ECE530: Cloud Computing

OpenStack IaaS Deployment

Suyog Joshi - 101846426 - sjoshi@unm.edu

Contents

List of Figures

Abstract

OpenStack is an open-source cloud computing platform that leverages virtualized resources to create and maintain both public and private cloud environments. In our first homework assignment, we were instructed to set up OpenStack across several virtual machines (VMs) on our system, focusing on the installation of key services. The objective was to deploy the following components.

1) Keystone**:**
   Keystone is the identity service used by OpenStack for authentication (authN) and high-level authorization (authZ).

2) Glance:
   The Image service (glance) project provides a service where users can upload and discover data assets that are meant to be used with other services.

3) Placement:
   Placement is an OpenStack service that provides an HTTP API for tracking cloud resource inventories and usages to help other services effectively manage and allocate their resources.

4) Nova:
   Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. KVM and Xen are available choices for hypervisor technology, together with Hyper-V and Linux container technology such as LXC.

5) Neutron:
   Neutron is an OpenStack project to provide "network connectivity as a service" between interface devices (e.g., vNICs) managed by other OpenStack services (e.g., nova).

6) Horizon:
   Dashboard provides administrators and users a graphical interface to access, provision, and automate cloud-based resources. The design accommodates third party products and services, such as billing, monitoring, and additional management tools. The dashboard is also brandable for service providers and other commercial vendors who want to make use of it. The dashboard is one of several ways users can interact with OpenStack resources. Developers can automate access or build tools to manage resources using the native OpenStack API or the EC2 compatibility API.

7) Cinder:
   Block Storage provides persistent block-level storage devices for use with OpenStack compute instances. The block storage system manages the creation, attaching and detaching of the block devices to servers. Block storage volumes are fully integrated into OpenStack Compute and the Dashboard allowing for cloud users to manage their own storage needs.

8) Swift:
   The OpenStack Object Store project, known as Swift, offers cloud storage software so that

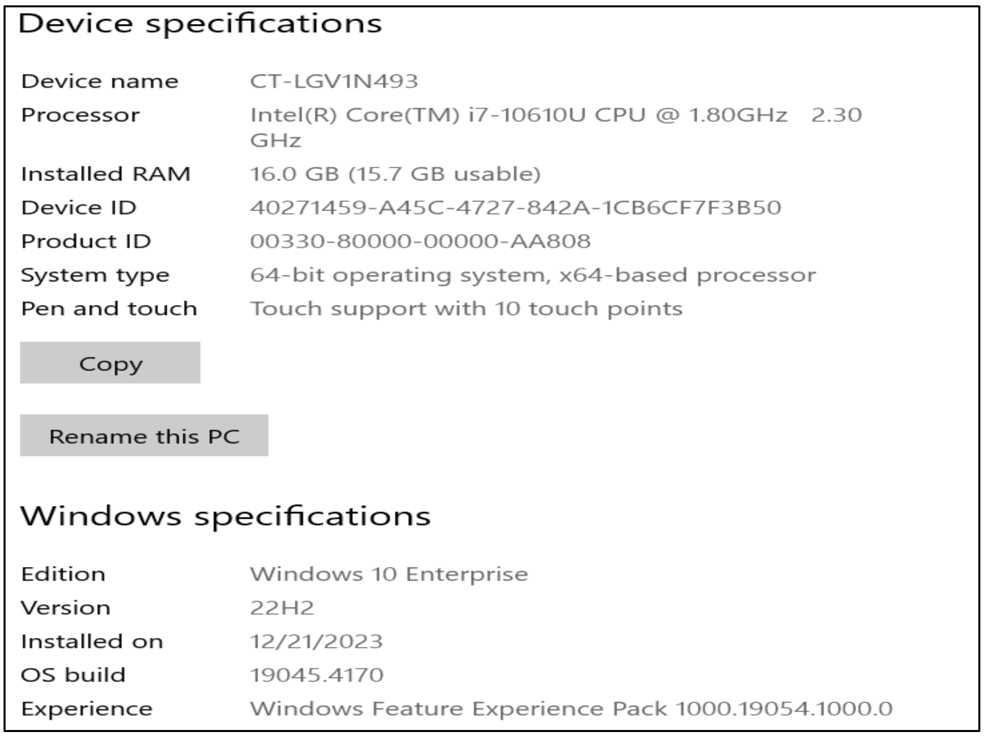
you can store and retrieve lots of data with a simple API. It's built for scale and optimized for durability, availability, and concurrency across the entire data set. Swift is ideal for storing unstructured data that can grow without bound.

Introduction

For our class, it was essential that we set up these services manually, without using any automated tools (like DevStack, Terraform) to ensure a deeper learning experience. We followed the official guidelines on the OpenStack website and supplemented our learning with Server World website. We decided to go with OpenStack Antelope.

After setting up the services, our next challenge was to create a subnet, a virtual machine (VM), and a storage volume. During the setup process, we realized the interdependence of the services; they needed to work together, which meant establishing a robust network was a fundamental step in the setup process.

Initially, we chose to deploy OpenStack on an Apple Mac Air M2. However, Apple's M2 being an ARM machine meant we were limited in our choices of virtualization software. After facing numerous crashes when working with VMware and a beta version of Orcale VM VirtualBox, we decided to forfeit implementation on Apple's M2 chip. Eventually, we chose to deploy on an intel device with Windows 10 that satisfied all the requirements for this project (CPU 6 cores, RAM 6 GB, Storage 200 GB). We used Oracle VM VirtualBox as our virtualization software.


Device specifications

Device name	CT-LGV1N493
Processor	Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz 2.30 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	40271459-A45C-4727-842A-1CB6CF7F3B50
Product ID	00330-80000-00000-AA808
System type	64-bit operating system, x64-based processor
Pen and touch	Touch support with 10 touch points

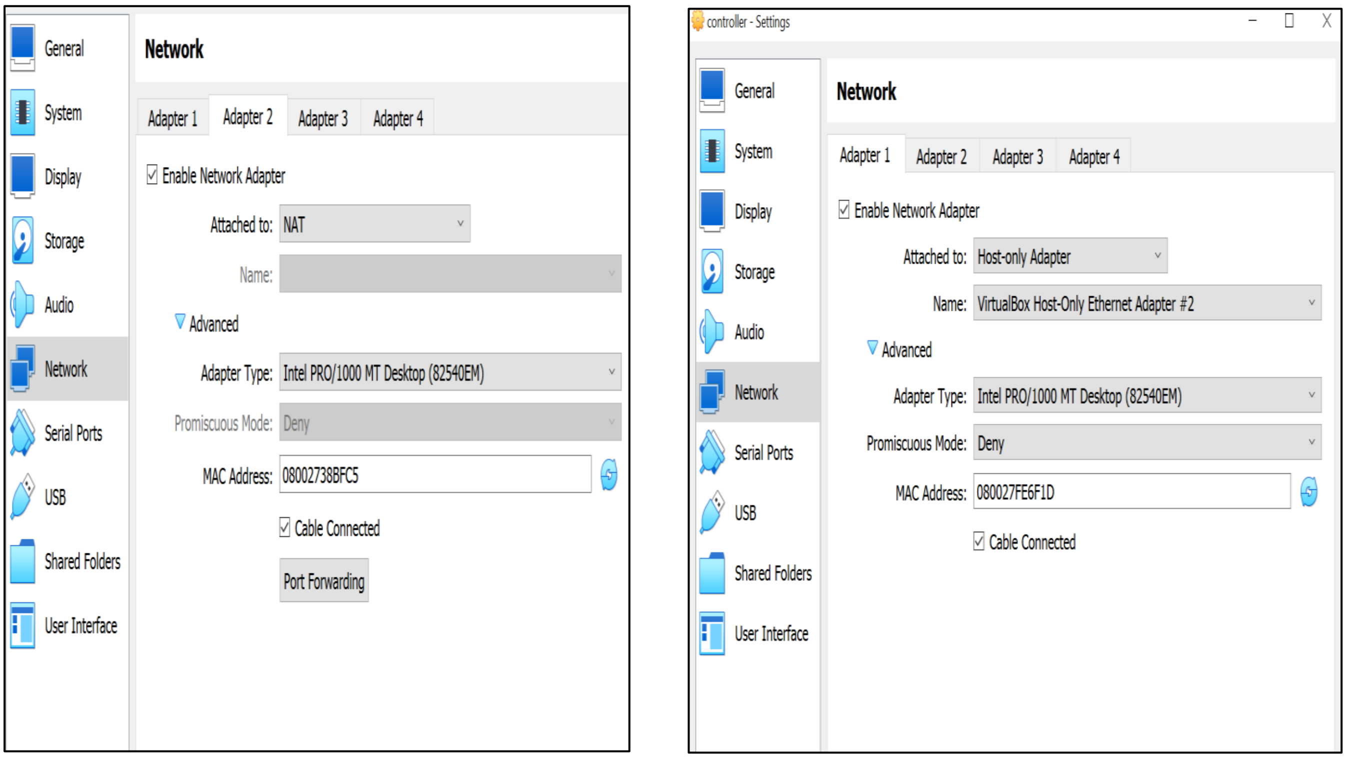
Windows specifications

Edition	Windows 10 Enterprise
Version	22H2
Installed on	12/21/2023
OS build	19045.4170
Experience	Windows Feature Experience Pack 1000.19054.1000.0

Figure 1: Device Specs

Deployment

For our deployment, we chose Ubuntu 22.04.4 LTS. We installed a fresh VM using Oracle VM VirtualBox and Ubuntu. We also configured the VMs with two network interfaces – we used NAT adapter for outside internet communication and host only adapter for internal communication between VMs. Our network configurations can be seen in Figure 2.



Network

Adapter 2

☑ Enable Network Adapter

Attached to: NAT

Name:

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 08002738BFC5

☑ Cable Connected

Network

Adapter 1

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #2

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027FE6F1D

☑ Cable Connected

Figure 2: Network Adapter Config

For this project, we required at least five virtual machines (VMs) to host the range of services needed: a controller node (controller), a compute node (compute), a block storage node (block), and two object storage nodes (object1 & object2). To streamline the setup, we first fully prepared controller by updating it and applying basic configurations. Then, we utilized full cloning for the VM, which preserved all installed packages and configurations, eliminating the need to configure each subsequent node from the ground up.

Next we setup the host networking part in the guide. We set the hostname of each clone to their respective names and configured name resolution by editing the /etc/hosts file in each node as show in Figure 3.

Installation and configuring components.

Installing on controller node:

```
apt install chrony
```

Edited the /etc/chrony/chrony.conf file:

```
server controller iburst
```

To enable other nodes to connect to the chrony daemon on the controller node, add this key to the same **chrony.conf** file mentioned above:

```
allow controller
```

Restarted the NTP service:

```
service chrony restart
```

Installing on other nodes:

```
apt install chrony
```

Edited the /etc/chrony/chrony.conf file on other nodes:

```
server controller iburst
```

Restarted the NTP service:

```
service chrony restart
```

Verify Operation:
Run these command on each nodes:

```
chronyc sources
```

Installing other packages:

```
add-apt-repository cloud-archive:antelope
apt install python3-openstackclient
apt install mariadb-server python3-pymysql
```

Next, we configured MariaDB by modifying /etc/mysql/mariadb.conf.d/99-openstack.cnf with controller IP

```
[mysqld]
bind-address = 192.168.72.3

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

We restarted and secured databased sercvice by running following commands:

```
service mysql restart
mysql_secure_installation
```

Next, we installed the message queue, RabbitMQ, on Ubuntu, which facilitates the coordination of operations and status information across services. It's important to note that we used the same password for all services, databases, and nodes, under the clear understanding that this is for a proof-of-concept deployment.

