Deployment of a Private/Hybrid Cloud IaaS OpenStack

Jathin Sreenivas
Jathin.Sreenivas@stud.fra-uas.de

Vidya Gopalakrishnarao
vidya.gopalakrishnarao@stud.fra-uas.de

Vineeth Bhat
vineeth.bhat@stud.fra-uas.de

Referent: Prof. Dr. Christian Baun
Department of Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt, Germany.

Abstract

Detailed instruction on deployment of a multi-node OpenStack[1] private cloud infrastructure and understanding the underlying components of OpenStack. To achieve this deployment, Microstack[2] will be used which is provided in a snap[13].

Table Of Contents

I. Introduction

OpenStack is a free open cloud computing platform, deployed as Infrastructure-as-a-Service (IaaS), where one can provide virtual services and resources as both public and private cloud.

OpenStack Architecture

The OpenStack architecture can be seen in Figure 1, where it shows all the components available in OpenStack and the nodes within which these components are present.

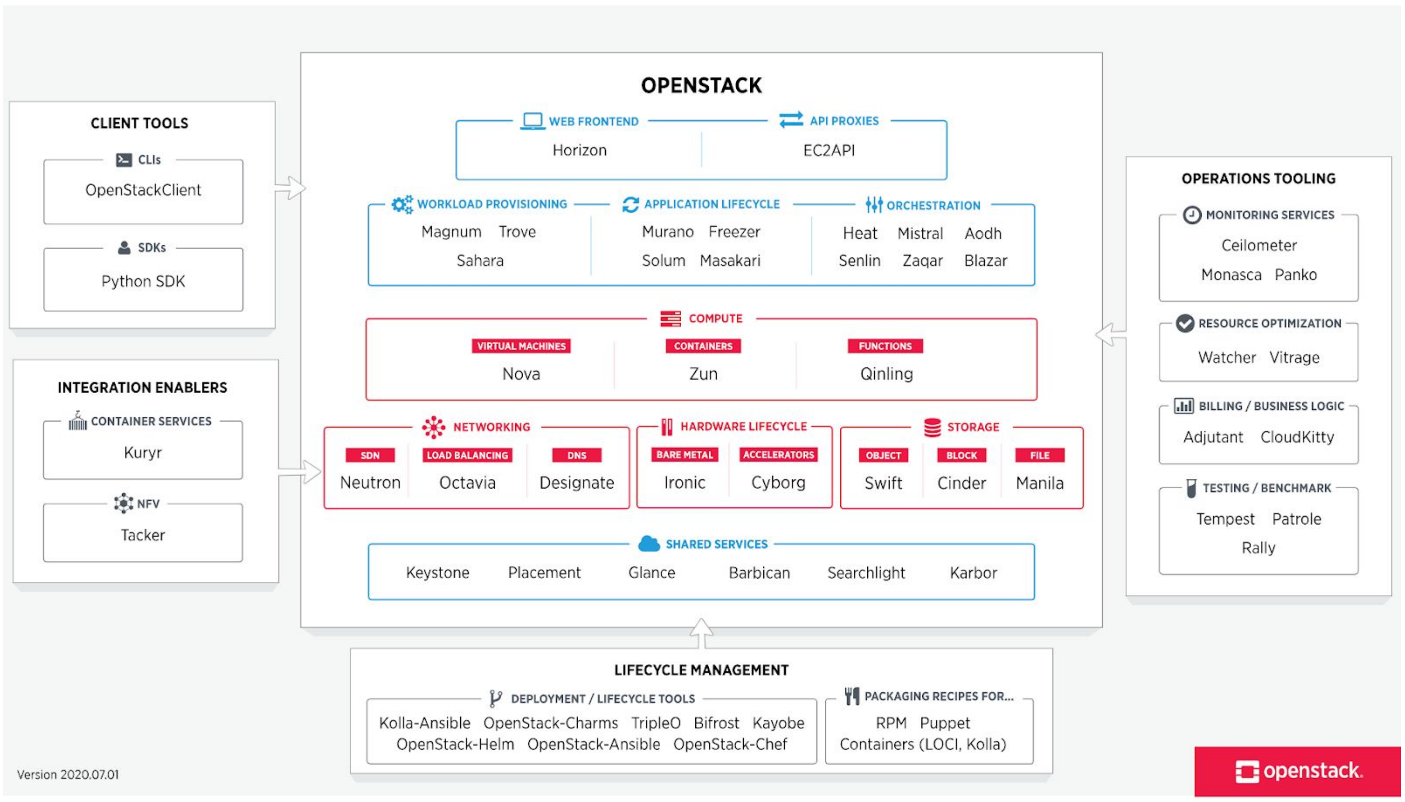


Figure 1. OpenStack Architecture

Nova
Nova provides the OpenStack compute service. It supports creating virtual machines, bare metal servers by using ironic. It runs as a set of daemons on top of existing Linux servers to provide that service.

Cinder
Cinder is the OpenStack block storage service for providing volumes to Nova virtual machines, Ironic bare metal hosts and containers. Cinder provides many useful advantages namely fault-tolerant, recoverable and open standards.

Neutron
Neutron provides the OpenStack network connectivity service between interfaces managed by other OpenStack services like vNICs and nova. It implements the neutron API.

Keystone
Keystone is the identity service used by OpenStack. It provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API.

Placement
The placement service in the OpenStack provides an HTTP API for tracking resource provider inventories and usages to help other services resource allocation and management. Placements also act as web services over data models.

Glance
Glance is the OpenStack image service which enables users to discover, register, and retrieve virtual machine data assets that are meant to be used with other services, this currently includes images.

Horizon
Horizon service is the OpenStack dashboard which provides a web based graphical interface to OpenStack services including Nova, Swift, Keystone, where users can access to manage OpenStack.

II. Deployment Instructions

Architecture

Figure 2 represents the architecture that can be achieved by following the deployment instructions provided in the document. Here two physical machines are used to host three virtual machines, where one of the VMs will act as the control node of OpenStack and the other two as compute nodes, thereby achieving multi-node deployment of OpenStack.

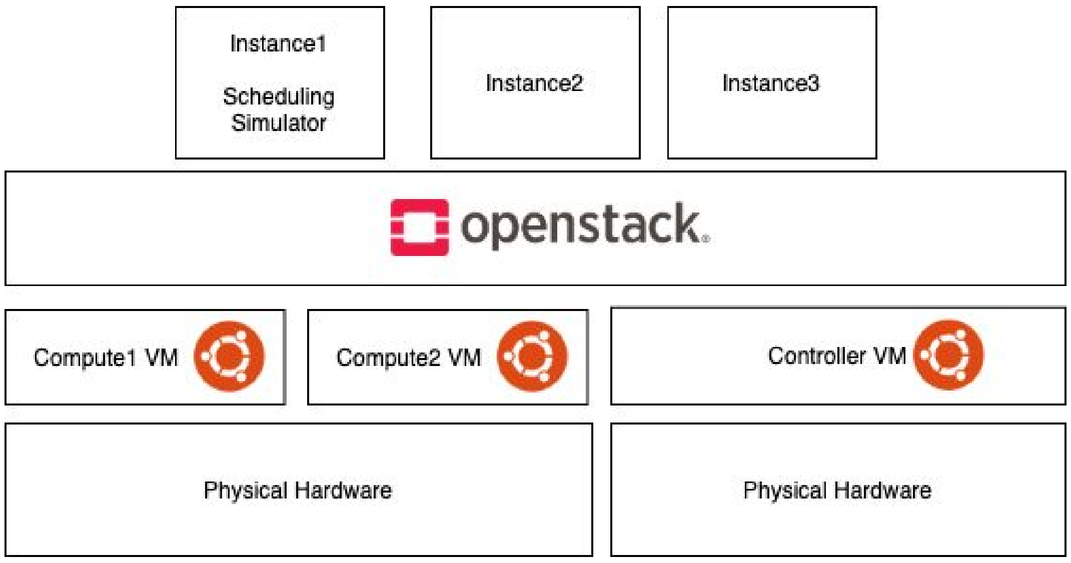


Figure 2. Architecture

Network Topology

The default network topology of OpenStack deployed using Microstack consists of an external network to which the internal network called test is connected via a Router as shown in Figure 3 and the instances are created in the test network.

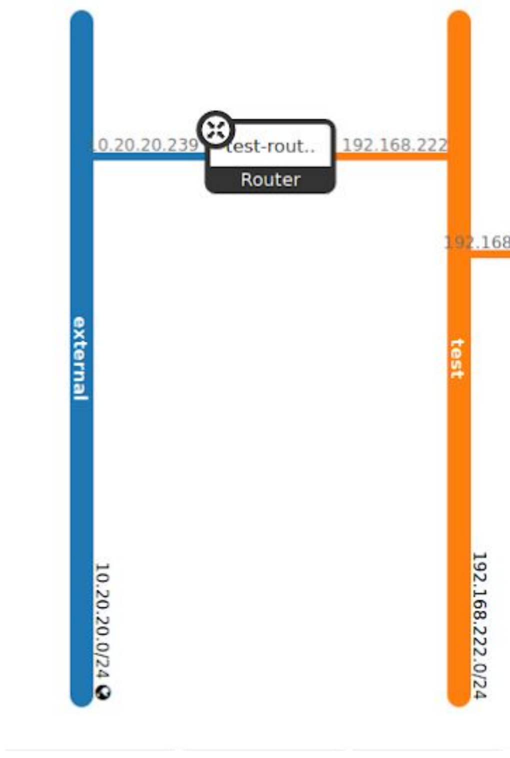


Figure 3. Network Topology

There are various tools available to deploy an OpenStack infrastructure like Devstack[3], Packstack[4], Microstack[5]. This document describes the installation using Microstack.

"MicroStack provides a single or multi-node OpenStack deployment which can run directly on your workstation. Although made for developers to prototype and test, it is also suitable for edge, IoT, and appliances. MicroStack is an OpenStack in a snap which means that all OpenStack services and supporting libraries are packaged together in a single package which can be easily installed, upgraded or removed. MicroStack includes all key OpenStack components: Keystone, Nova, Neutron, Glance, and Cinder." [2]

Prerequisites

To install OpenStack, the following prerequisites needs to be satisfied for a single node,

- A system with 16GB RAM
- Multi-core processor
- Atleast 50GB free disk space
- VMware
- Ubuntu 18.04 LTS or later (https://ubuntu.com/download/desktop)

Create three virtual machines with prerequisites mentioned above, The VMs used here are as follows,

- control-vm
- compute1-vm
- compute2-vm

Note: All the VMs must have connection to the internet, for this the VM's network must be configured to be a bridged network.

Install Microstack

To install Microstack in Ubuntu using snap, execute the following command in all the three VMs, i.e control-vm, compute1-vm and compute2-vm. This will download and install the Microstack from the specified channel and once the install is completed the following output will be displayed.

```
$ sudo snap install microstack --devmode --edge
```

```
Output: microstack (edge) ussuri from Canonical✓ installed
```

There are various channels available for example, microstack --devmode --beta, but the current installation is done using edge.

Setup Control Node

Set up control-vm as a control node by executing the following command in the control-vm.

```
$ sudo microstack init --auto --control
```

```
Output: microstack_init - INFO - Complete. Marked microstack as initialized!
```

Setup Compute Node

To initialize compute1-vm as a compute node. Run the following command on the control node to get a token that allows the compute node to join the control node.

```
$ sudo microstack add-compute
```

```
Output: Use the following connection string to add a new compute node to the cluster
(valid for 20 minutes from this moment):<connection-token-string>
```

Use the <connection-token-string> in the following command and execute it in the compute1-vm.

```
$ sudo microstack init --auto --compute --join <connection-token-string>
```

```
Output: microstack_init - INFO - Complete. Marked microstack as initialized!
```

Setup Multi-node Cluster

Repeat the steps from compute node setup again on compute2-vm respectively in order to setup compute2-vm as a compute node.

To check if Microstack is initialized. Open http://localhost in the browser of the control-vm to view the login page of OpenStack.

To view OpenStack dashboard in the compute node, node the <ip-address> of the control node using ifconfig and open http://<ip-address> in the browsers of the compute1-vm and compute2-vm.

Login

Execute the following command on any of the vm's to get the password to login to OpenStack.

```
$ sudo snap get microstack config.credentials.keystone-password
```

```
Output: <password>
```

Open OpenStack in a browser as explained above. And login as admin and use the <password> as password to login.

Enable/Disable Microstack

To disable microstack in the VMs before shutting down the VMs, execute the following command. It will save the changes made in the OpenStack before disabling.

```
$ sudo snap disable microstack
```

To enable the microstack again, execute the following command.

```
$ sudo snap enable microstack
```

This will bring up the microstack with the previously saved state.

III. Instance Creation

Now that a multi node OpenStack has been set up. The following sections explain the steps to create a virtual machine in OpenStack.

1. Image Creation

Firstly, the image of the OS for the virtual machine must be uploaded. To find the virtual machines images that works on the OpenStack visit https://docs.OpenStack.org/image-guide/obtain-images.html.The image can be uploaded to OpenStack in two ways:

1.1. Using the command line:

I. **Download Image**: Execute the following command in any of the VMs to download bionic-server-cloudimg-amd64-disk.img.

```
$ wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
```

II. **Create Image:** Execute the following command to create the image in OpenStack:

```
$ microstack.openstack image create --container-format bare --disk-format qcow2
--file bionic-server-cloudimg-amd64.img bionic
```

```
+------------------+-----------------------------------------------------------+
| Field            | Value                                                     |
+------------------+-----------------------------------------------------------+
| checksum         | 2f444b8c4d289747d1909998055e5e75                          |
|                  |                                                           |
| container_format | bare                                                      |
|                  |                                                           |
| created_at       | 2021-01-02T17:28:17Z                                      |
|                  |                                                           |
| disk_format      | qcow2                                                     |
|                  |                                                           |
| file             | /v2/images/f05a6a5d-0e97-4b5d-8880-9461eedf54bf/file      |
|                  |                                                           |
| id               | f05a6a5d-0e97-4b5d-8880-9461eedf54bf                      |
|                  |                                                           |
| min_disk         | 0                                                         |
|                  |                                                           |
| min_ram          | 0                                                         |
|                  |                                                           |
| name             | debian-9-openstack-amd64                                  |
|                  |                                                           |
| owner            | df2d2153582a419da31561593ca7a315                          |
|                  |                                                           |
| properties       | os_hash_algo='sha512', os_hash_value='65ca22e7625cc5c24001|
|                  | dd0e31c50042b44976c7dd0235a1d303b20f0bac6b133c44da91e5964e|
|                  | b019ec19e4c2f9b77e022b4976e8dd6e1aa7e53d38a60fcc19',      |
|                  | os_hidden='False', owner_specified.openstack.md5 =        |
|                  | '2f444b8c4d289747d1909998055e5e75',                       |
|                  | owner_specified.openstack.object =                        |
|                  | 'images/debian-9-openstack-amd64',                        |
|                  | owner_specified.openstack.sha256 =                        |
|                  | 'ed3bae5b661a17d5120831584b7aebd06e4290504f3e0463c83f00d  |
|                  | 83cbb4385', self='/v2/images/f05a6a5d-0e97-4b5d-8880-     |
|                  | 9461eedf54bf'                                             |
|                  |                                                           |
| protected        | False                                                     |
|                  |                                                           |
| schema           | /v2/schemas/image                                         |
|                  |                                                           |
| size             | 626980864                                                 |
|                  |                                                           |
| status           | active                                                    |
|                  |                                                           |
| tags             |                                                           |
|                  |                                                           |
| updated_at       | 2021-01-02T17:29:51Z                                      |
|                  |                                                           |
| visibility       | shared                                                    |
+------------------+-----------------------------------------------------------+
```

III. **Image List:** To check if the image has been created, execute the following command and check if an image with the name bionic is created:

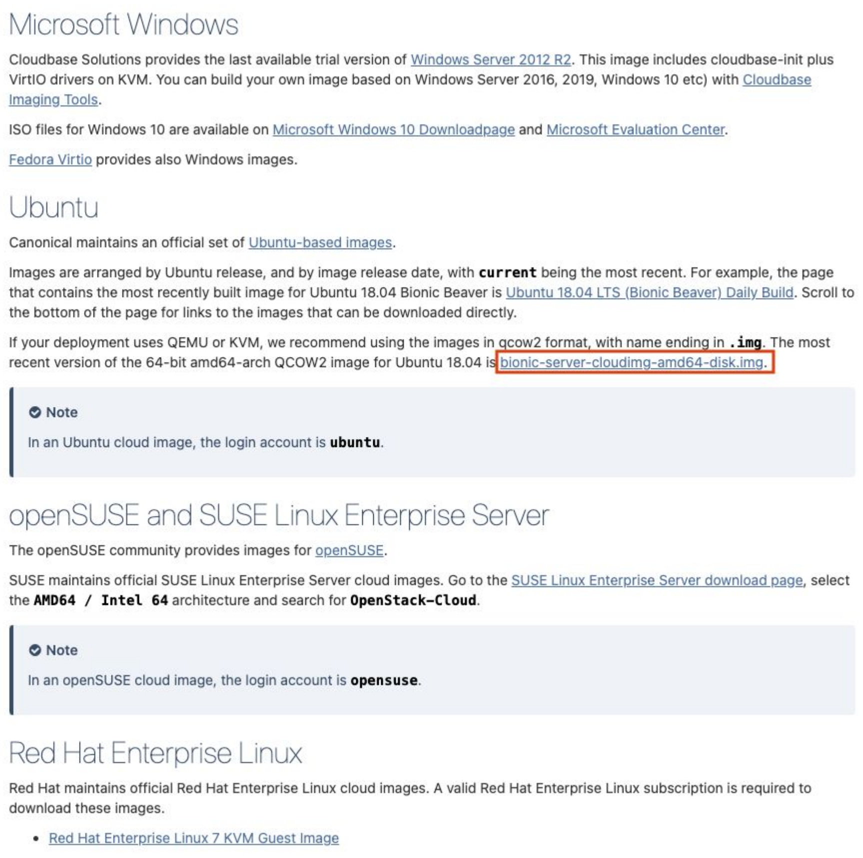
```
$ microstack.openstack image list
```

Sample Output:

```
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 54627c07-61c9-4185-b2ad-f8cea7be4aa5 | bionic | active |
| cbdfad7c-a5be-4335-93bd-c7be28c87a0c | cirros | active |
+--------------------------------------+--------+--------+
```

1.2. Using OpenStack Dashboard:

I. **Download Image**: Visit https://docs.OpenStack.org/image-guide/obtain-images.html and download bionic-server-cloudimg-amd64-disk.img which is the image of Ubuntu Bionic Server OS by clicking on bionic-server-cloudimg-amd64-disk.img as shown in Figure 4.



