OpenStack Tutorial

I. OpenStack Dashboard

The goal of this part is to familiarize yourself with OpenStack dashboard. 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

In order to perform these operations you need to login Ericsson's cloud via https://129.192.68.4/. This will give you access to your project(s). Once logged in, you will get a page similar to Fig 1.

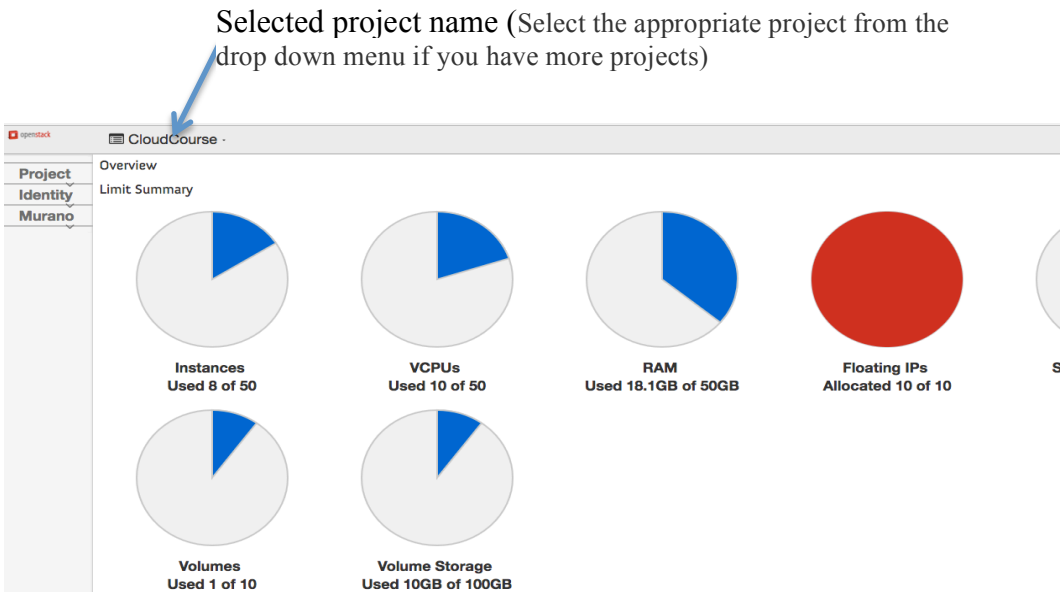

Fig 1. Project Dashboard

Setting up SSH keys

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) 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).