```
apt install rabbitmq-server
rabbitmqctl add_user openstack RABBIT_PASS
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Next, we installed memcached on Ubuntu to enable token caching and modified the configuration file /etc/memcached.conf with our management IP before restarting the memcached service. It's worth noting that although the OpenStack website recommends using python-memcache, our ARM configuration required us to use python3-memcache instead.

```
apt install memcached python3-memcache
```

```
-l 192.168.72.3
```

```
service memcached restart
```

Next we installed and configure etcd

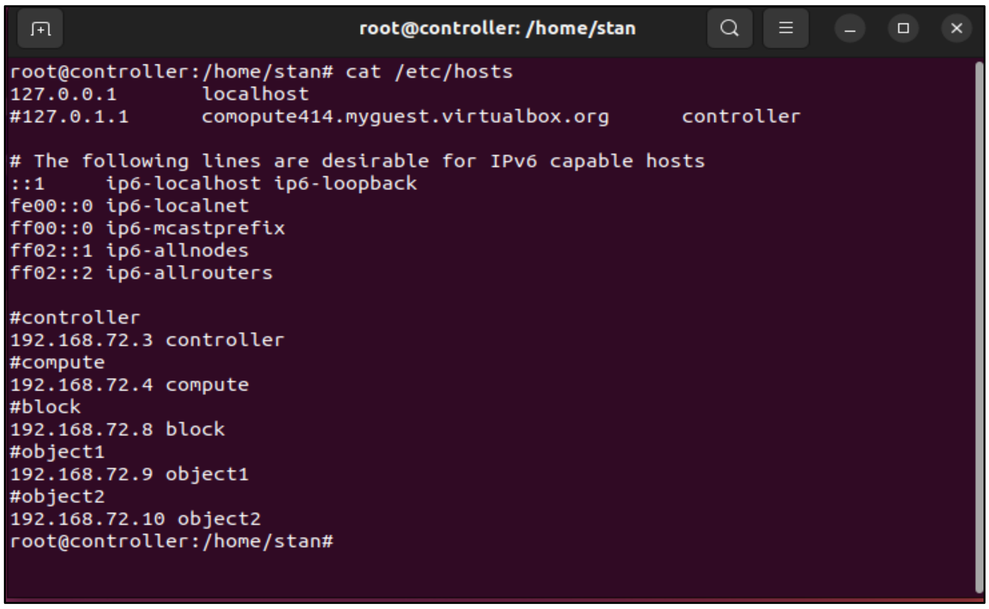
```
apt install etcd
```

Edited the /etc/default/etcd file:cloud

```
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://192.168.72.3:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.72.3:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.72.3:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.72.3:2379"
```

Enabled and restarted the etcd service:

```
systemctl enable etcd
systemctl restart etcd
```



```
root@controller:/home/stan# cat /etc/hosts
127.0.0.1       localhost
#127.0.1.1      comopute414.myguest.virtualbox.org      controller

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

#controller
192.168.72.3 controller
#compute
192.168.72.4 compute
#block
192.168.72.8 block
#object1
192.168.72.9 object1
#object2
192.168.72.10 object2
root@controller:/home/stan#
```

Figure 3: Configuring Name Resolution

Keystone

Next, we initiated the installation of OpenStack services, beginning with the Keystone identity service on the controller node. The steps were performed in the order they appear.
Create keystone database:

```
mysql
MariaDB [(none)]> CREATE DATABASE keystone; MariaDB [(none)]> GRANT ALL
PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY
'KEYSTONE_DBPASS'; MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.*
TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';
exit;
```

Install and set up Keystone:

```
apt install keystone
```

Edit the config file /etc/keystone/keystone.conf:

```
[database]
# ...
connection =
mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
[token]
# ...
provider = fernet
```

Populate the Identity service database:

```
su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Initialize Fernet key repositories:

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group
keystone
# keystone-manage credential_setup --keystone-user keystone --keystone-group
keystone
```

Bootstrap the Identity service:

```
keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:5000/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
```

Edit the **/etc/apache2/apache2.conf** file and configure the **ServerName** option to reference the controller node:

```
ServerName controller
```

Finalize the installation.

```
service apache2 restart
 export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
 export OS_PROJECT_NAME=admin
 export OS_USER_DOMAIN_NAME=Default
 export OS_PROJECT_DOMAIN_NAME=Default
 export OS_AUTH_URL=http://controller:5000/v3
 export OS_IDENTITY_API_VERSION=3
```

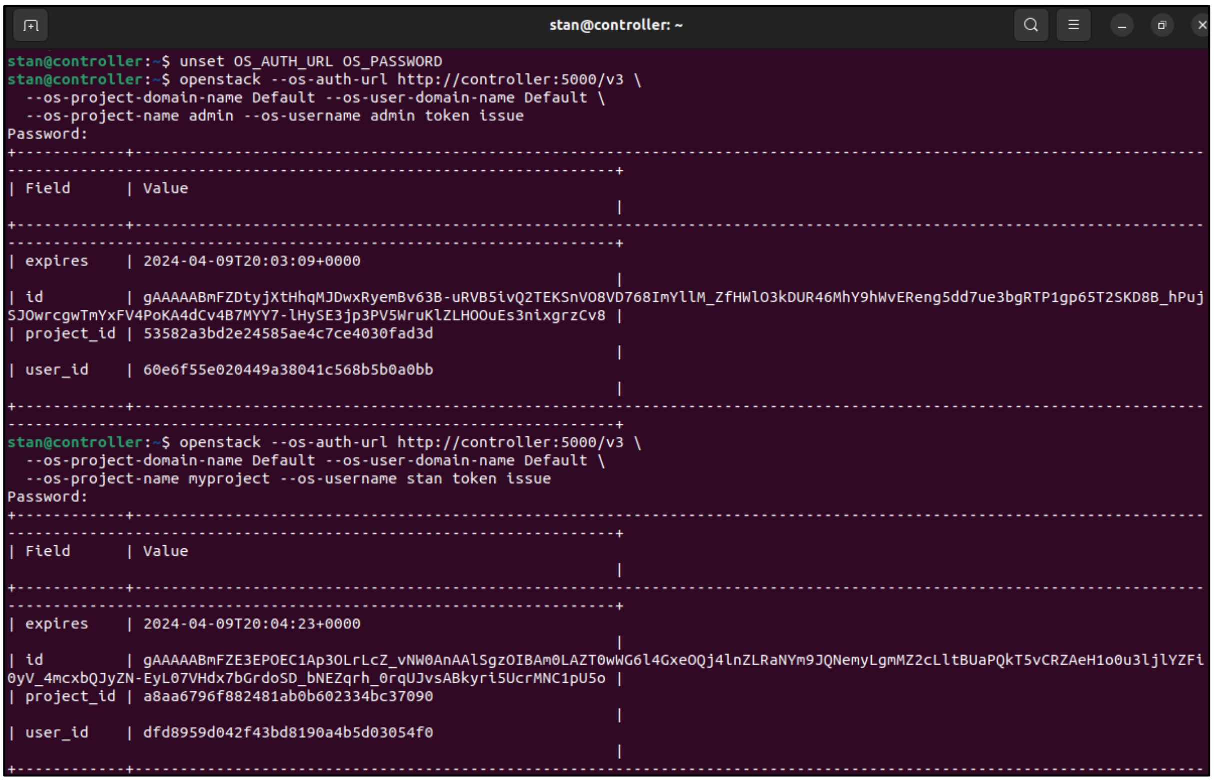
Create a domain, projects, users, and roles.

```
openstack domain create --description "An Example Domain" example
openstack project create --domain default  --description "Service Project" service
openstack project create --domain default --description "Demo Project" myproject
openstack user create --domain default  --password-prompt stan
openstack user create --domain default  --password-prompt stan
openstack role create myrole
openstack role add --project myproject --user myuser myrole
```

It is important to mention that we created a username stan. However, upon viewing the grading criteria, we created another username demo by doing the exact same thing, with the only exception being replacing stan with demo.

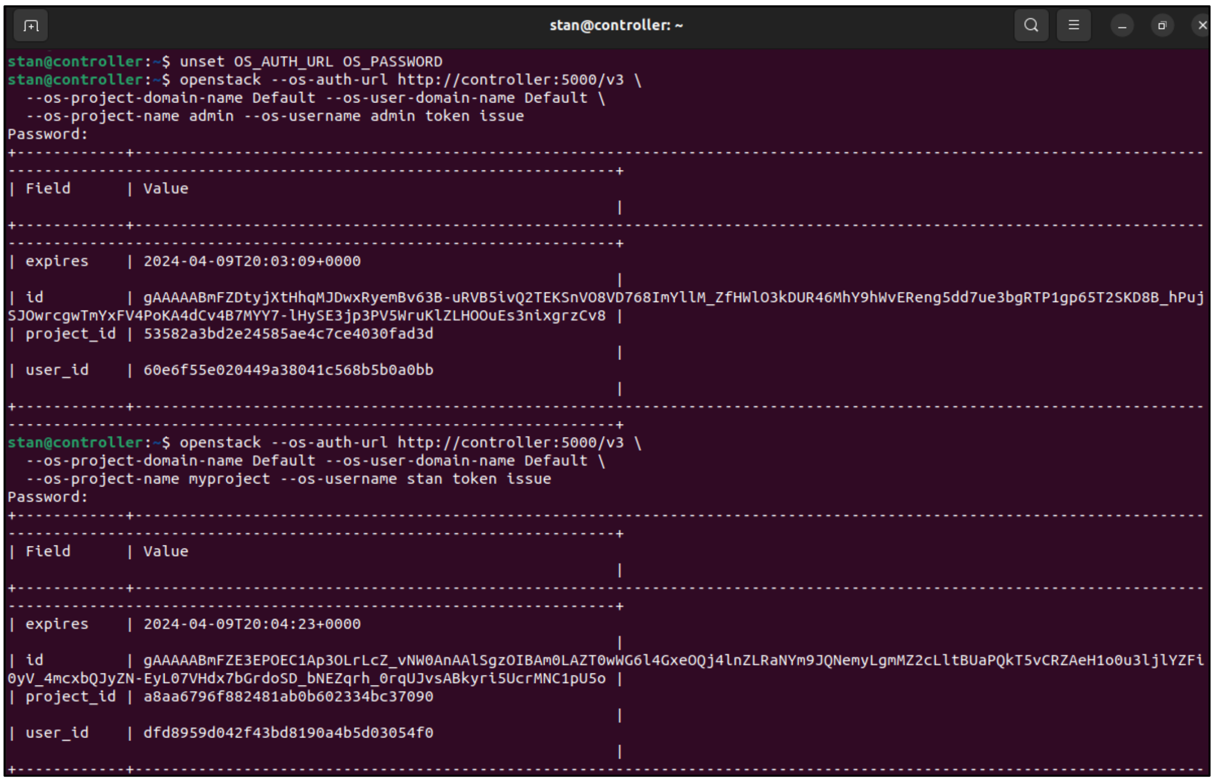
Next, we tested the Keystone installation by clearing the login credentials and requesting both an admin authentication token and a demo authentication token. **This procedure met the requirements of the Keystone grading checklist(see Figure 4, 5, 6).**

```
unset OS_AUTH_URL OS_PASSWORD
openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name admin --os-username admin token issue
openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name myproject --os-username myuser token issue
```



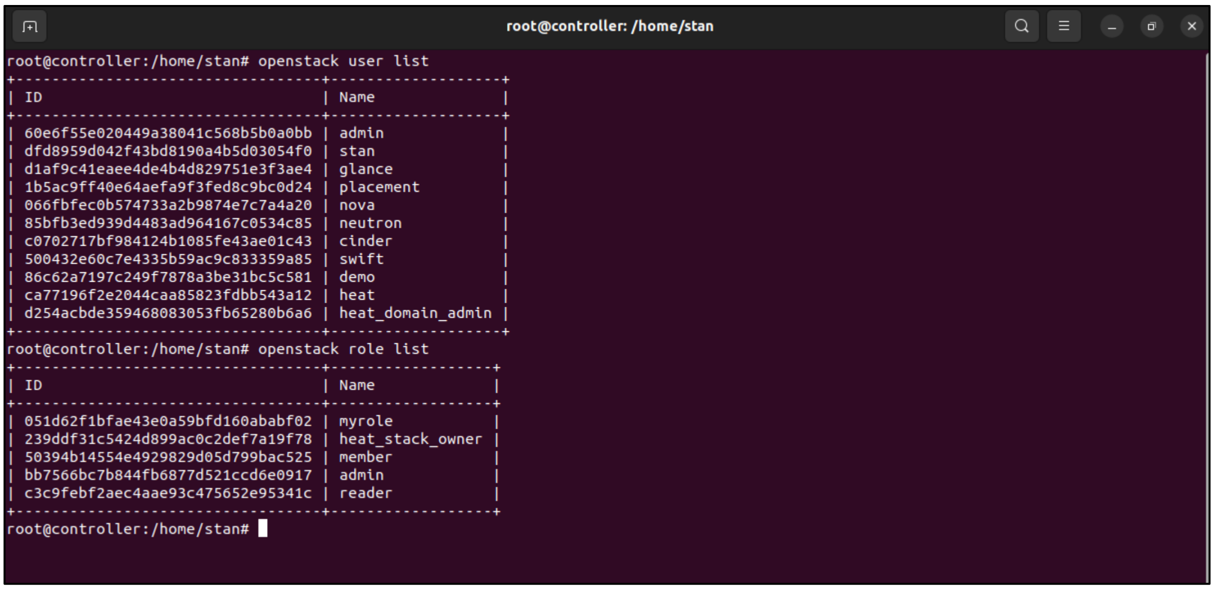
```
stan@controller:~$ unset OS_AUTH_URL OS_PASSWORD
stan@controller:~$ openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name admin --os-username admin token issue
Password:
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| Field      | Value
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| expires    | 2024-04-09T20:03:09+0000
                                                                   |
| id         | gAAAAABmFZDtyjXtHhqMJDwxRyemBv63B-uRVB5ivQ2TEKSnVO8VD768ImYllM_ZfHWlO3kDUR46MhY9hWvEReng5dd7ue3bgRTP1gp65T2SKD8B_hPuj
SJOwrcgwTmYxFV4PoKA4dCv4B7MYY7-lHySE3jp3PV5WruKlZLHOOuEs3nixgrzCv8 |
| project_id | 53582a3bd2e24585ae4c7ce4030fad3d
                                                                   |
| user_id    | 60e6f55e020449a38041c568b5b0a0bb
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
stan@controller:~$ openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name myproject --os-username stan token issue
Password:
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| Field      | Value
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| expires    | 2024-04-09T20:04:23+0000
                                                                   |
| id         | gAAAAABmFZE3EPOEC1Ap3OLrLcZ_vNW0AnAAlSgzOIBAm0LAZT0wWG6l4GxeOQj4lnZLRaNYm9JQNemyLgmMZ2cLltBUaPQkT5vCRZAeH1o0u3ljlYZFi
0yV_4mcxbQJyZN-EyL07VHdx7bGrdoSD_bNEZqrh_0rqUJvsABkyri5UcrMNC1pU5o |
| project_id | a8aa6796f882481ab0b602334bc37090
                                                                   |
| user_id    | dfd8959d042f43bd8190a4b5d03054f0
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
```

