PROJECT

Multi-tenant cybersecurity operations centre designed
as an open cloud service with machine learning elements

Linked to work package KPB1 – Design of the virt. solution
Linked to output
V3 – Description of the selected solution meeting the defined criteria
type – documentation

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

CONTENTS

List of figures 3
Introduction 4
1 DEPLOYMENT USING A BUNDLE 5
1.1 Manila deployment 5
1.2 Creating Manila relations 6
1.3 Manila Ganesha deployment 7
1.4 Creating Manila Ganesha relations 8
2 MANAGEMENT OF DEPLOYED CHARMS 9
2.1 Deploying the application without HAcluster on Manila 9
2.1.1 Managing Manila containers 9
2.1.2 Managing Manila Ganesha containers 11
2.1.3 Redirecting OpenStack Endpoints 12
3 CREATING A SHARE AND ITS MANAGEMENT 13
3.1 Creating a share using OpenStack CLI 13
3.2 Connecting an instance to the share 20
Conclusion 21

LIST OF FIGURES

Figure 1.1 Manila deployment 5
Figure 1.2 Creating Manila relations 6
Figure 1.3 Manila Ganesha deployment 7
Figure 1.4 Creating Manila Ganesha relations 8
Figure 2.1 Example of a correct start of the manila-api component 9
Figure 2.2 Example of correct operation of critical services in the manila-ganesha container 11
Figure 3.1 Example of creating a share type in OpenStack CLI 14
Figure 3.2 Listing information about the created share type 15
Figure 3.3 Share creation process in OpenStack CLI 16
Figure 3.4 Display of the created share 17
Figure 3.5 Display of the record of the created share on the CephFS backend 18
Figure 3.6 Illustration of creating access for VM HlavnaTest1 19
Figure 3.7 Output of the command displaying exports in the container serving the NFS- Ganesha server 20
Figure 3.8 Illustration of connecting to the share 20

INTRODUCTION

OpenStack Manila is an OpenStack component designed to provide shared file systems as a service (Shared File Systems as a Service). It enables the management and provision of network-accessible storage (e.g. NFS, CIFS) to multiple clients within a cloud environment. This functionality is essential for many applications and services that require shared access to data.

The aim of this document is to describe the process of deploying OpenStack Manila using Juju Charms – a tool for the declarative deployment and management of services in the cloud. Juju provides a simple and scalable way to integrate individual OpenStack components, including Manila, into a fully functional whole.

The document is intended for administrators and technical users who want to deploy Manila in a Charms-based OpenStack environment, whether in a testing phase or as part of a production cloud. The individual chapters address the infrastructure requirements, installation, configuration and validation of Manila functionality within the OpenStack environment.

1 DEPLOYMENT USING A BUNDLE

In the following section, we present the individual “bundles” for creating the Manila components and their relations.Manila deployment

```
manila-hacluster:
  charm: ch:hacluster
  channel: 2.4/stable
  bindings:
    "": openstack-mgmt
manila-mysql-router:
  charm: ch:mysql-router
  channel: 8.0/stable
  bindings:
    "": openstack-mgmt
manila:
  charm: ch:manila
  channel: 2023.2/stable
  num_units: 6
  options:
    openstack-origin: *openstack-origin
    default-share-backend: cephfsnfs1
    share-protocols: NFS
    vip: *manila-vip
  bindings:
    "": openstack-mgmt
    public: openstack-mgmt
    internal: openstack-mgmt
    admin: openstack-mgmt
    shared-db: openstack-mgmt
    amqp: openstack-mgmt
  to:
  - lxd:21
  - lxd:22
  - lxd:23
  - lxd:24
  - lxd:25
  - lxd:26
```

Figure 1.1 Manila deployment

1.1 Creating Manila relations

```
relations:
- - manila:ha
  - manila-hacluster:ha
- - manila:shared-db
  - manila-mysql-router:shared-db
- - manila-mysql-router:db-router
  - mysql-innodb-cluster:db-router
- - manila:amqp
  - rabbitmq-server:amqp
- - manila:identity-service
  - keystone:identity-service
- - manila:remote-manila-plugin
  - manila-ganesha:manila-plugin
- - manila:certificates
  - vault:certificates
```

