Neutron

OpenStack Lab 4

I. OpenStack Dashboard

The goal of this part is to familiarize yourself with OpenStack dashboard Fig 1. The focus will be on the following operations (for more information: https://docs.openstack.org/user-guide/dashboard.html):

- Setting up SSH keys
- Setting up networks
- Creating VM

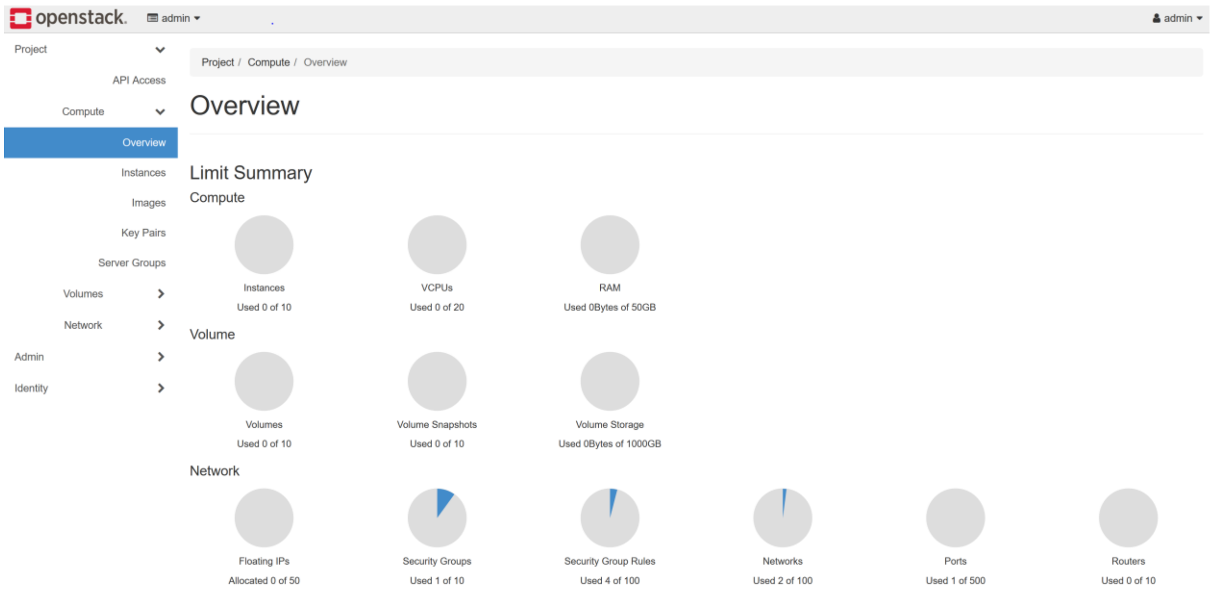


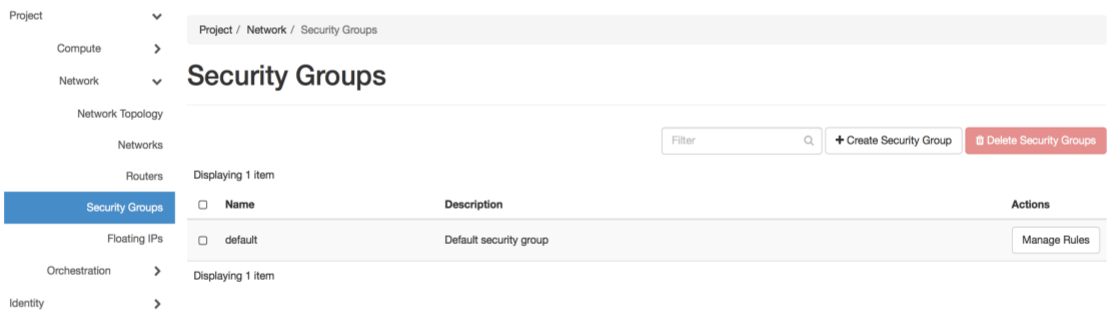
Fig 1. Project Dashboard

Setting up SSH keys

Setting up SSH keys in OpenStack allows you to securely log in to your virtual machines (VMs) without using a password. Instead of a password, you use a private key to log in to your VMs. In order to be able to SSH your instances, you need to create and download your keypair file. For that to happen you need to do the following two steps: enable ssh service and create keypair.

1) a) Dashboard: Enabling SSH

On the Project tab, open the Network tab, then Security Groups. You will get a window similar to Fig 2 under Security Group tab. Check if **ssh** service is enabled by clicking Manage Rules button on far right (see Fig 3).



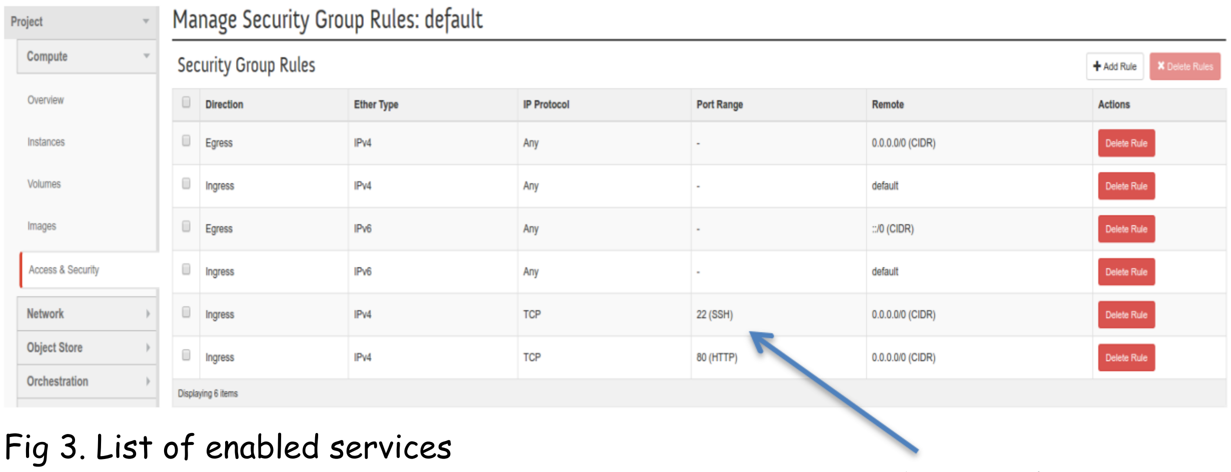
Security Groups

Displaying 1 item

	Name	Description	Actions
☐	default	Default security group	Manage Rules

Displaying 1 item

Fig 2. Security Group



Manage Security Group Rules: default

Security Group Rules

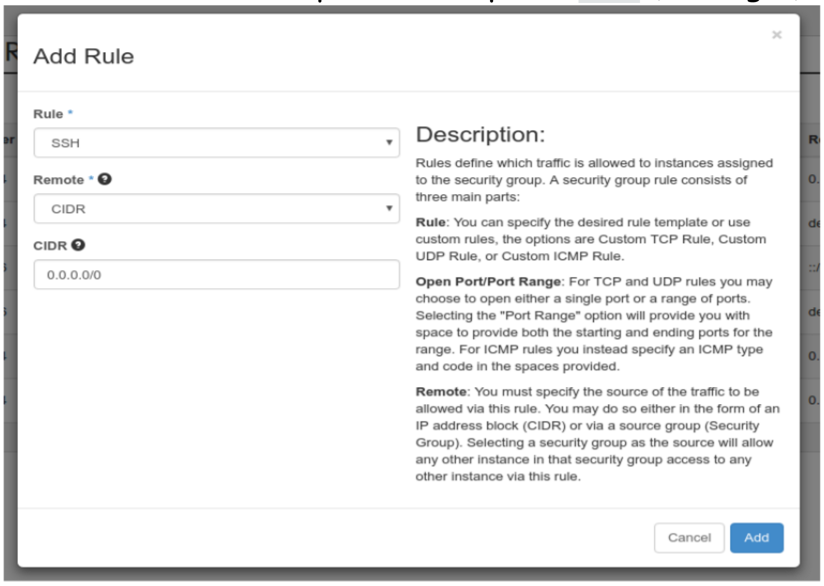
	Direction	Ether Type	IP Protocol	Port Range	Remote	Actions
☐	Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)	Delete Rule
☐	Ingress	IPv4	Any	-	default	Delete Rule
☐	Egress	IPv6	Any	-	::/0 (CIDR)	Delete Rule
☐	Ingress	IPv6	Any	-	default	Delete Rule
☐	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0 (CIDR)	Delete Rule
☐	Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0 (CIDR)	Delete Rule

Displaying 6 items

Fig 3. List of enabled services

ssh is enabled

If **ssh** is not in the list, select +Add Rule which will bring up a popup. Select **ssh** from the drop down and press Add (see fig 4).