Figure 4: Keystone Generate Token



```
stan@controller:~$ unset OS_AUTH_URL OS_PASSWORD
stan@controller:~$ openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name admin --os-username admin token issue
Password:
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| Field      | Value
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| expires    | 2024-04-09T20:03:09+0000
                                                                   |
| id         | gAAAAABmFZDtyjXtHhqMJDwxRyemBv63B-uRVB5ivQ2TEKSnVO8VD768ImYllM_ZfHWlO3kDUR46MhY9hWvEReng5dd7ue3bgRTP1gp65T2SKD8B_hPuj
SJOwrcgwTmYxFV4PoKA4dCv4B7MYY7-lHySE3jp3PV5WruKlZLHOOuEs3nixgrzCv8 |
| project_id | 53582a3bd2e24585ae4c7ce4030fad3d
                                                                   |
| user_id    | 60e6f55e020449a38041c568b5b0a0bb
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
stan@controller:~$ openstack --os-auth-url http://controller:5000/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name myproject --os-username stan token issue
Password:
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| Field      | Value
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| expires    | 2024-04-09T20:04:23+0000
                                                                   |
| id         | gAAAAABmFZE3EPOEC1Ap3OLrLcZ_vNW0AnAAlSgzOIBAm0LAZT0wWG6l4GxeOQj4lnZLRaNYm9JQNemyLgmMZ2cLltBUaPQkT5vCRZAeH1o0u3ljlYZFi
0yV_4mcxbQJyZN-EyL07VHdx7bGrdoSD_bNEZqrh_0rqUJvsABkyri5UcrMNC1pU5o |
| project_id | a8aa6796f882481ab0b602334bc37090
                                                                   |
| user_id    | dfd8959d042f43bd8190a4b5d03054f0
                                                                   |
+------------+----------------------------------------------------------------------------------------------------------------------
```

Figure 5: Keystone Create User



```
root@controller:/home/stan# openstack user list
+----------------------------------+-------------------+
| ID                               | Name              |
+----------------------------------+-------------------+
| 60e6f55e020449a38041c568b5b0a0bb | admin             |
| dfd8959d042f43bd8190a4b5d03054f0 | stan              |
| d1af9c41eaee4de4b4d829751e3f3ae4 | glance            |
| 1b5ac9ff40e64aefa9f3fed8c9bc0d24 | placement         |
| 066fbfec0b574733a2b9874e7c7a4a20 | nova              |
| 85bfb3ed939d4483ad964167c0534c85 | neutron           |
| c0702717bf984124b1085fe43ae01c43 | cinder            |
| 500432e60c7e4335b59ac9c833359a85 | swift             |
| 86c62a7197c249f7878a3be31bc5c581 | demo              |
| ca77196f2e2044caa85823fdbb543a12 | heat              |
| d254acbde359468083053fb65280b6a6 | heat_domain_admin |
+----------------------------------+-------------------+
root@controller:/home/stan# openstack role list
+----------------------------------+------------------+
| ID                               | Name             |
+----------------------------------+------------------+
| 051d62f1bfae43e0a59bfd160ababf02 | myrole           |
| 239ddf31c5424d899ac0c2def7a19f78 | heat_stack_owner |
| 50394b14554e4929829d05d799bac525 | member           |
| bb7566bc7b844fb6877d521ccd6e0917 | admin            |
| c3c9febf2aec4aae93c475652e95341c | reader           |
+----------------------------------+------------------+
root@controller:/home/stan# 
```

Figure 6: Keystone Retriever Users and Roles

Note: We initially went with minimal deployment for Openstack Antelope following the guide. Only after installing minimal features for deployment, we worked the grading requirements. That is why Figure 5 has those users and roles.

Glance

After installing Keystone, we started setting up the remaining services. Glance is the image service provided by OpenStack. The steps were performed in the order they appear.
Create glance database:

```
mysql
MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
exit;
```

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

```
. admin-openrc
```

Create the glance user:

```
openstack user create --domain default --password-prompt glance
```

Add the **admin** role to the **glance** user and **service** project:

```
openstack role add --project service --user glance admin
```

Create the **glance** service entity:

```
openstack service create --name glance \
 --description "OpenStack Image" image
```

Create the Image service API endpoints:

```
openstack endpoint create --region RegionOne \
  image public http://controller:9292
openstack endpoint create --region RegionOne \
  image internal http://controller:9292
openstack endpoint create --region RegionOne \
  image admin http://controller:9292
```

Install the packages:

```
apt install glance
```

Edit the **/etc/glance/glance-api.conf** file to have the following lines:

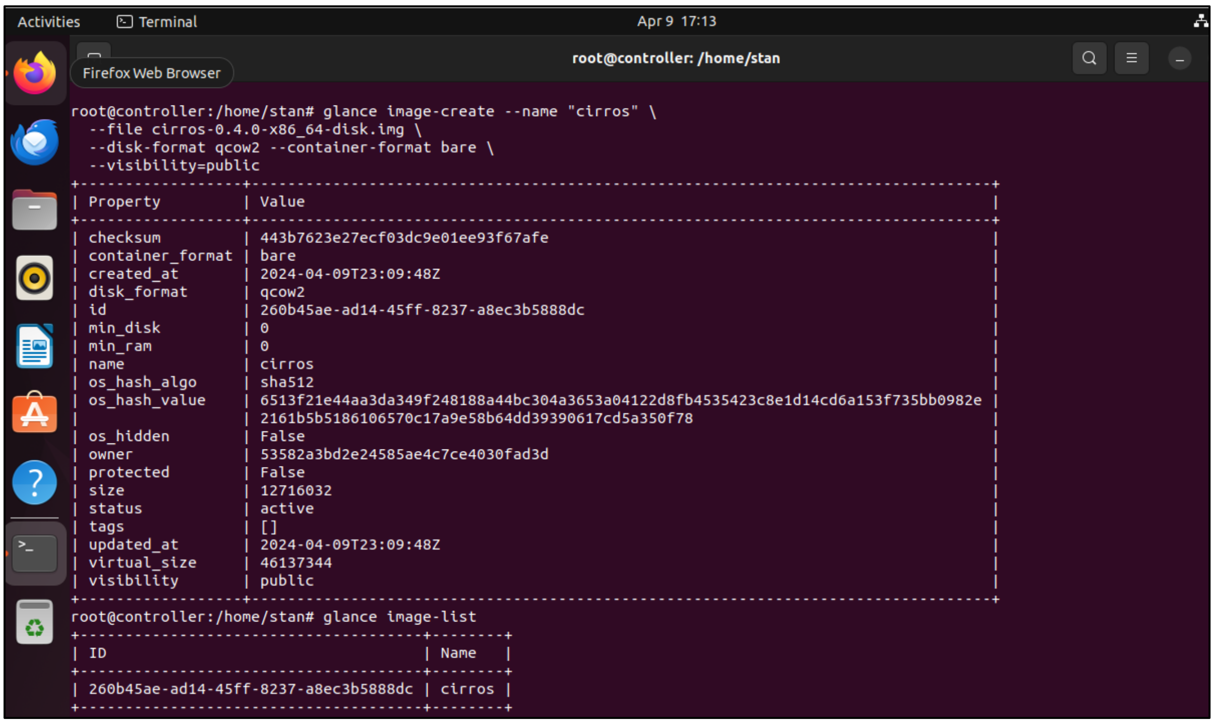
```
[database]
# ...
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
[keystone_authtoken]
# ...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
# ...
flavor = keystone

[glance_store]
# ...
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

[oslo_limit]
auth_url = http://controller:5000
auth_type = password
user_domain_id = default
username = glance
system_scope = all
password = GLANCE_PASS
endpoint_id = 340be3625e9b4239a6415d034e98aace
region_name = RegionOne
```

Make sure that the glance account has reader access to system-scope resources (like limits):

```
openstack role add --user glance --user-domain Default --system all reader
```

Populate the Image service database, restart the service and verify operation. **This action fulfilled the grading checklist for Cinder.**

```
su -s /bin/sh -c "glance-manage db_sync" glance
. admin-openrc
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
glance image-create --name "cirros" \
  --file cirros-0.4.0-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --visibility=public
glance image-list
```



```
root@controller:/home/stan# glance image-create --name "cirros" \
  --file cirros-0.4.0-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --visibility=public
+------------------+----------------------------------------------------------------------------------+
| Property         | Value                                                                            |
+------------------+----------------------------------------------------------------------------------+
| checksum         | 443b7623e27ecf03dc9e01ee93f67afe                                                 |
| container_format | bare                                                                             |
| created_at       | 2024-04-09T23:09:48Z                                                             |
| disk_format      | qcow2                                                                            |
| id               | 260b45ae-ad14-45ff-8237-a8ec3b5888dc                                             |
| min_disk         | 0                                                                                |
| min_ram          | 0                                                                                |
| name             | cirros                                                                           |
| os_hash_algo     | sha512                                                                           |
| os_hash_value    | 6513f21e44aa3da349f248188a44bc304a3653a04122d8fb4535423c8e1d14cd6a153f735bb0982e |
|                  | 2161b5b5186106570c17a9e58b64dd39390617cd5a350f78                                 |
| os_hidden        | False                                                                            |
| owner            | 53582a3bd2e24585ae4c7ce4030fad3d                                                 |
| protected        | False                                                                            |
| size             | 12716032                                                                         |
| status           | active                                                                           |
| tags             | []                                                                               |
| updated_at       | 2024-04-09T23:09:48Z                                                             |
| virtual_size     | 46137344                                                                         |
| visibility       | public                                                                           |
+------------------+----------------------------------------------------------------------------------+
root@controller:/home/stan# glance image-list
+--------------------------------------+--------+
| ID                                   | Name   |
+--------------------------------------+--------+
| 260b45ae-ad14-45ff-8237-a8ec3b5888dc | cirros |
+--------------------------------------+--------+
```

Figure 7: Cirros Image Upload and Retrieve Image List

Placement

Then we moved on to installing Placement. For most services, the initial steps are consistent—creating databases, establishing service credentials, and setting up endpoints.

```
mysql
MariaDB [(none)]> CREATE DATABASE placement;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO
'placement'@'localhost' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
  IDENTIFIED BY 'PLACEMENT_DBPASS';
exit;
```

```
. admin-openrc
openstack user create --domain default --password-prompt placement
openstack role add --project service --user placement admin
openstack service create --name placement \
  --description "Placement API" placement
openstack endpoint create --region RegionOne \
  placement public http://controller:8778
openstack endpoint create --region RegionOne \
  placement internal http://controller:8778
openstack endpoint create --region RegionOne \
  placement admin http://controller:8778
```

Install the packages:

```
apt install placement-api
```

Edit the **/etc/placement/placement.conf** file:

```
[placement_database]
# ...
connection =
mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
```

Populate the placement database and finalize installation:

```
su -s /bin/sh -c "placement-manage db sync" placement
service apache2 restart
```

Verify operation of the placement service:

```
. admin-openrc
placement-status upgrade check
```

Nova

After configuring the Placement Service, we moved on to installing nova. Again, the initial steps are consistent—creating databases, establishing service credentials, and setting up endpoints.

Configure Control Node

First setup the control node.

```
mysql
MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
 IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
 IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
 IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
 IDENTIFIED BY 'NOVA_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' \
 IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' \
 IDENTIFIED BY 'NOVA_DBPASS';
```

```
. admin-openrc
openstack user create --domain default --password-prompt nova
openstack role add --project service --user nova admin
openstack service create --name nova \
  --description "OpenStack Compute" compute
openstack endpoint create --region RegionOne \
  compute public http://controller:8774/v2.1
openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1
openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1
```

zxczxc

Install the packages:

```
apt install nova-api nova-conductor nova-novncproxy nova-scheduler
```

Edit the **/etc/nova/nova.conf** file and complete the following actions:

```
[api_database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
# ...
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
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
project_name = service
username = nova
password = NOVA_PASS
```

```
[service_user]
send_service_user_token = true
auth_url = https://controller/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = NOVA_PASS
[DEFAULT]
# ...
my_ip = 192.168.72.3
[vnc]
enabled = true
# ...
server_listen = $my_ip
server_proxyclient_address = $my_ip
[glance]
# ...
api_servers = http://controller:9292
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

Populate the **nova-api** database:

```
su -s /bin/sh -c "nova-manage api_db sync" nova
```

Register the **cell0** database:

```
su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
```

Create the **cell1** cell:

```
su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
```

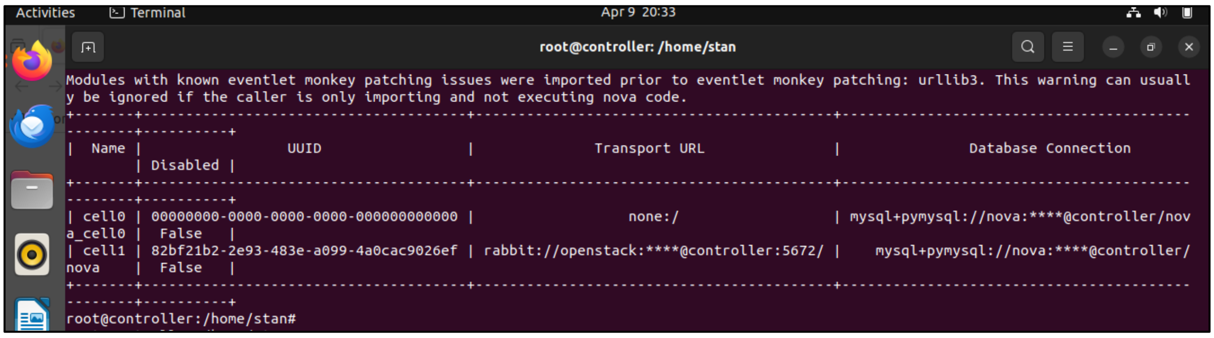
Populate the nova database:

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

Verify nova cell0 and cell1 are registered correctly.