Figure 1.2 Creating Manila relations

1.2 Manila Ganesha deployment

The Manila Ganesha deployment also remains essentially unchanged, but in this *case* too, certificates were added.

```
manila-ganesha-hacluster:
  charm: ch:hacluster
  channel: 2.4/stable
  bindings:
    "": openstack-mgmt
manila-ganesha-mysql-router:
  charm: ch:mysql-router
  channel: 8.0/stable
  bindings:
    "": openstack-mgmt
manila-ganesha:
  charm: ch:manila-ganesha
  channel: 2023.2/stable
  num_units: 6
  options:
    openstack-origin: *openstack-origin
    vip: *manila-vip
  bindings:
    "": openstack-mgmt
    public: openstack-mgmt
    internal: openstack-mgmt
    admin: openstack-mgmt
    shared-db: openstack-mgmt
    amqp: openstack-mgmt
    tenant-storage: openstack-mgmt
  to:
  - lxd:21
  - lxd:22
  - lxd:23
  - lxd:24
  - lxd:25
  - lxd:26
```

Figure 1.3 Manila Ganesha deployment

1.3 Creating Manila Ganesha relations

```
relations:
- - manila-ganesha:ha
  - manila-ganesha-hacluster:ha
- - manila-ganesha:shared-db
  - manila-ganesha-mysql-router:shared-db
- - manila-ganesha-mysql-router:db-router
  - mysql-innodb-cluster:db-router
- - manila-ganesha:amqp
  - rabbitmq-server:amqp
- - manila-ganesha:identity-service
  - keystone:identity-credentials
- - manila-ganesha:certificates
  - vault:certificates
- - manila:remote-manila-plugin
  - manila-ganesha:manila-plugin
```

Figure 1.4 Creating Manila Ganesha relations

2 MANAGEMENT OF DEPLOYED CHARMS

In this section we will discuss the management of deployed charms within containers. The main problem we encountered when creating the shared network file system was that not all changes made after adding relations were actually reflected in the configuration of our containers.

2.1 Deploying the application without HAcluster on Manila

Here, deployment using HAcluster did not succeed; it always occupied port 8786, on which the manila-api service listens, and the service was unable to accept requests; therefore we disabled it and worked with only one Manila container.

2.1.1 Managing Manila containers

The Manila charm, in combination with the Manila Ganesha charm, plays a role only in receiving requests to create, delete or modify a share, so the components that matter for it are manila-api, manila-scheduler and manila-data.

The manila-scheduler and manila-data components work practically always; we have never encountered problems with them, but for manila-api it is important that it looks like this:

```
root@juju-86d9a6-0-lxd-49:~# systemctl status manila-api manila-scheduler manila-data
● manila-api.service - OpenStack Manila API
     Loaded: loaded (/lib/systemd/system/manila-api.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2025-04-21 23:01:59 UTC; 5 days ago
       Docs: man:manila-api(1)
   Main PID: 230301 (manila-api)
      Tasks: 5 (limit: 314572)
     Memory: 328.0M
        CPU: 1h 54min 42.269s
     CGroup: /system.slice/manila-api.service
             ├─230301 /usr/bin/python3 /usr/bin/manila-api --config-file=/etc/manila/manila.conf --log-file=/var/log/manila/manila-api.log
             ├─230309 /usr/bin/python3 /usr/bin/manila-api --config-file=/etc/manila/manila.conf --log-file=/var/log/manila/manila-api.log
             ├─230310 /usr/bin/python3 /usr/bin/manila-api --config-file=/etc/manila/manila.conf --log-file=/var/log/manila/manila-api.log
             ├─230311 /usr/bin/python3 /usr/bin/manila-api --config-file=/etc/manila/manila.conf --log-file=/var/log/manila/manila-api.log
             └─230312 /usr/bin/python3 /usr/bin/manila-api --config-file=/etc/manila/manila.conf --log-file=/var/log/manila/manila-api.log
```

