_not_found()

test_migration_not_found()

test_migration_not_found_by_status()

test_not_found()

test_paste_app_not_found()

test_project_quota_not_found()

test_quota_class_not_found()

test_quota_not_found()

test_quota_resource_unknown()

test_quota_usage_not_found()

test_reservation_not_found()

test_scheduler_host_filter_not_found()

test_scheduler_host_weigher_not_found()

test_security_service_not_found()

test_service_not_found()

test_share_metadata_not_found()

test_share_network_not_found()

test_share_network_subnet_not_found()

test_share_not_found()

test_share_replica_not_found_exception()

test_share_resource_not_found()

test_share_server_not_found()

test_share_server_not_found_by_filters()

test_share_snapshot_not_found()

test_share_type_does_not_exist()

test_share_type_extra_specs_not_found()

test_share_type_not_found()

test_share_type_not_found_by_name()

test_snapshot_instance_not_found()

test_snapshot_resource_not_found()

test_storage_resource_not_found()

test_volume_not_found()

test_volume_snapshot_not_found()

class ManilaExceptionResponseCode413(*args, **kwds)

Bases: manila.test.TestCase

test_per_share_limit_exceeded()

test_port_limit_exceeded()

test_quota_error()

test_share_limit_exceeded()

test_share_networks_limit_exceeded()

test_share_size_exceeds_available_quota()

test_snapshot_limit_exceeded()

class ManilaExceptionTestCase(*args, **kwds)

Bases: manila.test.TestCase

test_default_error_code()

test_default_error_msg()

test_default_error_msg_with_kwargs()

test_error_code_from_kwarg()

test_error_msg()

test_error_msg_exception_with_kwargs()

test_error_msg_is_exception_to_string()

test_exception_kwargs_to_string()

test_exception_multi_kwargs_to_string()

test_exception_not_redundant_period_1_test_message_(*msg*)

test_exception_not_redundant_period_2_test_message____(*msg*)

test_exception_not_redundant_period_3__(*msg*)

test_exception_redundant_period()

test_manage_share_server_error()

test_replication_exception()

test_snapshot_access_already_exists()

Share Replication

As of the Mitaka release of OpenStack, *manila* supports replication of shares between different pools for drivers that operate with `driver_handles_share_servers=False` mode. These pools may be on different backends or within the same backend. This feature can be used as a disaster recovery solution or as a load sharing mirroring solution depending upon the replication style chosen, the capability of the driver and the configuration of backends.

This feature assumes and relies on the fact that share drivers will be responsible for communicating with ALL storage controllers necessary to achieve any replication tasks, even if that involves sending commands to other storage controllers in other Availability Zones (or AZs).

End users would be able to create and manage their replicas, alongside their shares and snapshots.

Storage availability zones and replication domains

Replication is supported within the same availability zone, but in an ideal solution, an Availability Zone should be perceived as a single failure domain. So this feature provides the most value in an inter-AZ replication use case.

The `replication_domain` option is a backend specific StrOpt option to be used within `manila.conf`. The value can be any ASCII string. Two backends that can replicate between each other would have the same `replication_domain`. This comes from the premise that manila expects Share Replication to be performed between backends that have similar characteristics.

When scheduling new replicas, the scheduler takes into account the `replication_domain` option to match similar backends. It also ensures that only one replica can be scheduled per pool. When backends report multiple pools, manila would allow for replication between two pools on the same backend.

The `replication_domain` option is meant to be used in conjunction with the `storage_availability_zone` (or back end specific `backend_availability_zone`) option to utilize this solution for Data Protection/Disaster Recovery.

Replication types

When creating a share that is meant to have replicas in the future, the user will use a `share_type` with an extra_spec, *replication_type* set to a valid replication type that manila supports. Drivers must report the replication type that they support as the *replication_type* capability during the `_update_share_stats()` call.

Three types of replication are currently supported:

writable Synchronously replicated shares where all replicas are writable. Promotion is not supported and not needed.

readable Mirror-style replication with a primary (writable) copy and one or more secondary (read-only) copies which can become writable after a promotion.

dr (for Disaster Recovery) Generalized replication with secondary copies that are inaccessible until they are promoted to become the `active` replica.

Note: The term *active* replica refers to the `primary` share. In *writable* style of replication, all replicas are *active*, and there could be no distinction of a `primary` share. In *readable* and *dr* styles of replication, a `secondary` replica may be referred to as `passive`, `non-active` or simply `replica`.

Health of a share replica

Apart from the `status` attribute, share replicas have the *replica_state* attribute to denote the state of the replica. The `primary` replica will have its *replica_state* attribute set to *active*. A `secondary` replica may have one of the following values as its *replica_state*:

in_sync The replica is up to date with the active replica (possibly within a backend specific *recovery point objective*).

out_of_sync The replica has gone out of date (all new replicas start out in this *replica_state*).

error When the scheduler failed to schedule this replica or some potentially irrecoverable damage occurred with regard to updating data for this replica.

Manila requests periodic update of the *replica_state* of all non-active replicas. The update occurs with respect to an interval defined through the `replica_state_update_interval` option in `manila.conf`.

Administrators have an option of initiating a `resync` of a secondary replica (for *readable* and *dr* types of replication). This could be performed before a planned failover operation in order to have the most up-to-date data on the replica.

Promotion

For *readable* and *dr* styles, we refer to the task of switching a `non-active` replica with the *active* replica as *promotion*. For the *writable* style of replication, promotion does not make sense since all replicas are *active* (or writable) at all given points of time.

The `status` attribute of the non-active replica being promoted will be set to *replication_change* during its promotion. This has been classified as a `busy` state and hence API interactions with the share are restricted while one of its replicas is in this state.

Promotion of replicas with *replica_state* set to `error` may not be fully supported by the backend. However, manila allows the action as an administrator feature and such an attempt may be honored by backends if possible.

When multiple replicas exist, multiple replication relationships between shares may need to be redefined at the backend during the promotion operation. If the driver fails at this stage, the replicas may be left in an inconsistent state. The share manager will set all replicas to have the `status` attribute set to `error`. Recovery from this state would require administrator intervention.

Snapshots

If the driver supports snapshots, the replication of a snapshot is expected to be initiated simultaneously with the creation of the snapshot on the *active* replica. Manila tracks snapshots across replicas as separate snapshot instances. The aggregate snapshot object itself will be in `creating` state until it is `available` across all of the shares replicas that have their *replica_state* attribute set to *active* or `in_sync`.

Therefore, for a driver that supports snapshots, the definition of being `in_sync` with the primary is not only that data is ensured (within the *recovery point objective*), but also that any available snapshots on the primary are ensured on the replica as well. If the snapshots cannot be ensured, the *replica_state* *must* be reported to manila as being `out_of_sync` until the snapshots have been replicated.

When a snapshot instance has its `status` attribute set to `creating` or `deleting`, manila will poll the respective drivers for a status update. As described earlier, the parent snapshot itself will be `available` only when its instances across the *active* and `in_sync` replicas of the share are `available`. The polling interval will be the same as `replica_state_update_interval`.

Access Rules

Access rules are not meant to be different across the replicas of the share. Manila expects drivers to handle these access rules effectively depending on the style of replication supported. For example, the *dr* style of replication does mean that the non-active replicas are inaccessible, so if read-write rules are expected, then the rules should be applied on the *active* replica only. Similarly, drivers that support *readable* replication type should apply any read-write rules as read-only for the non-active replicas.

Drivers will receive all the access rules in `create_replica`, `delete_replica` and `update_replica_state` calls and have ample opportunity to reconcile these rules effectively across replicas.