```
su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

Figure 8 verifies that the nova cell0 and cell1 have registered correctly.



```
Modules with known eventlet monkey patching issues were imported prior to eventlet monkey patching: urllib3. This warning can usuall
y be ignored if the caller is only importing and not executing nova code.
+-------+--------------------------------------+-------------------------------------------+------------------------------------------
--------+----------+
|  Name |                 UUID                 |               Transport URL               |            Database Connection
        | Disabled |
+-------+--------------------------------------+-------------------------------------------+------------------------------------------
--------+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                  none:/                   | mysql+pymysql://nova:****@controller/nov
a_cell0 |  False   |
| cell1 | 82bf21b2-2e93-483e-a099-4a0cac9026ef | rabbit://openstack:****@controller:5672/ |    mysql+pymysql://nova:****@controller/
nova    |  False   |
+-------+--------------------------------------+-------------------------------------------+------------------------------------------
--------+----------+
root@controller:/home/stan#
```

Figure 8: Nova Cell Registration Successful

Finalize installation by restarting controller services:

```
service nova-api restart
service nova-scheduler restart
 service nova-conductor restart
service nova-novncproxy restart
```

Configure Compute Node

Now setup the compute node for Nova.

Install the packages:

```
apt install nova-compute
```

Edit the /etc/nova/nova.conf file:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
[api]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
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
project_name = service
username = nova
password = NOVA_PASS
[service_user]
send_service_user_token = true
auth_url = https://controller/identity
auth_strategy = keystone
auth_type = password
project_domain_name = Default
project_name = service
user_domain_name = Default
username = nova
password = NOVA_PASS
[DEFAULT]
# ...
my_ip = 192.168.72.4
[vnc]
# ...
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html
[glance]
# ...
api_servers = http://controller:9292
```

```
[oslo_concurrency]
# ...
lock_path = /var/lib/nova/tmp
[placement]
# ...
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS
```

To finalize installation, determine whether our compute node supports hardware acceleration for virtual machines:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

The command returned a value of **zero**. So, our compute node does not support hardware acceleration and we had to configure **libvirt** to use QEMU instead of KVM. Edit the **[libvirt]** section in the **/etc/nova/nova-compute.conf** file as follows:

```
[libvirt]
# ...
virt_type = qemu
```

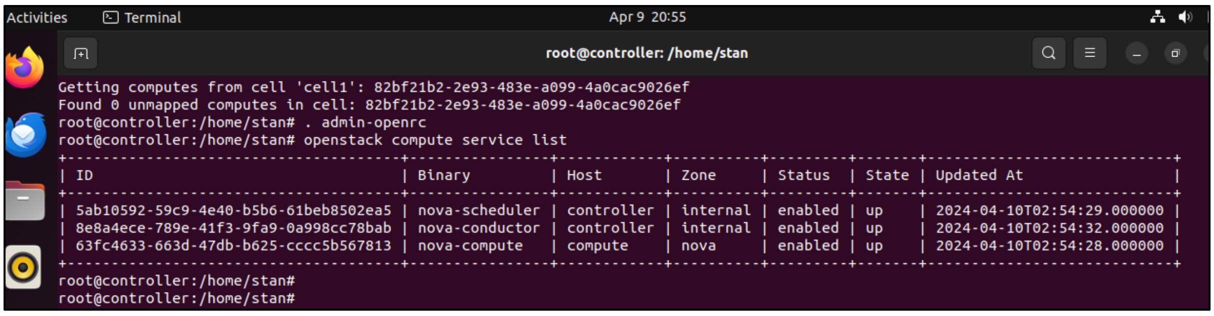
Restart the compute service and Next add the compute node to the cell database:

```
service nova-compute restart
. admin-openrc
openstack compute service list --service nova-compute
su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
```

Finally, verify Nova on compute node.

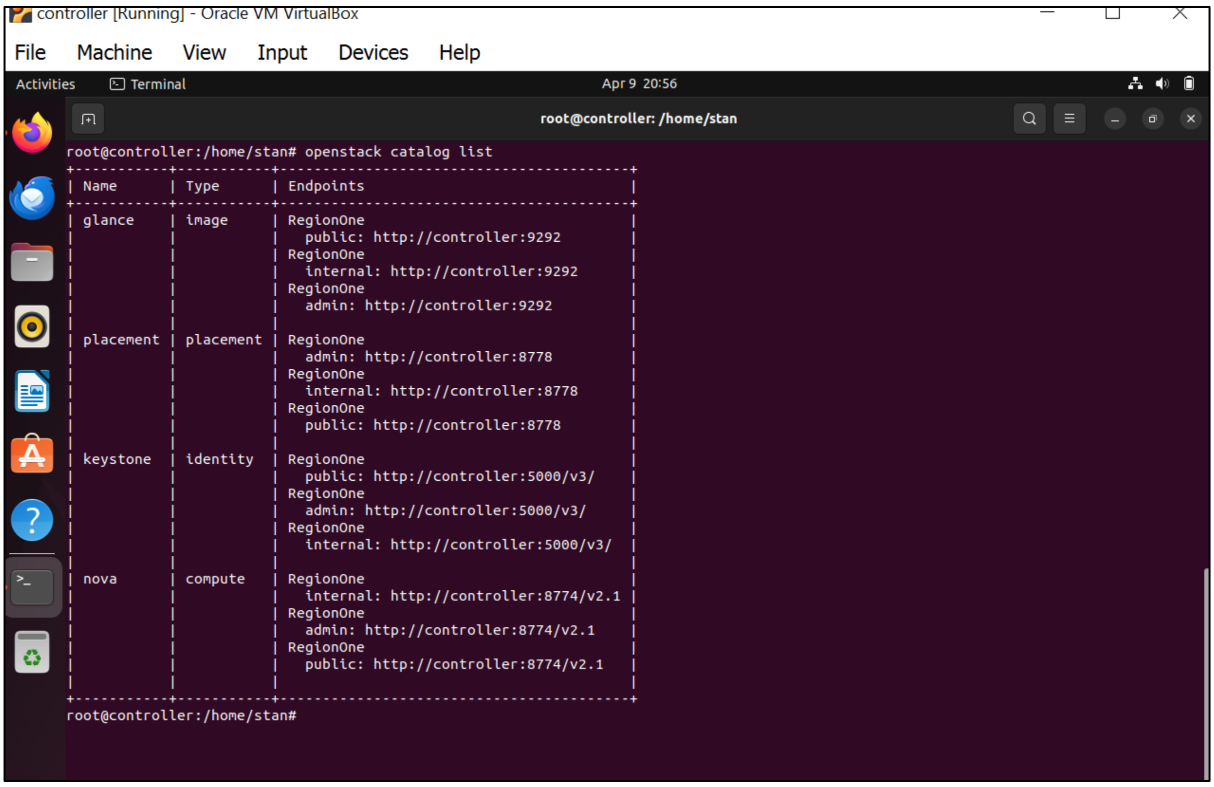
```
. admin-openrc
openstack compute service list
openstack catalog list
openstack image list
```

Figures 9,10, 11 confirm this. This is also shown in Figure 24 and Figure 25. **This action fulfilled the grading checklist for Nova.**



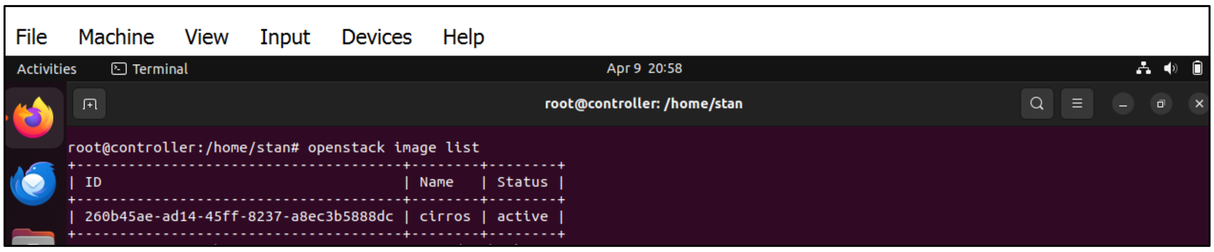
```
Getting computes from cell 'cell1': 82bf21b2-2e93-483e-a099-4a0cac9026ef
Found 0 unmapped computes in cell: 82bf21b2-2e93-483e-a099-4a0cac9026ef
root@controller:/home/stan# . admin-openrc
root@controller:/home/stan# openstack compute service list
+--------------------------------------+----------------+------------+----------+---------+-------+----------------------------+
| ID                                   | Binary         | Host       | Zone     | Status  | State | Updated At                 |
+--------------------------------------+----------------+------------+----------+---------+-------+----------------------------+
| 5ab10592-59c9-4e40-b5b6-61beb8502ea5 | nova-scheduler | controller | internal | enabled | up    | 2024-04-10T02:54:29.000000 |
| 8e8a4ece-789e-41f3-9fa9-0a998cc78bab | nova-conductor | controller | internal | enabled | up    | 2024-04-10T02:54:32.000000 |
| 63fc4633-663d-47db-b625-cccc5b567813 | nova-compute   | compute    | nova     | enabled | up    | 2024-04-10T02:54:28.000000 |
+--------------------------------------+----------------+------------+----------+---------+-------+----------------------------+
root@controller:/home/stan#
root@controller:/home/stan#
```

Figure 9: Compute Service List


controller [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
Activities
Terminal
Apr 9 20:56
root@controller: /home/stan

```
root@controller:/home/stan# openstack catalog list
+-----------+-----------+-----------------------------------------+
| Name      | Type      | Endpoints                               |
+-----------+-----------+-----------------------------------------+
| glance    | image     | RegionOne                               |
|           |           |   public: http://controller:9292        |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:9292      |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:9292         |
|           |           |                                         |
| placement | placement | RegionOne                               |
|           |           |   admin: http://controller:8778         |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:8778      |
|           |           | RegionOne                               |
|           |           |   public: http://controller:8778        |
|           |           |                                         |
| keystone  | identity  | RegionOne                               |
|           |           |   public: http://controller:5000/v3/    |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:5000/v3/     |
|           |           | RegionOne                               |
|           |           |   internal: http://controller:5000/v3/  |
|           |           |                                         |
| nova      | compute   | RegionOne                               |
|           |           |   internal: http://controller:8774/v2.1 |
|           |           | RegionOne                               |
|           |           |   admin: http://controller:8774/v2.1    |
|           |           | RegionOne                               |
|           |           |   public: http://controller:8774/v2.1   |
|           |           |                                         |
+-----------+-----------+-----------------------------------------+
root@controller:/home/stan#
```

Figure 10: Catalog List

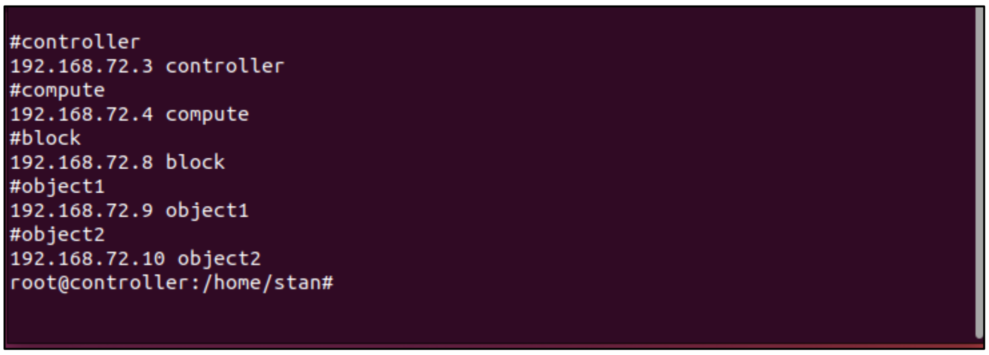


```
root@controller:/home/stan# openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 260b45ae-ad14-45ff-8237-a8ec3b5888dc | cirros | active |
+--------------------------------------+--------+--------+
```

Figure 11: Image List

Neutron

Before we could create a virtual machine as required by the homework instructions, we needed to install a networking service. First, we needed to configure name resolutions. In our case, much of this was done during the initial network setup by configuring the /etc/hosts file on the controller, compute, and block nodes. Figure 12 shows the configuration.