Figure 2.1 Example of a correct start of the manila-api component

The figure shows several processes, which indicates that the service is listening on the default ports for Manila. When there was only one process here, it usually meant that HA had occupied the manila-api ports and it was not working. We then verified which ports were running using:

sudo lsof -i :8786

If they were occupied by HAcluster, we always first stopped haproxy, then disabled it, and subsequently prevented it from being started by other applications as well, using the commands:

systemctl stop haproxy

systemctl disable haproxy

systemctl mask haproxy

After this step, we restarted the manila-api service so that it would re-occupy the relevant ports. It often happened that precisely the changes in relations so-called masked the application, so we were unable to start it; here it was necessary to unmask it, enable it and then start it using the commands:

systemctl unmask manila-api

systemctl enable manila-api

systemctl start manila-api

Since Manila Ganesha takes over the role of the manila-share component, it is important to disable this service in the manila container to avoid problems during creation. This step should be performed by the relation shown in the figure below, but for some reason this never happened and on some containers this service was active:

```
- manila:remote-manila-plugin
  - manila-ganesha:manila-plugin
```

Therefore it is important to check this on the specific containers and, if necessary, to stop, disable and mask it:

systemctl stop manila-share

systemctl disable manila-share

systemctl mask manila-share

2.1.2 Managing Manila Ganesha containers

After the series of previous steps, we can proceed to managing the manila-ganesha containers. Manila Ganesha is responsible for two services: the first is the manila-share component and the second is the nfs-ganesha service, which manages exports. In addition, here Ganesha also uses HAcluster, which runs on it without problems, but in the event of outages or when relations are being completed, conflicts may occur.

Here it is important to check that the two above-mentioned services + the active node managed by pacemaker are in the same container. We can do this using the command:

crm status

This lists the necessary information for us. As we can see in the figure below, all three of these services are running in the same container, so the application will work correctly this way.

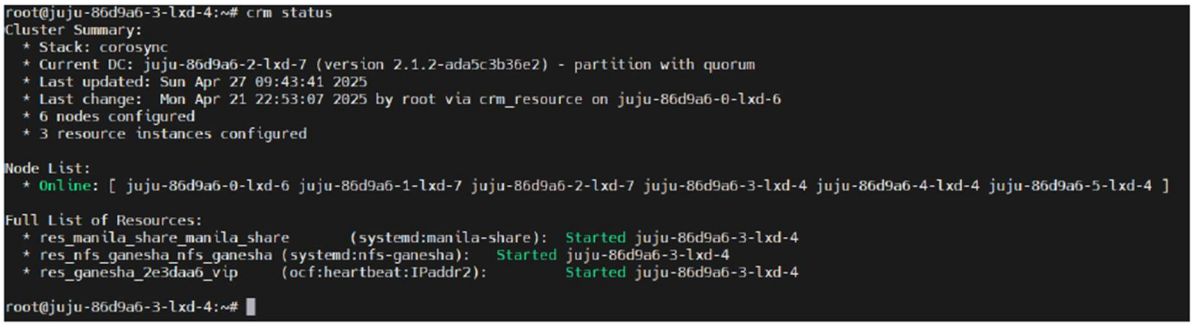


Figure 2.2 Example of correct operation of critical services in the manila-ganesha container

However, if a situation were to arise in which these services are not in the same place, they can be moved using the command:

crm resource move <resource_name> <node_name>

In our case, for example:

crm resource move res_manila_share_manila_share manila-ganesha/3

On the other containers these services should not be running. However, if errors were to occur, it is advisable to stop them by entering the given container and using the command:

crm resource stop res_manila_share_manila_share

crm resource disable res_manila_share_manila_share

2.1.3 Redirecting OpenStack Endpoints

Since we carried out the implementation without HAcluster for Manila but assigned it a Virtual IP when creating it, an endpoint was created in OpenStack that points to the relevant IP and port of manila-api. However, this implementation would not work, so we changed all the manila endpoints so that they point to the active manila container we selected that handles share management requests. (While working on the diploma thesis we tried various settings and not everything was removed correctly, so it is necessary to set absolutely all endpoints for it to work. This also applies to those that are no longer active, but without this they caused problems.) In a clean deployment of the entire Juju, it would be necessary in such an implementation to change only two endpoints, namely manila and manilav2, which are responsible for shares.