Understanding Replication Workflows

Creating a share that supports replication

Administrators can create a share type with extra-spec *replication_type*, matching the style of replication the desired backend supports. Users can use the share type to create a new share that allows/supports replication. A replicated share always starts out with one replica, the `primary` share itself.

The *manila-scheduler* service will filter and weigh available pools to find a suitable pool for the share being created. In particular,

- The `CapabilityFilter` will match the *replication_type* extra_spec in the request share_type with the `replication_type` capability reported by a pool.

- The `ShareReplicationFilter` will further ensure that the pool has a non-empty `replication_domain` capability being reported as well.
- The `AvailabilityZoneFilter` will ensure that the availability_zone requested matches with the pools availability zone.

Creating a replica

The user has to specify the share name/id of the share that is supposed to be replicated and optionally an availability zone for the replica to exist in. The replica inherits the parent shares share_type and associated extra_specs. Scheduling of the replica is similar to that of the share.

- **The *ShareReplicationFilter* will ensure that the pool is within** the same `replication_domain` as the *active* replica and also ensures that the pool does not already have a replica for that share.

Drivers supporting *writable* style **must** set the *replica_state* attribute to *active* when the replica has been created and is `available`.

Deleting a replica

Users can remove replicas that have their *status* attribute set to `error`, `in_sync` or `out_of_sync`. They could even delete an *active* replica as long as there is another *active* replica (as could be the case with *writable* replication style). Before the `delete_replica` call is made to the driver, an update_access call is made to ensure access rules are safely removed for the replica.

Administrators may also `force-delete` replicas. Any driver exceptions will only be logged and not re-raised; the replica will be purged from manilas database.

Promoting a replica

Users can promote replicas that have their *replica_state* attribute set to `in_sync`. Administrators can attempt to promote replicas that have their *replica_state* attribute set to `out_of_sync` or `error`. During a promotion, if the driver raises an exception, all replicas will have their *status* attribute set to *error* and recovery from this state will require administrator intervention.

Resyncing a replica

Prior to a planned failover, an administrator could attempt to update the data on the replica. The `update_replica_state` call will be made during such an action, giving drivers an opportunity to push the latest updates from the *active* replica to the secondaries.

Creating a snapshot

When a user takes a snapshot of a share that has replicas, manila creates as many snapshot instances as there are share replicas. These snapshot instances all begin with their *status* attribute set to *creating*. The driver is expected to create the snapshot of the `active` replica and then begin to replicate this snapshot as soon as the *active* replicas snapshot instance is created and becomes `available`.

Deleting a snapshot

When a user deletes a snapshot, the snapshot instances corresponding to each replica of the share have their `status` attribute set to `deleting`. Drivers must update their secondaries as soon as the *active* replicas snapshot instance is deleted.

Driver Interfaces

As part of the `_update_share_stats()` call, the base driver reports the `replication_domain` capability. Drivers are expected to update the *replication_type* capability.

Drivers must implement the methods enumerated below in order to support replication. `promote_replica`, `update_replica_state` and `update_replicated_snapshot` need not be implemented by drivers that support the *writable* style of replication. The snapshot methods `create_replicated_snapshot`, `delete_replicated_snapshot` and `update_replicated_snapshot` need not be implemented by a driver that does not support snapshots.

Each driver request is made on a specific host. Create/delete operations on secondary replicas are always made on the destination host. Create/delete operations on snapshots are always made on the *active* replicas host. `update_replica_state` and `update_replicated_snapshot` calls are made on the host that the replica or snapshot resides on.

Share Replica interfaces:

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)

Class defines interface of NAS driver.

create_replica(*context*, *replica_list*, *new_replica*, *access_rules*, *replica_snapshots*, *share_server=None*)

Replicate the active replica to a new replica on this backend.

Note: This call is made on the host that the new replica is being created upon.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share. This list also contains the replica to be created. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **new_replica** The share replica dictionary.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'creating',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'out_of_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
```

(continues on next page)

(continued from previous page)

```
    'share_server_id': 'e6155221-ea00-49ef-abf9-9f89b7dd900a',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules. These are rules that other instances of the share already obey. Drivers are expected to apply access rules to the new replica or disregard access rules that dont apply.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica that will need to exist on the new share replica that is being created. The driver needs to ensure that this snapshot instance is truly available before transitioning the replica from out_of_sync to in_sync. Snapshots instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the `update_replicated_snapshot` method.

Example:

```
[
 {
 'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
    },
 'share_replica_snapshot': {
    'id': '',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'status': 'available',
```

(continues on next page)

(continued from previous page)

```
    'provider_location': None,
        ...
    },
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica being created.

Returns None or a dictionary. The dictionary can contain export_locations replica_state and access_rules_status. export_locations is a list of paths and replica_state is one of active, in_sync, out_of_sync or error.

Important: A backend supporting writable type replication should return active as the replica_state.

Export locations should be in the same format as returned during the `create_share` call.

Example:

```
{
    'export_locations': [
        {
            'path': '172.16.20.22/sample/export/path',
            'is_admin_only': False,
            'metadata': {'some_key': 'some_value'},
        },
    ],
     'replica_state': 'in_sync',
     'access_rules_status': 'in_sync',
}
```

delete_replica(*context*, *replica_list*, *replica_snapshots*, *replica*, *share_server=None*)

Delete a replica.

Note: This call is made on the host that hosts the replica being deleted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be deleted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the share replica being deleted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'out_of_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '53099868-65f1-11e5-9d70-feff819cdc9f',
    'share_server': <models.ShareServer> or None,
```

(continues on next page)

(continued from previous page)

```
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. The dict contains snapshot instances that are associated with the share replica being deleted. No model updates to snapshot instances are possible in this method. The driver should return when the cleanup is completed on the backend for both, the snapshots and the replica itself. Drivers must handle situations where the snapshot may not yet have finished creating on this replica.

Example:

```
[
    {
    'id': '89dafd00-0999-4d23-8614-13eaa6b02a3b',
    'snapshot_id': '3ce1caf7-0945-45fd-a320-714973e949d3',
    'status: 'available',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'share_instance_id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f'
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be deleted.

Returns None.

Raises Exception. Any exception raised will set the share replicas status and replica_state attributes to error_deleting. It will not affect snapshots belonging to this replica.

promote_replica(*context*, *replica_list*, *replica*, *access_rules*, *share_server=None*)

Promote a replica to active replica state.

Note: This call is made on the host that hosts the replica being promoted.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be promoted. The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica to be promoted.

Example:

```
{
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS2',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'f6e146d0-65f0-11e5-9d70-feff819cdc9f',
    'export_locations': [
        models.ShareInstanceExportLocations
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
```

(continues on next page)

(continued from previous page)

```
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **share_server** <models.ShareServer> or None Share server of the replica to be promoted.

Returns updated_replica_list or None. The driver can return the updated list as in the request parameter. Changes that will be updated to the Database are: export_locations, access_rules_status and replica_state.

Raises Exception. This can be any exception derived from BaseException. This is re-raised by the manager after some necessary cleanup. If the driver raises an exception during promotion, it is assumed that all of the replicas of the share are in an inconsistent state. Recovery is only possible through the periodic update call and/or administrator intervention to correct the status of the affected replicas if they become healthy again.

update_replica_state(*context*, *replica_list*, *replica*, *access_rules*, *replica_snapshots*, *share_server=None*)

Update the replica_state of a replica.

Note: This call is made on the host which hosts the replica being updated.

Drivers should fix replication relationships that were broken if possible inside this method.

This method is called periodically by the share manager; and whenever requested by the administrator through the resync API.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share This list also contains the replica to be updated. The active replica will have its replica_state

attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': 'e82ff8b6-65f0-11e5-9d70-feff819cdc9f',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '07574742-67ea-4dfd-9844-9fbd8ada3d87',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica** Dictionary of the replica being updated Replica state will always be in_sync, out_of_sync, or error. Replicas in active state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
```

(continues on next page)

(continued from previous page)

```
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
}
```

Parameters **access_rules** A list of access rules These access rules are obeyed by other instances of the share. The driver could attempt to sync on any unapplied access_rules.