Microsoft Windows

Cloudbase Solutions provides the last available trial version of Windows Server 2012 R2. This image includes cloudbase-init plus VirtIO drivers on KVM. You can build your own image based on Windows Server 2016, 2019, Windows 10 etc) with Cloudbase Imaging Tools.

ISO files for Windows 10 are available on Microsoft Windows 10 Downloadpage and Microsoft Evaluation Center.

Fedora Virtio provides also Windows images.

Ubuntu

Canonical maintains an official set of Ubuntu-based images.

Images are arranged by Ubuntu release, and by image release date, with **current** being the most recent. For example, the page that contains the most recently built image for Ubuntu 18.04 Bionic Beaver is Ubuntu 18.04 LTS (Bionic Beaver) Daily Build. Scroll to the bottom of the page for links to the images that can be downloaded directly.

If your deployment uses QEMU or KVM, we recommend using the images in qcow2 format, with name ending in **.img**. The most recent version of the 64-bit amd64-arch QCOW2 image for Ubuntu 18.04 is bionic-server-cloudimg-amd64-disk.img.

Note

In an Ubuntu cloud image, the login account is **ubuntu**.

openSUSE and SUSE Linux Enterprise Server

The openSUSE community provides images for openSUSE.

SUSE maintains official SUSE Linux Enterprise Server cloud images. Go to the SUSE Linux Enterprise Server download page, select the **AMD64 / Intel 64** architecture and search for **OpenStack-Cloud**.

Note

In an openSUSE cloud image, the login account is **opensuse**.

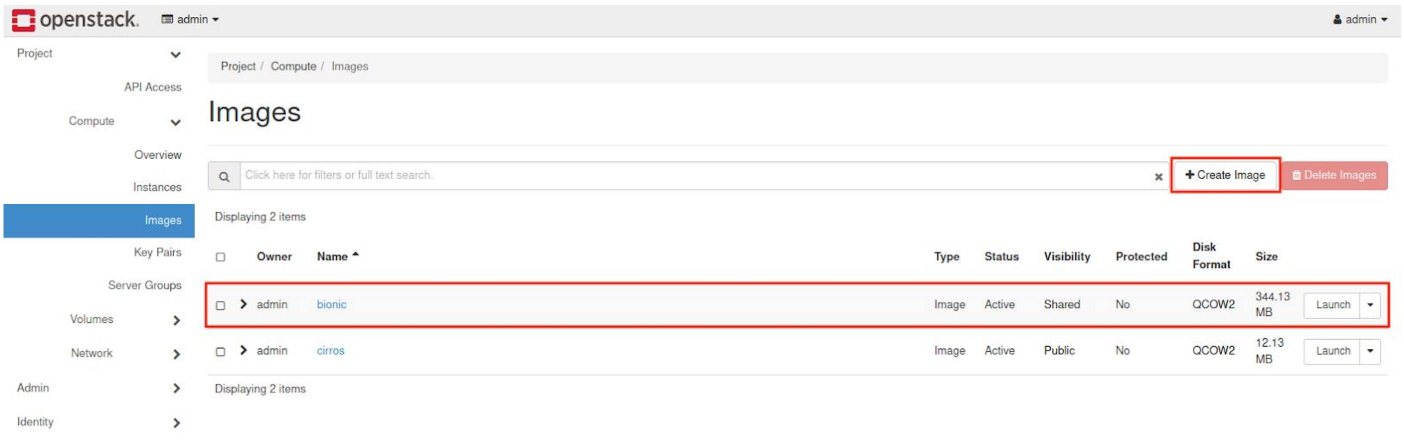
Red Hat Enterprise Linux

Red Hat maintains official Red Hat Enterprise Linux cloud images. A valid Red Hat Enterprise Linux subscription is required to download these images.

- Red Hat Enterprise Linux 7 KVM Guest Image

Figure 4. Download Bionic Server

II. **Create Image:** Go to the Images tab under the Compute tab then select Create image as shown in Figure 5.



Images

Click here for filters or full text search.

+ Create Image | Delete Images

Displaying 2 items

	Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size	
☐	admin	bionic	Image	Active	Shared	No	QCOW2	344.13 MB	Launch
☐	admin	cirros	Image	Active	Public	No	QCOW2	12.13 MB	Launch

Displaying 2 items

Figure 5. Create Image

- Now enter the image details as shown in Figure 6 and click Create Image.

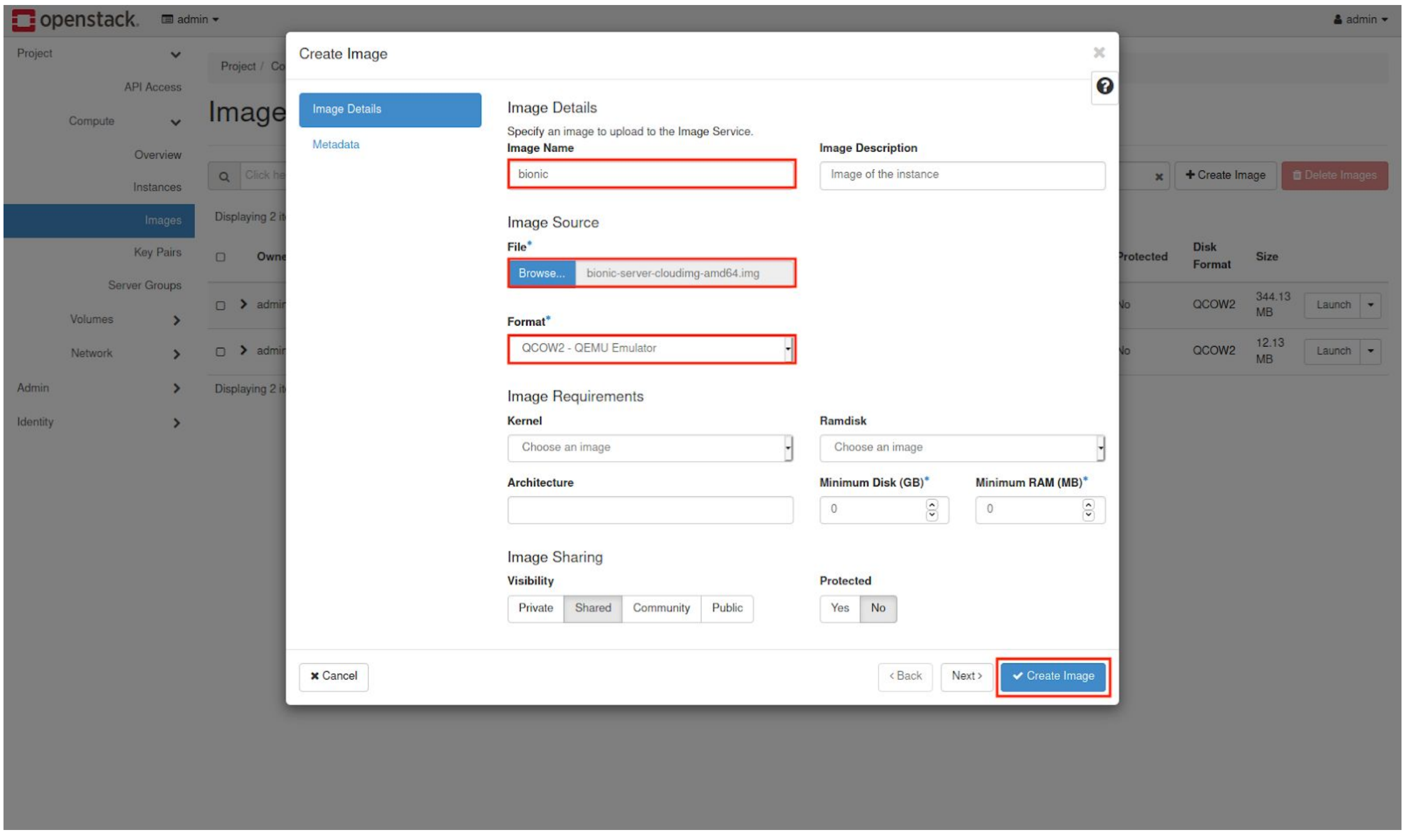


Image Details

Specify an image to upload to the Image Service.

Image Name: bionic

Image Description: Image of the instance

Image Source

File*: Browse... bionic-server-cloudimg-amd64.img

Format*: QCOW2 - QEMU Emulator

Image Requirements

Kernel: Choose an image

Ramdisk: Choose an image

Architecture:

Minimum Disk (GB)*: 0

Minimum RAM (MB)*: 0

Image Sharing

Visibility: Private [✓ Shared] Community Public

Protected: Yes [✓ No]

Figure 6. Image Details

Troubleshoot: In case there is an error while creating an image: “Request entity too large, nginx”. This is caused due to nginx limiting the size of the file being uploaded. This can be corrected by increasing the size in the nginx.conf file. Follow the steps below to correct the error in the control-vm,

```
$ sudo vi /var/snap/microstack/common/etc/nginx/snap/nginx.conf
```

In the http section, add the following line,

```
client_max_body_size 32768M;
```

That increases the maximum file size to 32GB. After the file is saved, restart microstack or enable and disable the microstack.

```
$ sudo snap restart microstack
```

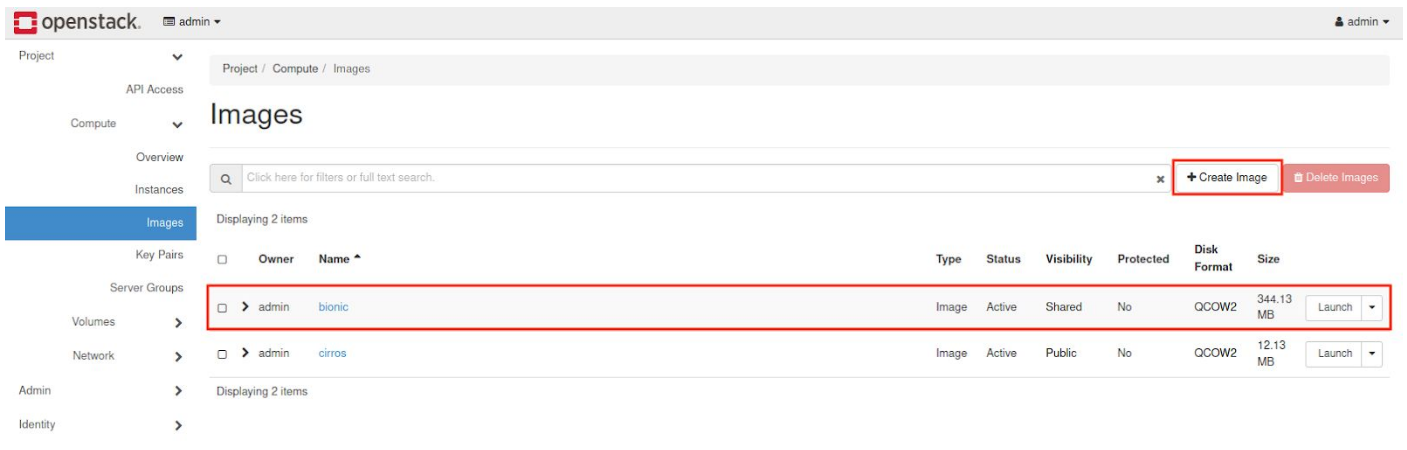
Or

```
$ sudo snap disable microstack
```

Once disabled enable using the following command,

```
$ sudo snap enable microstack
```

III. **Image List:** Now in the Images tab under the Compute tab, the bionic image should be added as shown in Figure 7.



Displaying 2 items

	Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size	
☐	admin	bionic	Image	Active	Shared	No	QCOW2	344.13 MB	Launch
☐	admin	cirros	Image	Active	Public	No	QCOW2	12.13 MB	Launch

Displaying 2 items

Figure 7. Image Created

2. Instance Creation

To create the instance of the image can be done in two ways:

2.1. Using the command line:

I. **Create a new Key-pair:** Execute the following commands to create a new SSH key which then can be used to login to the instance

```
$ ssh-keygen -q -N ""
```

```
Enter file in which to save the key (/home/ubuntu/.ssh/id\_rsa):
```

Note: A file name need not be provided. Press Enter to continue

II. **Create key-pair in OpenStack:**

```
$ microstack.openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
```

```
+-----------------+-------------------------------------------------+
| Field           | Value                                           |
+-----------------+-------------------------------------------------+
|fingerprint      | ab:eb:bc:55:9e:c2:4f:0b:ad:f0:62:7b:02:f0:89:e7 |
|name             | mykey                                           |
|user_id          | cd22ff23ece040bca3d12639abddd726                |
+-----------------+-------------------------------------------------+
```