We changed the endpoints according to the relevant id and a command in the form:

openstack endpoint set <id> --url <http://<IP>:8786/<version>%\(tenant_id\)s

In our case as follows:

openstack endpoint set 14d4d858809849c687fdae51ef688376 --url http://10.11.1.189:8786/v1/%\(tenant_id\)s

openstack endpoint set 6e74f471147a465a98066f0d675e1fad --url http://10.11.1.189:8786/v1/%\(tenant_id\)s

openstack endpoint set 590c00c7d4c84858a27cdcf291ed056a --url http://10.11.1.189:8786/v2/%\(tenant_id\)s

openstack endpoint set 6b456cbbb10d43edaa95410c2ebbe74d --url http://10.11.1.189:8786/v2/%\(tenant_id\)s

openstack endpoint set d84e28cdae074e949a39e8a3e6b2f4da --url http://10.11.1.189:8786/v1/%\(tenant_id\)s

openstack endpoint set bd4ca0d2b26f4fc580d56b3cc1d35758 --url http://10.11.1.189:8786/v2/%\(tenant_id\)s

3 CREATING A SHARE AND ITS MANAGEMENT

This chapter is taken from the relevant diploma thesis, since it is described there in great detail in the chapter “7 TESTING THE CREATED SOLUTION”, in subsection 7.1 Creating a share using OpenStack CLI.

3.1 Creating a share using OpenStack CLI

The first step is to create a share type, which defines the parameters and properties of the shared storage. It is important because without it Manila would not know which backend to use when creating shares. We created it using the command:

share type create cephfsnfstype false

There are two parameters within this command.

Cephfsnfstype – specifies the given share type (the one we chose defines that this is the case of CephFS via NFS-Ganesha).

False – in turn indicates that the backend does not require Manila to manage the share servers and network infrastructure – the backend handles everything itself.

The figure below shows the creation of such a share type.

Field	Value
id	c098fac3-0b3d-4717-93f4-b6aa88bbc1fa
name	cephfsnfstype
visibility	public
is_default	False
required_extra_specs	driver_handles_share_servers : False
optional_extra_specs	
description	None

Figure 3.1 Example of creating a share type in OpenStack CLI

After this step, it is still necessary to specify it additionally, namely by using the command:

share type set cephfsnfstype --extra-specs vendor_name=Ceph storage_protocol=NFS

Here we assigned additional metadata (so-called extra specs) to the created type, which further specify the characteristics of the backend Manila will work with.

vendor_name=Ceph – defines a parameter that serves as an identifier of the storage the backend uses; in our case it is Ceph. This variable changes in case we had multiple backends.

storage_protocol=NFS – specifies which protocol will be used for storing data. Here it needs to be defined, since we export the share via the NFS Ganesha server.

The figure below shows how such a created share type can be listed, using the command:

share type list

```
(openstack) share type list
+--------------------------------------+--------------+------------+------------+--------------------------------------+-----------------------+-------------+
| id                                   | name         | visibility | is_default | required_extra_specs                 | optional_extra_specs  | description |
+--------------------------------------+--------------+------------+------------+--------------------------------------+-----------------------+-------------+
| c098fac3-0b3d-4717-93f4-b6aa88bbc1fa | cephfsnfstype | public     | False      | driver_handles_share_servers : False | vendor_name : Ceph    | None        |
|                                      |              |            |            |                                      | storage_protocol : NFS |             |
+--------------------------------------+--------------+------------+------------+--------------------------------------+-----------------------+-------------+
```

Figure 3.2 Listing information about the created share type

After performing these steps, we can proceed to creating the share itself, using the command:

share create --share-type cephfsnfstype --name cephnfsshare2 nfs 1

--share-type cephfsnfstype – in this case specifies that we will use the previously created cephfsnfstype as the share type.