Example:

```
[
 {
    'id': 'f0875f6f-766b-4865-8b41-cccb4cdf1676',
    'deleted' = False,
    'share_id' = 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'access_type' = 'ip',
    'access_to' = '172.16.20.1',
    'access_level' = 'rw',
 }
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. This includes snapshot instances of every snapshot of the share whose aggregate_status property was reported to be available when the share manager initiated this request. Each list member will have two sub dictionaries: active_replica_snapshot and share_replica_snapshot. The active replica snapshot corresponds to the instance of the snapshot on any of the active replicas of the share while share_replica_snapshot corresponds to the snapshot instance for the specific replica being updated. The driver needs to ensure that this snapshot instance is truly available before transitioning from out_of_sync to in_sync. Snapshots instances for snapshots that have an aggregate_status of creating or deleting will be polled for in the update_replicated_snapshot method.

Example:

```
 [
  {
'active_replica_snapshot': {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'share_instance_id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'status': 'available',
    'provider_location': '/newton/share-snapshot-10e49c3e-aca9',
    ...
   },
 'share_replica_snapshot': {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
```

(continues on next page)

(continued from previous page)

```
    'status': 'creating',
    'provider_location': None,
        ...
    },
  }
]
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_state: a str value denoting the replica_state. Valid values are in_sync and out_of_sync or None (to leave the current replica_state unchanged).

Replicated Snapshot interfaces:

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)

Class defines interface of NAS driver.

create_replicated_snapshot(*context*, *replica_list*, *replica_snapshots*, *share_server=None*)

Create a snapshot on active instance and update across the replicas.

Note: This call is made on the active replicas host. Drivers are expected to transfer the snapshot created to the respective replicas.

The driver is expected to return model updates to the share manager. If it was able to confirm the creation of any number of the snapshot instances passed in this interface, it can set their status to available as a cue for the share manager to set the progress attr to 100%.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
```

(continues on next page)

(continued from previous page)

```
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to creating.

Example:

```
[
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'creating',
    'progress': '0%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being created.

Raises Exception. Any exception in this method will set all instances to error.

delete_replicated_snapshot(*context*, *replica_list*, *replica_snapshots*, *share_server=None*)

Delete a snapshot by deleting its instances across the replicas.

Note: This call is made on the active replicas host, since drivers may not be able to delete the snapshot from an individual replica.

The driver is expected to return model updates to the share manager. If it was able to confirm the removal of any number of the snapshot instances passed in this interface, it can set their status to deleted as a cue for the share manager to clean up that instance from the database.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'in_sync',
        ...
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
    'share_server': <models.ShareServer> or None,
    },
    {
    'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'replica_state': 'active',
        ...
    'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
    'share_server': <models.ShareServer> or None,
    },
    ...
]
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. All the instances will have their status attribute set to deleting.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status': 'deleting',
    'progress': '100%',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'status: 'deleting',
    'progress': '100%',
        ...
    },
    ...
]
```

Parameters **share_server** <models.ShareServer> or None

Returns List of dictionaries of snapshot instances. The dictionaries can contain values that need to be updated on the database for the snapshot instances being deleted. To confirm the deletion of the snapshot instance, set the status attribute of the instance to deleted (constants.STATUS_DELETED)

Raises Exception. Any exception in this method will set the status attribute of all snapshot instances to error_deleting.

update_replicated_snapshot(*context*, *replica_list*, *share_replica*, *replica_snapshots*, *replica_snapshot*, *share_server=None*)

Update the status of a snapshot instance that lives on a replica.

Note: For DR and Readable styles of replication, this call is made on the replicas host and not the active replicas host.

This method is called periodically by the share manager. It will query for snapshot instances that track the parent snapshot across non-active replicas. Drivers can expect the status of the instance to be creating or deleting. If the driver sees that a snapshot instance has been removed from the replicas backend and the instance status was set to deleting, it is expected to raise a SnapshotResourceNotFound exception. All other exceptions will set the snapshot instance status to error. If the instance was not in deleting state, raising a SnapshotResourceNotFound will set the instance status to error.

Parameters

- **context** Current context
- **replica_list** List of all replicas for a particular share The active replica will have its replica_state attr set to active.

Example:

```
[
    {
     'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'in_sync',
     ...
     'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
     'share_server': <models.ShareServer> or None,
    },
    {
     'id': '10e49c3e-aca9-483b-8c2d-1c337b38d6af',
     'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
     'replica_state': 'active',
     ...
     'share_server_id': 'f63629b3-e126-4448-bec2-03f788f76094',
     'share_server': <models.ShareServer> or None,
    },
     ...
]
```

Parameters **share_replica** Share replica dictionary. This replica is associated

with the snapshot instance whose status is being updated. Replicas in active replica_state will not be passed via this parameter.

Example:

```
{
    'id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_id': 'f0e4bb5e-65f0-11e5-9d70-feff819cdc9f',
    'deleted': False,
    'host': 'openstack2@cmodeSSVMNFS1',
    'status': 'available',
    'scheduled_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'launched_at': datetime.datetime(2015, 8, 10, 0, 5, 58),
    'terminated_at': None,
    'replica_state': 'in_sync',
    'availability_zone_id': 'e2c2db5c-cb2f-4697-9966-c06fb200cb80',
    'export_locations': [
        models.ShareInstanceExportLocations,
    ],
    'access_rules_status': 'in_sync',
    'share_network_id': '4ccd5318-65f1-11e5-9d70-feff819cdc9f',
    'share_server_id': '4ce78e7b-0ef6-4730-ac2a-fd2defefbd05',
}
```

Parameters **replica_snapshots** List of dictionaries of snapshot instances. These snapshot instances track the snapshot across the replicas. This will include the snapshot instance being updated as well.

Example:

```
 [
    {
    'id': 'd3931a93-3984-421e-a9e7-d9f71895450a',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    {
    'id': '8bda791c-7bb6-4e7b-9b64-fefff85ff13e',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
        ...
    },
    ...
]
```

Parameters **replica_snapshot** Dictionary of the snapshot instance. This is the instance to be updated. It will be in creating or deleting state when sent via this parameter.

Example:

```
{
    'name': 'share-snapshot-18825630-574f-4912-93bb-af4611ef35a2',
    'share_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'share_name': 'share-d487b88d-e428-4230-a465-a800c2cce5f8',
    'status': 'creating',
    'id': '18825630-574f-4912-93bb-af4611ef35a2',
    'deleted': False,
    'created_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share': <models.ShareInstance>,
    'updated_at': datetime.datetime(2016, 8, 3, 0, 5, 58),
    'share_instance_id': 'd487b88d-e428-4230-a465-a800c2cce5f8',
    'snapshot_id': '13ee5cb5-fc53-4539-9431-d983b56c5c40',
    'progress': '0%',
    'deleted_at': None,
    'provider_location': None,
}
```

Parameters **share_server** <models.ShareServer> or None

Returns replica_snapshot_model_update: a dictionary. The dictionary must contain values that need to be updated on the database for the snapshot instance that represents the snapshot on the replica.

Raises exception.SnapshotResourceNotFound Raise this exception for snapshots that are not found on the backend and their status was deleting.