III. **Create the instance:** Execute the following command to create the instance

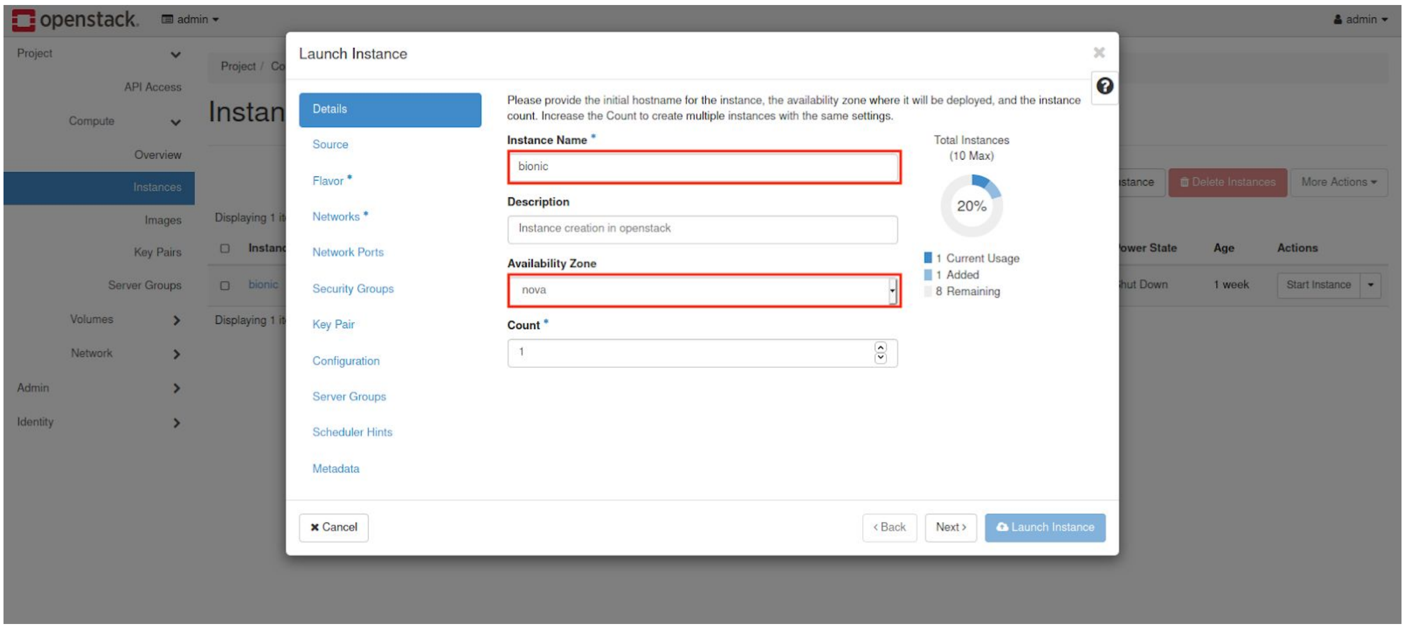
```
$ microstack.openstack server create --flavor <flavor-id> --image <image-id>
--security-group <default-security-group-id> --key-name mykey --nic
net-id=<test-network-id> bionic
```

```
+-------------------------------------+-----------------------------------------------------------------+
| Field                               | Value                                                           |
+-------------------------------------+-----------------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                                          |
| OS-EXT-AZ:availability_zone         |                                                                 |
| OS-EXT-SRV-ATTR:host                | None                                                            |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                                            |
| OS-EXT-SRV-ATTR:instance_name       |                                                                 |
| OS-EXT-STS:power_state              | NOSTATE                                                         |
| OS-EXT-STS:task_state               | scheduling                                                      |
| OS-EXT-STS:vm_state                 | building                                                        |
| OS-SRV-USG:launched_at              | None                                                            |
| OS-SRV-USG:terminated_at            | None                                                            |
| accessIPv4                          |                                                                 |
| accessIPv6                          |                                                                 |
| addresses                           |                                                                 |
| adminPass                           | t86Ncrk6GnjS                                                    |
| config_drive                        |                                                                 |
| created                             | 2021-01-02T17:46:42Z                                            |
| flavor                              | m1.small (2)                                                    |
| hostId                              |                                                                 |
| id                                  | 3017568b-8aa4-44da-8e84-9efc0bf9ee79                            |
| image                               | debian-9-openstack-amd64 (f05a6a5d-0e97-4b5d-8880-9461eedf54bf) |
| key_name                            | mykey                                                           |
| name                                | Debianserver                                                    |
| progress                            | 0                                                               |
| project_id                          | df2d2153582a419da31561593ca7a315                                |
| properties                          |                                                                 |
| security_groups                     | name='71b2f9f3-07ed-485f-88ba-d80f04c2eb5a'                     |
| status                              | BUILD                                                           |
| updated                             | 2021-01-02T17:46:43Z                                            |
| user_id                             | cd22ff23ece040bca3d12639abddd726                                |
| volumes_attached                    |                                                                 |
+-------------------------------------+-----------------------------------------------------------------+
```

2.2. Using OpenStack Dashboard:

I. **Create the instance:** Follow the steps in order to create an instance.

- Provide the name and select the zone available and click on next as shown in Figure 8.



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *: bionic

Description: Instance creation in openstack

Availability Zone: nova

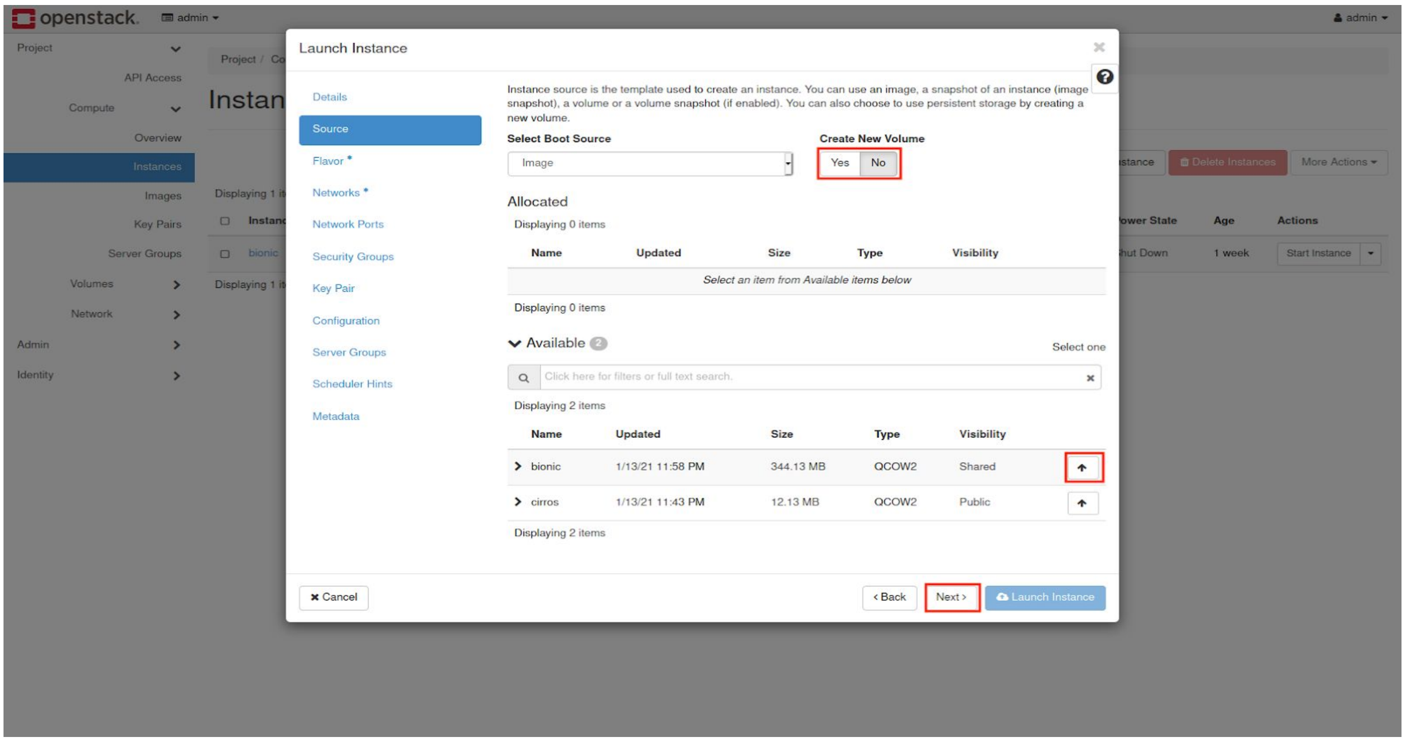
Count *: 1

Total Instances (10 Max): 20%

- 1 Current Usage
- 1 Added
- 8 Remaining

Figure 8. Instance Details

- Select “No” for creating a new volume as we do not need any volume and select the image from which an instance needs to be created as shown in Figure 9.



Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source: Image

Create New Volume: Yes [✓ No]

Allocated

Displaying 0 items

Name	Updated	Size	Type	Visibility
Select an item from Available items below				

Displaying 0 items

Available 2 — Select one

Click here for filters or full text search.

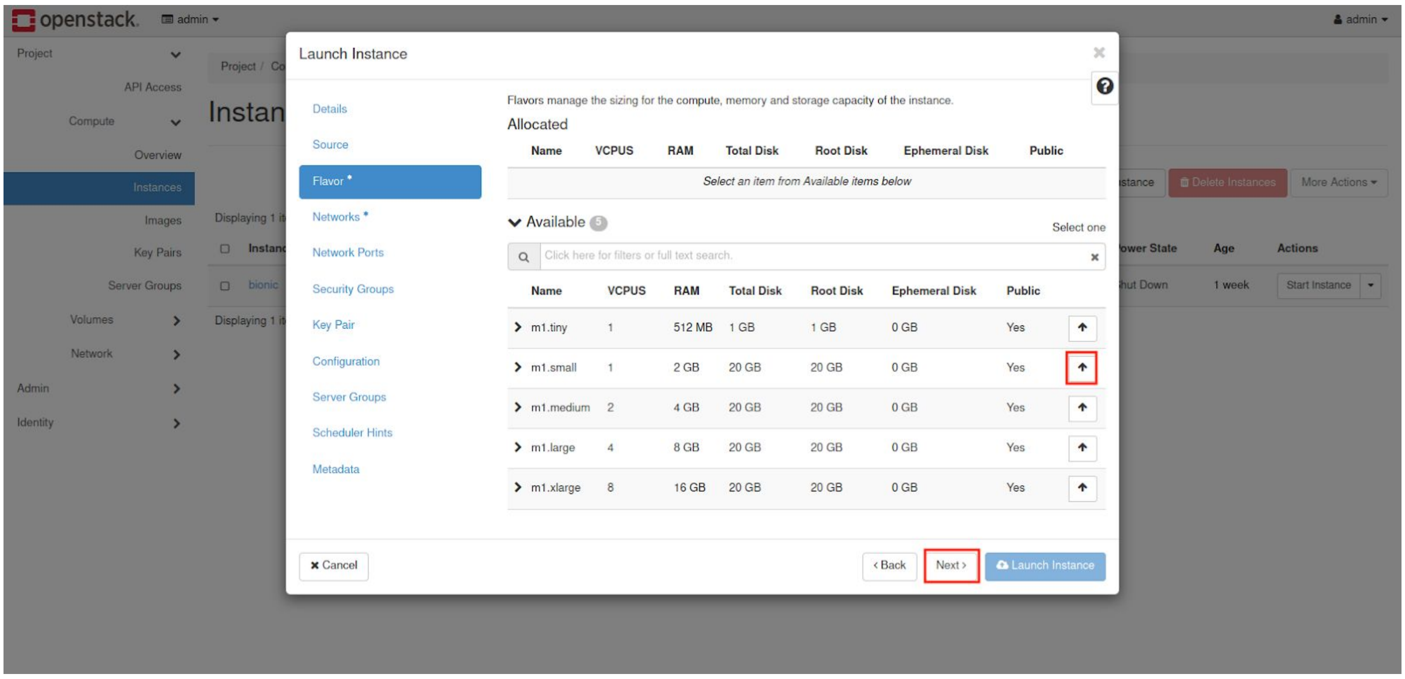
Displaying 2 items

Name	Updated	Size	Type	Visibility
bionic	1/13/21 11:58 PM	344.13 MB	QCOW2	Shared
cirros	1/13/21 11:43 PM	12.13 MB	QCOW2	Public

Displaying 2 items

Figure 9. Instance Source

- Select the appropriate flavour highlighted in Figure 10. If the flavour is expanded, details about the flavour will be shown. Depending on the size and capability needed for the instance, flavour with greater capacity has to be selected.


Launch Instance

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

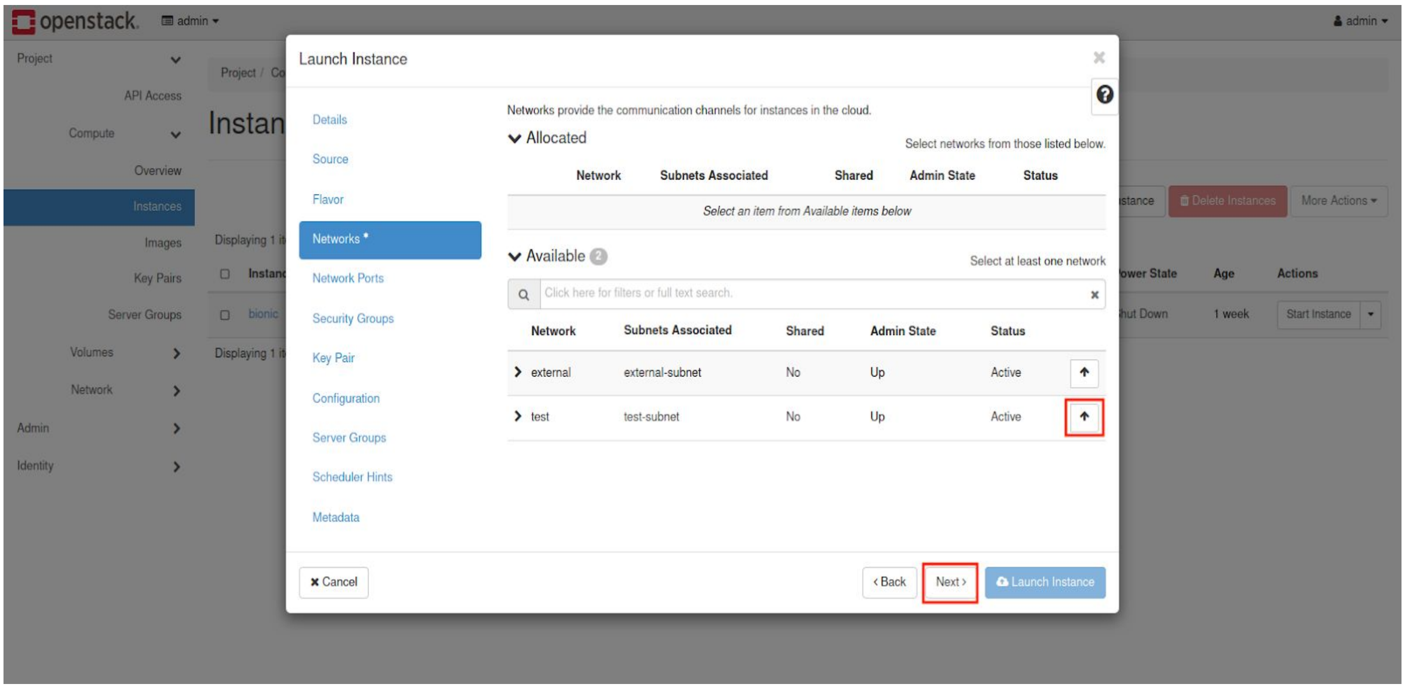
Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
Select an item from Available items below						

Available 5 — Select one

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
m1.small	1	2 GB	20 GB	20 GB	0 GB	Yes
m1.medium	2	4 GB	20 GB	20 GB	0 GB	Yes
m1.large	4	8 GB	20 GB	20 GB	0 GB	Yes
m1.xlarge	8	16 GB	20 GB	20 GB	0 GB	Yes

Figure 10. Instance Flavor

- Select the test network as shown in Figure 11, as it is the one to which the instance must be connected.


Launch Instance

Networks provide the communication channels for instances in the cloud.

Allocated — Select networks from those listed below.