--name cephnfsshare2 – represents the name that will be assigned to the new share and serves for easy identification of the share within the system. This name can be chosen freely.

nfs – the variable specifies the protocol that will be used for access to the share.

1 – specifies the size of the share in GB. For testing purposes such a size is sufficient, but in production this parameter would need to be adjusted to the given needs.

After entering this command, creation of the relevant share begins, and OpenStack displays information about it, which can be seen in the figure below.

```
(openstack) share create --share-type cephfsnfstype --name cephnfsshare2 nfs 1
+---------------------------------------+--------------------------------------+
| Field                                 | Value                                |
+---------------------------------------+--------------------------------------+
| access_rules_status                   | active                               |
| availability_zone                     | None                                 |
| create_share_from_snapshot_support    | False                                |
| created_at                            | 2025-04-22T11:23:52.700529           |
| description                           | None                                 |
| has_replicas                          | False                                |
| host                                  |                                      |
| id                                    | 4af4084f-d253-43d2-abe4-4214ea847769 |
| is_public                             | False                                |
| metadata                              | {}                                   |
| mount_snapshot_support                | False                                |
| name                                  | cephnfsshare2                        |
| progress                              | None                                 |
| project_id                            | 0db553d7901246a3afacf58ef269ec2a     |
| replication_type                      | None                                 |
| revert_to_snapshot_support            | False                                |
| share_group_id                        | None                                 |
| share_network_id                      | None                                 |
| share_proto                           | NFS                                  |
| share_server_id                       | None                                 |
| share_type                            | c098fac3-0b3d-4717-93f4-b6aa88bbc1fa |
| share_type_name                       | cephfsnfstype                        |
| size                                  | 1                                    |
| snapshot_id                           | None                                 |
| snapshot_support                      | False                                |
| source_share_group_snapshot_member_id | None                                 |
| status                                | creating                             |
| task_state                            | None                                 |
| user_id                               | ecb16915b3774396be1be59701bd3365     |
| volume_type                           | cephfsnfstype                        |
+---------------------------------------+--------------------------------------+
```

Figure 3.3 Share creation process in OpenStack CLI

After a while the share creation completes, and we can display it based on the *id* parameter using the command:

Share show <id>

```
(openstack) share show 4af4084f-d253-43d2-abe4-4214ea847769
+---------------------------------------+---------------------------------------------------------------------------------------------------------------------------------------+
| Field                                 | Value                                                                                                                                 |
+---------------------------------------+---------------------------------------------------------------------------------------------------------------------------------------+
| access_rules_status                   | active                                                                                                                                |
| availability_zone                     | nova                                                                                                                                  |
| create_share_from_snapshot_support    | False                                                                                                                                 |
| created_at                            | 2025-04-22T11:23:52.700529                                                                                                            |
| description                           | None                                                                                                                                  |
| export_locations                      |                                                                                                                                       |
|                                       | id = 1fb809f5-9046-47e0-aa75-af8b30304a3e                                                                                             |
|                                       | path = 10.11.0.100:/volumes/_nogroup/5b49a652-4823-4cff-9a52-6113d34bf003/70d0d9f4-8665-4fcf-a494-7f4d7f35438e                        |
|                                       | preferred = False                                                                                                                     |
|                                       | share_instance_id = 5b49a652-4823-4cff-9a52-6113d34bf003                                                                              |
|                                       | is_admin_only = False                                                                                                                 |
| has_replicas                          | False                                                                                                                                 |
| host                                  | 10.11.0.100@cephfsnfs1#cephfs                                                                                                         |
| id                                    | 4af4084f-d253-43d2-abe4-4214ea847769                                                                                                  |
| is_public                             | False                                                                                                                                 |
| mount_snapshot_support                | False                                                                                                                                 |
| name                                  | cephnfsshare2                                                                                                                         |
| progress                              | 100%                                                                                                                                  |
| project_id                            | 0db553d7901246a3afacf58ef269ec2a                                                                                                      |
| properties                            |                                                                                                                                       |
| replication_type                      | None                                                                                                                                  |
| revert_to_snapshot_support            | False                                                                                                                                 |
| share_group_id                        | None                                                                                                                                  |
| share_network_id                      | None                                                                                                                                  |
| share_proto                           | NFS                                                                                                                                   |
| share_server_id                       | None                                                                                                                                  |
| share_type                            | c098fac3-0b3d-4717-93f4-b6aa88bbc1fa                                                                                                  |
| share_type_name                       | cephfsnfstype                                                                                                                         |
| size                                  | 1                                                                                                                                     |
| snapshot_id                           | None                                                                                                                                  |
| snapshot_support                      | False                                                                                                                                 |
| source_share_group_snapshot_member_id | None                                                                                                                                  |
| status                                | available                                                                                                                             |
| task_state                            | None                                                                                                                                  |
| user_id                               | ecb16915b3774396be1be59701bd3365                                                                                                      |
| volume_type                           | cephfsnfstype                                                                                                                         |
+---------------------------------------+---------------------------------------------------------------------------------------------------------------------------------------+
```

Figure 3.4 Display of the created share

In the figure above we see several important parameters. The most important of them is *export_locations*, which shows us the path to the created share within NFS Ganesha. It is in the form {NFS-Ganesha server ip address }:{path where it is stored}, and the individual clients then connect to the given share according to this path. The share_instance_id parameter specifies under which id the created share will be stored within subvolumes on the CephFS backend. In addition, we can see *progress 100%* and *status available*, which inform us of its successful creation. The other parameters are not needed for the purposes of this diploma thesis, so we will omit their explanation.

Successful creation of the relevant share can then also be checked after logging in to one of the ceph monitors, where, after entering the command:

Ceph fs subvolume ls <file system name> (in our case ceph-fs)

We can list all the stored shares on the backend storage. In the figure below there are several of them due to testing, but we are interested specifically in the middle one, which belongs to the share from the example we are presenting. We can easily match it by the share_instance_id, which matches the one shown in the output in Figure 19.

```
root@juju-86d9a6-0-lxd-2:~# ceph fs subvolume ls ceph-fs
[
    {
        "name": "23c394a8-0639-4674-9cc1-a79cc072ab46"
    },
    {
        "name": "5b49a652-4823-4cff-9a52-6113d34bf003"
    },
    {
        "name": "b90c2809-23b6-4ade-baf0-9584c496f227"
    }
]
```

Figure 3.5 Display of the record of the created share on the CephFS backend

After this step, it is necessary to define who can access the created share. We do this using an ACL, in which we define which IP addresses will be allowed.

In the figures below, we can see the process of creating such access, from entering the command, through its creation, to the successful assignment of the rule, for an instance with the IP address 158.193.154.30.

```
(openstack) share access create cephnfsshare2 ip 158.193.154.50
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| id           | 75df74dc-33e5-4a71-ac72-a07d8e4c6fba |
| share_id     | 4af4084f-d253-43d2-abe4-4214ea847769 |
| access_level | rw                                   |
| access_to    | 158.193.154.50                       |
| access_type  | ip                                   |
| state        | queued_to_apply                      |
| access_key   | None                                 |
| created_at   | 2025-04-23T21:06:08.520036           |
| updated_at   | None                                 |
| properties   |                                      |
+--------------+--------------------------------------+
(openstack) share access show 75df74dc-33e5-4a71-ac72-a07d8e4c6fba
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| id           | 75df74dc-33e5-4a71-ac72-a07d8e4c6fba |
| share_id     | 4af4084f-d253-43d2-abe4-4214ea847769 |
| access_level | rw                                   |
| access_to    | 158.193.154.50                       |
| access_type  | ip                                   |
| state        | active                               |
| access_key   | None                                 |
| created_at   | 2025-04-23T21:06:08.520036           |
| updated_at   | 2025-04-23T21:06:11.482062           |
| properties   |                                      |
+--------------+--------------------------------------+
```