Configure and use driver filter and weighing for scheduler

OpenStack manila enables you to choose a share back end based on back-end specific properties by using the DriverFilter and GoodnessWeigher for the scheduler. The driver filter and weigher scheduling can help ensure that the scheduler chooses the best back end based on requested share properties as well as various back-end specific properties.

What is driver filter and weigher and when to use it

The driver filter and weigher give you the ability to more finely control how manila scheduler chooses the best back end to use when handling a share provisioning request. One example scenario where using the driver filter and weigher can be if a back end that utilizes thin-provisioning is used. The default filters use the `free capacity` property to determine the best back end, but that is not always perfect. If a back end has the ability to provide a more accurate back-end specific value, it can be used as part of the weighing process to find the best possible host for a new share. Some more examples of the use of these filters could be with respect to back end specific limitations. For example, some back ends may be limited by the number of shares that can be created on them, or by the minimum or maximum size allowed per share or by the fact that provisioning beyond a particular capacity affects their performance. The driver filter and weigher can provide a way for these limits to be accounted for during scheduling.

Defining your own filter and goodness functions

You can define your own filter and goodness functions through the use of various capabilities that manila exposes. Capabilities exposed include information about the share request being made, `share_type` settings, and back-end specific information about drivers. All of these allow for a lot of control over how the ideal back end for a share request will be decided.

The `filter_function` option is a string defining a function that will determine whether a back end should be considered as a potential candidate by the scheduler.

The `goodness_function` option is a string defining a function that will rate the quality of the potential host (0 to 100, 0 lowest, 100 highest).

Important: The driver filter and weigher will use default values for filter and goodness functions for each back end if you do not define them yourself. If complete control is desired then a filter and goodness function should be defined for each of the back ends in the `manila.conf` file.

Supported operations in filter and goodness functions

Below is a table of all the operations currently usable in custom filter and goodness functions created by you:

Operations	Type
+, -, *, /, ^	standard math
not, and, or, &, |, !	logic
>, >=, <, <=, ==, <>, !=	equality
+, -	sign
x ? a : b	ternary
abs(x), max(x, y), min(x, y)	math helper functions

Caution: Syntax errors in filter or goodness strings are thrown at a share creation time.

Available capabilities when creating custom functions

There are various properties that can be used in either the `filter_function` or the `goodness_function` strings. The properties allow access to share info, qos settings, extra specs, and so on.

The following capabilities are currently available for use:

Host capabilities for a back end

host The hosts name

share_backend_name The share back end name

vendor_name The vendor name

driver_version The driver version

storage_protocol The storage protocol

qos Boolean signifying whether QoS is supported

total_capacity_gb The total capacity in gibibytes

allocated_capacity_gb The allocated capacity in gibibytes

free_capacity_gb The free capacity in gibibytes

reserved_percentage The reserved storage percentage

driver_handles_share_server The driver mode used by this host

thin_provisioning Whether or not thin provisioning is supported by this host

updated Last time this hosts stats were updated

dedupe Whether or not dedupe is supported by this host

compression Whether or not compression is supported by this host

snapshot_support Whether or not snapshots are supported by this host

replication_domain The replication domain of this host

replication_type The replication type supported by this host

provisioned_capacity_gb The provisioned capacity of this host in gibibytes

pools This hosts storage pools

max_over_subscription_ratio This hostss over subscription ratio for thin provisioning

Capabilities specific to a back end

These capabilities are determined by the specific back end you are creating filter and goodness functions for. Some back ends may not have any capabilities available here.

Requested share capabilities

availability_zone_id ID of the availability zone of this share

share_network_id ID of the share network used by this share

share_server_id ID of the share server of this share

host Host name of this share

is_public Whether or not this share is public

snapshot_support Whether or not snapshots are supported by this share

status Status for the requested share

share_type_id The share type ID

share_id The share ID

user_id The shares user ID

project_id The shares project ID

id The share instance ID

replica_state The shares replication state

replication_type The replication type supported by this share

snapshot_id The ID of the snapshot of which this share was created from

size The size of the share in gibibytes

share_proto The protocol of this share

metadata General share metadata

The most used capability from this list will most likely be the `size`.

Extra specs for the requested share type

View the available properties for share types by running:

```
$ manila extra-specs-list
```

Driver filter and weigher usage examples

Below are examples for using the filter and weigher separately, together, and using driver-specific properties.

Example `manila.conf` file configuration for customizing the filter function:

```
[default]
enabled_backends = generic1, generic2

[generic1]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC1
filter_function = "share.size < 10"

[generic2]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC2
filter_function = "share.size >= 10"
```

The above example will filter share to different back ends depending on the size of the requested share. Shares with a size less than 10 GB are sent to generic1 and shares with a size greater than or equal to 10 GB are sent to generic2.

Example `manila.conf` file configuration for customizing the goodness function:

```
[default]
enabled_backends = generic1, generic2

[generic1]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC1
goodness_function = "(share.size < 5) ? 100 : 50"

[generic2]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC2
goodness_function = "(share.size >= 5) ? 100 : 25"
```

The above example will determine the goodness rating of a back end based on the requested shares size. The example shows how the ternary if statement can be used in a filter or goodness function. If a requested share is of size 10 GB then generic1 is rated as 50 and generic2 is rated as 100. In this case generic2 wins. If a requested share is of size 3 GB then generic1 is rated 100 and generic2 is rated 25. In this case generic1 would win.

Example `manila.conf` file configuration for customizing both the filter and goodness functions:

```
[default]
enabled_backends = generic1, generic2

[generic1]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC1
filter_function = "stats.total_capacity_gb < 500"
goodness_function = "(share.size < 25) ? 100 : 50"

[generic2]
share_driver = manila.share.drivers.generic.GenericShareDriver
share_backend_name = GENERIC2
filter_function = "stats.total_capacity_gb >= 500"
goodness_function = "(share.size >= 25) ? 100 : 75"
```

The above example combines the techniques from the first two examples. The best back end is now decided based on the total capacity of the back end and the requested shares size.

Example `manila.conf` file configuration for accessing driver specific properties:

```
[default]
enabled_backends = example1, example2, example3

[example1]
share_driver = manila.share.drivers.example.ExampleShareDriver
share_backend_name = EXAMPLE1
filter_function = "share.size < 5"
goodness_function = "(capabilities.provisioned_capacity_gb < 30) ? 100 : 50"

[example2]
```

(continues on next page)

(continued from previous page)

```
share_driver = manila.share.drivers.example.ExampleShareDriver
share_backend_name = EXAMPLE2
filter_function = "shares.size < 5"
goodness_function = "(capabilities.provisioned_capacity_gb < 80) ? 100 : 50"

[example3]
share_driver = manila.share.drivers.example.ExampleShareDriver
share_backend_name = EXAMPLE3
goodness_function = "55"
```

The above is an example of how back-end specific capabilities can be used in the filter and goodness functions. In this example, the driver has a `provisioned_capacity_gb` capability that is being used to determine which back end gets used during a share request. In the above example, `example1` and `example2` will handle share requests for all shares with a size less than 5 GB. `example1` will have priority until the provisioned capacity of all shares on it hits 30 GB. After that, `example2` will have priority until the provisioned capacity of all shares on it hits 80 GB. `example3` will collect all shares greater or equal to 5 GB as well as all shares once `example1` and `example2` lose priority.

Share Migration

As of the Ocata release of OpenStack, *manila* supports migration of shares across different pools through an experimental API. Since it was first introduced, several enhancements have been made through the subsequent releases while still in experimental state. This developer document reflects the latest version of the experimental Share Migration API.