```
#controller
192.168.72.3 controller
#compute
192.168.72.4 compute
#block
192.168.72.8 block
#object1
192.168.72.9 object1
#object2
192.168.72.10 object2
root@controller:/home/stan#
```

Figure 12: Name Resolution Configuration

After verifying all nodes capability to talk to each other by ping-ing each other and websites on the internet, we moved forward to install Neutron.

Configure Control Node

```
mysql -u root -p
MariaDB [(none)]> CREATE DATABASE neutron;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
\
  IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
exit;
```

```
. admin-openrc
openstack user create --domain default --password-prompt neutron
openstack role add --project service --user neutron admin
openstack service create --name neutron \
  --description "OpenStack Networking" network
openstack endpoint create --region RegionOne \
  network public http://controller:9696
openstack endpoint create --region RegionOne \
  network internal http://controller:9696
openstack endpoint create --region RegionOne \
  network admin http://controller:9696
```

Then we chose our networking option of choice. We decided to go with Provider Network.
First, install components:

```
apt install neutron-server neutron-plugin-ml2 \
  neutron-openvswitch-agent neutron-dhcp-agent \
  neutron-metadata-agent
```

Edit the **/etc/neutron/neutron.conf** file:

```
[database]
# ...
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
[DEFAULT]
# ...
core_plugin = ml2
service_plugins =
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller
[DEFAULT]
# ...
auth_strategy = keystone
```

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

[keystone_authtoken]
# ...
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
project_name = service
username = neutron
password = NEUTRON_PASS
[DEFAULT]
# ...
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true

[nova]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

Edit the **/etc/neutron/plugins/ml2/ml2_conf.ini** file:

```
[ml2]
# ...
type_drivers = flat,vlan
[ml2]
# ...
tenant_network_types =
[ml2]
# ...
mechanism_drivers = openvswitch
[ml2]
# ...
extension_drivers = port_security
[ml2_type_flat]
# ...
flat_networks = provider
```

Edit the **/etc/neutron/plugins/ml2/openvswitch_agent.ini** file:

```
[ovs]
bridge_mappings = provider:PROVIDER_INTERFACE_NAME
[securitygroup]
# ...
enable_security_group = true
firewall_driver = openvswitch
```

Edit the **/etc/neutron/dhcp_agent.ini** file:

```
[DEFAULT]
# ...
interface_driver = openvswitch
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
```

After this we came back to installing Neutron.

Edit the **/etc/neutron/metadata_agent.ini** file:

```
[DEFAULT]
# ...
nova_metadata_host = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Next Configure the Compute service to use the Networking service.
Edit the **/etc/nova/nova.conf** file:

```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET
```

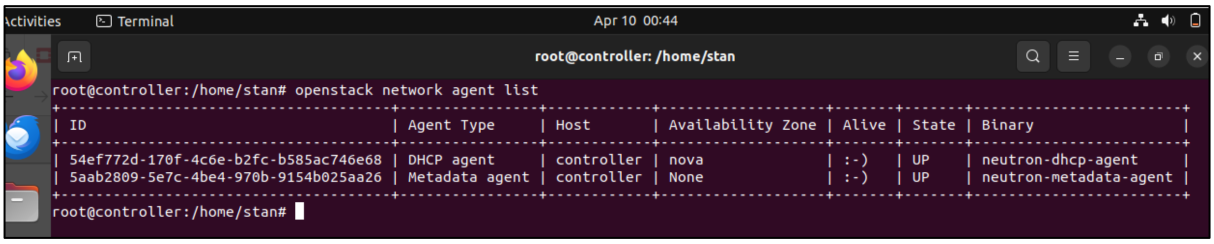
Populate the database, restart the Compute API service, and restart the networking services:

```
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
service nova-api restart
service neutron-server restart
 service neutron-openvswitch-agent restart
 service neutron-dhcp-agent restart
 service neutron-metadata-agent restart
```

Finally, list agents to verify successful launch of the neutron agents:

```
openstack network agent list
```

Figure 12 verifies the successful launch of neutron agents.



```
root@controller:/home/stan# openstack network agent list
+--------------------------------------+----------------+------------+-------------------+-------+-------+------------------------+
| ID                                   | Agent Type     | Host       | Availability Zone | Alive | State | Binary                 |
+--------------------------------------+----------------+------------+-------------------+-------+-------+------------------------+
| 54ef772d-170f-4c6e-b2fc-b585ac746e68 | DHCP agent     | controller | nova              | :-)   | UP    | neutron-dhcp-agent     |
| 5aab2809-5e7c-4be4-970b-9154b025aa26 | Metadata agent | controller | None              | :-)   | UP    | neutron-metadata-agent |
+--------------------------------------+----------------+------------+-------------------+-------+-------+------------------------+
root@controller:/home/stan#
```

Figure 13: Neutron Agent List

Configure Compute Node

Next step is to configure the compute node.

```
apt install neutron-openvswitch-agent
```

Edit the **/etc/neutron/neutron.conf** file:

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

[keystone_authtoken]
# ...
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
project_name = service
username = neutron
password = NEUTRON_PASS
[oslo_concurrency]
# ...
lock_path = /var/lib/neutron/tmp
```

Edit the **/etc/nova/nova.conf** file:

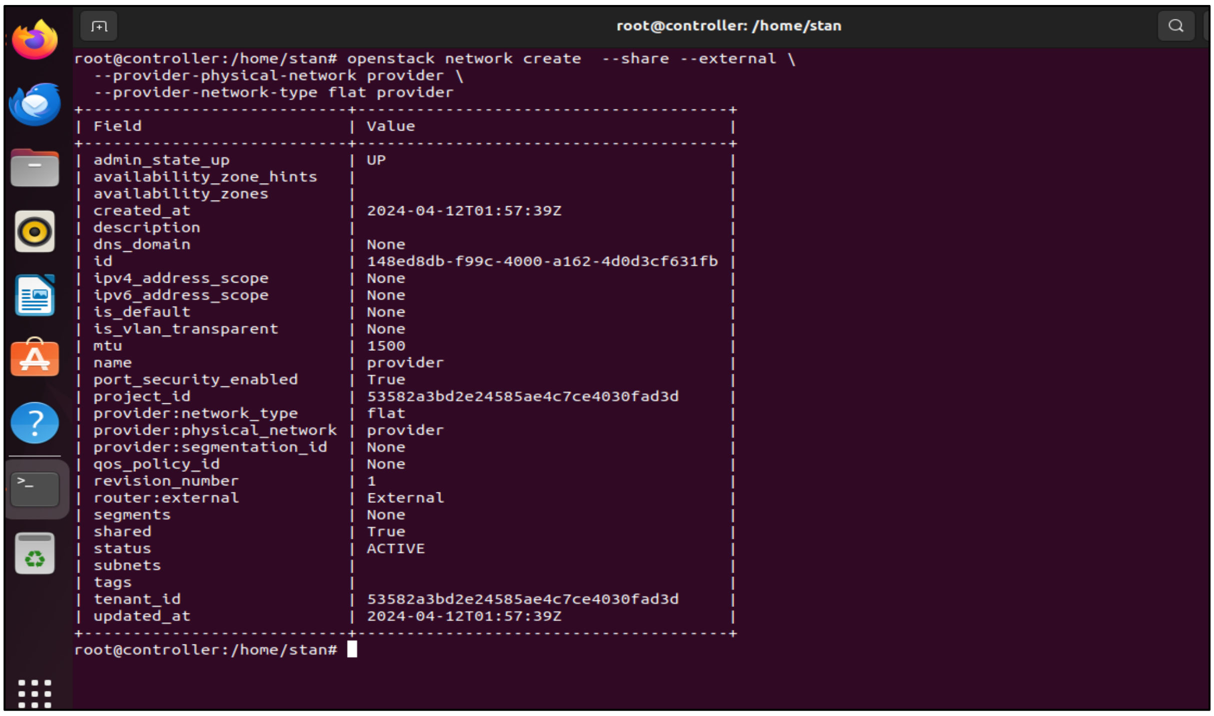
```
[neutron]
# ...
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Restart the Compute service and the Linux bridge agent:

```
service nova-compute restart
service neutron-openvswitch-agent restart
```

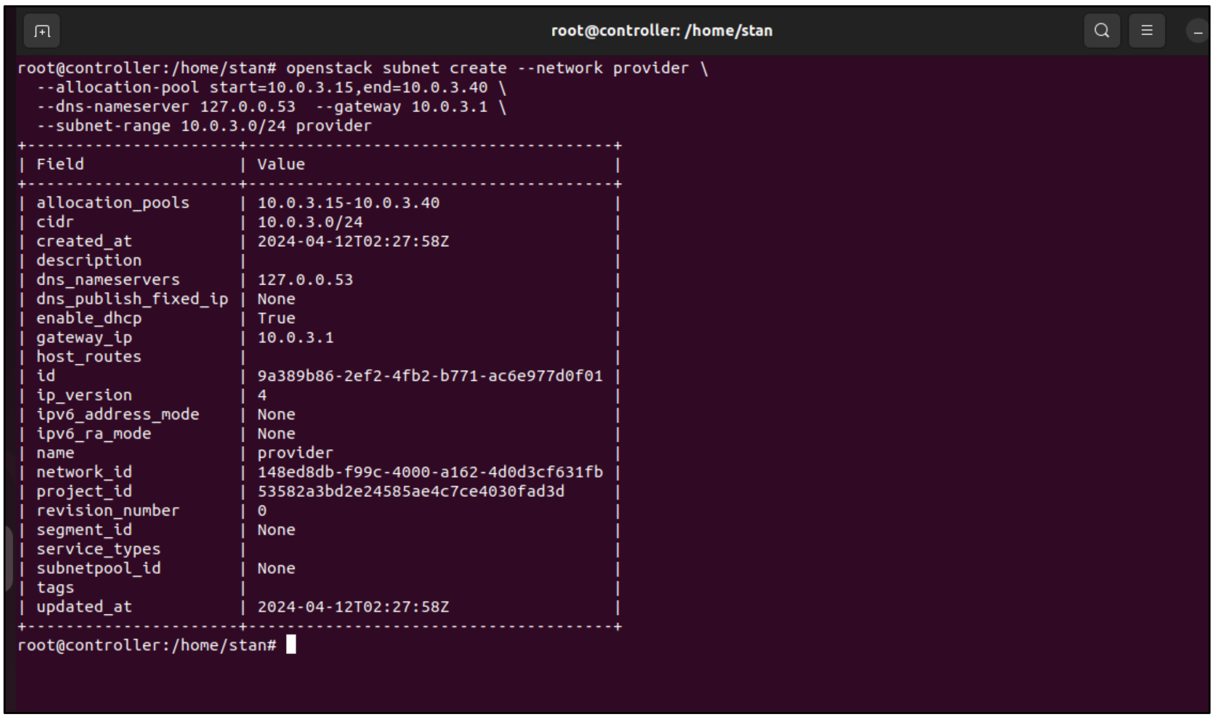
This now allows us to create a network and subnet for use with our images. In this example we have named the network as provider as shown in Figure 14, 15. This satisfies the grading requirement for Neutron. **This action fulfilled the grading checklist for Cinder.**

```
openstack network agent list
openstack network create  --share --external \
  --provider-physical-network provider \
  --provider-network-type flat provider
openstack subnet create --network provider \
  --allocation-pool start= 10.0.3.15 end=10.0.3.40 \
  --dns-nameserver 127.0.0.53 --gateway 10.0.3.1 \
  --subnet-range 10.0.3.0/24 provider
```



```
root@controller:/home/stan# openstack network create  --share --external \
  --provider-physical-network provider \
  --provider-network-type flat provider
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2024-04-12T01:57:39Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 148ed8db-f99c-4000-a162-4d0d3cf631fb |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | provider                             |
| port_security_enabled     | True                                 |
| project_id                | 53582a3bd2e24585ae4c7ce4030fad3d     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 53582a3bd2e24585ae4c7ce4030fad3d     |
| updated_at                | 2024-04-12T01:57:39Z                 |
+---------------------------+--------------------------------------+
root@controller:/home/stan#
```

Figure 14: Neutron Network Creation



```
root@controller:/home/stan# openstack subnet create --network provider \
  --allocation-pool start=10.0.3.15,end=10.0.3.40 \
  --dns-nameserver 127.0.0.53  --gateway 10.0.3.1 \
  --subnet-range 10.0.3.0/24 provider
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 10.0.3.15-10.0.3.40                  |
| cidr                 | 10.0.3.0/24                          |
| created_at           | 2024-04-12T02:27:58Z                 |
| description          |                                      |
| dns_nameservers      | 127.0.0.53                           |
| dns_publish_fixed_ip | None                                 |
| enable_dhcp          | True                                 |
| gateway_ip           | 10.0.3.1                             |
| host_routes          |                                      |
| id                   | 9a389b86-2ef2-4fb2-b771-ac6e977d0f01 |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | provider                             |
| network_id           | 148ed8db-f99c-4000-a162-4d0d3cf631fb |
| project_id           | 53582a3bd2e24585ae4c7ce4030fad3d     |
| revision_number      | 0                                    |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | None                                 |
| tags                 |                                      |
| updated_at           | 2024-04-12T02:27:58Z                 |
+----------------------+--------------------------------------+
root@controller:/home/stan#
```

Figure 15: Neutron Network Creationg 2

Horizon

So far, we have done everything in terminal. However, Horizon gives us a GUI dashboard through which we can create instances, manage users, and install images.