Add Rule

Rule *: SSH

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.

Fig 4. Adding ssh rule.

Note that you can follow the same step to enable any service, for example http.

b) CLI: Enabling SSH

To enable SSH using OpenStack CLI, you can use the following steps:

1. Login to your OpenStack environment using the OpenStack CLI.

2. Create a new security group that allows SSH traffic.

   ```
   sql
   openstack security group create ssh
   openstack security group rule create --proto tcp --dst-port 22 ssh
   ```

   openstack security group rule create --ingress --protocol tcp --dst-port 22 ssh

3. Launch a new instance and associate the newly created security group with it.

   to verify: openstack security group rule list ssh

   ```
   css
   openstack server create --flavor <flavor-name> --image <image-name> --
   security-group ssh <instance-name>
   ```

4. Get the floating IP address for the instance.

   ```
   php
   openstack floating ip create <network-name>
   openstack server add floating ip <instance-name> <floating-ip-address>
   ```

5. SSH into the instance using the floating IP address.

   ```
   php
   ssh <user-name>@<floating-ip-address>
   ```

Note: Replace the placeholders <flavor-name>, <image-name>, <instance-name>, <network-name>, <floating-ip-address>, and <user-name> with your own values.

c) OpenStack Python SDK: Enabling SSH

To enable SSH using OpenStack Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

```
pip install openstacksdk
```

2. Import the necessary modules in your Python script:

```
python
import openstack
```

3. Connect to your OpenStack environment:

```
python
conn = openstack.connect(cloud='<cloud-name>')
```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Create a new security group that allows SSH traffic:

```
python
security_group = conn.network.create_security_group(name='ssh')
conn.network.create_security_group_rule(
                                      security_group_id=security_group.id,
                                       direction='ingress',
                                       protocol='tcp',
                                       port_range_min=22,
                                       port_range_max=22)
```

5. Launch a new instance and associate the newly created security group with it:

```
python
instance = conn.compute.create_server(name='<instance-name>',
                        image='<image-id>',
                        flavor='<flavor-id>',
                        security_groups=[security_group.id])
```

Replace the placeholders <instance-name>, <image-id>, and <flavor-id> with your own values.

6. Get the floating IP address for the instance:

```
python
floating_ip = conn.network.create_ip(floating_network_id='<network-id>')
```

Neutron

```
conn.compute.add_floating_ip_to_server(instance,
floating_ip['floating_ip_address'])
```

Replace the placeholder <network-id> with the ID of the network to which you want to assign the floating IP.

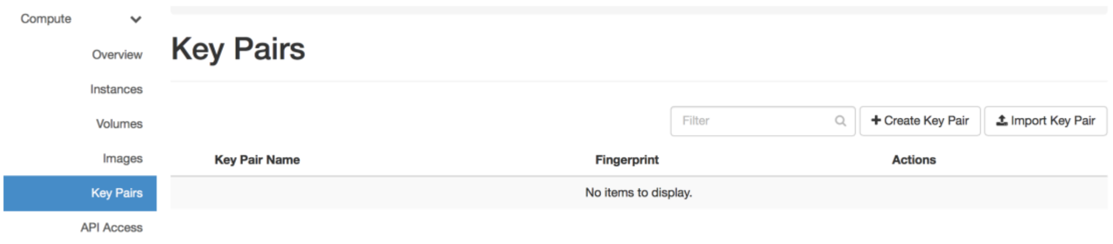
7. SSH into the instance using the floating IP address:

```
php
ssh <user-name>@<floating-ip-address>
```

Replace the placeholders <user-name> and <floating-ip-address> with your own values.

2) a) Dashboard: Creating key pair

On the Project tab, open the Compute tab, then Key Pairs. You will get similar to Fig 5 under Key Pairs tab. Select +Create Key Pair and provide a name to your key pair. The file will be downloaded automatically (if not please download the file manually). Now you can use the **ssh** command to make a secure connection to your instance (We will see how that later once a VM is created).



Key Pair Name	Fingerprint	Actions
No items to display.		

Fig 5. Creating Key Pair.

b) CLI: Creating key pair

To create a key pair using OpenStack CLI, you can use the following steps:

1. Login to your OpenStack environment using the OpenStack CLI.
2. Create a new key pair and save the private key to a file:

```
vbnet
openstack keypair create --public-key <public-key-file> <key-pair-name> >
<private-key-file>
```

Replace the placeholders <public-key-file>, <key-pair-name>, and <private-key-file> with your own values.