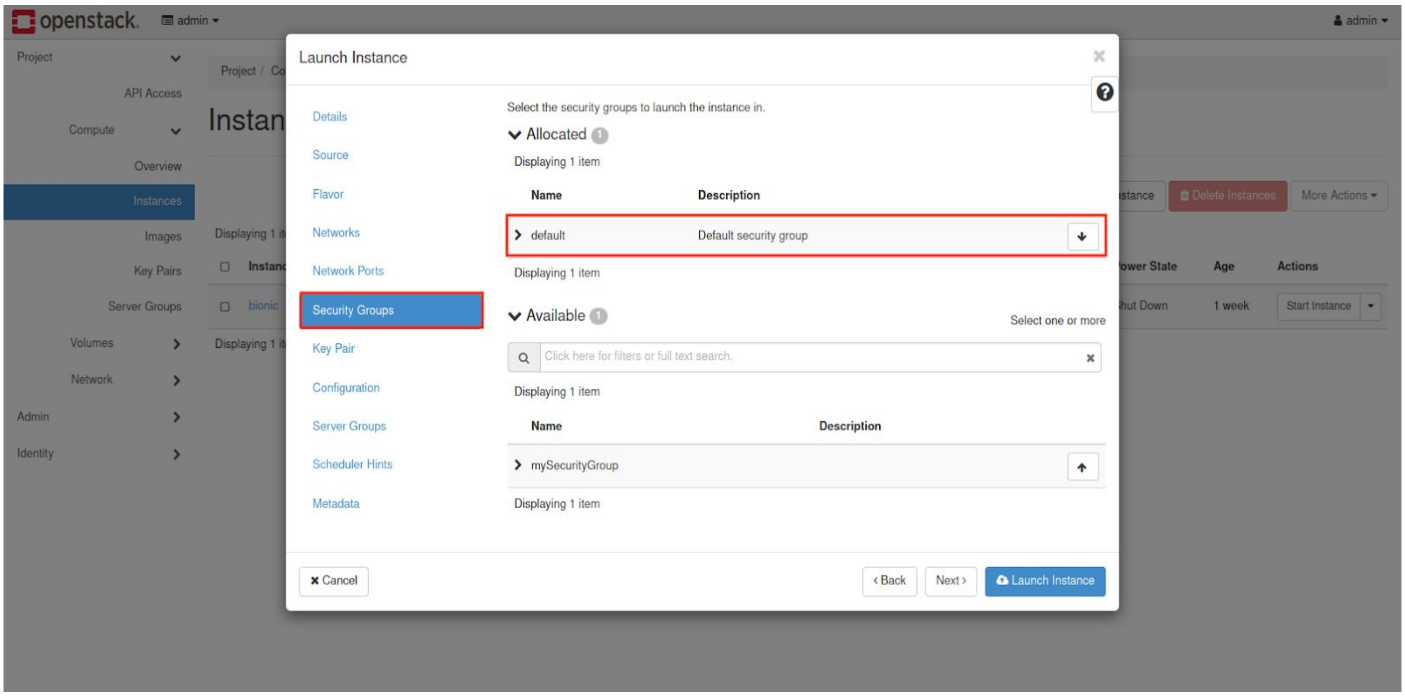
Network	Subnets Associated	Shared	Admin State	Status
Select an item from Available items below				

Available 2 — Select at least one network

Network	Subnets Associated	Shared	Admin State	Status
external	external-subnet	No	Up	Active
test	test-subnet	No	Up	Active

Figure 11. Instance Networks

- The default security group will be selected as highlighted in Figure 12, if not select the security group.



Select the security groups to launch the instance in.

Allocated 1

Displaying 1 item

Name	Description
default	Default security group

Displaying 1 item

Available 1 — Select one or more

Click here for filters or full text search.

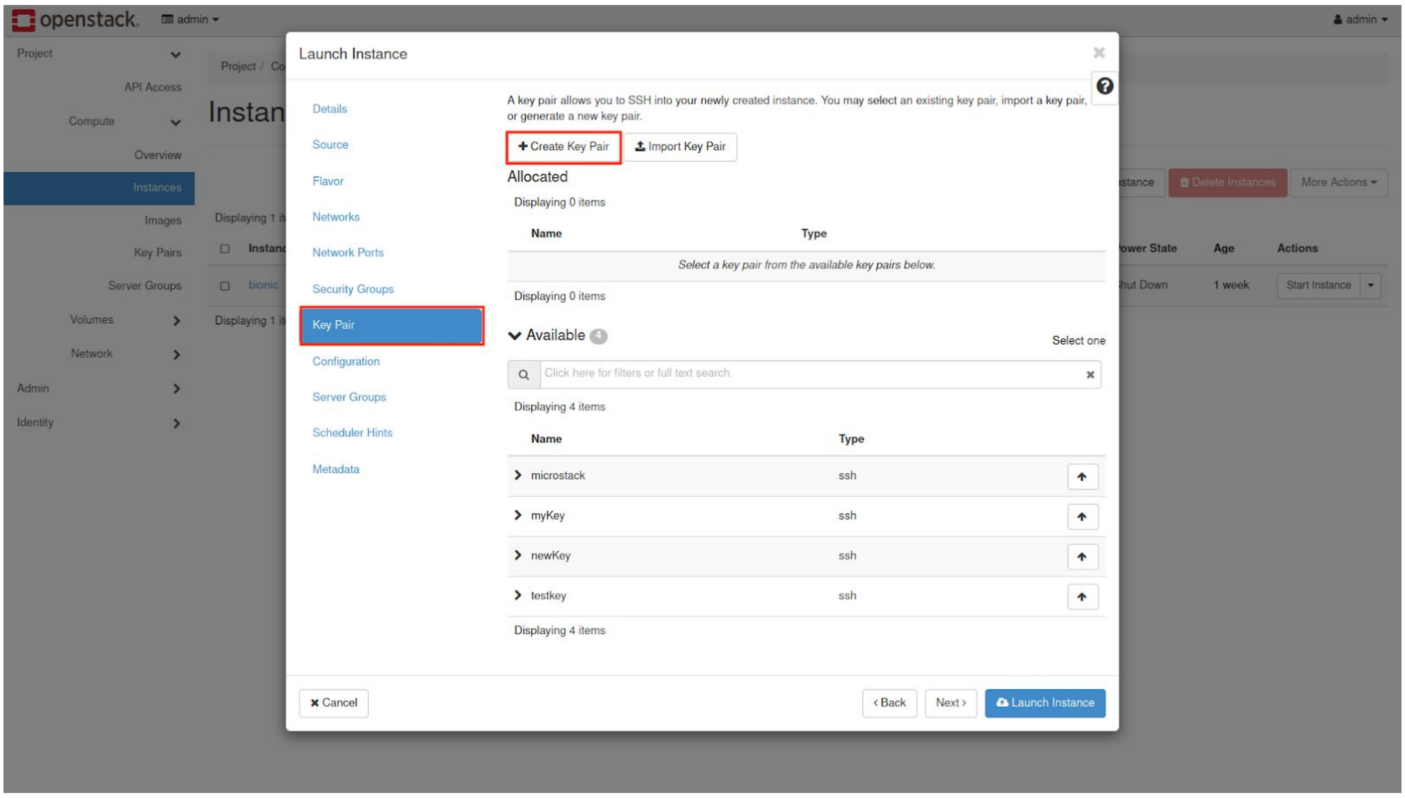
Displaying 1 item

Name	Description
mySecurityGroup	

Displaying 1 item

Figure 12. Instance Security Groups

- Create the key-pair as it will be needed later to login using SSH, by clicking on “Create Key Pair”, which is highlighted in Figure 13.



A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.

Allocated

Displaying 0 items

Name	Type
Select a key pair from the available key pairs below.	

Displaying 0 items

Available 4 — Select one

Click here for filters or full text search.

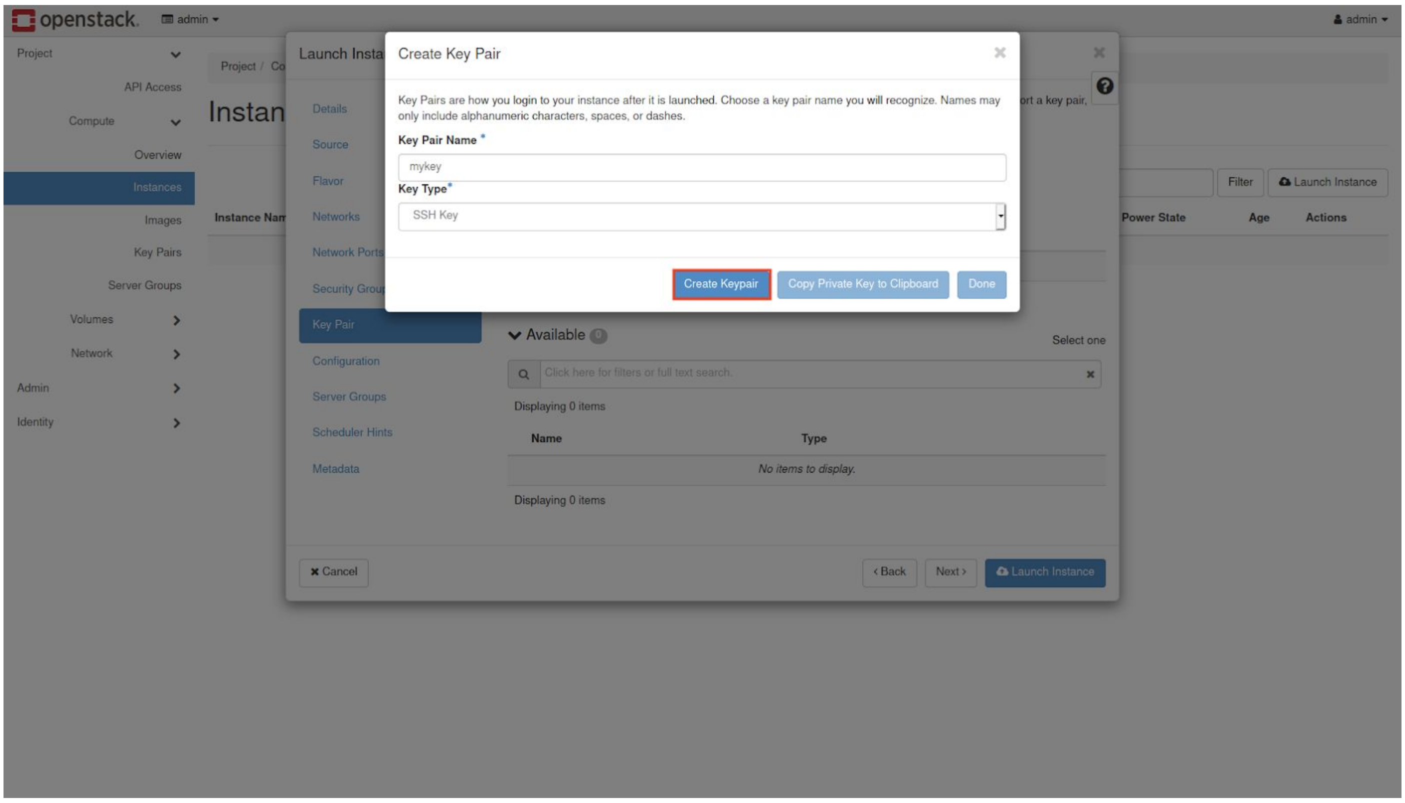
Displaying 4 items

Name	Type
microstack	ssh
myKey	ssh
newKey	ssh
testkey	ssh

Displaying 4 items

Figure 13. Instance Key Pair

- As shown in Figure 14, a pop-up window will appear where the name of key pair and type must be selected and then click on “Create Keypair”.



Create Key Pair

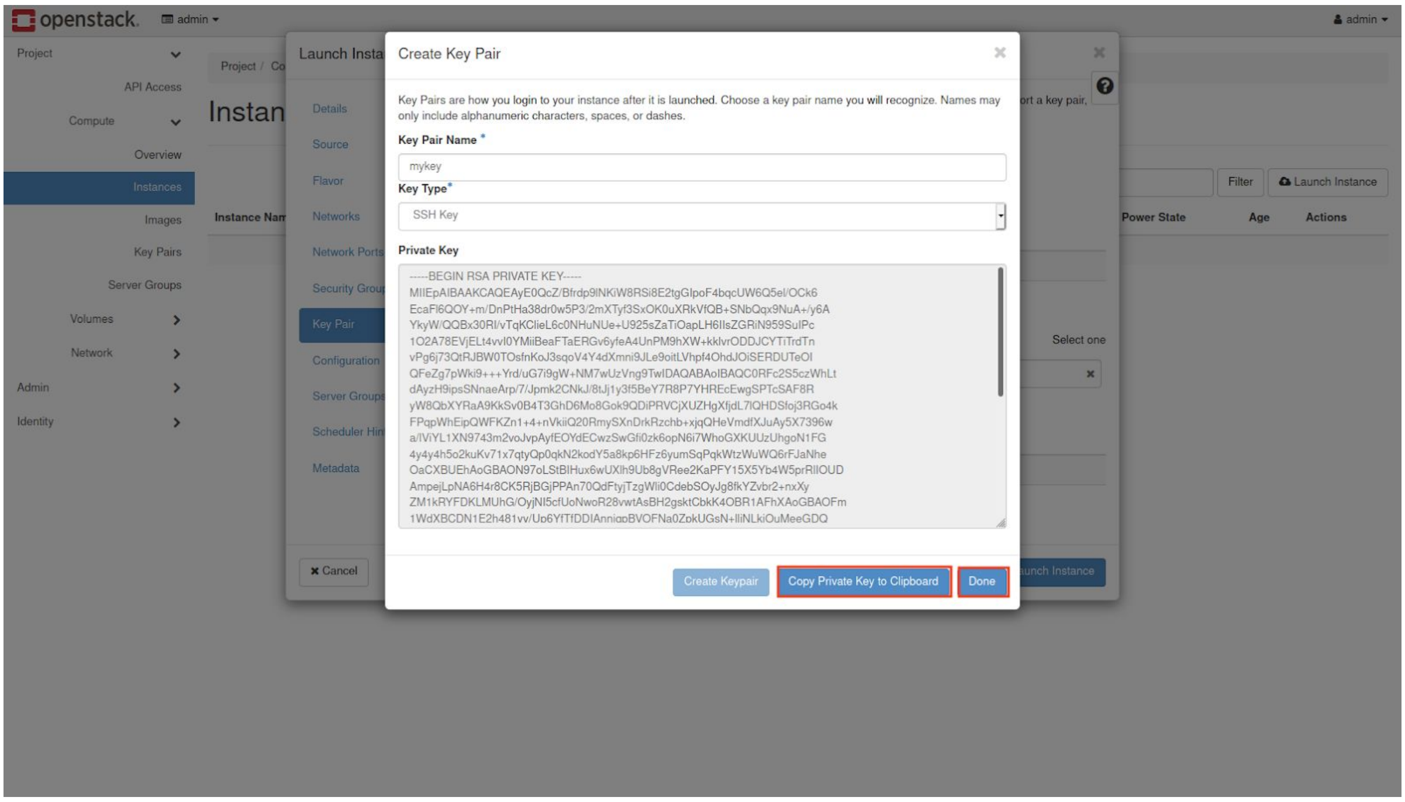
Key Pairs are how you login to your instance after it is launched. Choose a key pair name you will recognize. Names may only include alphanumeric characters, spaces, or dashes.

Key Pair Name *: mykey

Key Type *: SSH Key

Figure 14. Instance Key Pair creation

- Private Key is created as shown in Figure 15. Copy the key and save it in a file for later use and click on “Done”.



Create Key Pair

Key Pairs are how you login to your instance after it is launched. Choose a key pair name you will recognize. Names may only include alphanumeric characters, spaces, or dashes.