Install the packages:

```
apt install openstack-dashboard
```

Edit the **/etc/openstack-dashboard/local_settings.py** file:

```
apt install openstack-dashboard
OPENSTACK_HOST = "controller"
ALLOWED_HOSTS = [*]
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

CACHES = {
  'default': {
     'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
     'LOCATION': 'controller:11211',
  }
}
OPENSTACK_KEYSTONE_URL = "http://%s/identity/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
  "identity": 3,
  "image": 2,
  "volume": 3,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_NEUTRON_NETWORK = {
  ...
  'enable_router': False,
  'enable_quotas': False,
  'enable_ipv6': False,
  'enable_distributed_router': False,
  'enable_ha_router': False,
  'enable_fip_topology_check': False,
}
TIME_ZONE = "TIME_ZONE"
```

Add the following line to **/etc/apache2/conf-available/openstack-dashboard.conf**

```
WSGIApplicationGroup %{GLOBAL}
```

Finalize the installation:

```
systemctl reload apache2.service
```

We then verified the operation by going to `http://controller/horizon/` on our browser within the VirtualBox as well as outside the VirtualBox as shown in Figure 16, 17. **This action fulfilled the grading checklist for Cinder.**

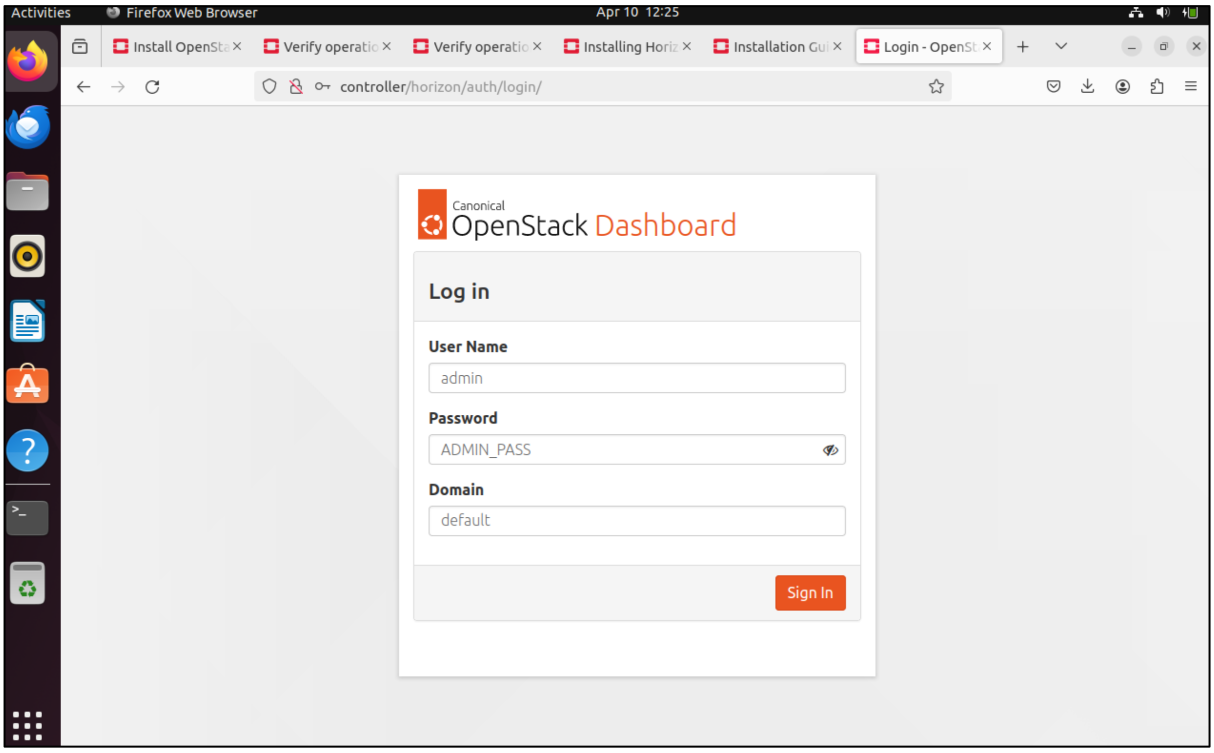


Figure 16: Horizon Login

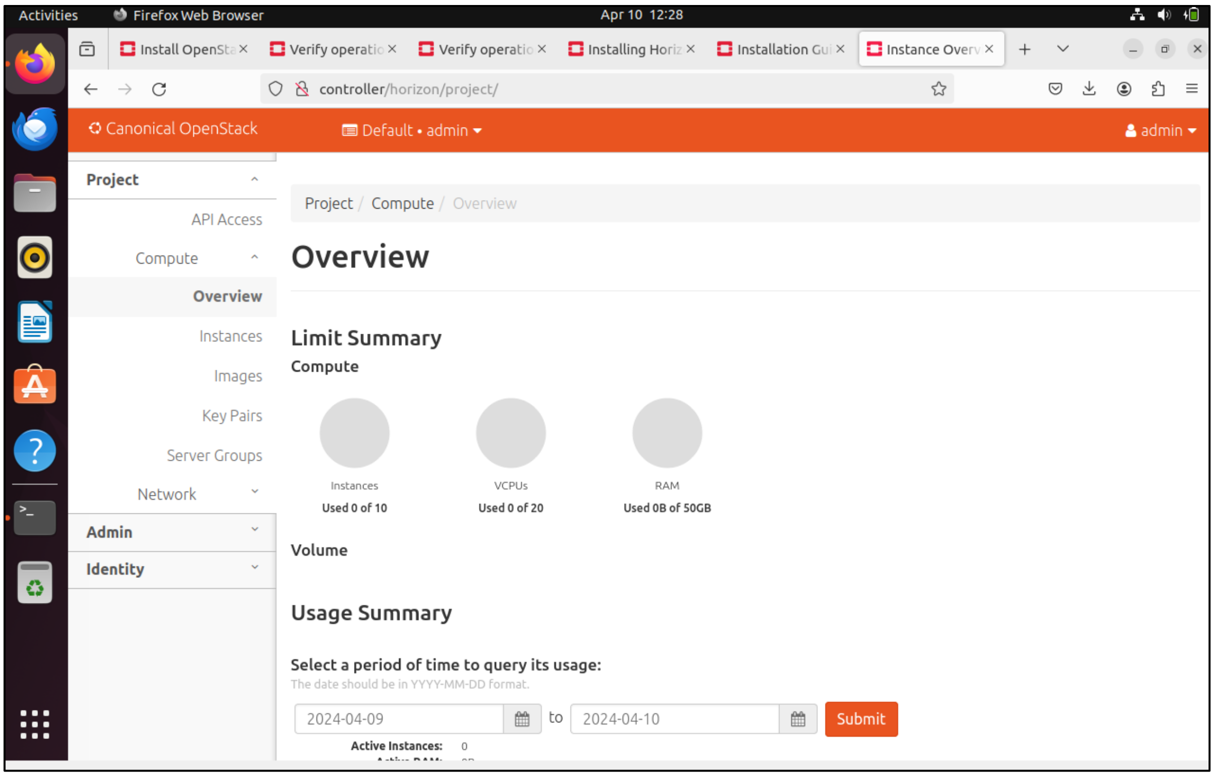


Figure 17: Horizon Login Success

Cinder

This section outlines the steps to set up and configure storage nodes for the block storage service on the block node. Like other nodes, this process includes configuring MariaDB and Keystone.

```
mysql
MariaDB [(none)]> CREATE DATABASE cinder;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
  IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
  IDENTIFIED BY 'CINDER_DBPASS';
exit;
```

```
. admin-openrc
openstack user create --domain default --password-prompt cinder
openstack role add --project service --user cinder admin
openstack service create --name cinderv3 \
  --description "OpenStack Block Storage" volumev3
openstack endpoint create --region RegionOne \
  volumev3 public http://controller:8776/v3/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev3 internal http://controller:8776/v3/%\(project_id\)s
openstack endpoint create --region RegionOne \
  volumev3 admin http://controller:8776/v3/%\(project_id\)s
```

On our controller node, we install and configure the Cinder service by modifying the config file **/etc/cinder/cinder.conf**

```
apt install cinder-api cinder-scheduler
```

```
[database]
# ...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

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

[DEFAULT]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
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
project_name = service
username = cinder
password = Nepal123!

[DEFAULT]
# ...
my_ip = 192.168.72.3

[oslo_concurrency]
# ...
lock_path = /var/lib/cinder/tmp
```

We populate the database with these changes.

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

Next, we set up the compute node for Cinder by making changes to the /etc/nova/nova.conf file.

```
[cinder]
os_region_name = RegionOne
```

Restarted the compute api and block storage services

```
# service nova-api restart
# service cinder-scheduler restart
# service apache2 restart
```

The subsequent actions took place on the block1 node. We installed a local volume management tool, established an LVM physical volume, set up volume groups, and configured permissions.

```
# apt install lvm2 thin-provisioning-tools
# pvcreate /dev/sdb
# vgcreate cinder-volumes /dev/sdb
```

Edited the filr /etc/lvm/lvm.conf

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

After creating the physical volume, installing cinder on block node.

```
# apt install cinder-volume tgt
```

Edited the /etc/cinder/cinder.conf file.

```
[database]
# ...
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder

[DEFAULT]
# ...
auth_strategy = keystone

[keystone_authtoken]
# ...
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
project_name = service
username = cinder
password = Nepal123!

[DEFAULT]
# ...
my_ip = 192.168.72.8

[lvm]
# ...
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
```

```
target_protocol = iscsi
target_helper = tgtadm

[DEFAULT]
# ...
enabled_backends = lvm

[DEFAULT]
# ...
glance_api_servers = http://controller:9292
[oslo_concurrency]
# ...
lock_path = /var/lib/cinder/tmp
```

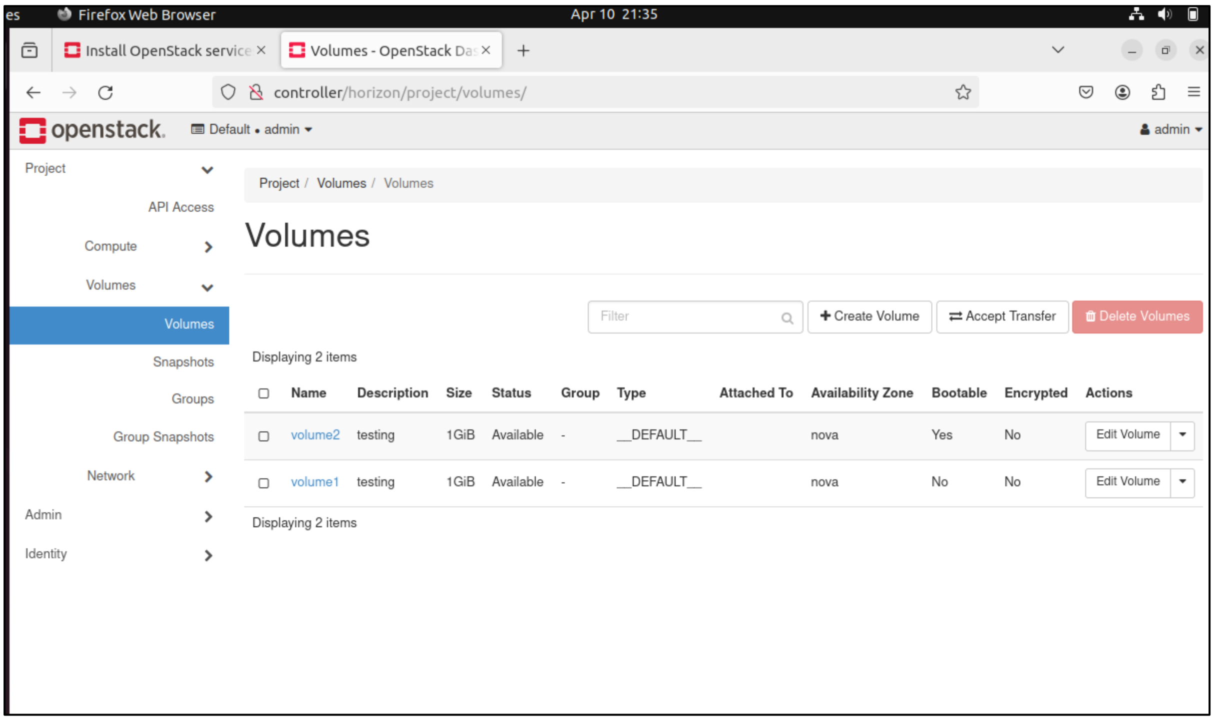
Once these changes are done, restarting the block storage volume service with its dependencies.

```
# service tgt restart
# service cinder-volume restart
```

After the services were restarted, we verified the successful launch of each process:

```
. admin-openrc
openstack volume service list
```

After restarting the services, we created a volume through the newly installed Horizon dashboard. **This action fulfilled the grading checklist for Cinder.**



Volumes

Displaying 2 items

	Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
☐	volume2	testing	1GiB	Available	-	__DEFAULT__		nova	Yes	No	Edit Volume
☐	volume1	testing	1GiB	Available	-	__DEFAULT__		nova	No	No	Edit Volume

Displaying 2 items

Figure 18: Cinder Create and List Volumes

Swift

The final service we needed to set up was Swift, OpenStack's container service. Using the same methods for database and Keystone user creation as with the other services, we established two additional nodes for object storage. Similar to the block storage node, each of these nodes required the creation of two physical volumes. The use of multiple nodes and volumes enables us to distribute our loads according to demand.

```
. admin-openrc
openstack user create --domain default --password-prompt swift
openstack role add --project service --user swift admin
openstack service create --name swift \
 --description "OpenStack Object Storage" object-store
openstack endpoint create --region RegionOne \
 object-store public http://controller:8080/v1/AUTH_%\(project_id\)s
openstack endpoint create --region RegionOne \
 object-store internal http://controller:8080/v1/AUTH_%\(project_id\)s
openstack endpoint create --region RegionOne \
 object-store admin http://controller:8080/v1
```

Installing the packages

```
apt-get install swift swift-proxy python3-swiftclient \
 python3-keystoneclient python3-keystonemiddleware \
 memcached
```

Created a /etc/swift directory and obtained the proxy service configuration file from the Object Storage source repository.

```
mkdir /etc/swift
curl -o /etc/swift/proxy-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/proxy-server.conf-sample
```

After that edited the config file /etc/swift/proxy-server.conf

```
[DEFAULT]
...
bind_port = 8080
user = swift
swift_dir = /etc/swift
[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache container_sync bulk
ratelimit authtoken keystoneauth container-quotas account-quotas slo dlo
versioned_writes proxy-logging proxy-server

[app:proxy-server]
use = egg:swift#proxy
```

```
...
account_autocreate = True

[filter:keystoneauth]
use = egg:swift#keystoneauth
...
operator_roles = admin,user

[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = swift
password = Nepal123!
delay_auth_decision = True [filter:cache]
use = egg:swift#memcache
...
memcache_servers = controller:11211
```

This concluded the setup of the controller node, leading us to the establishment of the separate object storage nodes. Given that each node was to be identical, we configured one node completely, then cloned it. We then changed the hostname and updated all configuration files with the appropriate IP addresses.

```
apt-get install xfsprogs rsync
mkfs.xfs /dev/sdb
mkfs.xfs /dev/sdc
mkdir -p /srv/node/sdb
mkdir -p /srv/node/sdc

blkid
UUID="<see-the-picture-below>" /srv/node/sdb xfs noatime 0 2
UUID="<see-the-picture-below>" /srv/node/sdc xfs noatime 0 2

mount /srv/node/sdb
mount /srv/node/sdc
```

Created the /etc/rsyncd.conf file and added the following and making sure to adjust the IP based on which object node we were working

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
```

```
pid file = /var/run/rsyncd.pid
address = 192.168.72.8

[account]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = False
lock file = /var/lock/object.lock
```

After that we edited the /etc/default/rsync and enable the rsync service:

```
RSYNC_ENABLE=true
```

Started the rsync service and installing the packages needed.