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/mykey
openstack keypair create --public-key ~/.ssh/mykey.pub mykey
openstack keypair list
```

Note: The <public-key-file> should contain the public key content, which is used to verify the authenticity of the private key. The private key content will be written to <private-key-file>.

After creating the key pair, you can use it when launching instances in your OpenStack environment. When you launch an instance with a key pair, you can log into the instance using the private key.

c) OpenStack Python SDK: Creating key pair

To create a key pair using OpenStack Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

    ```
    pip install openstacksdk
    ```

2. Import the necessary modules in your Python script:

    ```
    python
    import openstack
    ```

3. Connect to your OpenStack environment:

    ```
    python
    conn = openstack.connect(cloud='<cloud-name>')
    ```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Generate a new key pair and save the private key to a file:

    ```
    python
    keypair = conn.compute.create_keypair(name='<key-pair-name>',
    public_key='<public-key-file>')
    with open('<private-key-file>', 'w') as f:
        f.write(keypair['private_key'])
    ```

Replace the placeholders <key-pair-name>, <public-key-file>, and <private-key-file> with your own values.

Note: The <public-key-file> should contain the public key content, which is used to verify the authenticity of the private key. The private key content will be written to <private-key-file>.

After creating the key pair, you can use it when launching instances in your OpenStack environment. When you launch an instance with a key pair, you can log into the instance using the private key.

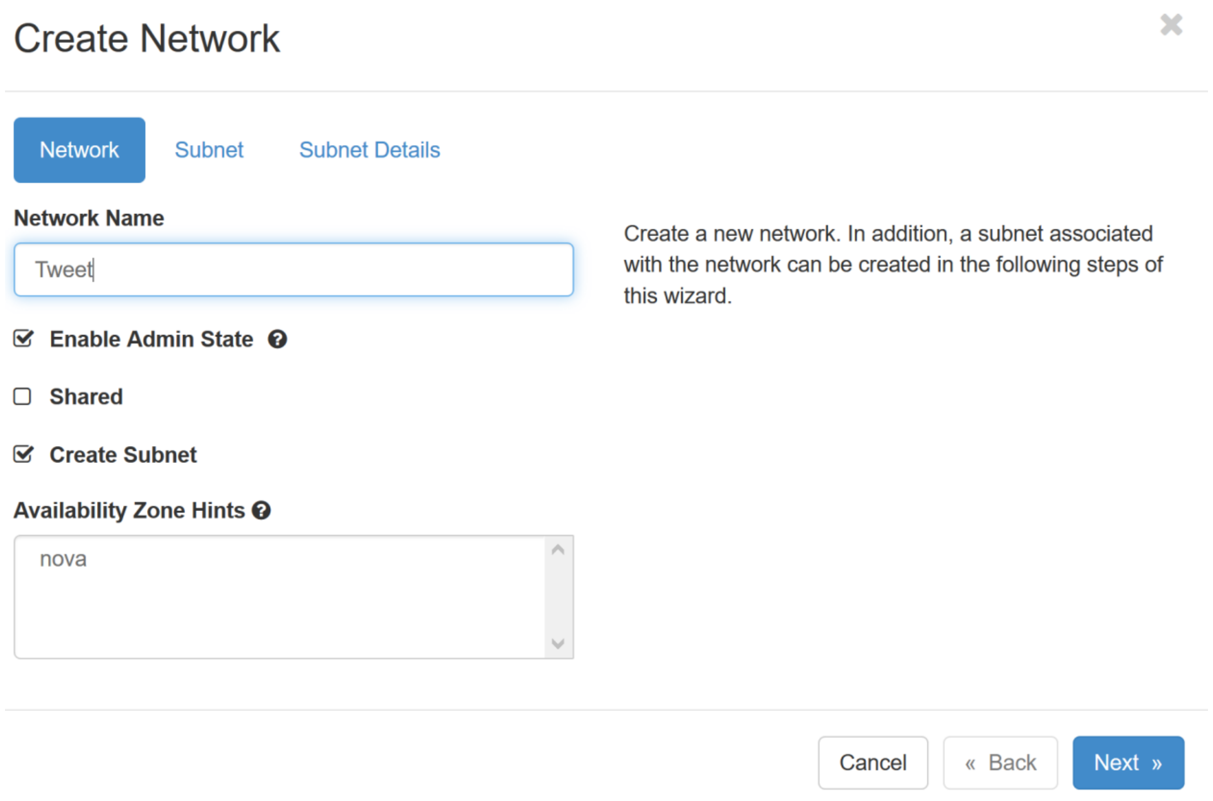
Setting up a network

Setting up a network in OpenStack is a necessary step in creating a virtual infrastructure for your application. It involves creating a virtual network, subnets, and routers, and connecting them to the external network. This section will show you how to create a new network, set up a subnet associated with the network and create router.

1) a) Dashboard: Creating network and its associated subnet

On the Network tab, open the Networks tab, then press +Create Network on the top right side. Provide the required information and press Next (see Fig 6).

Fig 6. Creating network.



Create Network

Network | Subnet | Subnet Details

Network Name: Tweet

Create a new network. In addition, a subnet associated with the network can be created in the following steps of this wizard.

☑ Enable Admin State

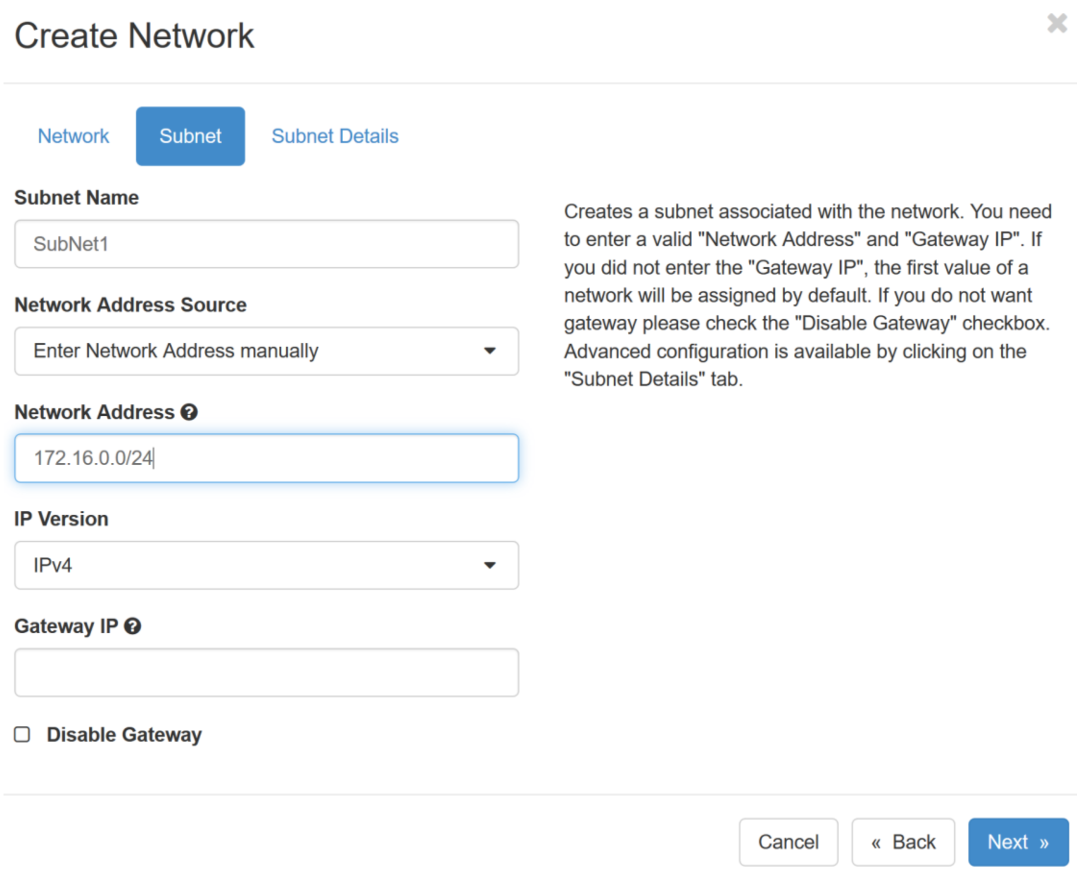
☐ Shared

☑ Create Subnet

Availability Zone Hints: nova

Cancel | « Back | Next »

The UI shown in Fig 7 will be displayed after pressing Next in Fig 6. Provide the necessary information and press Next. It is advisable to use private IP address ranges either from class A, B, or C in Network Address field. You do not have to specify a subnet when you create a network, but if you do not specify a subnet, the network cannot be attached to an instance.



Subnet Name: SubNet1

Network Address Source: Enter Network Address manually

Network Address: 172.16.0.0/24

IP Version: IPv4

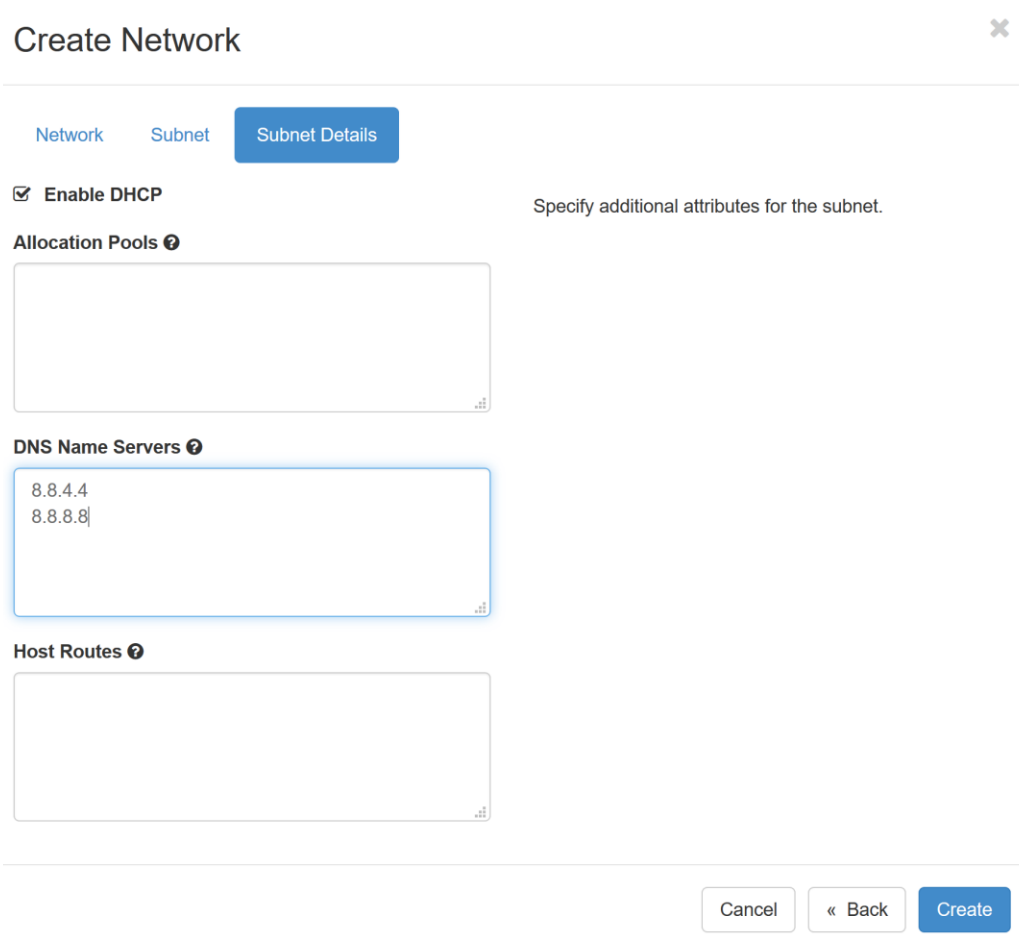
Gateway IP:

☐ Disable Gateway

Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.

Fig 7: Creating subnet.

Press Next in Fig 7 and press Create in Fig 8. You have now created a network with subnet!!


Create Network

☑ Enable DHCP

Specify additional attributes for the subnet.

Allocation Pools

DNS Name Servers

8.8.4.4
8.8.8.8

Host Routes

Fig 8: Creating subnet final step.

b) CLI: Creating network and its associated subnet

To create a network and its associated subnet using OpenStack CLI, you can use the following steps:

1. Login to your OpenStack environment using the OpenStack CLI.
2. Create a new network:

   ```
   lua
   openstack network create <network-name>