Key Pair Name *: mykey

Key Type *: SSH Key

Private Key

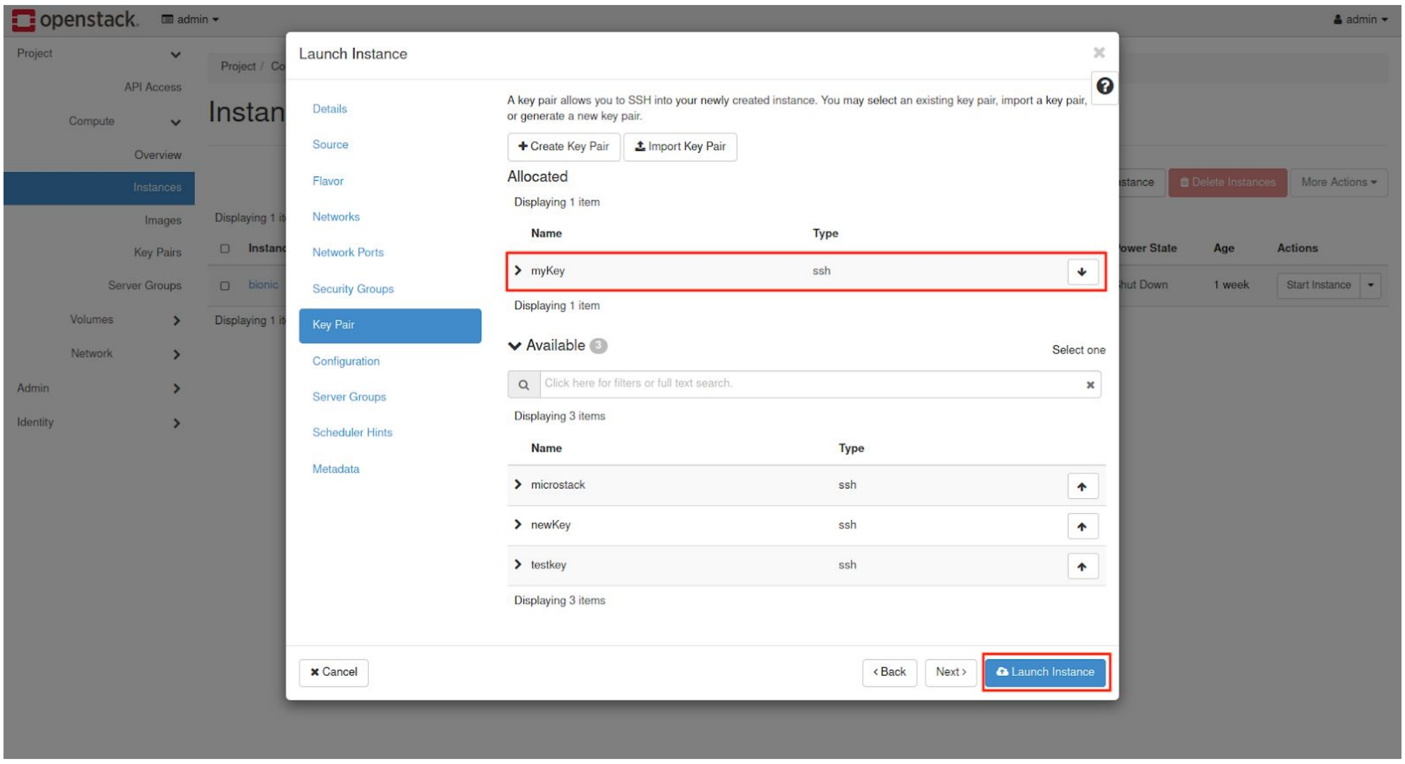
```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyE0QcZ/Bfrdp9INKiW8RSi8E2tgGIpoF4bqcUW6Q5el/OCk6
EcaFl6QOY+m/DnPtHa38dr0w5P3/2mXTyf3SxOK0uXRkVfQB+SNbQqx9NuA+/y6A
YkyW/QQBx30RI/vTqKClieL6c0NHuNUe+U925sZaTiOapLH6IIsZGRiN959SuIPc
1O2A78EVjELt4vvI0YMiiBeaFTaERGv6yfeA4UnPM9hXW+kklvrODDJCYTiTrdTn
vPg6j73QtRJBW0TOsfnKoJ3sqoV4Y4dXmni9JLe9oitLVhpf4OhdJOiSERDUTeOI
QFeZg7pWki9+++Yrd/uG7i9gW+NM7wUzVng9TwIDAQABAoIBAQC0RFc2S5czWhLt
dAyzH9ipsSNnaeArp/7/Jpmk2CNkJ/8tJj1y3f5BeY7R8P7YHREcEwgSPTcSAF8R
yW8QbXYRaA9KkSv0B4T3GhD6Mo8Gok9QDiPRVCjXUZHgXfjdL7lQHDSfoj3RGo4k
FPqpWhEipQWFKZn1+4+nVkiiQ20RmySXnDrkRzchb+xjqQHeVmdfXJuAy5X7396w
a/IViYL1XN9743m2voJvpAyfEOYdECwzSwGfi0zk6opN6i7WhoGXKUUzUhgoN1FG
4y4y4h5o2kuKv71x7qtyQp0qkN2kodY5a8kp6HFz6yumSqPqkWtzWuWQ6rFJaNhe
OaCXBUEhAoGBAON97oLStBIHux6wUXlh9Ub8gVRee2KaPFY15X5Yb4W5prRIlOUD
AmpejLpNA6H4r8CK5RjBGjPPAn70QdFtyjTzgWli0CdebSOyJg8fkYZvbr2+nxXy
ZM1kRYFDKLMUhG/OyjNI5cfUoNwoR28vwtAsBH2gsktCbkK4OBR1AFhXAoGBAOFm
1WdXBCDN1E2h481vv/Up6YfTfDDIAnniqpBVOFNa0ZpkUGsN+lliNLkiOuMeeGDQ
```

Figure 15. Instance SSH Key Pair Creation

- Save the keypair created in a file, for example “bionic-id.pub” in the control-vm. Then change the permissions of the file to read only by running the following command with the specific access permissions. This is because the SSH will not accept the file if it's editable or executable.

```
$ chmod 400 <filename>
```

- Once a new keypair is created click on “Launch instance” as shown in Figure 16 to create the instance and a new instance will be created.



A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.

Allocated

Displaying 1 item

Name	Type
myKey	ssh

Displaying 1 item

Available 3 — Select one

Click here for filters or full text search.

Displaying 3 items

Name	Type
microstack	ssh
newKey	ssh
testkey	ssh

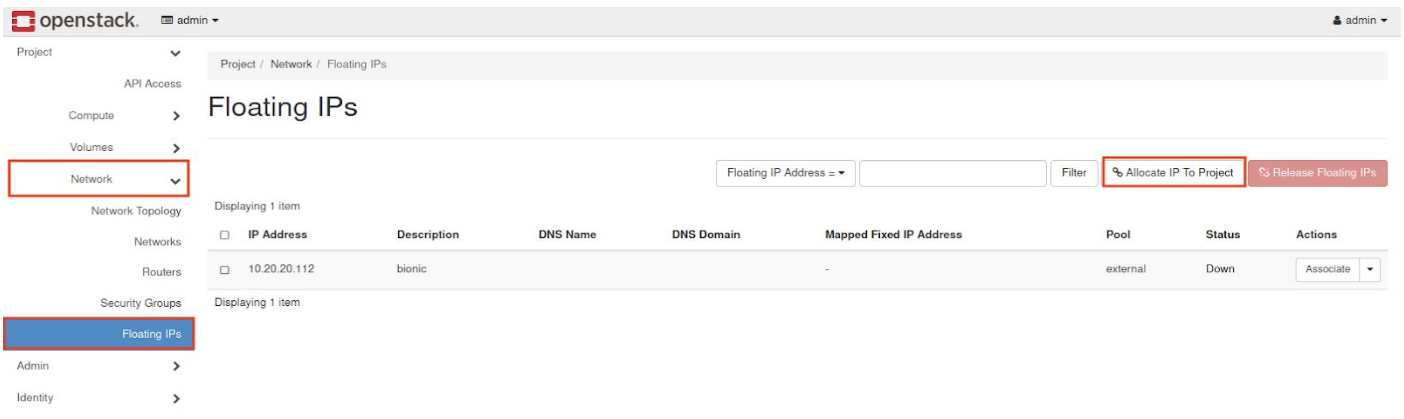
Displaying 3 items

Figure 16. Instance Key Pair allocation

2.3. Allocating Floating IP:

Once the instance is created, a floating IP has to be allocated to it. The following steps explain how to associate a floating IP to an instance.

- First create a floating IP by clicking on “Network”, then select “Floating IPs” and then click on “Allocate IP To Project” on the page as shown in Figure 17.



Floating IPs

Displaying 1 item

IP Address	Description	DNS Name	DNS Domain	Mapped Fixed IP Address	Pool	Status	Actions
10.20.20.112	bionic			-	external	Down	Associate

Displaying 1 item

Fig 17. Floating IP Creation

- A pop-up will appear as shown in Figure 18, to create a floating IP. Choose “external” for the Pool, Provide a description and then click on Allocate IP. A new floating IP will be created.

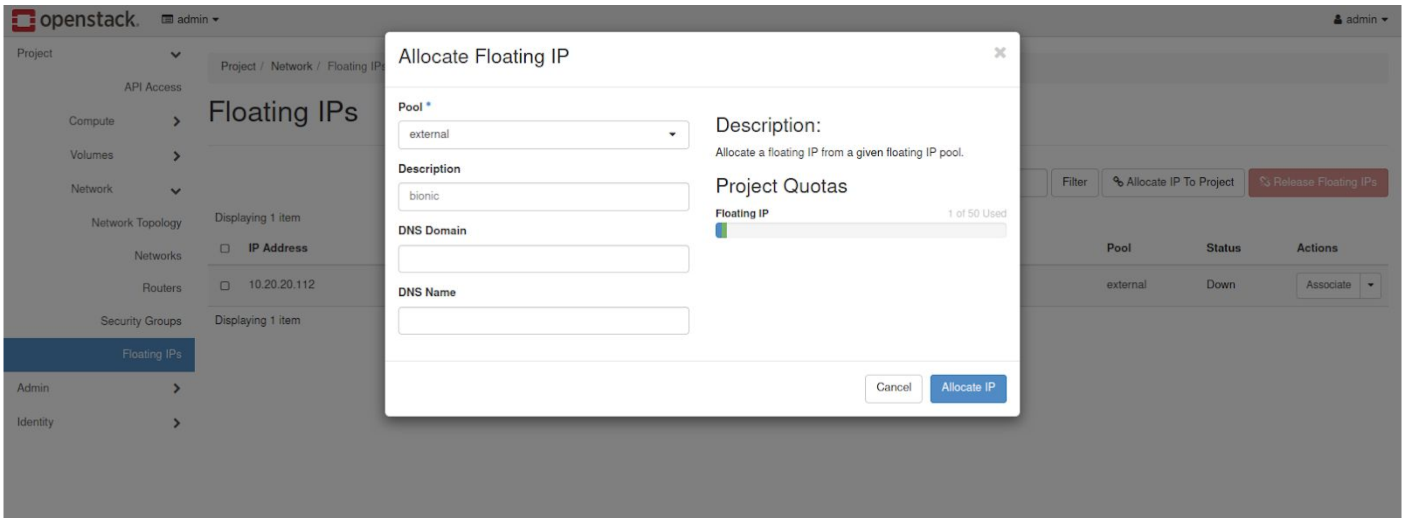


Figure 18. Allocate Floating IP

- Now move back to the Instance tab within the Compute Tab. Then click on the drop down on the provided in the instance you want to associate floating IP to as shown in Figure 19. Then click on the option of “Associate Floating IP”.

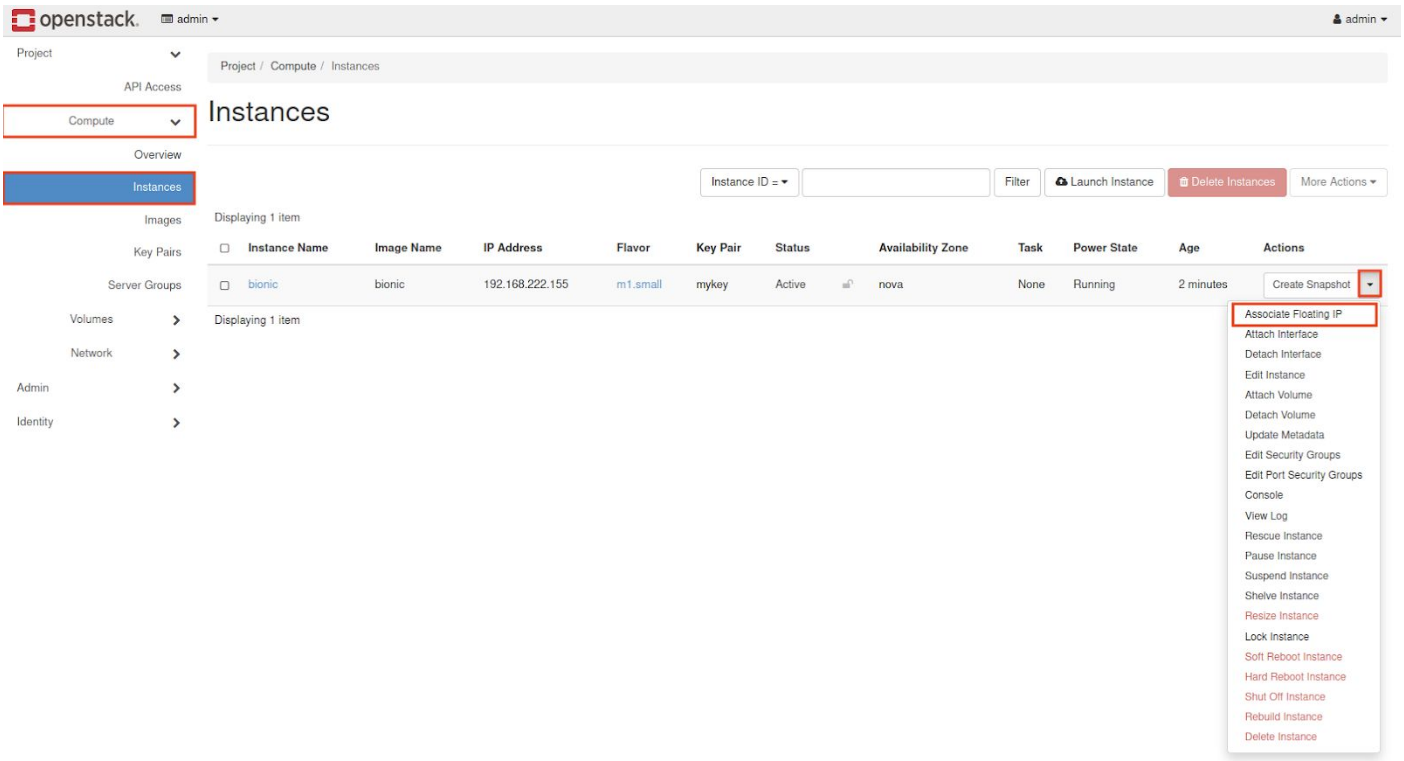
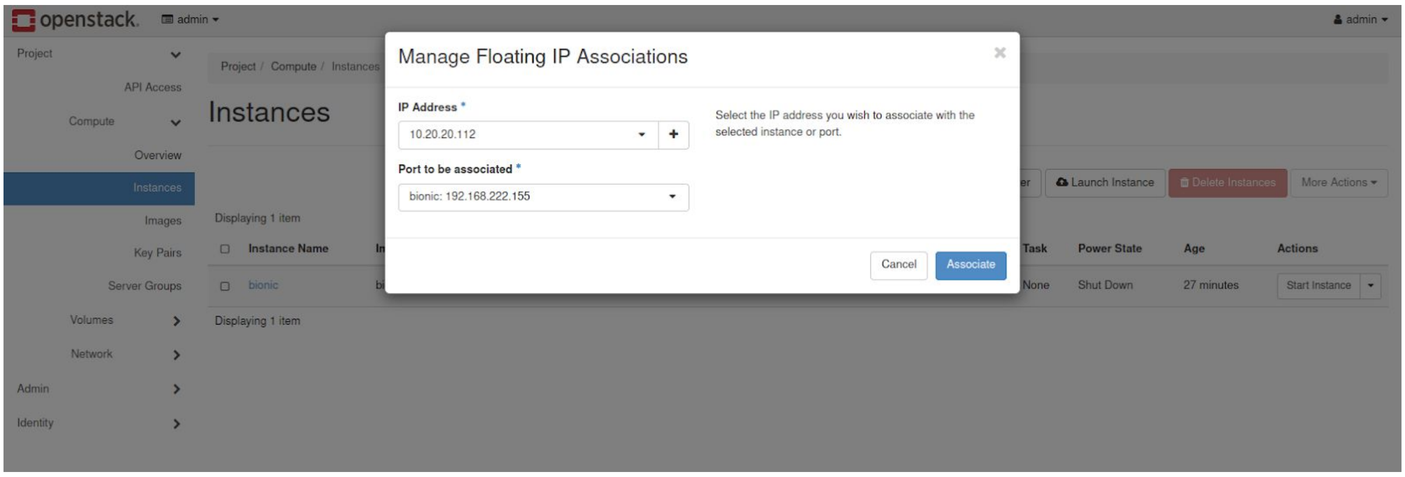


Figure 19. Associate Floating IP to instance

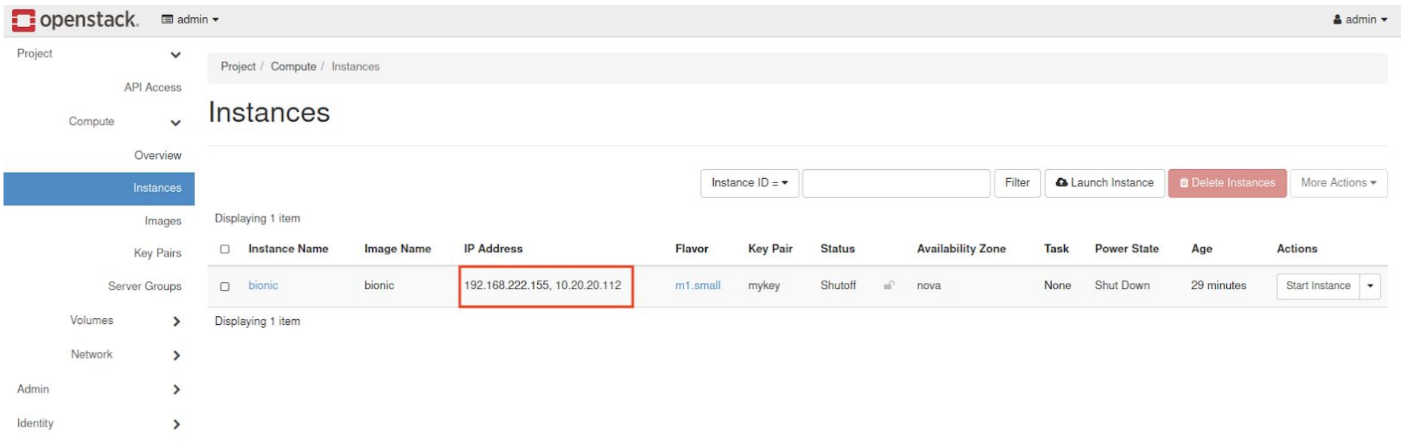
- A window “Manage Floating IP Allocations” will open as shown in Figure 20, select the floating IP created from the drop down and click on “Allocate”. This will allocate the floating IP to the instance. This will be displayed in the dashboard as shown in Figure 21.