Feature definition

The Share Migration API is an administrator-only experimental API that allows the invoker to select a destination pool to migrate a share to, while still allowing clients to access the source share instance during the migration. Migration of data is generally expected to be disruptive for users accessing the source, because at some point it will cease to exist. For this reason, the share migration feature is implemented in a 2-phase approach, for the purpose of controlling the timing of that expected disruption of migrating shares.

The first phase of migration is when operations that take the longest are performed, such as data copying or replication. After the first pass of data copying is complete, it is up to the administrator to trigger the second phase, often referred to as switchover phase, which may perform operations such as a last sync and deleting the source share instance.

During the data copy phase, users remain connected to the source, and may have to reconnect after the switchover phase. In order to migrate a share, manila may employ one of two mechanisms which provide different capabilities and affect how the disruption occurs with regards to user access during data copy phase and disconnection during switchover phase. Those two mechanisms are:

driver-assisted migration This mechanism uses the underlying driver running in the manila-share service node to coordinate the migration. The migration itself is performed directly on the storage. In order for this mechanism to be used, it requires the driver to implement this functionality, while also requiring that the driver which manages the destination pool is compatible with driver-assisted migration. Typically, drivers would be able to assist migration of shares within storage systems from the same vendor. It is likely that this will be the most efficient and reliable mechanism to migrate a

given share, as the storage back end may be able to migrate the share while remaining writable, preserving all file system metadata, snapshots, and possibly perform this operation non-disruptively. When this mechanism cannot be used, the host-assisted migration will be attempted.

host-assisted migration This mechanism uses the Data Service (manila-data) to copy the source shares data to a new destination share created in the given destination pool. For this mechanism to work, it is required that the Data Service is properly configured in the cloud environment and the migration operation for the source shares protocol and access rule type combination is supported by the Data Service. This is the most suited mechanism to migrate shares when the two pools are from different storage vendors. Given that this mechanism is a rough copy of files and the back ends are unaware that their share contents are being copied over, the optimizations found in the driver-assisted migration are not present here, thus the source share remains read-only, snapshots cannot be transferred, some file system metadata such as permissions and ownership may be lost, and users are expected to be disconnected by the end of migration.

Note that during a share migration, access rules cannot be added or removed.

As of Ocata release, this feature allows several concurrent migrations (driver-assisted or host-assisted) to be performed, having a best-effort type of scalability.

API description

The migration of a share is started by invoking the `migration_start` API. The parameters are:

share The share to be migrated. This parameter is mandatory.

destination The destination pool in `host@backend#pool` representation. This parameter is mandatory.

force_host_assisted_migration Forces the host-assisted mechanism to be used, thus using the Data Service to copy data across back ends. This parameter value defaults to *False*. When set to *True*, it skips the driver-assisted approach which would otherwise be attempted first. This parameter is optional.

preserve_metadata Specifies whether migration should enforce the preservation of all file system metadata. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of ensuring preservation of file system metadata, migration will result in an error status. As of Ocata release, host-assisted migration cannot provide any guarantees of preserving file system metadata. This parameter is mandatory.

preserve_snapshots Specifies whether migration should enforce the preservation of all existing snapshots at the destination. In other words, the existing snapshots must be migrated along with the share data. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of migrating the snapshots, migration will result in an error status. As of Ocata release, host-assisted migration cannot provide this capability. This parameter is mandatory.

nondisruptive Specifies whether migration should only be performed without disrupting clients during migration. For such, it is also expected that the export location does not change. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of allowing the share to remain accessible through the two phases of the migration, migration will result in an error status. As of Ocata release, host-assisted migration cannot provide this capability. This parameter is mandatory.

writable Specifies whether migration should only be performed if the share can remain writable. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of allowing the share to remain writable, migration will result in an error status. If drivers are not capable

of performing a nondisruptive migration, manila will ensure that the share will remain writable through the data copy phase of migration. However, during the switchover phase the share will be re-exported at the destination, causing the share to be rendered inaccessible for the duration of this phase. As of Ocata release, host-assisted migration cannot provide this capability. This parameter is mandatory.

new_share_type If willing to retype the share so it can be allocated in the desired destination pool, the invoker may supply a new share type to be used. This is often suited when the share is to be migrated to a pool which operates in the opposite driver mode. This parameter is optional.

new_share_network If willing to change the shares share-network so it can be allocated in the desired destination pool, the invoker may supply a new share network to be used. This is often suited when the share is to be migrated to a pool which operates in a different availability zone or managed by a driver that handles share servers. This parameter is optional.

After started, a migration may be cancelled through the `migration_cancel` API, have its status obtained through the `migration_get_progress` API, and completed through the `migration_complete` API after reaching a certain state (see `Workflows` section below).

Workflows

Upon invoking `migration_start`, several validations will be performed by the API layer, such as:

- If supplied API parameters are valid.
- If the share does not have replicas.
- If the share is not member of a share group.
- If the access rules of the given share are not in error status.
- If the driver-assisted parameters specified do not conflict with *force_host_assisted_migration* parameter.
- If *force_host_assisted_migration* parameter is set to True while snapshots do not exist.
- If share status is *available* and is not busy with other tasks.
- If the destination pool chosen to migrate the share to exists and is running.
- If share service or Data Service responsible for performing the migration exists and is running.
- If the combination of share network and share type resulting is compatible with regards to driver modes.

If any of the above validations fail, the API will return an error. Otherwise, the *task_state* field value will transition to *migration_starting* and the shares status will transition to *migrating*. Past this point, all validations, state transitions and errors will not produce any notifications to the user. Instead, the given shares *task_state* field value will transition to *migration_error*.

Following API validation, the scheduler will validate if the supplied destination is compatible with the desired share type according to the pools capabilities. If this validation fails, the *task_state* field value will transition to *migration_error*.

The scheduler then invokes the source share pools manager to proceed with the migration, transitioning the *task_state* field value to *migration_in_progress*. If *force-host-assisted-migration* API parameter is not set, then a driver-assisted migration will be attempted first.

Note that whichever mechanism is employed, there will be a new share instance created in the database, referred to as the destination instance, with a status field value *migrating_to*. This share instance will not have its export location displayed during migration and will prevail instead of the original instance database entry when migration is complete.

Driver-assisted migration data copy phase

A share server will be created as needed at the destination pool. Then, the share server details are provided to the driver to report the set of migration capabilities for this destination. If the API parameters *writable*, *nondisruptive*, *preserve_metadata* and *preserve_snapshots* are satisfied by the reported migration capabilities, the *task_state* field value transitions to *migration_driver_starting* and the driver is invoked to start the migration.

The drivers migration_start method should start a job in the storage back end and return, allowing the *task_state* field value to transition to *migration_driver_in_progress*. If any of the API parameters described previously are not satisfied, or the driver raises an exception in *migration_start*, the driver-assisted migration ends setting the *task_state* field value to *migration_error*, all allocated resources will be cleaned up and migration will proceed to the host-assisted migration mechanism.

Once the *migration_start* driver method succeeds, a periodic task that checks for shares with *task_state* field value *migration_driver_in_progress* will invoke the drivers *migration_continue* method, responsible for executing the next steps of migration until the data copy phase is completed, transitioning the *task_state* field value to *migration_driver_phase1_done*. If this step fails, the *task_state* field value transitions to *migration_error* and all allocated resources will be cleaned up.

Host-assisted migration data copy phase

A new share will be created at the destination pool and the source shares access rules will be changed to read-only. The *task_state* field value transitions to *data_copying_starting* and the Data Service is then invoked to mount both shares and copy data from the source to the destination.

In order for the Data Service to mount the shares, it will ask the storage driver to allow access to the node where the Data Service is running. It will then attempt to mount the shares via their respective administrator-only export locations that are served in the administrator network when available, otherwise the regular export locations will be used.