   ```

Replace the placeholder <network-name> with your own network name.

3. Create a new subnet associated with the network:

   ```
   php
   openstack subnet create --network <network-name> --subnet-range
   <subnet-range> <subnet-name>
   ```

Replace the placeholders <network-name>, <subnet-range>, and <subnet-name> with your own values.

The <subnet-range> is specified in CIDR notation, such as 192.168.0.0/24.

Note: The above commands create a basic network and subnet in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the IP version (IPv4 or IPv6), the gateway IP address, or the allocation pool for IP addresses in the subnet. For more information, refer to the OpenStack CLI documentation.

c) OpenStack Python SDK: Creating network and its associated subnet

To create a network and its associated subnet using OpenStack Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

   ```
   pip install openstacksdk
   ```

2. Import the necessary modules in your Python script:

   ```
   python
   import openstack
   ```

3. Connect to your OpenStack environment:

   ```
   python
   conn = openstack.connect(cloud='<cloud-name>')
   ```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Create a new network:

   ```
   python
   network = conn.network.create_network(name='<network-name>')
   ```

Replace the placeholder <network-name> with your own network name.

5. Create a new subnet associated with the network:

   ```
   python
   subnet = conn.network.create_subnet(network_id=network.id,
   ip_version='4', cidr='<subnet-range>', name='<subnet-name>')
   ```

Replace the placeholders <subnet-range> and <subnet-name> with your own values.

The <subnet-range> is specified in CIDR notation, such as 192.168.0.0/24.

Note: The above commands create a basic network and subnet in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the gateway IP address, or the allocation pool for IP addresses in the subnet. For more information, refer to the OpenStack Python SDK documentation.

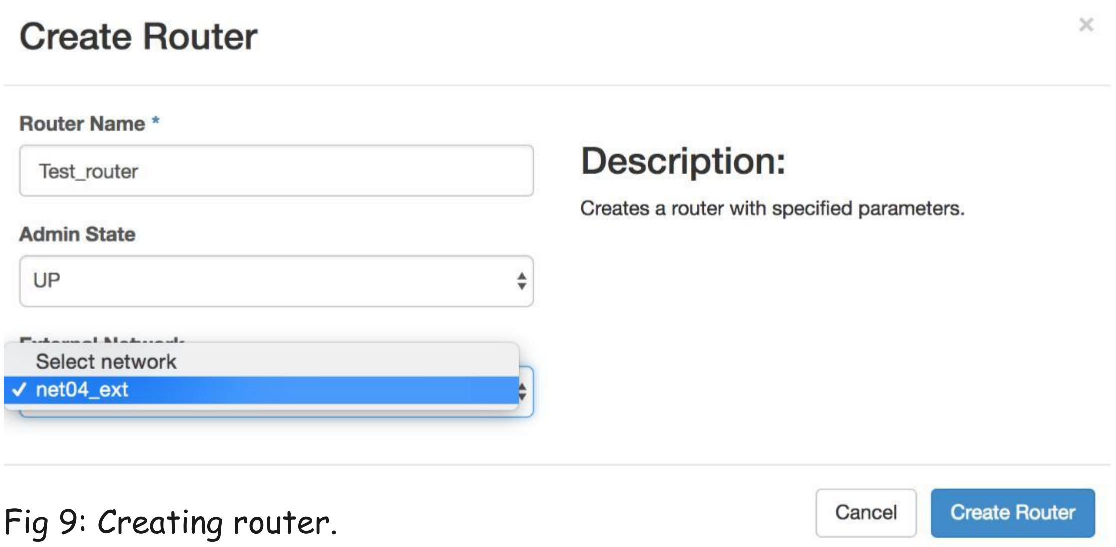
Creating Router

Creating a router in OpenStack is a necessary step in setting up a virtual network infrastructure. A router acts as a gateway between the virtual network and the external network, allowing VMs within the virtual network to communicate with the outside world.

1) a) Dashboard: Creating Router

For VMs to communicate with the external world you need to set up a router.

On the Project tab, open the Network tab and click Routers category and press +Create Router. Specify a name for the router and External Network, and click Create Router(see Fig 9).


Create Router

Router Name *: Test_router

Admin State: UP

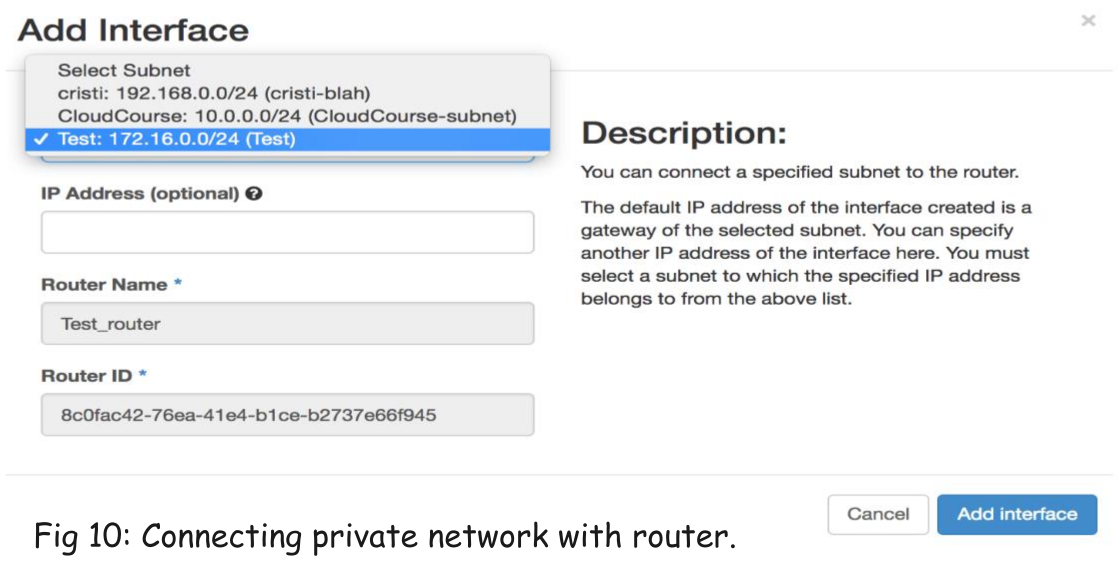
External Network: ✓ net04_ext

Description:
Creates a router with specified parameters.

Fig 9: Creating router.

To connect the private network created above to the newly created router, perform the following steps:

a. On the Routers tab, click the name of the router you created.
1. On the Router Details page, click the Interfaces tab, and then click Add Interface.In the Add Interface dialog box, select the
2. Subnet you created above (see Fig 10).



Add Interface

Select Subnet
cristi: 192.168.0.0/24 (cristi-blah)
CloudCourse: 10.0.0.0/24 (CloudCourse-subnet)
✓ Test: 172.16.0.0/24 (Test)

IP Address (optional)

Router Name *: Test_router

Router ID *: 8c0fac42-76ea-41e4-b1ce-b2737e66f945

Description:

You can connect a specified subnet to the router.

The default IP address of the interface created is a gateway of the selected subnet. You can specify another IP address of the interface here. You must select a subnet to which the specified IP address belongs to from the above list.

Fig 10: Connecting private network with router.

b) CLI: Creating Router

To create a router in OpenStack using the CLI, you can use the following steps:

1. Login to your OpenStack environment using the OpenStack CLI.
2. Create a new router:

```
lua
openstack router create <router-name>
```

Replace the placeholder <router-name> with your own router name.

3. Set the router's external network as its gateway:

```
kotlin
openstack router set --external-gateway <external-network-name>
<router-name>
```

Replace the placeholders <external-network-name> and <router-name> with your own values.

The <external-network-name> is the name of the external network that you want to use as the gateway for the router.

4. Add an internal subnet to the router:

```
php
openstack router add subnet <router-name> <subnet-name>
```

Replace the placeholders <router-name> and <subnet-name> with your own values. The <subnet-name> is the name of the internal subnet that you want to add to the router.

Note: The above commands create a basic router in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the router type or the IP address for the router's gateway interface. For more information, refer to the OpenStack CLI documentation.

c) OpenStack Python SDK: Creating Router

To create a router in OpenStack using the Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

```
pip install openstacksdk
```

2. Import the necessary modules in your Python script:

```
python
import openstack
```

3. Connect to your OpenStack environment:

```
python
conn = openstack.connect(cloud='<cloud-name>')
```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Create a new router:

```
python
router = conn.network.create_router(name='<router-name>')
```

Replace the placeholder <router-name> with your own router name.

5. Set the router's external network as its gateway:

```
python
conn.network.update_router(router,
external_gateway_info={'network_id': '<external-network-id>'})
```

Replace the placeholder <external-network-id> with the ID of the external network that you want to use as the gateway for the router.

6. Add an internal subnet to the router:

```
python
subnet = conn.network.find_subnet(name_or_id='<subnet-name>')
conn.network.add_interface_to_router(router, subnet_id=subnet.id)
```

Replace the placeholder <subnet-name> with the name of the internal subnet that you want to add to the router.

Note: The above commands create a basic router in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the router type or the IP address for the router's gateway interface. For more information, refer to the OpenStack Python SDK documentation.

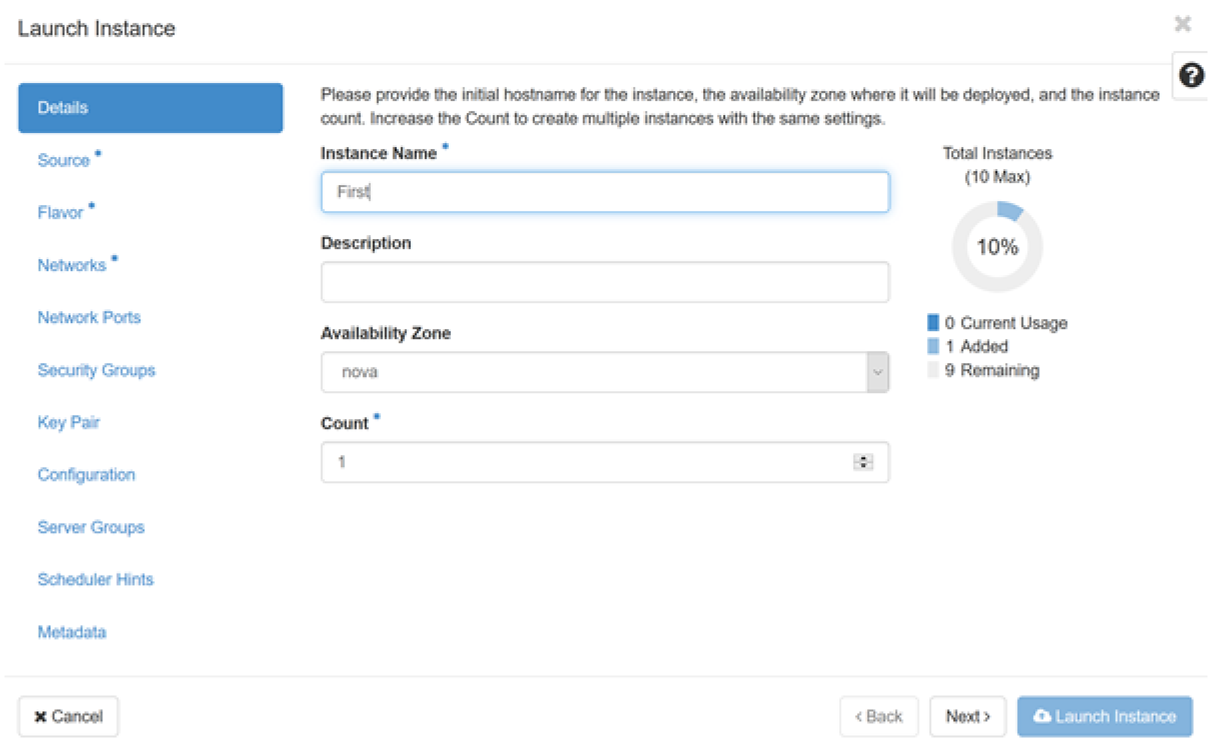
Setting up a VM

Setting up a virtual machine (VM) in OpenStack involves launching an instance of a pre-configured image and allocating resources such as CPU, memory, and disk storage to it. It is now time to create a VM and play with it!!

1) Dashboard: Creating VM

On the Project tab, open the Compute tab and click Instances category. The dashboard shows the instances with its name, its private and floating IP addresses (we will come to this later), size, status, task, power state, and so on.

Click Launch Instance in the top right corner and provide the necessary information (see figs 11-15). To see the progress on how the VM is initializing, click the name of the VM instance in the list of instances, then click 'Log'.


Launch Instance

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 *: First

Description:

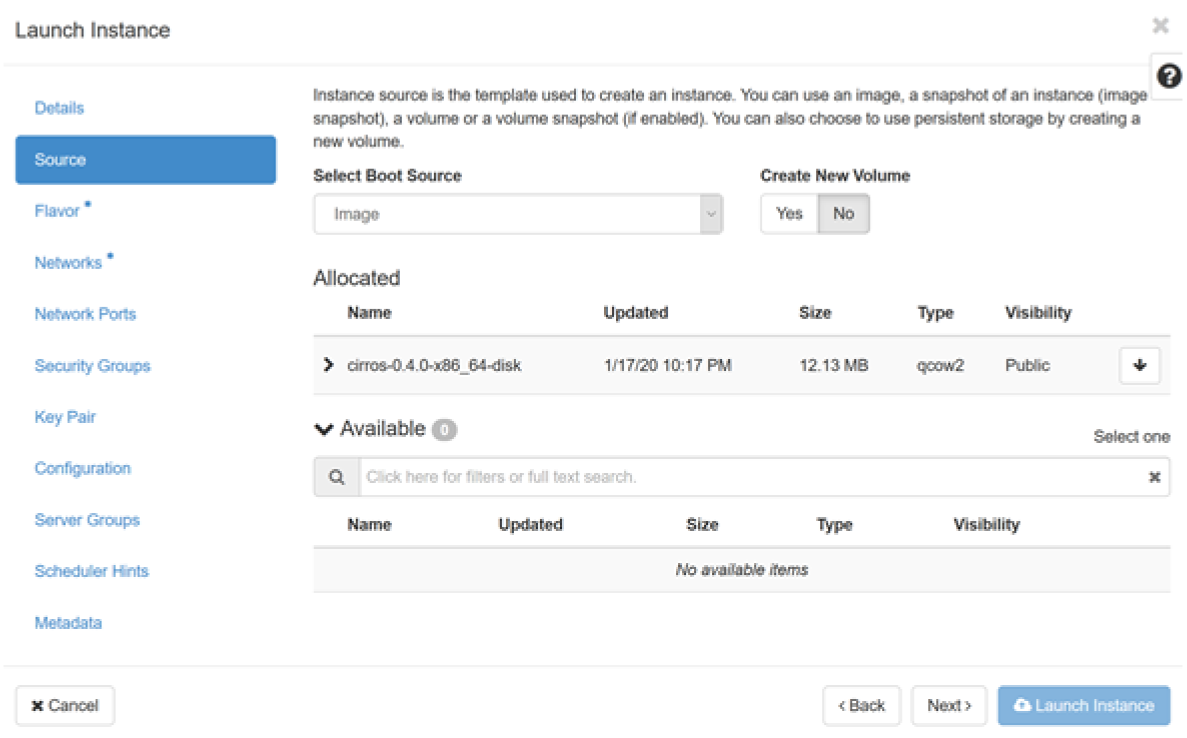
Availability Zone: nova

Count *: 1

Total Instances (10 Max): 10%

0 Current Usage
1 Added
9 Remaining

Fig 11: Creating VM.



Launch Instance

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

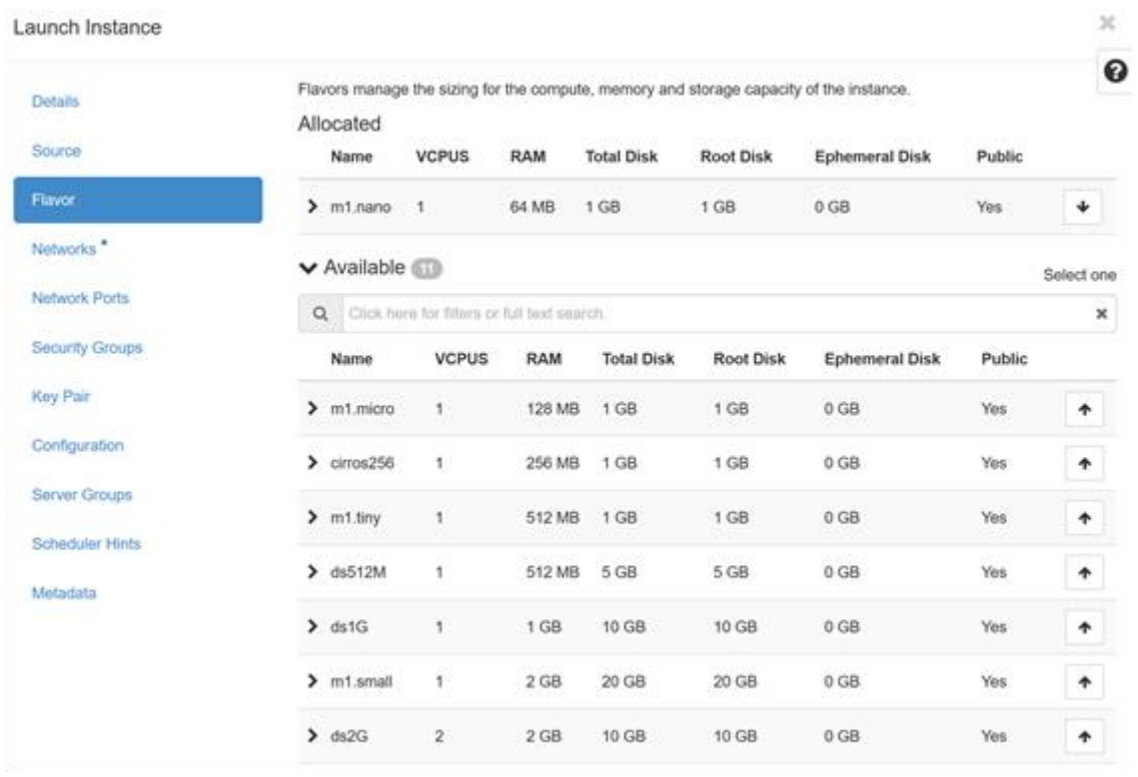
Name	Updated	Size	Type	Visibility
cirros-0.4.0-x86_64-disk	1/17/20 10:17 PM	12.13 MB	qcow2	Public

Available 0 — Select one

Name	Updated	Size	Type	Visibility
No available items				

Fig 12. Selecting Source:

Select **No** for ***Create New Volume*** and Move the cirros disk to ***Allocated*** from ***Available***.



Launch Instance

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

Allocated

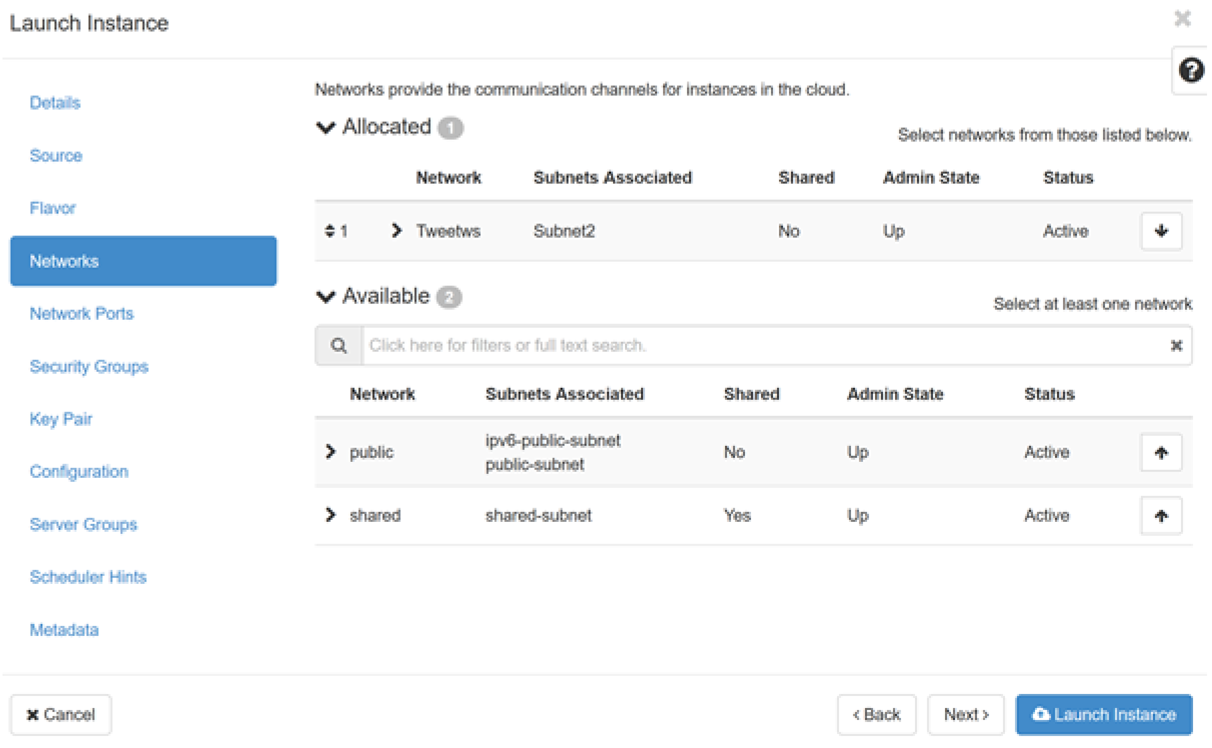
Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.nano	1	64 MB	1 GB	1 GB	0 GB	Yes

Available 11 — Select one

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.micro	1	128 MB	1 GB	1 GB	0 GB	Yes
cirros256	1	256 MB	1 GB	1 GB	0 GB	Yes
m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
ds512M	1	512 MB	5 GB	5 GB	0 GB	Yes
ds1G	1	1 GB	10 GB	10 GB	0 GB	Yes
m1.small	1	2 GB	20 GB	20 GB	0 GB	Yes
ds2G	2	2 GB	10 GB	10 GB	0 GB	Yes

Fig 13. Selecting Flavor.

Move **m1.nano** from **Available** to **Allocated**.



Networks provide the communication channels for instances in the cloud.

Allocated 1 — Select networks from those listed below.

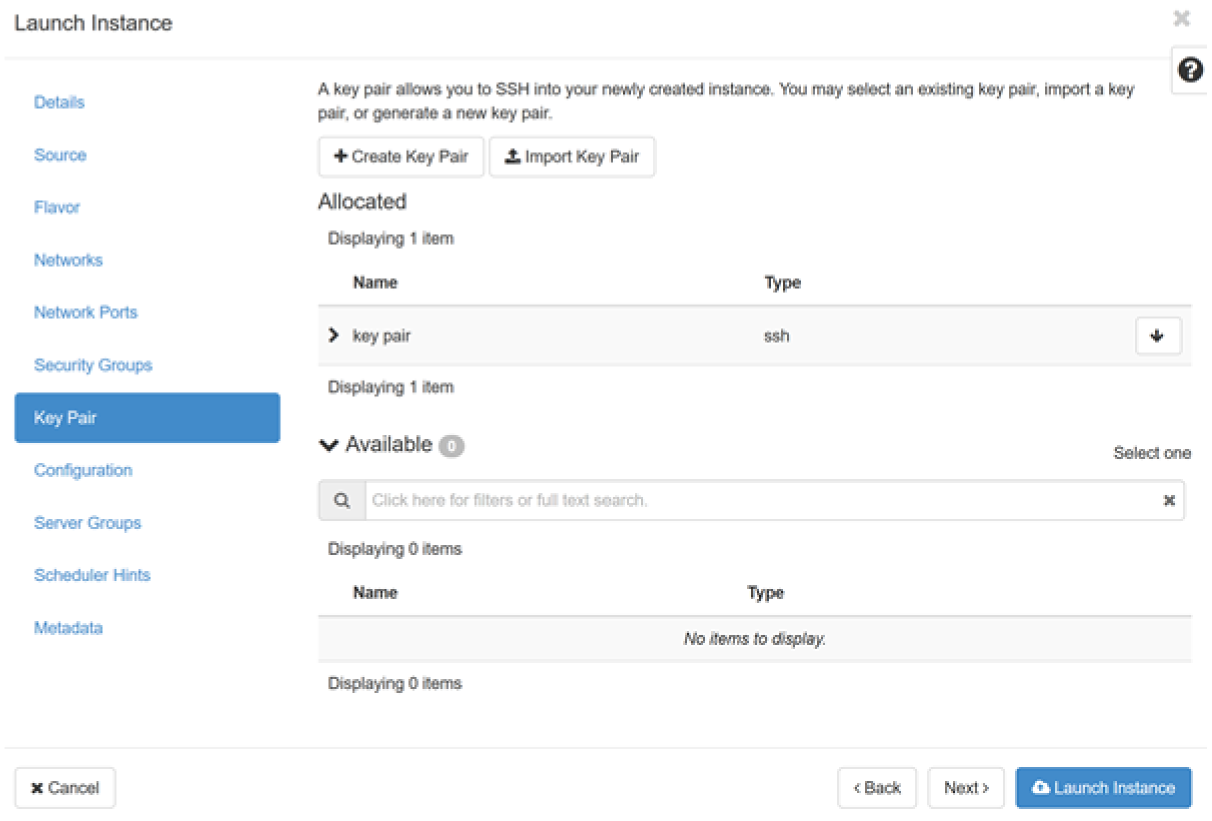
	Network	Subnets Associated	Shared	Admin State	Status
1	Tweetws	Subnet2	No	Up	Active

Available 2 — Select at least one network

Click here for filters or full text search.

Network	Subnets Associated	Shared	Admin State	Status
public	ipv6-public-subnet public-subnet	No	Up	Active
shared	shared-subnet	Yes	Up	Active

Fig 14: Network setup (choose the network you created above)



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
key pair	ssh

Displaying 1 item

Available 0 — Select one

Click here for filters or full text search.

Displaying 0 items

Name	Type
No items to display.	

Displaying 0 items

Fig 15: Ensure the SSH key pair you created above is allocated.

Select **Launch Instance**.

b) CLI: Creating VM

To create a virtual machine (VM) in OpenStack using the CLI, you can use the following steps:

1. Login to your OpenStack environment using the OpenStack CLI.
2. Choose an image for your VM:

```
openstack image list
```

This command lists the available images that you can use to create your VM. Note the ID of the image that you want to use.

3. Choose a flavor for your VM:

```
openstack flavor list
```

This command lists the available flavors that you can use to create your VM. Note the ID of the flavor that you want to use.

4. Choose a network for your VM:

```
openstack network list
```

This command lists the available networks that you can use to create your VM. Note the ID of the network that you want to use.

5. Create a new VM:

php

```
openstack server create --image <image-id> --flavor <flavor-id> --network <network-id> <vm-name>
```

Replace the placeholders <image-id>, <flavor-id>, <network-id>, and <vm-name> with your own values.

6. Verify the status of your VM:

```
openstack server list
```

This command lists all the VMs in your OpenStack environment. Check the status of your new VM to make sure it has been created successfully.

Note: The above commands create a basic VM in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the availability zone, security groups, or the key pair for accessing the VM. For more information, refer to the OpenStack CLI documentation.

c) OpenStack Python SDK: Creating VM

To create a virtual machine (VM) in OpenStack using the Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

```
pip install openstacksdk
```

2. Import the necessary modules in your Python script:

```
python
import openstack
```

3. Connect to your OpenStack environment:

```
python
conn = openstack.connect(cloud='<cloud-name>')
```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Choose an image for your VM:

```
python
image = conn.compute.find_image(name_or_id='<image-name>')
```

Replace the placeholder <image-name> with the name of the image that you want to use to create your VM.

5. Choose a flavor for your VM:

```
python
flavor = conn.compute.find_flavor(name_or_id='<flavor-name>')
```

Replace the placeholder <flavor-name> with the name of the flavor that you want to use to create your VM.

6. Choose a network for your VM:

```
python
network = conn.network.find_network(name_or_id='<network-name>')
```

Replace the placeholder <network-name> with the name of the network that you want to use to create your VM.

7. Create a new VM:

```
python
server = conn.compute.create_server(
    name='<vm-name>',
    image_id=image.id,
    flavor_id=flavor.id,
    networks=[{"uuid": network.id}]
)
```

Replace the placeholder <vm-name> with your own VM name.

8. Verify the status of your VM:

```
python
server = conn.compute.get_server(server.id)
```

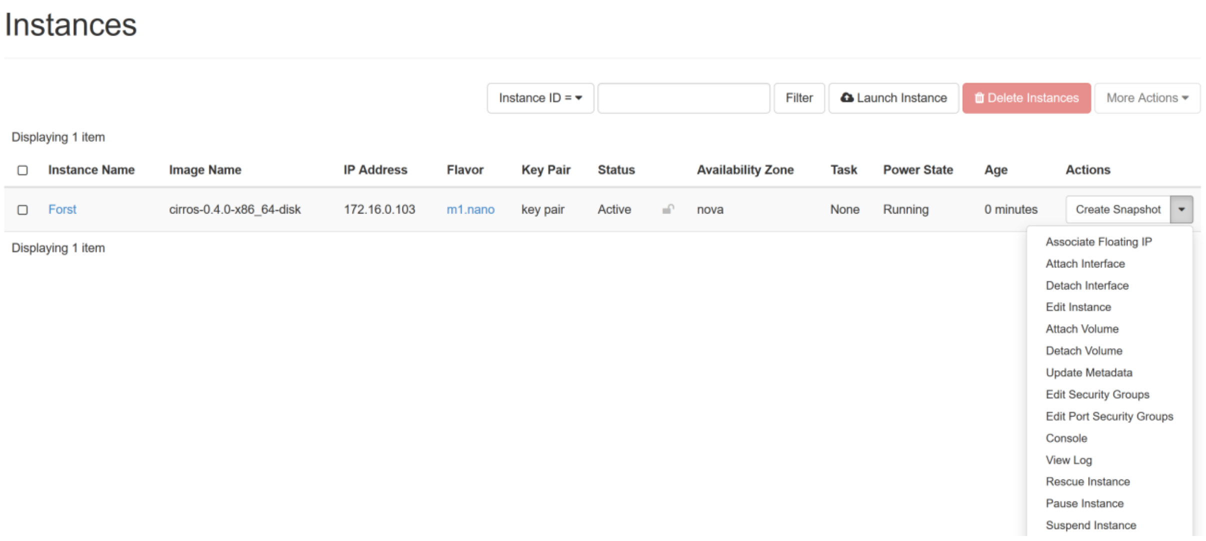
This command retrieves the details of your new VM. Check the status of your new VM to make sure it has been created successfully.

Note: The above commands create a basic VM in OpenStack. Depending on your use case, you may need to specify additional parameters, such as the availability zone, security groups, or the key pair for accessing the VM. For more information, refer to the OpenStack Python SDK documentation.

2) a) Dashboard: Associated floating IP to a VM

Associating floating IP to a VM helps to associate public IP address to your VM so that it can be accessed externally.

On the Project tab, open the Compute tab and click Instances category. On the far right parallel to the instance click the drop down menu and select Associate Floating IP. Choose from the list (see Fig 16).


Instances

Instance ID = ▾ | Filter | Launch Instance | Delete Instances | More Actions ▾

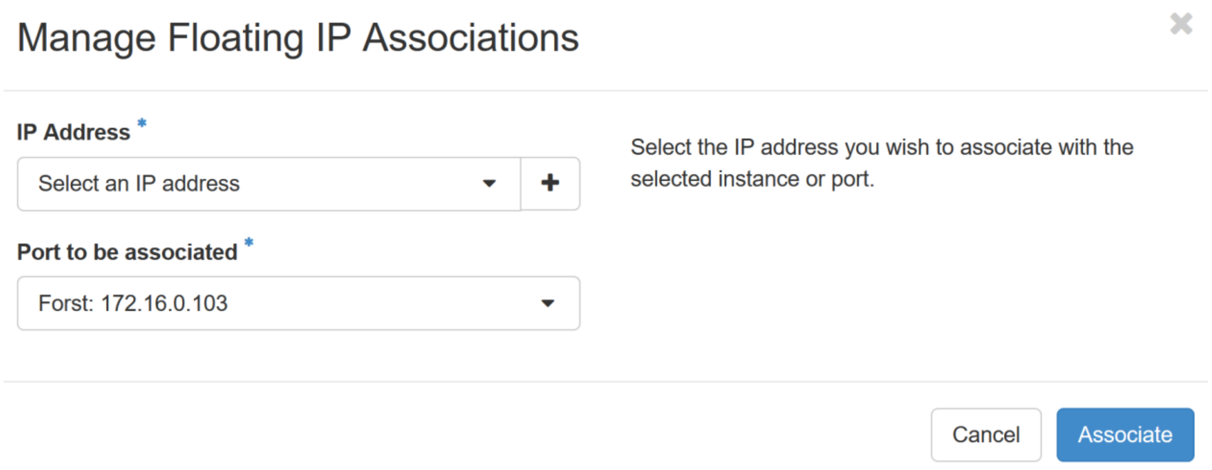
Displaying 1 item

	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status		Availability Zone	Task	Power State	Age	Actions
☐	Forst	cirros-0.4.0-x86_64-disk	172.16.0.103	m1.nano	key pair	Active		nova	None	Running	0 minutes	Create Snapshot ▾

Displaying 1 item

Figure 16. Associate Floating IP

Click the '+' sign next to 'Select an IP address' textbox in Figure 17 below.


Manage Floating IP Associations

IP Address *

Select an IP address ▾ +

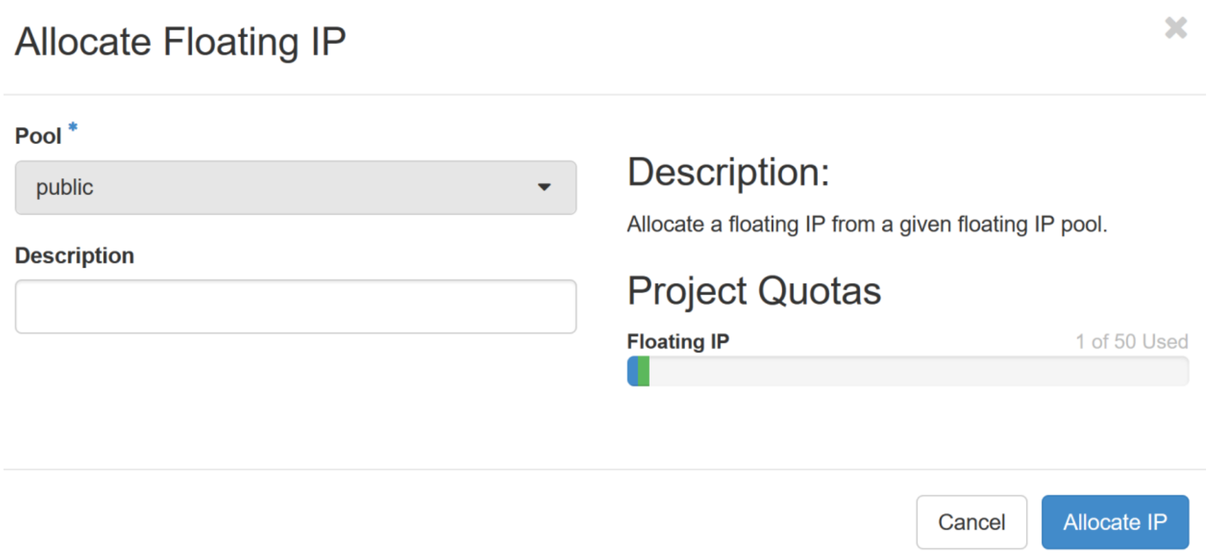
Select the IP address you wish to associate with the selected instance or port.

Port to be associated *

Forst: 172.16.0.103 ▾

Figure 17. Manage Floating IP Associations

Click the 'Allocate IP' button in Figure 18 below.


Allocate Floating IP

Pool *
public

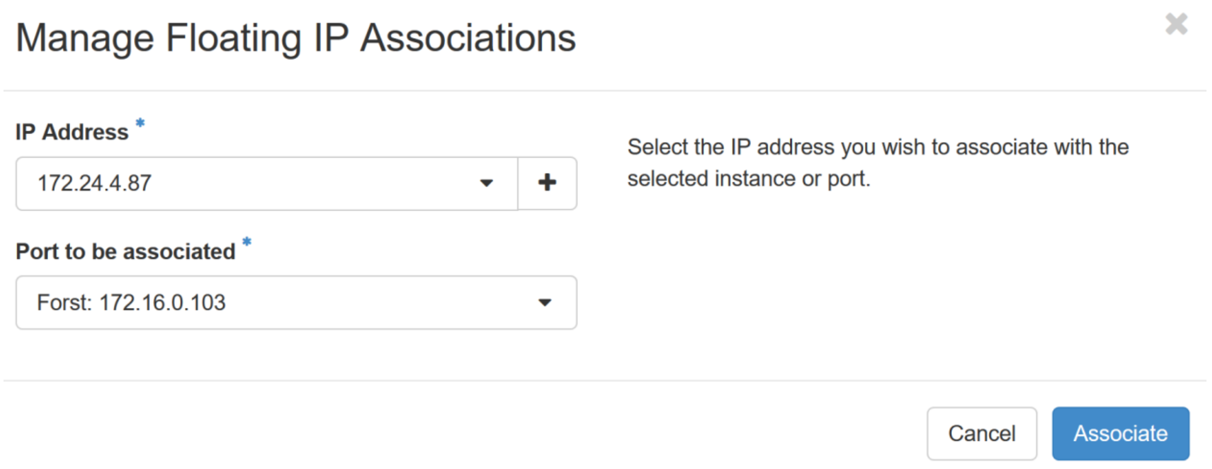
Description

Description:
Allocate a floating IP from a given floating IP pool.

Project Quotas
Floating IP 1 of 50 Used

Figure 18. Alocate Floating IP

Click the 'Associate' button in Figure 19.


Manage Floating IP Associations

IP Address *
172.24.4.87

Port to be associated *
Forst: 172.16.0.103

Select the IP address you wish to associate with the selected instance or port.

Figure 19: Associate IP

b) CLI: Associated floating IP to a VM

To associate a floating IP to a virtual machine (VM) in OpenStack using the CLI, you can use the following steps:

1. Create a new floating IP:

```
lua
openstack floating ip create <network-name>
```

Replace the placeholder <network-name> with the name of the network that you want to create the floating IP in.

2. Get the ID of the VM that you want to associate the floating IP with:

```
openstack server list
```

This command will list all the VMs in your OpenStack environment along with their ID.

3. Associate the floating IP with the VM:

```
php
openstack server add floating ip <vm-id> <floating-ip-address>
```

Replace the placeholder <vm-id> with the ID of the VM that you want to associate the floating IP with and the placeholder <floating-ip-address> with the IP address of the floating IP that you created in step 1.

4. Verify the association:

```
python
openstack server show <vm-id>
```

This command will show the details of your VM, including the associated floating IP address. Check that the floating IP is listed under "addresses".

Note: To disassociate a floating IP from a VM, use the following command:

```
php
openstack server remove floating ip <vm-id> <floating-ip-address>
```

c) OpenStack Python SDK: Associated floating IP to a VM

To associate a floating IP to a virtual machine (VM) in OpenStack using the Python SDK with Python 3.x, you can use the following steps:

1. Install the OpenStack Python SDK using pip:

```
pip install openstacksdk
```

2. Import the necessary modules in your Python script:

```
python
import openstack
```

3. Connect to your OpenStack environment:

```
python
conn = openstack.connect(cloud='<cloud-name>')
```

Replace the placeholder <cloud-name> with your OpenStack cloud name.

4. Get the details of the VM that you want to associate the floating IP with:

```
python
server = conn.compute.find_server(name_or_id='<vm-name>')
```

Replace the placeholder <vm-name> with the name of the VM that you want to associate the floating IP with.

5. Create a new floating IP:

```
python
floating_ip = conn.network.create_ip(floating_ip_network='<network-
name>')
```

Replace the placeholder <network-name> with the name of the network that you want to create the floating IP in.

6. Associate the floating IP with the VM:

```
css
conn.compute.add_floating_ip_to_server(server=server,
address=floating_ip['floating_ip_address'])
```

7. Verify the association:

```
python
updated_server = conn.compute.get_server(server.id)
```

This command retrieves the updated details of your VM, including the associated floating IP address. Check that the floating IP is listed in the "addresses" field.

Note: To disassociate a floating IP from a VM, use the following code:

```
css
conn.compute.remove_floating_ip_from_server(server=server,
address=floating_ip['floating_ip_address'])
```

END