Manage Floating IP Associations
IP Address *
10.20.20.112
Port to be associated *
bionic: 192.168.222.155
Select the IP address you wish to associate with the selected instance or port.
Cancel
Associate

Figure 20. Manage Floating IP Associations

- Once allocated it will be shown in the instance.



	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
☐	bionic	bionic	192.168.222.155, 10.20.20.112	m1.small	mykey	Shutoff	nova	None	Shut Down	29 minutes	Start Instance

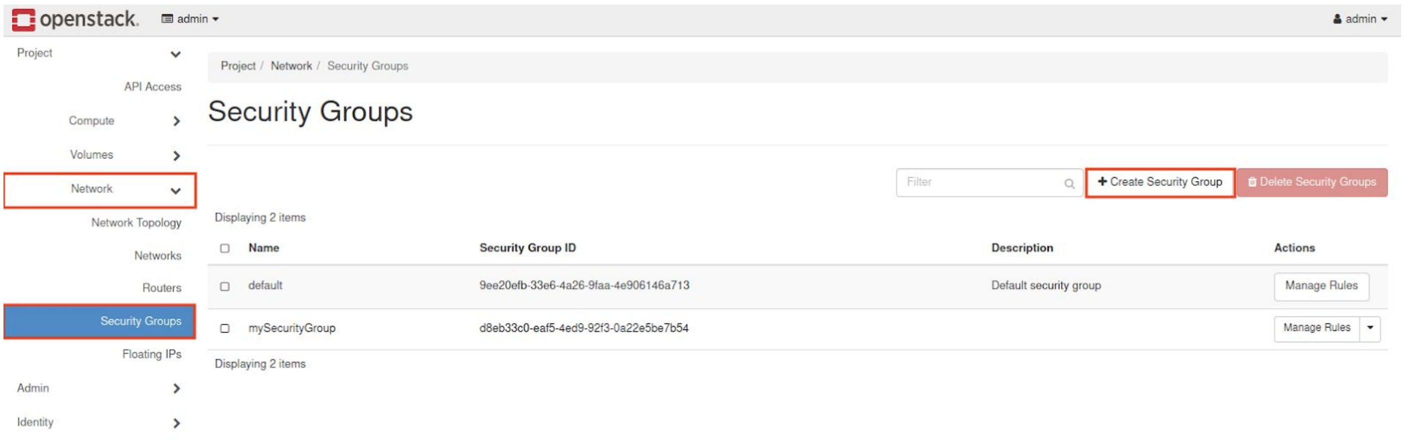
Figure 21. Associated Floating IP

3. Security Group

While creating the instance a default security group is assigned to the instance. The purpose of the security group is to handle the traffic and provide security to the instance. The default security group provided by OpenStack will restrict the traffic to and from the instance.

For this purpose a new security group that allows the traffic flow to and from the instance is created in the dashboard and assigned to the instance.

3.1. Go to the Network tab within the Project tab. Then select the "Security Groups". The screen will be displayed as shown in Figure 22. Then select "Create Security Group" to create a new security group.



Security Groups

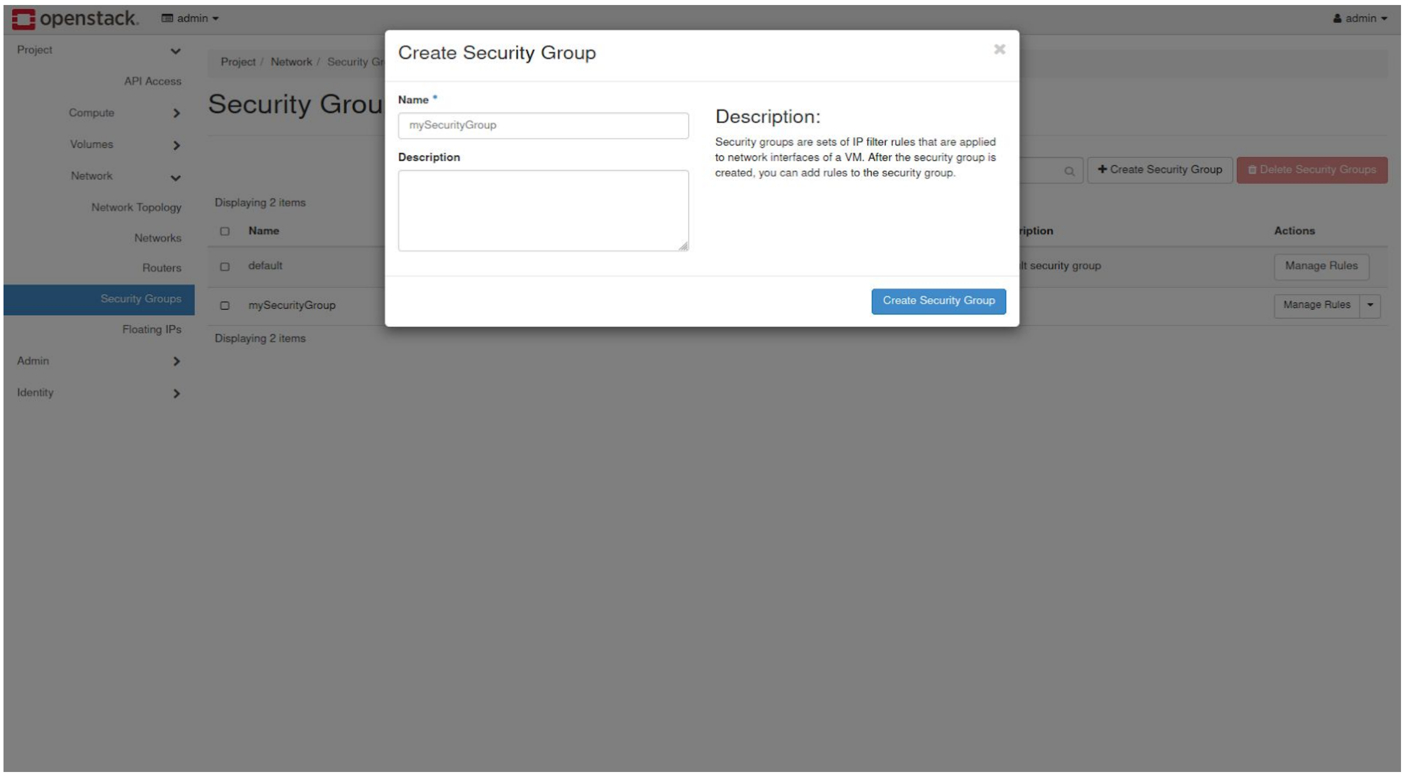
Displaying 2 items

Name	Security Group ID	Description	Actions
default	9ee20efb-33e6-4a26-9faa-4e906146a713	Default security group	Manage Rules
mySecurityGroup	d8eb33c0-eaf5-4ed9-92f3-0a22e5be7b54		Manage Rules

Displaying 2 items

Figure 22 .Security Groups

3.1.1. Provide a name for the security group. There should be no spaces in the name as shown in Figure 23, then click on "Create Security Group"



Create Security Group

Name *: mySecurityGroup

Description:

Description:

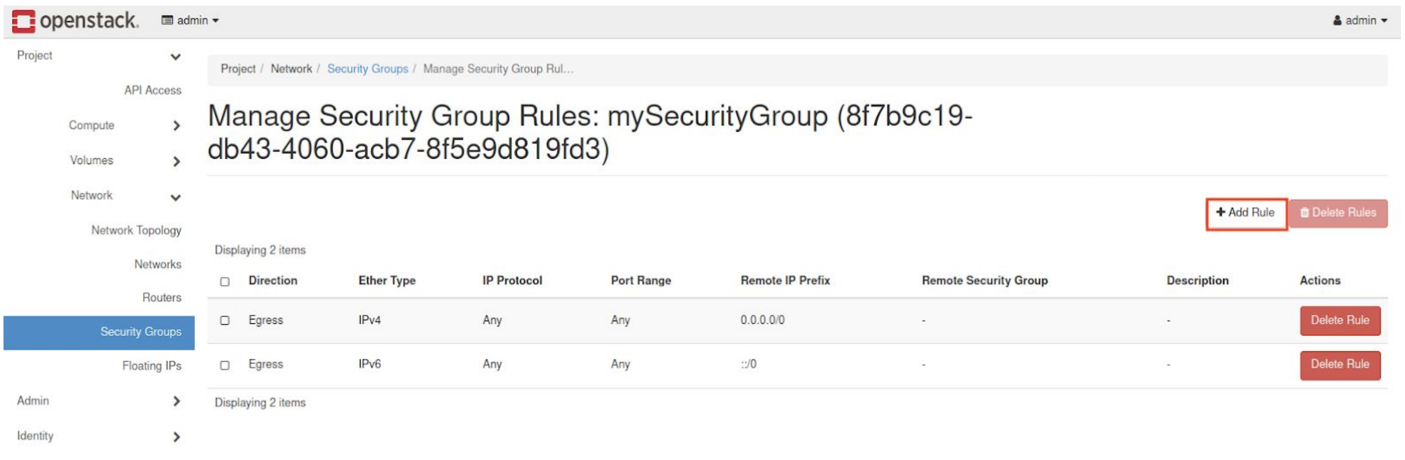
Security groups are sets of IP filter rules that are applied to network interfaces of a VM. After the security group is created, you can add rules to the security group.

Figure 23 .Security Groups Create

3.1.2. A new window will open to show the rules available in the security group created. Add new rules to enable the traffic flow. Click on "Add Rule" as shown in Figure 24. A pop-up window will appear as shown in Figure 25. Create a rule with following specification,

- Rule : Custom TCP Rule
- Direction : Ingress
- Port : Provide a port number using which the application in the instance will be accessed

Then click on "Add". The Figure 26 shows the adding of new rule in dashboard.



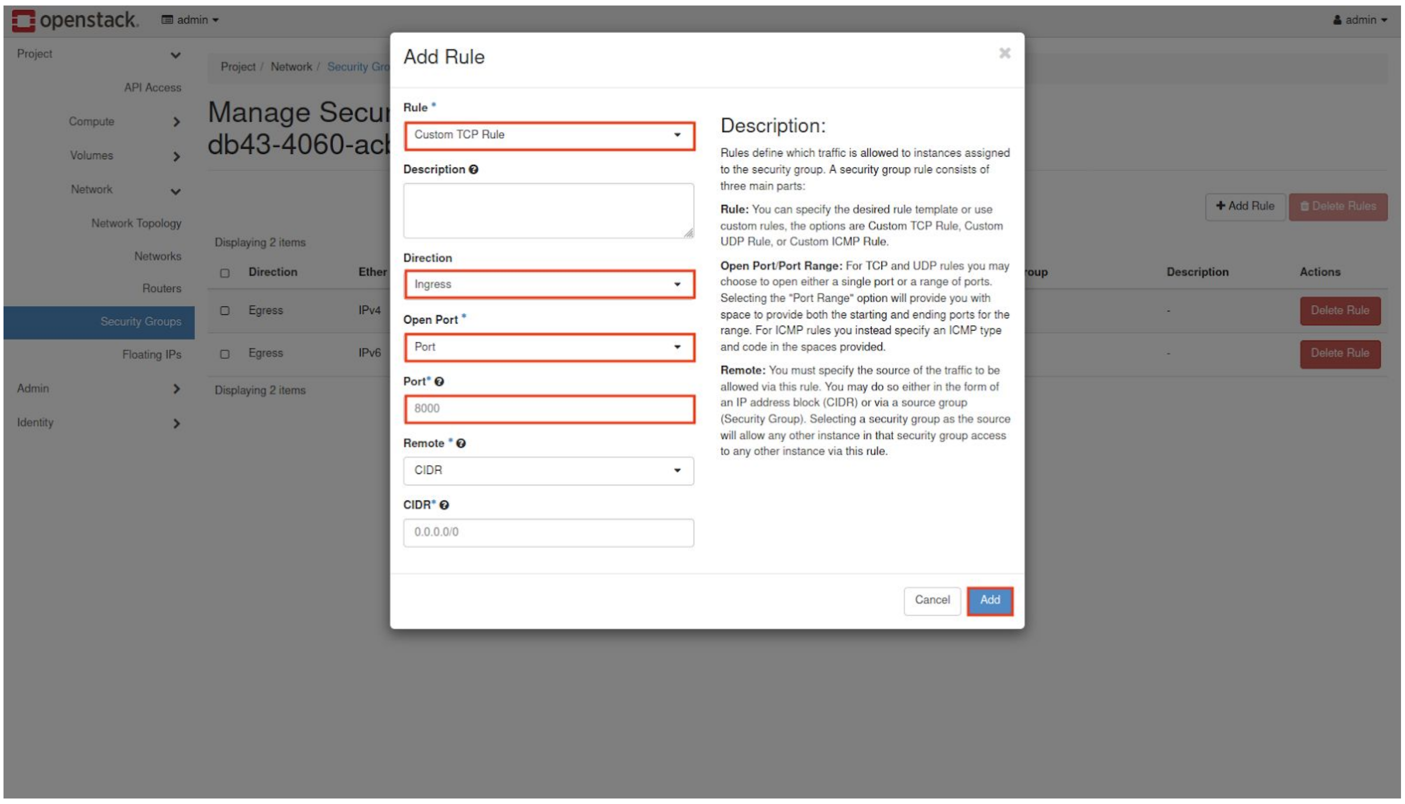
Manage Security Group Rules: mySecurityGroup (8f7b9c19-db43-4060-acb7-8f5e9d819fd3)

Displaying 2 items

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule

Displaying 2 items

Figure 24. Security Group Rules



Add Rule

Rule *: Custom TCP Rule

Description:

Direction: Ingress

Open Port *: Port

Port*: 8000

Remote *: CIDR

CIDR*: 0.0.0.0/0

Description:

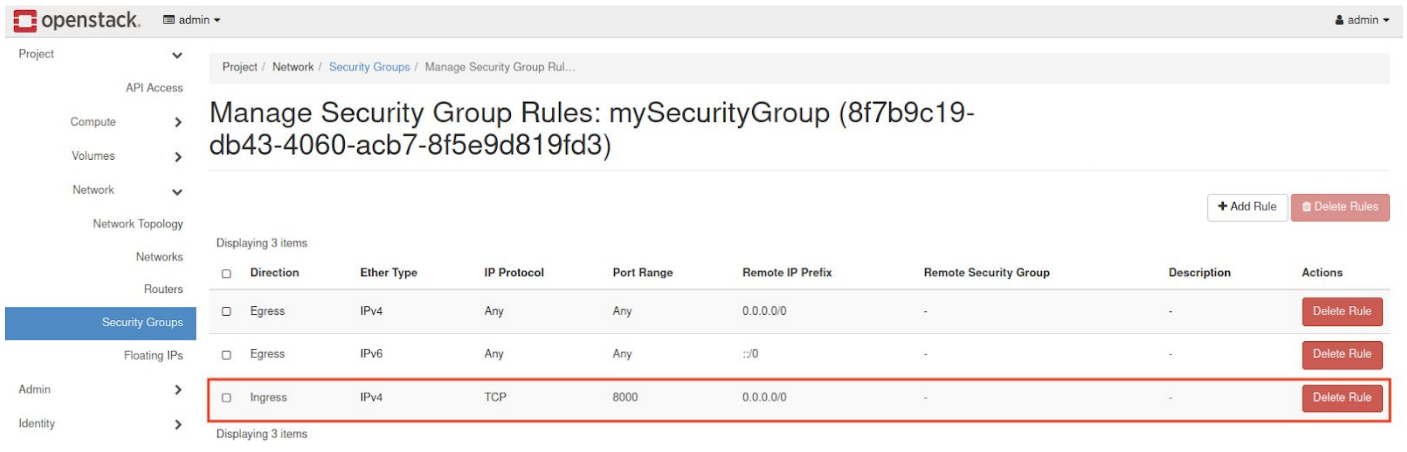
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Figure 25. Adding New Rule