```
service rsync start
apt-get install swift swift-account swift-container swift-object
curl -o /etc/swift/account-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/account-server.conf-
sample
curl -o /etc/swift/container-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/container-server.conf-
sample
curl -o /etc/swift/object-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/object-server.conf-sample
```

Edited the /etc/swift/account-server.conf

```
[DEFAULT]
…
bind_ip = 192.168.72.8
bind_port = 6202
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True
```

```
[pipeline:main]
pipeline = healthcheck recon account-server

[filter:recon]
use = egg:swift#recon
…
recon_cache_path = /var/cache/swift
```

Edited the /etc/swift/container-server.conf

```
[DEFAULT]
…
bind_ip = 192.168.72.8
bind_port = 6201
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True

[pipeline:main]
pipeline = healthcheck recon container-server

[filter:recon]
use = egg:swift#recon
…
recon_cache_path = /var/cache/swift
```

Edited the /etc/swift/object-server.conf

```
[DEFAULT]
…
bind_ip = 192.168.72.8
bind_port = 6200
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = True

[pipeline:main]
pipeline = healthcheck recon object-server

[filter:recon]
use = egg:swift#recon
…
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

We ensured proper ownership of the mount point directory structure and after that created the recon directory and ensured proper ownership of it.

```
chown -R swift:swift /srv/node
mkdir -p /var/cache/swift
chown -R root:swift /var/cache/swift
chmod -R 775 /var/cache/swift
```

After properly configuring the nodes, we proceeded to generate and distribute the initial account, container, and object rings. The ring builder tool creates configuration files that dictate the storage architecture for each node. We ran the ring builder on each physical volume of every object storage node, totaling four executions. Following this, we checked the contents of the ring and rebalanced it. These actions were repeated across the object server, container server, and account server.

```
cd /etc/swift
swift-ring-builder account.builder create 10 3 1
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.72.9--port 6202 --device sdb --weight 100
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.72.9--port 6202 --device sdc --weight 100
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.72.10--port 6202 --device sdb --weight 100
swift-ring-builder account.builder add \
  --region 1 --zone 1 --ip 192.168.72.10--port 6202 --device sdc --weight 100
swift-ring-builder account.builder
swift-ring-builder account.builder rebalance

swift-ring-builder container.builder create 10 3 1
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.72.9 --port 6201 --device sdb --weight 100
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.72.9 --port 6201 --device sdc --weight 100
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.72.10 --port 6201 --device sdb --weight 100
swift-ring-builder container.builder add \
  --region 1 --zone 1 --ip 192.168.72.10 --port 6201 --device sdc --weight 100
swift-ring-builder container.builder
swift-ring-builder container.builder rebalance

swift-ring-builder object.builder create 10 3 1
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.72.9 --port 6200 --device sdb --weight 100
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.72.9 --port 6200 --device sdc --weight 100
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.72.10 --port 6200 --device sdb --weight 100
swift-ring-builder object.builder add \
  --region 1 --zone 1 --ip 192.168.72.10 --port 6200 --device sdc --weight 100
```

```
swift-ring-builder object.builder
swift-ring-builder object.builder rebalance
```

After distributing the ring configuration files—account.ring.gz, container.ring.gz, and object.ring.gz—to the /etc/swift directory on each storage node, we moved on to finalize the installation. We retrieved and modified the /etc/swift/swift.conf file from the object storage source repository. Once the modifications were made as detailed below, we replicated this file across all the storage nodes.

```
curl -o /etc/swift/swift.conf \
  https://opendev.org/openstack/swift/raw/branch/master/etc/swift.conf-sample
```

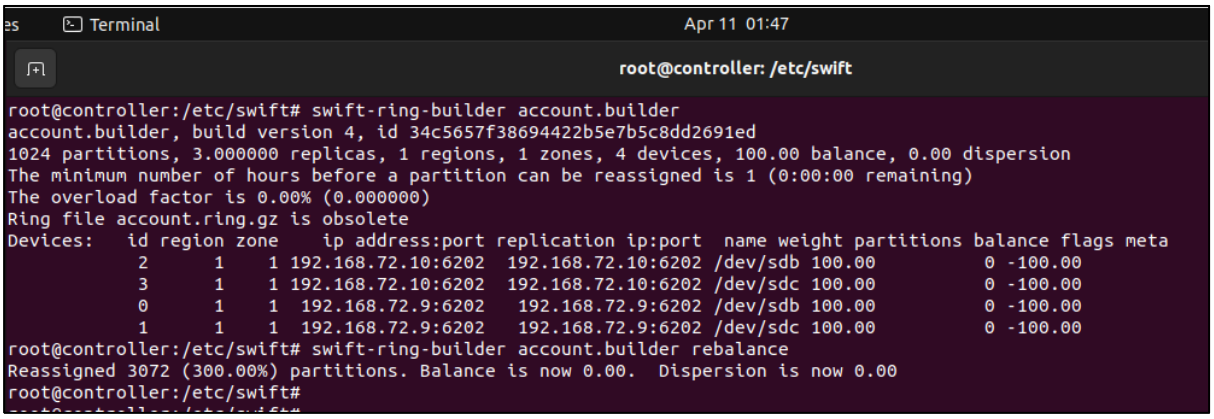
Edited the /etc/swift/swift.conf

```
[swift-hash]
...
swift_hash_path_suffix = HASH_PATH_SUFFIX
swift_hash_path_prefix = HASH_PATH_PREFIX

[storage-policy:0]
...
name = Policy-0
default = yes
```

On all nodes, ensure proper ownership of the configuration directory and restarted all storage services.

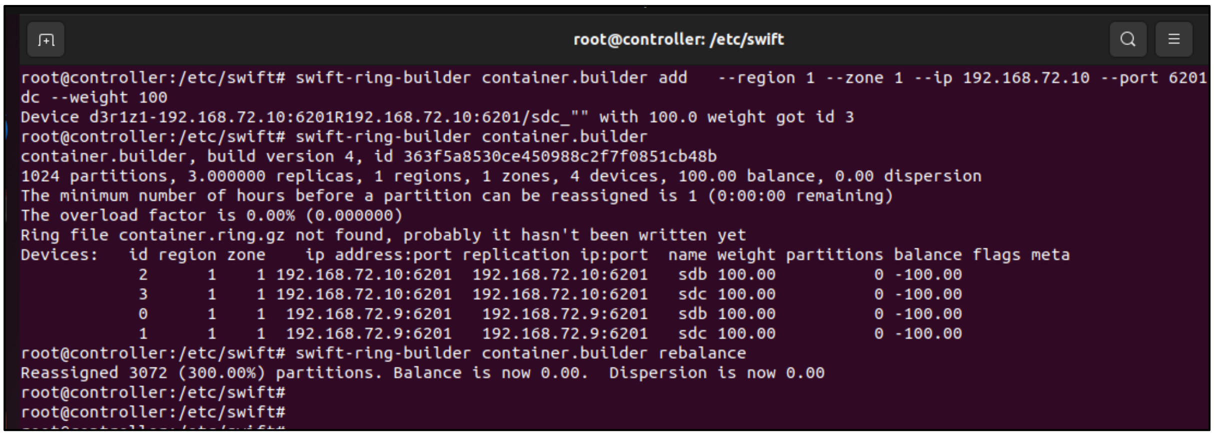
```
chown -R root:swift /etc/swift
service memcached restart
service swift-proxy restart
swift-init all start
```

Regrettably, following the restart, both object nodes experienced corruption and were unable to boot into an operating system. Consequently, we couldn't perform tasks such as creating a container or uploading/downloading files as required for the homework. We plan to continue working on reconstructing the virtual machines; however, this won't be completed by the deadline. Figures 19, 20, 21, 22 show our progress and crash report.



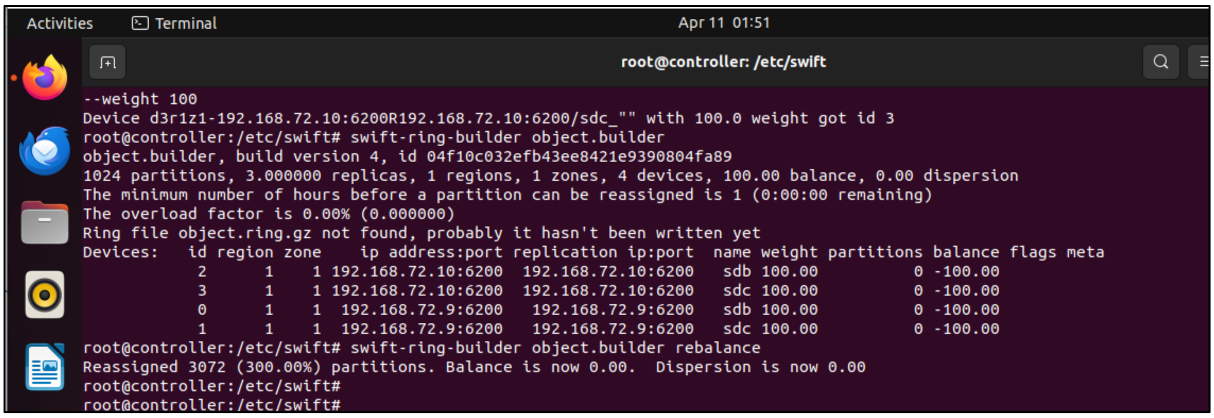
```
root@controller:/etc/swift# swift-ring-builder account.builder
account.builder, build version 4, id 34c5657f38694422b5e7b5c8dd2691ed
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file account.ring.gz is obsolete
Devices:   id region zone    ip address:port replication ip:port  name weight partitions balance flags meta
            2      1    1 192.168.72.10:6202  192.168.72.10:6202 /dev/sdb 100.00          0 -100.00
            3      1    1 192.168.72.10:6202  192.168.72.10:6202 /dev/sdc 100.00          0 -100.00
            0      1    1  192.168.72.9:6202   192.168.72.9:6202 /dev/sdb 100.00          0 -100.00
            1      1    1  192.168.72.9:6202   192.168.72.9:6202 /dev/sdc 100.00          0 -100.00
root@controller:/etc/swift# swift-ring-builder account.builder rebalance
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
root@controller:/etc/swift#
```

Figure 19: Swift 1



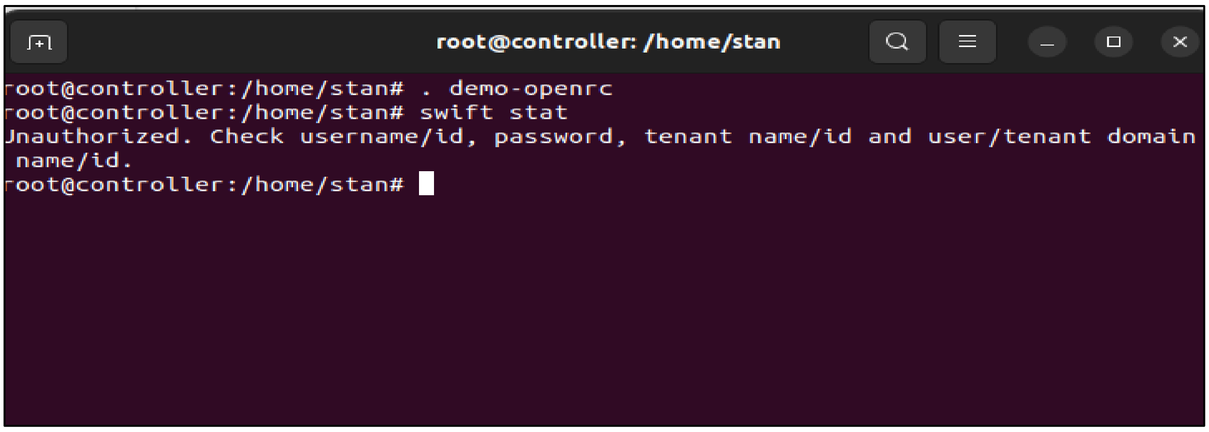
```
root@controller:/etc/swift# swift-ring-builder container.builder add   --region 1 --zone 1 --ip 192.168.72.10 --port 6201
dc --weight 100
Device d3r1z1-192.168.72.10:6201R192.168.72.10:6201/sdc_"" with 100.0 weight got id 3
root@controller:/etc/swift# swift-ring-builder container.builder
container.builder, build version 4, id 363f5a8530ce450988c2f7f0851cb48b
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file container.ring.gz not found, probably it hasn't been written yet
Devices:   id region zone    ip address:port replication ip:port  name weight partitions balance flags meta
            2      1    1 192.168.72.10:6201  192.168.72.10:6201   sdb 100.00          0 -100.00
            3      1    1 192.168.72.10:6201  192.168.72.10:6201   sdc 100.00          0 -100.00
            0      1    1  192.168.72.9:6201   192.168.72.9:6201   sdb 100.00          0 -100.00
            1      1    1  192.168.72.9:6201   192.168.72.9:6201   sdc 100.00          0 -100.00
root@controller:/etc/swift# swift-ring-builder container.builder rebalance
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
root@controller:/etc/swift#
root@controller:/etc/swift#
```