In order for the access and mount procedures to succeed, the administrator-only export location must be reachable from the Data Service and the access parameter properly configured in the Data Service configuration file. For instance, a NFS share should require an IP configuration, whereas a CIFS share should require a username credential. Those parameters should be previously set in the Data Service configuration file by the administrator.

The data copy routine runs commands as root user for the purpose of setting the correct file metadata to the newly created files at the destination share. It can optionally verify the integrity of all files copied through a configuration parameter. Once copy is completed, the shares are unmounted, their access from the Data Service are removed and the *task_state* field value transitions to *data_copying_completed*, allowing the switchover phase to be invoked.

Share migration switchover phase

When invoked, the *task_state* field value transitions to *migration_completing*. Whichever migration mechanism is used, the source share instance is deleted and the access rules are applied to the destination share instance. In the driver-assisted migration, the driver is first invoked to perform a final sync.

The last step is to update the share models optional capability fields, such as *create_share_from_snapshot_support*, *revert_to_snapshot_support* and *mount_snapshot_support*, according to the *new_share_type*, if it had been specified when the migration was initiated.

At last, the *task_state* field value transitions to *migration_success*. If the *nondisruptive* driver-assisted capability is not supported or the host-assisted migration mechanism is used, the export location will change and clients will need to remount the share.

Driver interfaces

All drivers that implement the driver-assisted migration mechanism should be able to perform all required steps from the source share instance back end within the implementation of the interfaces listed in the section below. Those steps include:

- Validating compatibility and connectivity between the source and destination back end;
- Start the migration job in the storage back end. Return after the job request has been submitted;
- Subsequent invocations to the driver to monitor the job status, cancel it and obtain its progress in percentage value;
- Complete migration by performing a last sync if necessary and delete the original share from the source back end.

For host-assisted migration, drivers may override some methods defined in the base class in case it is necessary to support it.

Additional notes

- In case of an error in the storage back end during the execution of the migration job, the driver should raise an exception within the `migration_continue` method.
- If the manila-share service is restarted during a migration, in case it is a driver-assisted migration, the drivers `migration_continue` will be invoked continuously with an interval configured in the share manager service (`migration_driver_continue_interval`). The invocation will stop when the driver finishes the data copy phase. In case of host-assisted migration, the migration job is disrupted only if the manila-data service is restarted. In such event, the migration has to be restarted from the beginning.
- To be compatible with host-assisted migration, drivers must also support the `update_access` interface, along with its *recovery mode* mechanism.

Share Migration driver-assisted interfaces:

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)
Class defines interface of NAS driver.

migration_cancel(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)
Cancels migration of a given share to another host.

Note: Is called in source shares backend to cancel migration.

If possible, driver can implement a way to cancel an in-progress migration.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

migration_check_compatibility(*context*, *source_share*, *destination_share*, *share_server=None*, *destination_share_server=None*)
Checks destination compatibility for migration of a given share.

Note: Is called to test compatibility with destination backend.

Driver should check if it is compatible with destination backend so driver-assisted migration can proceed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the share to be migrated.
- **destination_share** Reference to the share model to be used by migrated share.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

A dictionary containing values indicating if destination backend is compatible, if share can remain writable during migration, if it can preserve all file metadata and if it can perform migration of given share non-disruptively.

Example:

```
{
    'compatible': True,
    'writable': True,
    'preserve_metadata': True,
    'nondisruptive': True,
    'preserve_snapshots': True,
}
```

migration_complete(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Completes migration of a given share to another host.

Note: Is called in source shares backend to complete migration.

If driver is implementing 2-phase migration, this method should perform the disruptive tasks related to the 2nd phase of migration, thus completing it. Driver should also delete all original share data from source backend.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns

If the migration changes the share export locations, snapshot provider locations or snapshot export locations, this method should return a dictionary with the relevant info. In such case, a dictionary containing a list of export locations and a list of model updates for each snapshot indexed by their IDs.

Example:

```
{
    'export_locations':
    [
        {
```

(continues on next page)

(continued from previous page)

```
            'path': '1.2.3.4:/foo',
            'metadata': {},
            'is_admin_only': False
            },
            {
            'path': '5.6.7.8:/foo',
            'metadata': {},
            'is_admin_only': True
            },
        ],
        'snapshot_updates':
        {
            'bc4e3b28-0832-4168-b688-67fdc3e9d408':
            {
            'provider_location': '/snapshots/foo/bar_1',
            'export_locations':
            [
                {
                'path': '1.2.3.4:/snapshots/foo/bar_1',
                'is_admin_only': False,
                },
                {
                'path': '5.6.7.8:/snapshots/foo/bar_1',
                'is_admin_only': True,
                },
            ],
            },
            '2e62b7ea-4e30-445f-bc05-fd523ca62941':
            {
            'provider_location': '/snapshots/foo/bar_2',
            'export_locations':
            [
                {
                'path': '1.2.3.4:/snapshots/foo/bar_2',
                'is_admin_only': False,
                },
                {
                'path': '5.6.7.8:/snapshots/foo/bar_2',
                'is_admin_only': True,
                },
            ],
            },
        },
}
```

migration_continue(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Continues migration of a given share to another host.

Note: Is called in source shares backend to continue migration.

Driver should implement this method to continue monitor the migration progress in storage and perform following steps until 1st phase is completed.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns Boolean value to indicate if 1st phase is finished.

migration_get_progress(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Obtains progress of migration of a given share to another host.

Note: Is called in source shares backend to obtain migration progress.

If possible, driver can implement a way to return migration progress information.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Returns A dictionary with at least total_progress field containing the percentage value.

migration_start(*context*, *source_share*, *destination_share*, *source_snapshots*, *snapshot_mappings*, *share_server=None*, *destination_share_server=None*)

Starts migration of a given share to another host.

Note: Is called in source shares backend to start migration.

Driver should implement this method if willing to perform migration in a driver-assisted way, useful for when source shares backend driver is compatible with destination backend driver. This method should start the migration procedure in the backend and end. Following steps should be done in migration_continue.

Parameters

- **context** The context.RequestContext object for the request.
- **source_share** Reference to the original share model.
- **destination_share** Reference to the share model to be used by migrated share.
- **source_snapshots** List of snapshots owned by the source share.
- **snapshot_mappings** Mapping of source snapshot IDs to destination snapshot models.
- **share_server** Share server model or None.
- **destination_share_server** Destination Share server model or None.

Share Migration host-assisted interfaces:

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)

Class defines interface of NAS driver.

connection_get_info(*context*, *share*, *share_server=None*)

Is called to provide necessary generic migration logic.

Parameters

- **context** The context.RequestContext object for the request.
- **share** Reference to the share being migrated.
- **share_server** Share server model or None.

Returns A dictionary with migration information.

Share Server Migration

As of the Victoria release of OpenStack, Manila supports migration of share servers across different pools through an experimental API. This developer document reflects the latest version of the experimental Share Server Migration API.

Feature definition

The Share Server Migration API is an administrator-only API that allows the invoker to select a destination backend to migrate a share server to, while still allowing clients to access the source share server resources during the migration. Migration of data is expected to be disruptive for users accessing the source, because at some point it will cease to exist. For this reason, the share server migration feature is implemented in a 2-phase approach, for the purpose of controlling the timing of that expected disruption of migrating share servers.

The first phase of the migration is when operations that take the longest are performed, such as data copying or replication. After the first phase of data copying is complete, it is up to the administrator to trigger the second phase, often referred to as switchover phase, which may perform operations such as a last sync and changing the source share server to inactive.

During the data copy phase, users remain connected to the source, and may have to reconnect after the switchover phase.