Manage Security Group Rules: mySecurityGroup (8f7b9c19-db43-4060-acb7-8f5e9d819fd3)

Displaying 3 items

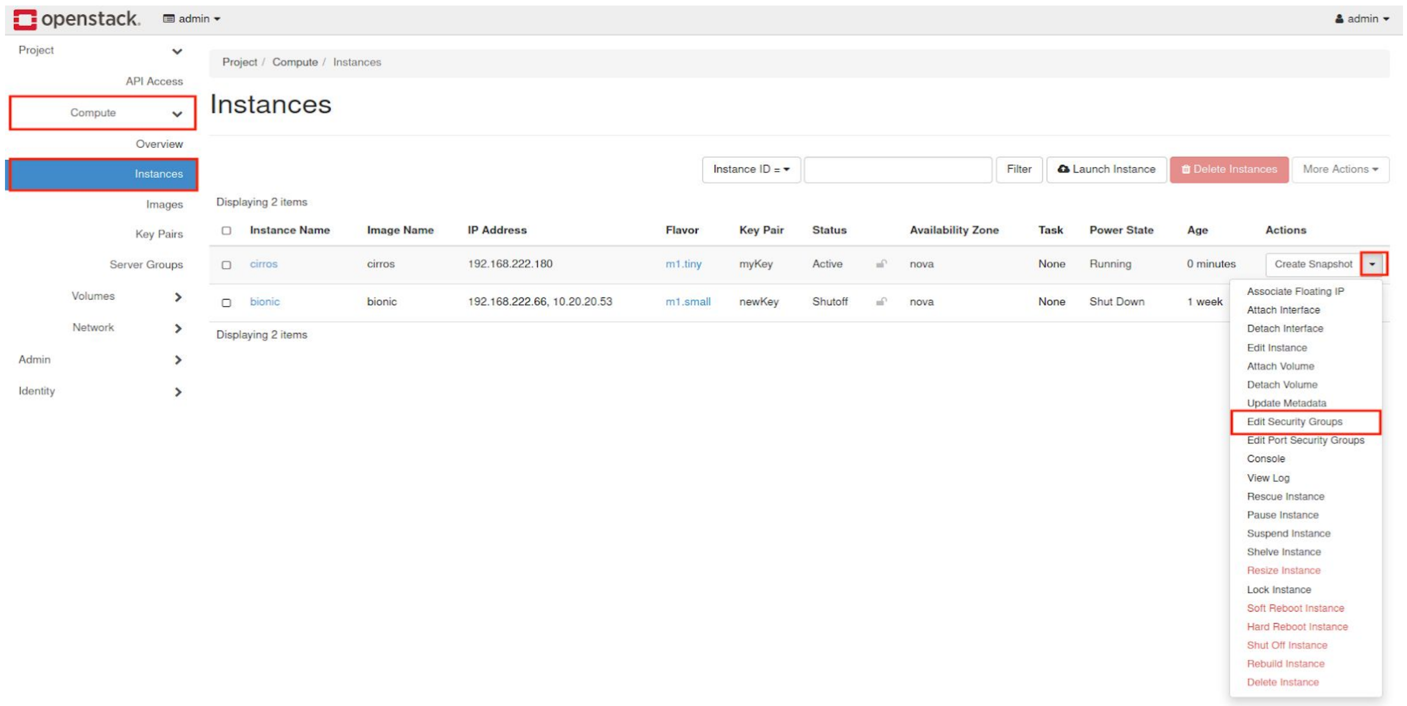
Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
Ingress	IPv4	TCP	8000	0.0.0.0/0	-	-	Delete Rule

Displaying 3 items

Figure 26. New Rule Added

3.1.3. Move to the "Compute" tab and then select "Instance" tab as shown in Figure 27

3.1.4. Select the dropdown at the end of an instance in which the security group has to be updated as displayed in figure(Instances Security Group). Click on "Edit Security Group".



Instances

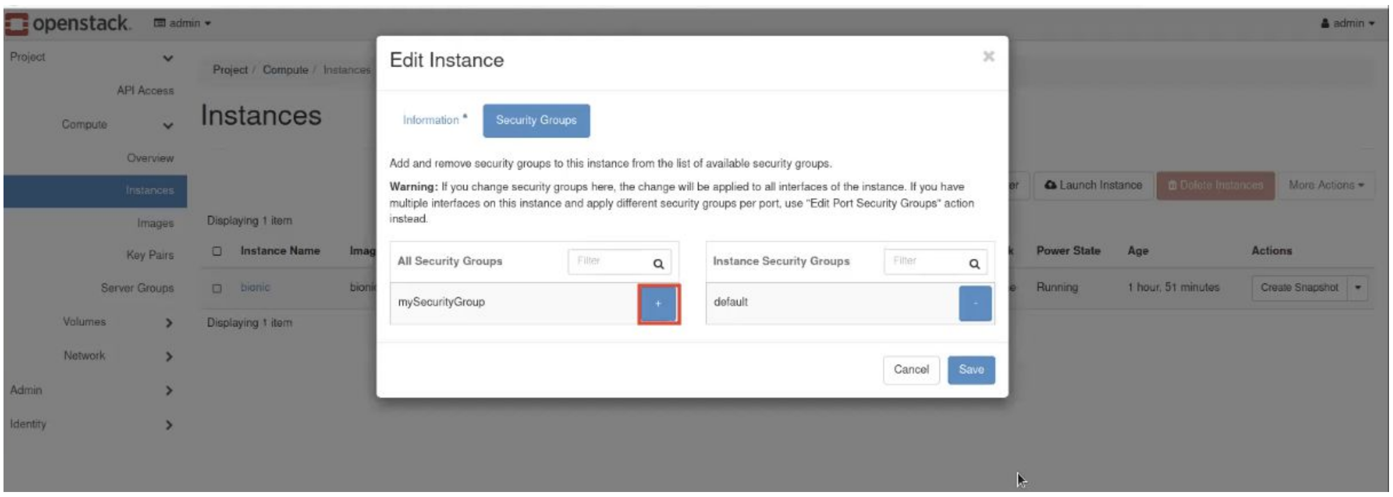
Displaying 2 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
cirros	cirros	192.168.222.180	m1.tiny	myKey	Active	nova	None	Running	0 minutes	Create Snapshot
bionic	bionic	192.168.222.66, 10.20.20.53	m1.small	newKey	Shutoff	nova	None	Shut Down	1 week	

Displaying 2 items

Figure 27. Instances Edit Security Group

3.1.5. A pop-up window will open as shown in Figure 28. Add the security group that is needed from left to right and click on “Save”. This will update the security group for the instance.



Add and remove security groups to this instance from the list of available security groups.

Warning: If you change security groups here, the change will be applied to all interfaces of the instance. If you have multiple interfaces on this instance and apply different security groups per port, use "Edit Port Security Groups" action instead.

Figure 28. Adding new security group to Instance

4. Enabling Internet in the instance

By default the instance created will not have connection to the internet. To enable this the control node should act as a router. This can be achieved by executing the following commands in control-vm terminal.

```
$ sudo iptables -t nat -A POSTROUTING -s 10.20.20.1/24 ! -d 10.20.20.1/24 -j MASQUERADE
```

```
$ sudo sysctl net.ipv4.ip\_forward=1
```

Note: This will not persist if the system is restarted. The commands have to be executed in case the changes are required after restart of control-vm.

5. SSH to the instance

Once all the necessary configuration is done for the instance. The instance can be connected by SSH to the instance from the control-vm machine.

To perform this action, use the key-pair file created and saved in the control-vm (explained in Key-pair creation section) and execute the following command in the control-vm terminal.

```
$ ssh -i <filename> <user@ip address>
```

For example:

```
$ ssh -i bionic-id.pub ubuntu@10.20.20.33
```

Troubleshooting: Incase if there is an issue that the instance is not booted up correctly. The instance can be rebuilt using the same image file. This will erase everything from the instance boot up new instance. Following instructions will explain how to rebuild an instance.

- Click on the drop down provided on the instance you want to rebuild.
- Select the appropriate image file and click on rebuild. This will take a few minutes to rebuild. Once completed the status will turn to Active.

IV. OpenStack Client Tools

OpenStack client[17] is a unified command-line client, which enables access to the project API through easy-to-use commands. The MicroStack CLI syntax is identical to the client delivered by the python-openstack client package. Following commands will interact with the OpenStack to perform the required actions

1. **Service Catalog**: OpenStack keystone service catalog allows API clients to dynamically discover and navigate to cloud services. The service catalog may differ from deployment-to-deployment, user-to-user, and project-to-project[16]. The service catalog itself is composed of a list of services and each service is associated with one or more related endpoints. For additional commands - https://docs.openstack.org/python-openstackclient/latest/cli/command-objects/catalog.html

```
$ microstack.openstack catalog list
```

```
+-----------+-----------+---------------------------------------------------------------------------+
| Name      | Type      | Endpoints                                                                 |
+-----------+-----------+---------------------------------------------------------------------------+
| placement | placement | microstack                                                                |
|           |           |   admin: http://192.168.0.110:8778                                        |
|           |           | microstack                                                                |
|           |           |   public: http://192.168.0.110:8778                                       |
|           |           | microstack                                                                |
|           |           |  internal: http://192.168.0.110:8778                                      |
|           |           |                                                                           |
| nova      | compute   | microstack                                                                |
|           |           |   internal: http://192.168.0.110:8774/v2.1                                |
|           |           | microstack                                                                |
|           |           |   admin: http://192.168.0.110:8774/v2.1                                   |
|           |           | microstack                                                                |
|           |           |   public: http://192.168.0.110:8774/v2.1                                  |
|           |           |                                                                           |
| neutron   | network   | microstack                                                                |
|           |           |   public: http://192.168.0.110:9696                                       |
|           |           | microstack                                                                |
|           |           |   admin: http://192.168.0.110:9696                                        |
|           |           | microstack                                                                |
|           |           |   internal: http://192.168.0.110:9696                                     |
|           |           |                                                                           |
| cinderv3  | volumev3  | microstack                                                                |
|           |           |  internal:http://192.168.0.110:8776/v3/c2bd9d300b5340b79ef5e7798b6f77a4   |
|           |           | microstack                                                                |
|           |           |   admin: http://192.168.0.110:8776/v3/c2bd9d300b5340b79ef5e7798b6f77a4    |
|           |           | microstack                                                                |
|           |           |   public: http://192.168.0.110:8776/v3/c2bd9d300b5340b79ef5e7798b6f77a4   |
|           |           |                                                                           |
| keystone  | identity  | microstack                                                                |
|           |           |   public: http://192.168.0.110:5000/v3/                                   |
|           |           | microstack                                                                |
|           |           |   internal: http://192.168.0.110:5000/v3/                                 |
|           |           | microstack                                                                |
|           |           |   admin: http://192.168.0.110:5000/v3/                                    |
|           |           |                                                                           |
| cinderv2  | volumev2  | microstack                                                                |
|           |           |   admin: http://192.168.0.110:8776/v2/c2bd9d300b5340b79ef5e7798b6f77a4    |
|           |           | microstack                                                                |
|           |           |   public: http://192.168.0.110:8776/v2/c2bd9d300b5340b79ef5e7798b6f77a4   |
|           |           | microstack                                                                |
|           |           |   internal: http://192.168.0.110:8776/v2/c2bd9d300b5340b79ef5e7798b6f77a4 |
|           |           |                                                                           |
| glance    | image     | microstack                                                         |
|           |           |   internal: http://192.168.0.110:9292                              |
|           |           | microstack                                                         |
|           |           |   admin: http://192.168.0.110:9292                                 |
|           |           | microstack                                                         |
|           |           |   public: http://192.168.0.110:9292                                |
|           |           |                                                                    |
+-----------+-----------+--------------------------------------------------------------------+
```

2. **Compute services[18] -** OpenStack Compute is used to host and manage cloud computing systems. OpenStack Compute interacts with OpenStack Identity for authentication, OpenStack Placement for resource inventory tracking and selection, OpenStack Image service for disk and server images, and OpenStack Dashboard for the user and administrative interface. Image access is limited by projects, and by users. For additional commands - https://docs.openstack.org/nova/latest/admin/services.html

```
$ microstack.openstack compute service list
```

```
+----+----------------+--------------------------+----------+---------+-------+----------------------------+
| ID | Binary         | Host                     | Zone     | Status  | State | Updated At                 |
+----+----------------+--------------------------+----------+---------+-------+----------------------------+
|  3 | nova-conductor | node2                    | internal | enabled | up    | 2021-01-21T19:26:29.000000 |
|  4 | nova-scheduler | node2                    | internal | enabled | up    | 2021-01-21T19:26:27.000000 |
|  7 | nova-compute   | node2                    | nova     | enabled | up    | 2021-01-21T19:26:27.000000 |
|  9 | nova-compute   | controller-virtual-machine | nova   | enabled | down  | 2021-01-14T03:11:29.000000 |
| 10 | nova-compute   | compute-virtual-machine  | nova     | enabled | up    | 2021-01-21T19:26:21.000000 |
| 11 | nova-compute   | compute2                 | nova     | enabled | up    | 2021-01-21T19:26:28.000000 |
+----+----------------+--------------------------+----------+---------+-------+----------------------------+
```

3. **Flavors -** Flavors[15] define the compute, memory, and storage capacity of nova computing instances. It specifies the hardware configuration for a server. Execute the following command to list all the flavors. For additiona commands to create and manage flavors - https://docs.openstack.org/nova/latest/user/flavors.html

```
$ microstack.openstack flavor list
```

```
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   20 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   20 |         0 |     4 | True      |
| 5  | m1.xlarge | 16384 |   20 |         0 |     8 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

4. **Floating IP -** Each instance has a private, fixed IP address and can also have a public, or floating IP address. Private IP addresses are used for communication between instances, and public addresses are used for communication with networks outside the cloud, including the Internet. Execute the following command to view the floating IP of the instances. For additional commands to manage IP addresses - https://docs.openstack.org/ocata/user-guide/cli-manage-ip-addresses.html

```
$ microstack.openstack floating ip list
```

```
+------------------------------------+-------------------+---------------+------------------------------------+------------------------------------+--------------------------------+
| ID                                 |Floating IP Address|Fixed IP Address| Port                              |Floating Network                    | Project                        |
+------------------------------------+-------------------+---------------+------------------------------------+------------------------------------+--------------------------------+
|a3845b2b-5b84-4979-bed6-74e213fd0915|10.20.20.53        | 192.168.222.66 |51b1969b-1de6-4225-8592-bdbc05d51092|2d039649-b494-40ef-b02c-028dcc7f2417|c2bd9d300b5340b79ef5e7798b6f77a4|
+------------------------------------+--------------------+------------------+------------------------------------+------------------------------------+--------------------------------+
```

5. **Hypervisor** - OpenStack Compute supports many hypervisors such as KVM, LXC, QEMU etc.[19]

```
$ microstack.openstack hypervisor list
```

```
+----+---------------------------+-----------------+---------------+-------+
| ID | Hypervisor Hostname       | Hypervisor Type | Host IP       | State |
+----+---------------------------+-----------------+---------------+-------+
|  1 | node2                     | QEMU            | 192.168.0.110 | up    |
|  2 | controller-virtual-machine | QEMU           | 192.168.0.104 | down  |
|  3 | compute-virtual-machine   | QEMU            | 192.168.0.105 | up    |
|  4 | compute2                  | QEMU            | 192.168.0.106 | up    |
+----+---------------------------+-----------------+---------------+-------+
```

6. **Image -** A virtual machine image is a single file which contains a virtual disk that has a bootable operating system installed on it. The Following command retrieves the list of images. To get further details about a single image, use **openstack image show <image-name>** command[20].

```
$ microstack.openstack image list
```

```
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| 54627c07-61c9-4185-b2ad-f8cea7be4aa5 | bionic | active |
| cbdfad7c-a5be-4335-93bd-c7be28c87a0c | cirros | active |
+--------------------------------------+--------+--------+
```

7. **Keypair -** After launching a virtual machine, a key pair has to be injected, which allows SSH access to the instance. A single key pair can be used for multiple instances that belong to that project. Execute the following command to list the key pair.