Figure 20: Swift 2



```
--weight 100
Device d3r1z1-192.168.72.10:6200R192.168.72.10:6200/sdc_"" with 100.0 weight got id 3
root@controller:/etc/swift# swift-ring-builder object.builder
object.builder, build version 4, id 04f10c032efb43ee8421e9390804fa89
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 4 devices, 100.00 balance, 0.00 dispersion
The minimum number of hours before a partition can be reassigned is 1 (0:00:00 remaining)
The overload factor is 0.00% (0.000000)
Ring file object.ring.gz not found, probably it hasn't been written yet
Devices:   id region zone    ip address:port replication ip:port  name weight partitions balance flags meta
            2      1    1 192.168.72.10:6200  192.168.72.10:6200   sdb 100.00          0 -100.00
            3      1    1 192.168.72.10:6200  192.168.72.10:6200   sdc 100.00          0 -100.00
            0      1    1  192.168.72.9:6200   192.168.72.9:6200   sdb 100.00          0 -100.00
            1      1    1  192.168.72.9:6200   192.168.72.9:6200   sdc 100.00          0 -100.00
root@controller:/etc/swift# swift-ring-builder object.builder rebalance
Reassigned 3072 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
root@controller:/etc/swift#
root@controller:/etc/swift#
```

Figure 21: Swift 3

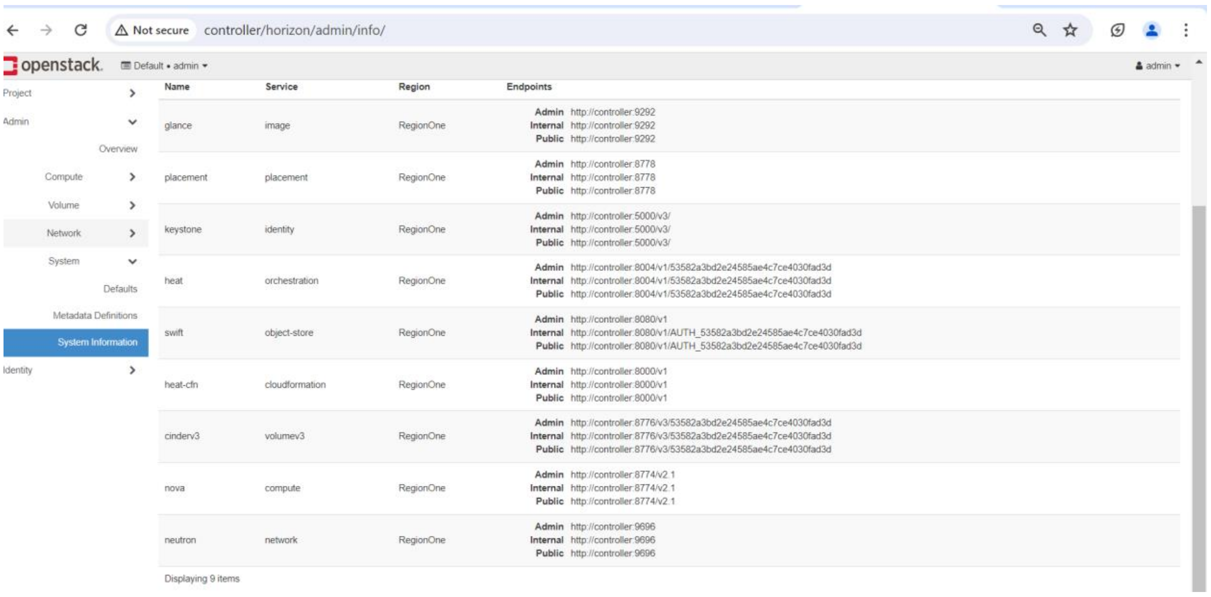


```
root@controller:/home/stan# . demo-openrc
root@controller:/home/stan# swift stat
Unauthorized. Check username/id, password, tenant name/id and user/tenant domain
 name/id.
root@controller:/home/stan# 
```

Figure 22: Swift 4

Heat (Extra Credit)

We were able to install an extra orchestration service, Heat. Figure 23 shows that heat has successfully been installed.



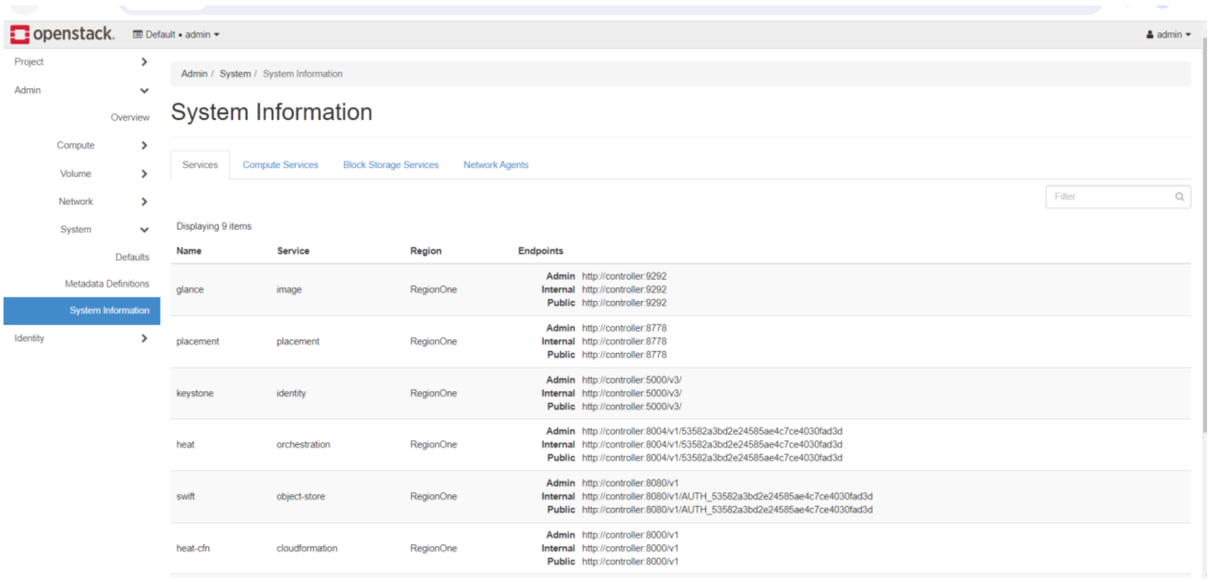
Name	Service	Region	Endpoints
glance	image	RegionOne	**Admin** http://controller:9292 **Internal** http://controller:9292 **Public** http://controller:9292
placement	placement	RegionOne	**Admin** http://controller:8778 **Internal** http://controller:8778 **Public** http://controller:8778
keystone	identity	RegionOne	**Admin** http://controller:5000/v3/ **Internal** http://controller:5000/v3/ **Public** http://controller:5000/v3/
heat	orchestration	RegionOne	**Admin** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Internal** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d
swift	object-store	RegionOne	**Admin** http://controller:8080/v1 **Internal** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d
heat-cfn	cloudformation	RegionOne	**Admin** http://controller:8000/v1 **Internal** http://controller:8000/v1 **Public** http://controller:8000/v1
cinderv3	volumev3	RegionOne	**Admin** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d **Internal** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d
nova	compute	RegionOne	**Admin** http://controller:8774/v2.1 **Internal** http://controller:8774/v2.1 **Public** http://controller:8774/v2.1
neutron	network	RegionOne	**Admin** http://controller:9696 **Internal** http://controller:9696 **Public** http://controller:9696

Displaying 9 items

Figure 23: Heat: Orchestration Services

Conclusion

In conclusion, we successfully got most services up and running in our OpenStack deployment, with the only exception being Swift. This manual deployment process was highly educational, providing not just insight into the installation of various services, but also into troubleshooting them. We often had to delete and reinstall services, users, groups, etc., to resolve issues. This was by far the toughest assignment that we've ever done. Despite the challenges, this homework was extremely valuable for gaining a comprehensive understanding of OpenStack. We have also included in Fig 24, 25 screenshots of dashboard showing all the services that we installed.



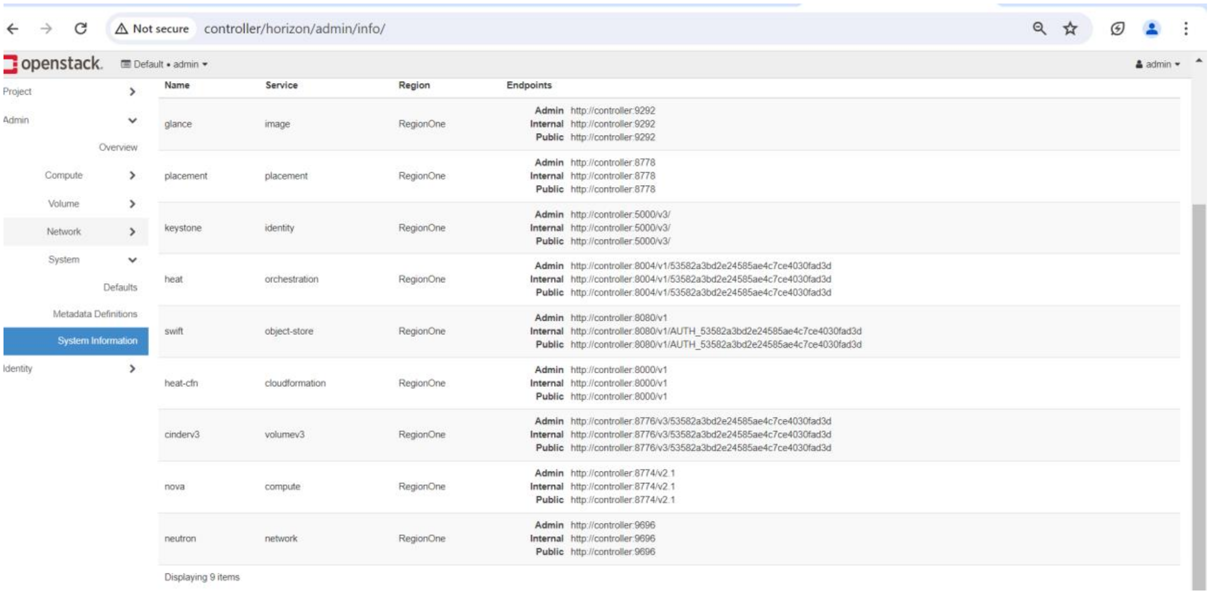
Admin / System / System Information

System Information

Displaying 9 items

Name	Service	Region	Endpoints
glance	image	RegionOne	**Admin** http://controller:9292 **Internal** http://controller:9292 **Public** http://controller:9292
placement	placement	RegionOne	**Admin** http://controller:8778 **Internal** http://controller:8778 **Public** http://controller:8778
keystone	identity	RegionOne	**Admin** http://controller:5000/v3/ **Internal** http://controller:5000/v3/ **Public** http://controller:5000/v3/
heat	orchestration	RegionOne	**Admin** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Internal** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d
swift	object-store	RegionOne	**Admin** http://controller:8080/v1 **Internal** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d
heat-cfn	cloudformation	RegionOne	**Admin** http://controller:8000/v1 **Internal** http://controller:8000/v1 **Public** http://controller:8000/v1

Figure 24: System Info 1



controller/horizon/admin/info/

Name	Service	Region	Endpoints
glance	image	RegionOne	**Admin** http://controller:9292 **Internal** http://controller:9292 **Public** http://controller:9292
placement	placement	RegionOne	**Admin** http://controller:8778 **Internal** http://controller:8778 **Public** http://controller:8778
keystone	identity	RegionOne	**Admin** http://controller:5000/v3/ **Internal** http://controller:5000/v3/ **Public** http://controller:5000/v3/
heat	orchestration	RegionOne	**Admin** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Internal** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8004/v1/53582a3bd2e24585ae4c7ce4030fad3d
swift	object-store	RegionOne	**Admin** http://controller:8080/v1 **Internal** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8080/v1/AUTH_53582a3bd2e24585ae4c7ce4030fad3d
heat-cfn	cloudformation	RegionOne	**Admin** http://controller:8000/v1 **Internal** http://controller:8000/v1 **Public** http://controller:8000/v1
cinderv3	volumev3	RegionOne	**Admin** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d **Internal** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d **Public** http://controller:8776/v3/53582a3bd2e24585ae4c7ce4030fad3d
nova	compute	RegionOne	**Admin** http://controller:8774/v2.1 **Internal** http://controller:8774/v2.1 **Public** http://controller:8774/v2.1
neutron	network	RegionOne	**Admin** http://controller:9696 **Internal** http://controller:9696 **Public** http://controller:9696

Displaying 9 items

Figure 25: System Info 2

References

"OpenStack Installation Guide." OpenStack, docs.openstack.org/install-guide/. Accessed 11 Apr. 2024.

"Server World." Server World, www.server-world.info/en/. Accessed 11 Apr. 2024.