Share server migration only supports driver-assisted migration. This mechanism uses the underlying driver running in the manila-share service node to coordinate the migration. The migration is performed directly in the storage. In order to use this mechanism, the driver should implement this functionality. Also, the driver managing the destination back end should support driver-assisted migration. Typically, drivers would be able to assist migration of share servers within storage systems from the same vendor. It is likely that this will be the most efficient and reliable mechanism to migrate a given share server, as the storage back end may be able to migrate the share server while remaining writable, snapshots, and possibly perform this operation non-disruptively.

Note that during a share server migration, access rules cannot be added or removed. Also, it is not possible to modify existent access rules for shares and share snapshots created upon the share server being migrated.

API description

The migration of a share server is started by invoking the `migration_start` API. The parameters are:

share_server_id The share server to be migrated. This parameter is mandatory.

destination The destination backend in `host@backend` representation. This parameter is mandatory.

preserve_snapshots Specifies whether migration should enforce the preservation of all existing snapshots at the destination. In other words, the existing snapshots must be migrated along with the share server data. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of migrating the snapshots, migration will result in an error status. This parameter is mandatory.

nondisruptive Specifies whether the migration should only be performed without disrupting clients during migration. For such, it is also expected that the export location does not change. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of allowing the share server shares to remain accessible through the two phases of the migration, migration will result in an error status. This parameter is mandatory.

writable Specifies whether migration should only be performed if the share server shares can remain writable. When this behavior is expected (i.e, this parameter is set to *True*) and drivers are not capable of allowing the share server shares to remain writable, migration will result in an error status. If drivers are not capable of performing a nondisruptive migration, manila will ensure that the share server shares will remain writable through the data copy phase of migration. However,

during the switchover phase the shares will be re-exported at the destination, causing the share to be rendered inaccessible for the duration of this phase. This parameter is mandatory.

new_share_network_id If willing to change the share servers share-network so it can be allocated in the desired destination backend, the invoker may supply a new share network to be used. This is often suited when the share server is to be migrated to a backend which operates in a different availability zone or managed by a driver that handles share servers. This parameter is optional.

After started, a migration may be cancelled through the `migration_cancel` API, have its status obtained through the `migration_get_progress` API, and completed through the `migration_complete` API after reaching a certain state (see `Workflows` section below).

Workflows

Upon invoking `migration_start`, several validations will be performed by the API layer, such as:

- If supplied API parameters are valid.
- If share server status is *active*.
- If there are share groups related to the share server.
- If a new share network id was provided and is compatible with the destination.
- If a new host and share network id were provided and theyre different from the source share server.
- If the share server to be migrated serves as destination to another share server.
- If all the availability zones match with all shares share types within the share server.
- If the share servers shares do not have replicas.
- If the share servers shares are not member of a share group.
- If the access rules of the given share servers shares are not in error status.
- If the snapshots of all share server shares are in *available* state.
- If the destination backend chosen to migrate the share server to exists, as well as it and its share service are running.

If any of the above validations fail, the API will return an error. Otherwise, the *task_state* field value will transition to *migration_starting* and the share servers status will transition to *server_migrating*. Past this point, all validations, state transitions and errors will not produce any notifications to the user. Instead, the given share servers *task_state* field value will transition to *migration_error*.

Right after the API validations, a driver call will be performed in the destination backend in order to validate if the destination host is compatible within the requested operation. The driver will then determine the compatibility between source and destination hosts for the share server migration.

A new share server will be created in the database, referred to as the destination share server, with a status field value *server_migrating_to*.

Share server migration data copy phase

A share server will be created as needed at the destination backend. Then, the share server details are provided to the driver to report the set of migration capabilities for this destination. If the API parameters *writable*, *nondisruptive*, *preserve_metadata* and *preserve_snapshots* are satisfied by the reported migration capabilities, the *task_state* field value transitions to *migration_driver_starting* and the driver is invoked to start the migration.

The drivers `share_server_migration_start` method should start a job in the storage back end and return, allowing the *task_state* field value to transition to *migration_driver_in_progress*. If any of the API parameters described previously are not satisfied, or the driver raises an exception in *share_server_migration_start*, the migration ends setting the *task_state* field value to *migration_error*, and the created share server will have its status set to error.

Once the `share_server_migration_start` driver method succeeds, a periodic task that checks for shares with *task_state* field value *migration_driver_in_progress* will invoke the drivers `share_server_migration_continue` method, responsible for executing the next steps of migration until the data copy phase is completed, transitioning the *task_state* field value to *migration_driver_phase1_done*. If this step fails, the *task_state* field value transitions to *migration_error* and all allocated resources will be cleaned up.

Share server migration switchover phase

When invoked, the *task_state* field value transitions to *migration_completing*. In this phase, these operations will happen: * The source share instances are deleted * The source share server will have its status set to inactive * The access rules are applied to the shares of the destination share server * A final sync is also performed.

At last, the *task_state* field value transitions to *migration_success*. If the *nondisruptive* capability is not supported, the export locations will change and clients will need to remount the shares.

Driver interfaces

All drivers that implement the migration mechanism should be able to perform all required steps from the source share server back end within the implementation of the interfaces listed in the section below. Those steps include:

- Validating compatibility and connectivity between the source and destination back end;
- Start the migration job in the storage back end. Return after the job request has been submitted;
- Subsequent invocations to the driver to monitor the job status.
- Complete migration by performing a last sync if necessary and delete the original shares from the source back end.

Note: The implementation of the `share_server_migration_cancel` and `share_server_migration_get_progress` operations is not mandatory. If the driver is able to perform such operations, make sure to set `share_server_migration_cancel` and `share_server_migration_get_progress` equal to `True` in the response of the `share_server_migration_check` operation.

Additional notes

- In case of an error in the storage back end during the execution of the migration job, the driver should raise an exception within the `share_server_migration_continue` method.
- If the manila-share service is restarted during a migration, the drivers `share_server_migration_continue` will be invoked periodically with an interval configured in the share manager service (`share_server_migration_driver_continue_interval`). The invocation will stop when the driver finishes the data copy phase.

Share Server Migration interfaces:

class ShareDriver(*driver_handles_share_servers*, **args*, ***kwargs*)

Class defines interface of NAS driver.

share_server_migration_cancel(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Cancels migration of a given share server to another host.

Note: Is called in destination share servers backend to continue migration.

If possible, driver can implement a way to cancel an in-progress migration.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

share_server_migration_check_compatibility(*context*, *share_server*, *dest_host*, *old_share_network*, *new_share_network*, *shares_request_spec*)

Checks destination compatibility for migration of a share server.

Note: Is called in destination share servers backend to continue migration. Can be called by an admin to check if a given host is compatible or by the share manager to test compatibility with destination backend.

Driver should check if it is compatible with destination backend so driver-assisted migration can proceed.

Parameters

- **context** The context.RequestContext object for the request.
- **share_server** Share server model.

- **dest_host** Reference to the hos to be used by the migrated share server.
- **old_share_network** Share network model where the source share server is placed.
- **new_share_network** Share network model where the share server is going to be migrated to.
- **shares_request_spec** Dict. Contains information about all shares and share types that belong to the source share server. The drivers can use this information to check if the capabilities match with the destination backend and if there is available space to hold the new share server and all its resource.