```
$ microstack.openstack keypair list
```

```
+------------+-------------------------------------------------+
| Name       | Fingerprint                                     |
+------------+-------------------------------------------------+
| microstack | 3d:43:ec:21:37:0a:11:21:e2:ae:b1:3d:6f:1d:ae:db |
| myKey      | 9c:49:6e:fc:36:fa:39:3a:47:47:48:03:7f:f0:a0:f0 |
| newKey     | cb:80:5a:a9:5b:79:af:e4:46:c0:20:24:3d:03:3c:81 |
| testkey    | 22:68:e6:7d:bf:6b:91:41:61:42:dc:46:03:e5:79:fd |
+------------+-------------------------------------------------+
```

8. **Networks** - OpenStack Networking handles the creation and management of a virtual networking infrastructure, including networks, switches, subnets, and routers for devices managed by the OpenStack

Compute service (nova). A network is an isolated Layer 2 networking segment. There are two types of networks, project and provider networks. Project networks are fully isolated and are not shared with other projects. Only an OpenStack administrator can create provider networks. Networks can be connected via routers. Execute the following commands to list the networks. For additional commands to manage networks - https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/network.html

```
$ microstack.openstack network list
```

```
+--------------------------------------+----------+--------------------------------------+
| ID                                   | Name     | Subnets                              |
+--------------------------------------+----------+--------------------------------------+
| 2d039649-b494-40ef-b02c-028dcc7f2417 | external | dfe00b34-077e-49e6-b254-227ed175e522 |
| 9a96c71e-2ea8-4b57-8fce-0ccc9016e319 | test     | bfbfc303-0281-4d6d-b501-0da5572eed1a |
+--------------------------------------+----------+--------------------------------------+
```

9. **Security Groups -** Security groups are sets of IP filter rules that are applied to all project instances, which define networking access to the instance. Group rules are project specific; project members can edit the default rules for their group and add new rule sets.

```
$ microstack.openstack security group list
```

```
+--------------------------------------+-----------------+------------------------+----------------------------------+------+
| ID                                   | Name            | Description            | Project                          | Tags |
+--------------------------------------+-----------------+------------------------+----------------------------------+------+
| 9ee20efb-33e6-4a26-9faa-4e906146a713 | default         | Default security group | c2bd9d300b5340b79ef5e7798b6f77a4 | []   |
| d8eb33c0-eaf5-4ed9-92f3-0a22e5be7b54 | mySecurityGroup |                        | c2bd9d300b5340b79ef5e7798b6f77a4 | []   |
| de059903-71ab-416b-970c-08f8681118d9 | default         | Default security group | d6d822f4ef67469fbf64bc4b8379461c | []   |
+--------------------------------------+-----------------+------------------------+----------------------------------+------+
```

10. **Server -** A server[14] is a virtual machine (VM) instance, a physical machine or a container. Execute the following command to view the list of servers. For additional commands to create and manage servers - https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/server.html#server-list

```
$ microstack.openstack server list
```

```
+--------------------------------------+--------+---------+----------------------------------+--------+----------+
| ID                                   | Name   | Status  | Networks                         | Image  | Flavor   |
+--------------------------------------+--------+---------+----------------------------------+--------+----------+
| 35479735-8b26-447d-a07f-d97c65ff0397 | bionic | SHUTOFF | test=192.168.222.66, 10.20.20.53 | bionic | m1.small |
+--------------------------------------+--------+---------+----------------------------------+--------+----------+
```

Additional commands can be found by executing this command,

```
$ microstack.openstack command list
```

V. API

- Install CURL using the following command:

```
$ sudo apt install curl
```

- Environment Variables:

```
$ export OS_PROJECT_NAME=admin
$ export OS_USERNAME=adminAPI
$ export OS_PASSWORD=<password>
$ export OS_USER_DOMAIN_NAME=Default
$ export OS_PROJECT_DOMAIN_NAME=Default
$ export OS_AUTH_URL=http://192.168.64.2:5000/v3/
$ export OS_HORIZON_URL=http://192.168.64.2:8774/v2.1/
```

- API to get the auth token:

```
$  curl  -v  -s  -X  POST  $OS_AUTH_URL/auth/tokens?nocatalog     -H  "Content-Type:
application/json"
-d  '{  "auth":  {  "identity":  {  "methods":  ["password"],"password":  {"user":
{"domain":    {"name":    "'"$OS_USER_DOMAIN_NAME"'"},"name":    "'"$OS_USERNAME"'",
"password": "'"$OS_PASSWORD"'"} } },
"scope":  {  "project":  {  "domain":  {  "name":  "'"$OS_PROJECT_DOMAIN_NAME"'"  },
"name":
"'"$OS_PROJECT_NAME"'" } } }}' \ | python3 -m json.tool
```

Output :

```
*   Trying 192.168.0.110:5000...
* TCP_NODELAY set
* Connected to 192.168.0.110 (192.168.0.110) port 5000 (#0)
> POST /v3//auth/tokens?nocatalog HTTP/1.1
> Host: 192.168.0.110:5000
> User-Agent: curl/7.68.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 252
>
} [252 bytes data]
* upload completely sent off: 252 out of 252 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 201 CREATED
< Server: nginx/1.19.0
< Date: Thu, 21 Jan 2021 19:29:53 GMT
< Content-Type: application/json
< Content-Length: 648
< Connection: keep-alive
< X-Subject-Token: gAAAAABgCdYwP0u9ZfF9TVLBrG9h4I8brbaUpcYWlk3lG2eh3p7Og2R9bABLd34jHpJtNTkj36OIzp_rR-
<                  MNPsbfhPjKqMntsrI99dmsLeY6xP0dFrH0NPthyF2Pj7xXYk4Jb5elxsBVYGaf889ToBJSDiC4XyZVac-Y
<                  pGVXkznWuh-Z04AlGSQ
< Vary: X-Auth-Token
< x-openstack-request-id: req-44038b92-f718-437c-8662-be6f2c04c09a
<
{ [648 bytes data]
* Connection #0 to host 192.168.0.110 left intact
* Closing connection -1
{
    "token": {
        "methods": [
            "password"
        ],
        "user": {
            "domain": {
                "id": "default",
                "name": "Default"
            },
            "id": "c543b521d98d494d85fc362c9185cd84",
            "name": "admin",
            "password_expires_at": null
        },
        "audit_ids": [
            "KsooIzFzTVqQy2I0WwCY6g"
        ],
        "expires_at": "2021-01-21T20:29:52.000000Z",
        "issued_at": "2021-01-21T19:29:52.000000Z",
        "project": {
            "domain": {
                "id": "default",
                "name": "Default"
            },
            "id": "c2bd9d300b5340b79ef5e7798b6f77a4",
            "name": "admin"
        },
        "is_domain": false,
        "roles": [
            {
                "id": "d710ac3171e24af39f5e19758d1def8b",
                "name": "member"
            },
            {
                "id": "ba0fb7c382924ad39dd5c909c7ee9343",
                "name": "admin"
            },
            {
                "id": "dbcadf4ca8d84962b252a10f382b85f2",
                "name": "reader"
            }
        ]
    }
}
```

- Copy the X-Subject-Token from the response header and export to the environment variable as

```
$ export OS_TOKEN=<X-Subject-Token>
```

- Copy the project id (project/id) from the response JSON and export to the environment variable as

```
$ export OS_PROJECT_ID=<project-id>
```

- API to retrieve flavors

```
$ curl -s -H "X-Auth-Token: $OS_TOKEN" $OS_HORIZON_URL/$OS_PROJECT_ID/flavors |
python3 -m json.tool
```

Output:

```
{
    "flavors": [
        {
            "id": "1",
            "name": "m1.tiny",
            "links": [
                {
                    "rel": "self",
                    "href": "http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/1"
                },
                {
                    "rel": "bookmark",
                    "href": "http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/1"
                }
            ]
        },
        {
            "id": "2",
            "name": "m1.small",
            "links": [
                {
                    "rel": "self",
                    "href": "http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/2"
                },
                {
                    "rel": "bookmark",
                    "href": "http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/2"
                }
            ]
        },
        {
            "id": "3",
            "name": "m1.medium",
            "links": [
                {
                    "rel": "self",
                    "href": "http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/3"
                },
                {
                    "rel": "bookmark",
                    "href": "http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/3"
                }
            ]
        },
        {
            "id": "4",
            "name": "m1.large",
            "links": [
                {
                    "rel": "self",
                    "href": "http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/4"
                },
                {
                    "rel": "bookmark",
                    "href": "http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/4"
                }
            ]
        },
        {
            "id": "5",
            "name": "m1.xlarge",
            "links": [
                {
                    "rel": "self",
                    "href": "http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/5"
                },
                {
                    "rel": "bookmark",
                    "href": "http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/flavors/5"
                }
            ]
        }
    ]
}
```

- API to retrieve images

```
$ curl -s -H "X-Auth-Token: $OS_TOKEN" $OS_HORIZON_URL/$OS_PROJECT_ID/images |
python3 -m json.tool
```

Output:

```
{
    "images": [
        {
            "id": "54627c07-61c9-4185-b2ad-f8cea7be4aa5",
            "name": "bionic",
            "links": [
                {
                    "rel": "self",
                    "href":
"http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/images/54627c07-61c9-4185-b2ad-f8cea7be4aa
5"
                },
                {
                    "rel": "bookmark",
                    "href":
"http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/images/54627c07-61c9-4185-b2ad-f8cea7be4aa5"
                },
                {
                    "rel": "alternate",
                    "type": "application/vnd.openstack.image",
                    "href": "http://192.168.0.110:9292/images/54627c07-61c9-4185-b2ad-f8cea7be4aa5"
                }
            ]
        },
        {
            "id": "cbdfad7c-a5be-4335-93bd-c7be28c87a0c",
            "name": "cirros",
            "links": [
                {
                    "rel": "self",
                    "href":
"http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/images/cbdfad7c-a5be-4335-93bd-c7be28c87a0
c"
                },
                {
                    "rel": "bookmark",
                    "href":
"http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/images/cbdfad7c-a5be-4335-93bd-c7be28c87a0c"
                },
                {
                    "rel": "alternate",
                    "type": "application/vnd.openstack.image",
                    "href": "http://192.168.0.110:9292/images/cbdfad7c-a5be-4335-93bd-c7be28c87a0c"
                }
            ]
        }
    ]
}
```

- API to retrieve servers

```
$ curl -s -H "X-Auth-Token: $OS_TOKEN" $OS_HORIZON_URL/$OS_PROJECT_ID/images |
python3 -m json.tool
```

Output:

```
{
    "servers": [
        {
            "id": "35479735-8b26-447d-a07f-d97c65ff0397",
            "name": "bionic",
            "links": [
                {
                    "rel": "self",
                                                                                            "href":
"http://192.168.0.110:8774/v2.1/c2bd9d300b5340b79ef5e7798b6f77a4/servers/35479735-8b26-447d-a07f-d97
c65ff0397"
                },
                {
                    "rel": "bookmark",
                                                                                            "href":
"http://192.168.0.110:8774/c2bd9d300b5340b79ef5e7798b6f77a4/servers/35479735-8b26-447d-a07f-d97c65ff
0397"
                }
            ]
        }
    ]
}
```

For more APIs:
https://docs.openstack.org/api-quick-start/index.html#current-api-versions

VI. Web Application Deployment

The instance created can be used to deploy a web application or any other PaaS. Following instructions will explain the deployment of a web application that provides a simulation on real-time scheduling algorithms[11].

1. Download the source code using the following command.

```
$ git clone https://github.com/bhatvineeth/SchedulingSimulation.git
```

2. Install pip3 using python3. Before installing pip3 update the ubuntu by running following command,

```
$ sudo apt-get update
```

Now install pip3 by running the following command,

```
$ sudo apt install python3-pip
```

3. Install a virtual environment.

```
$ pip3 install virtualenv
```

4. Change the directory to SchedulingSimulator app

```
$ cd SchedulingSimulation/SchedulingSimulator/
```

5. Install Virtual env within this folder by executing following command

```
$ sudo apt install virtualenv
```

6. To create a virtual env, it has to find the pyhton3 files so execute this to find the python3 source folder,

```
$ which python3
```

7. Now execute the below command using the python3 path obtained in the previous step.

```
$  virtualenv env -p <python3 path>
$ . env/bin/activate
```

8. Install django

```
$ pip3 install django
```

9. Change the directory to SchedulingSimulator

```
$ cd SchedulingSimulator
```

10. In settings.py edit the ALLOWED_HOSTS: [ ] to ALLOWED_HOSTS: [<ip address of inscance>] then save the file.

11. Execute manage.py to start the server. Note that this port 8000 must be added in the security groups while creating a new security group.

```
$ python manage.py runserver 0.0.0.0:8000
```

12. Application will be available on localhost:8000 in the browser of control-vm.
To know more about the application and its usage refer the following literature, https://github.com/bhatvineeth/SchedulingSimulation/blob/master/Documentation/Paper/Scheduling_Simulator.pdf

VII. Conclusion

To conclude, OpenStack is a good open source Infrastructure as a Service (IaaS), that provides huge potential in scalability, by allowing a large number of nodes interconnected to provide the necessary services. Also, providing flexibility by having modular components that interact to form the final infrastructure. These modular components can be added or removed when the necessity arises. In this document it shows the deployment of OpenStack using Microstack, through which one can easily deploy the infrastructure. Micostack deploys OpenStack with minimal system requirements and also handles the burden of configuration of OpenStack and its network before deployment. The main intention of Microstack is to provide an OpenStack environment in a developer's system for testing or development purposes and also support IoT applications. Microstack is a part of Canonical and it only works on Ubuntu. There are various tools available that help in deployment like Devstack. Which is provided by OpenStack for deployment. The downside of this is it takes a considerable amount of time to deploy and the configurations are to be done manually and requires the system to be highly capable.

VIII. References

[1] "OpenStack: Open Source Cloud Computing Infrastructure" - https://www.openstack.org/, Accessed On: 29/01/2021

[2] "Microstack Overview" - https://ubuntu.com/tutorials/microstack-get-started#1-overview/, Accessed On: 29/01/2021

[3] "OpenStack Docs: DevStack Overview" - https://docs.openstack.org/devstack/latest/, Accessed On: 29/01/2021

[4] "Packstack — RDO" - https://www.rdoproject.org/install/packstack/, Accessed On: 29/01/2021

[5] "Single-node OpenStack deployment" - https://ubuntu.com/openstack/install#single-node-deployment/, Accessed On: 29/01/2021

[6] "Get Ubuntu Server" - https://ubuntu.com/download/server/, Accessed On: 29/01/2021

[7] "Fedora" - https://getfedora.org/, Accessed On: 29/01/2021

[8] "CentOS Download" - https://www.centos.org/download/, Accessed On: 29/01/2021

[9] "openSUSE TOOLS" - https://www.opensuse.org/, Accessed On: 29/01/2021

[10]"Scheduling Simulator Codebase Github" - https://github.com/bhatvineeth/SchedulingSimulation/, Accessed On: 29/01/2021

[11]"Scheduling Simulator Report" - https://github.com/bhatvineeth/SchedulingSimulation/blob/master/Documentation/Paper/Scheduling_Simulator.pdf, Accessed On: 29/01/2021

[12]"Snap" - https://snapcraft.io/, Accessed On: 29/01/2021

[13]"OpenStack Server concepts" - https://docs.openstack.org/api-guide/compute/server_concepts.html/, Accessed On: 30/01/2021

[14]"OpenStack Flavors" - https://docs.openstack.org/nova/latest/user/flavors.html/, Accessed On: 30/01/2021

[15]"Service Catalog Overview" - https://docs.openstack.org/keystone/latest/contributor/service-catalog.html/, Accessed On: 30/01/2021

[16]"OpenStack Client" - https://docs.openstack.org/python-openstackclient/latest/, Accessed On:30/01/2021

[17]"OpenStack Compute Service" - https://docs.openstack.org/nova/latest/install/index.html, Accessed On: 30/01/2021

[18]"OpenStack Hypervisors" - https://docs.openstack.org/ocata/config-reference/compute/hypervisors.html, Accessed On: 30/01/2021

[19]"OpenStack Virtual Machine Image Guide" - https://docs.openstack.org/image-guide/index.html, Accessed On: 30/01/2021

[20]"OpenStack Networking" - https://docs.openstack.org/mitaka/networking-guide/intro.html, Accessed On: 30/01/2021

[21]"Manage project security" - https://docs.openstack.org/nova/latest/admin/security-groups.html, Accessed On: 30/01/2021