Figure 3.6 Illustration of creating access for VM HlavnaTest1

After such access is created, this information is also propagated to our NFS Ganesha server, where, by entering the command:

busctl call org.ganesha.nfsd /org/ganesha/nfsd/ExportMgr org.ganesha.nfsd.exportmgr ShowExports

We can list all available exports. We are specifically interested in this one:

"/volumes/_nogroup/5b49a652-4823-4cff-9a52-6113d34bf003/70d0d9f4-8665-4fcf-a494-7f4d7f35438e", which means that if it is displayed after this command, Ganesha allows access to it. The output of this command can be seen in the figure below.

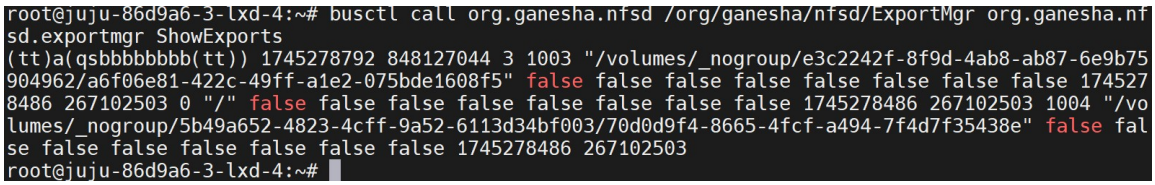


Figure 3.7 Output of the command displaying exports in the container serving the NFS-Ganesha server

3.2 Connecting an instance to the share

The final step is connecting the instance to such a share, for which it is necessary to perform two steps. The first is downloading the nfs-common package, which provides tools needed for working with NFS. After its successful installation, it is also necessary to create a location where we will mount the share, for example */mnt/manilatest1.*

We can then proceed to connecting to the created share at the corresponding address using the command:

mount -v -t nfs <share address> (10.11.0.100/volumes/_nogroup/5b49a652-4823- 4cff-9a52-6113d34bf003/70d0d9f4-8665-4fcf-a494-7f4d7f35438e)

/mnt/manilatest1

In the figure below, we can then see that this was successful (it did not return any error)

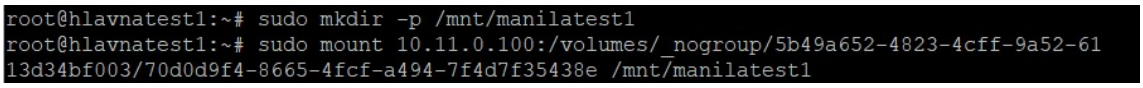


Figure 3.8 Illustration of the connection to the share

After this step, any management of the given share is possible, such as creating folders, adding information to them and so on. In essence, everyone should connect to the given share using the mount command, so that the method of access to such shares should be separate for administrators and its users.

CONCLUSION

In this document, we have described in detail the process of deploying the OpenStack Manila service using Juju Charms. We went through the individual steps, from preparing the environment and configuring the necessary components, to the successful deployment and verification of the functionality of the shared file system service.

The use of Charms significantly simplifies the management and scaling of OpenStack components, including Manila, while also enabling configuration automation and the adaptation of the service to specific needs. Such an approach increases the reliability of the entire solution and reduces the need for manual intervention by administrators.

OpenStack Manila deployed via Juju provides a robust and flexible solution for providing shared storage in a cloud environment. It is suitable for a wide range of scenarios — from test environments to production deployments in enterprise environments.

A further step may be the integration of Manila with specific back-end storage systems (e.g. CephFS, NFS, NetApp), the implementation of advanced access policies and the monitoring of operation using tools such as Prometheus or Grafana.