Example:

```
{
'shares_size': 100,
'snapshots_size': 100,
'shares_req_spec':
[
    {
    'share_properties':
        {
        'size': 10
        'user_id': '2f5c1df4-5203-444e-b68e-1e60f3f26fc3'
        'project_id': '0b82b278-51d6-4357-b273-0d7263982c31'
        'snapshot_support': True
        'create_share_from_snapshot_support': True
        'revert_to_snapshot_support': False
        'mount_snapshot_support': False
        'share_proto': NFS
        'share_type_id': '360e01c1-a4f7-4782-9676-dc013f1a2f21'
        'is_public': False
        'share_group_id': None
        'source_share_group_snapshot_member_id': None
        'snapshot_id': None
        },
    'share_instance_properties':
        {
        'availability_zone_id':
            '02377ad7-381c-4b25-a04c-6fd218f22a91',
        'share_network_id': '691544aa-da83-4669-8522-22719f236e16',
        'share_server_id': 'cd658413-d02c-4d1b-ac8a-b6b972e76bac',
        'share_id': 'e42fec45-781e-4dcc-a4d2-44354ad5ae91',
        'host': 'hostA@backend1#pool0',
        'status': 'available',
        },
    'share_type':
        {
        'id': '360e01c1-a4f7-4782-9676-dc013f1a2f21',
        'name': 'dhss_false',
        'is_public': False,
        'extra_specs':
```

(continues on next page)

(continued from previous page)

```
                {
                'driver_handles_share_servers': False,
                }
            },
        'share_id': e42fec45-781e-4dcc-a4d2-44354ad5ae91,
        },
],
}
```

Returns

A dictionary containing values indicating if destination backend is compatible, if share can remain writable during migration, if it can preserve all file metadata and if it can perform migration of given share non-disruptively.

Example:

```
{
    'compatible': True,
    'writable': True,
    'nondisruptive': True,
    'preserve_snapshots': True,
    'migration_cancel': True,
    'migration_get_progress': False,
}
```

share_server_migration_complete(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*, *new_network_info*)

Completes migration of a given share server to another host.

Note: Is called in destination share servers backend to complete migration.

If driver is implementing 2-phase migration, this method should perform the disruptive tasks related to the 2nd phase of migration, thus completing it. Driver should also delete all original data from source backend.

It expected that all shares and snapshots will be available at the destination share server in the end of the migration complete and all updates provided in the returned model update.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.
- **new_network_info** Network allocation associated to the destination share server.

Returns

If the migration changes the shares export locations, snapshots provider locations or snapshots export locations, this method should return a dictionary containing a list of share instances and snapshot instances indexed by their ids, where each instance should provide a dict with the relevant information that need to be updated.

Example:

```
{
    'share_updates':
    {
        '4363eb92-23ca-4888-9e24-502387816e2a':
        {
        'export_locations':
        [
            {
            'path': '1.2.3.4:/foo',
            'metadata': {},
            'is_admin_only': False
            },
            {
            'path': '5.6.7.8:/foo',
            'metadata': {},
            'is_admin_only': True
            },
        ],
        'pool_name': 'poolA',
        },
    },
    'snapshot_updates':
    {
        'bc4e3b28-0832-4168-b688-67fdc3e9d408':
        {
        'provider_location': '/snapshots/foo/bar_1',
        'export_locations':
        [
            {
            'path': '1.2.3.4:/snapshots/foo/bar_1',
            'is_admin_only': False,
            },
            {
            'path': '5.6.7.8:/snapshots/foo/bar_1',
            'is_admin_only': True,
            },
        ],
        },
        '2e62b7ea-4e30-445f-bc05-fd523ca62941':
        {
        'provider_location': '/snapshots/foo/bar_2',
        'export_locations':
```

(continues on next page)

(continued from previous page)

```
            [
                {
                'path': '1.2.3.4:/snapshots/foo/bar_2',
                'is_admin_only': False,
                },
                {
                'path': '5.6.7.8:/snapshots/foo/bar_2',
                'is_admin_only': True,
                },
            ],
            },
    }
    'backend_details':
    {
        'new_share_server_info_key':
            'new_share_server_info_value',
    },
}
```

share_server_migration_continue(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Continues migration of a given share server to another host.

Note: Is called in destination share servers backend to continue migration.

Driver should implement this method to continue monitor the migration progress in storage and perform following steps until 1st phase is completed.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns Boolean value to indicate if 1st phase is finished.

share_server_migration_get_progress(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Obtains progress of migration of a share server to another host.

Note: Is called in destination shares backend to obtain migration progress.

If possible, driver can implement a way to return migration progress information.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns A dictionary with at least total_progress field containing the percentage value.

share_server_migration_start(*context*, *src_share_server*, *dest_share_server*, *shares*, *snapshots*)

Starts migration of a given share server to another host.

Note: Is called in destination share servers backend to start migration.

Driver should implement this method if willing to perform a server migration in driver-assisted way, useful when source share servers backend driver is compatible with destination backend driver. This method should start the migration procedure in the backend and return immediately. Following steps should be done in share_server_migration_continue.

Parameters

- **context** The context.RequestContext object for the request.
- **src_share_server** Reference to the original share server.
- **dest_share_server** Reference to the share server to be used by as destination.
- **shares** All shares in the source share server that should be migrated.
- **snapshots** All snapshots in the source share server that should be migrated.

Returns

Dict with migration information to be set in the destination share server.

Example:

```
{
    'backend_details': {
        'migration_info_key':  'migration_info_value',
    }
}
```

4.2 Additional reference

Contents:

4.2.1 Reference

Glossary

manila OpenStack project to provide Shared Filesystems as a service.

manila-api Service that provides a stable RESTful API. The service authenticates and routes requests throughout the Shared Filesystem service. There is *python-manilaclient* to interact with the API.

python-manilaclient Command line interface to interact with *manila* via *manila-api* and also a Python module to interact programmatically with *manila*.

manila-scheduler Responsible for scheduling/routing requests to the appropriate *manila-share* service. It does that by picking one back-end while filtering all except one back-end.

manila-share Responsible for managing Shared File Service devices, specifically the back-end devices.

DHSS Acronym for driver handles share servers. It defines two different share driver modes when they either do handle share servers or not. Each driver is allowed to work only in one mode at once. Requirement is to support, at least, one mode.

replication_type Type of replication supported by a share driver. If the share driver supports replication it will report a valid value to the *manila-scheduler*. The value of this capability can be one of *readable*, *writable* or *dr*.

readable A type of replication supported by *manila* in which there is one *active* replica (also referred to as *primary* share) and one or more non-active replicas (also referred to as *secondary* shares). All share replicas have at least one export location and are mountable. However, the non-active replicas cannot be written to until after promotion.

writable A type of replication supported by *manila* in which all share replicas are writable. There is no requirement of a promotion since replication is synchronous. All share replicas have one or more export locations each and are mountable.

dr Acronym for *Disaster Recovery*. It is a type of replication supported by *manila* in which there is one *active* replica (also referred to as *primary* share) and one or more non-active replicas (also referred to as *secondary* shares). Only the *active* replica has one or more export locations and can be mounted. The non-active replicas are inaccessible until after promotion.

active In *manila*, an *active* replica refers to a share that can be written to. In *readable* and *dr* styles of replication, there is only one *active* replica at any given point in time. Thus, it may also be referred to as the *primary* share. In *writable* style of replication, all replicas are writable and there may be no distinction of a *primary* share.

replica_state An attribute of the Share Instance (Share Replica) model in *manila*. If the value is *active*, it refers to the type of the replica. If the value is one of *in_sync* or *out_of_sync*, it refers to the state of consistency of data between the *active* replica and the share replica. If the value is *error*, a potentially irrecoverable error may have occurred during the update of data between the *active* replica and the share replica.

replication_change State of a non-active replica when it is being promoted to become the *active* replica.

recovery point objective Abbreviated as RPO, recovery point objective is a target window of time between which a storage backend may guarantee that data is consistent between a primary and a secondary replica. This window is **not** managed by *manila*.