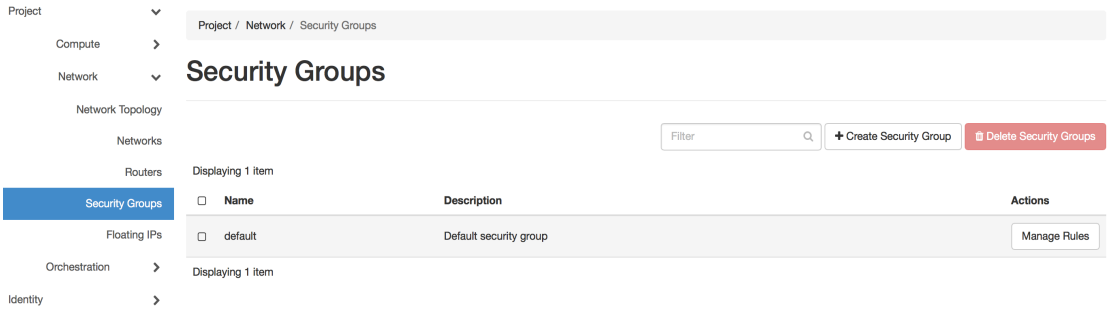

Fig 2. Security Group

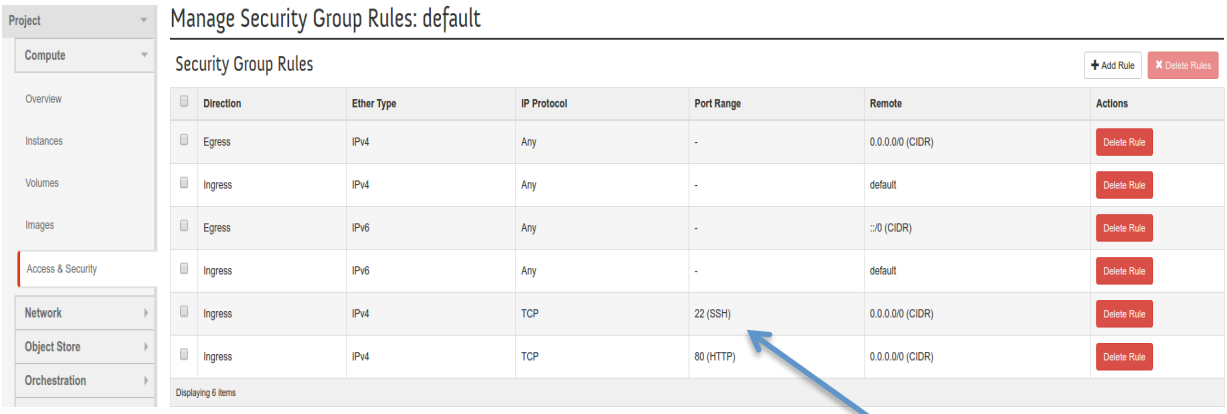

Fig 3. List of enabled services

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).

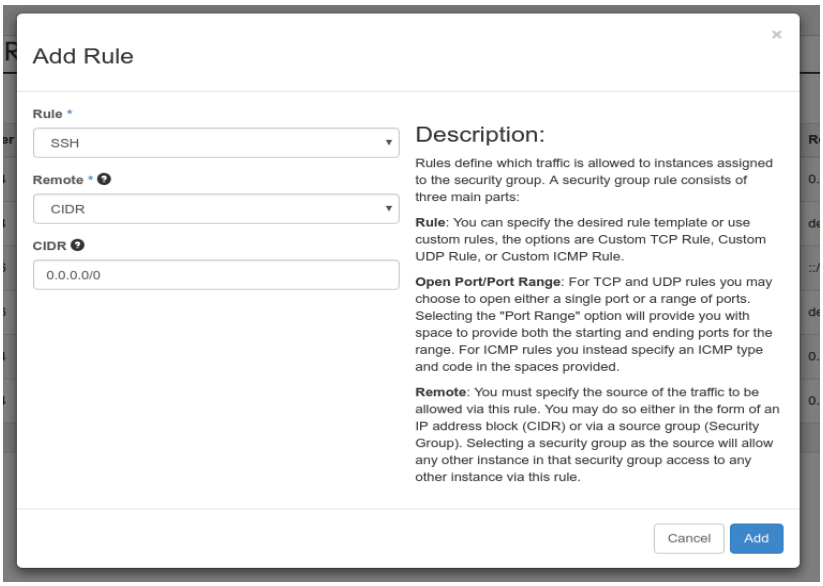

Fig 4. Adding ssh rule.

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

2)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).

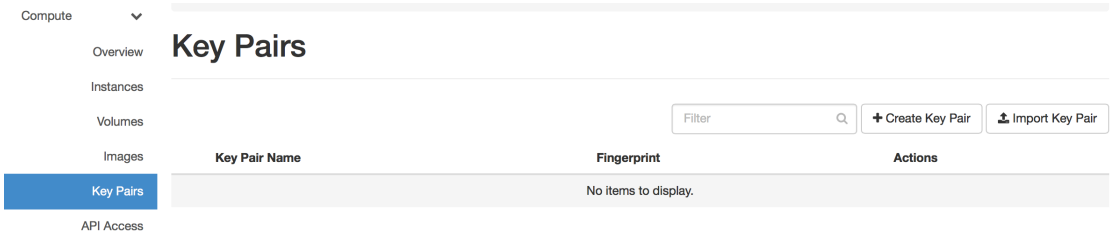

Fig 5. Creating Key Pair.

```
$ ssh -i MyKey.pem ubuntu@10.0.0.2 #make sure that the permission to
Mykey.pem is set to owner.
```

For **Windows** user please check:
https://github.com/davidheijkamp/docs/wiki/Howto:-Creating-and-using-OpenStack---SSH-keypairs-on-Windows

Setting up Network

This section will show you how to create a new network, set up a subnet associated with the network and create router.

1)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).

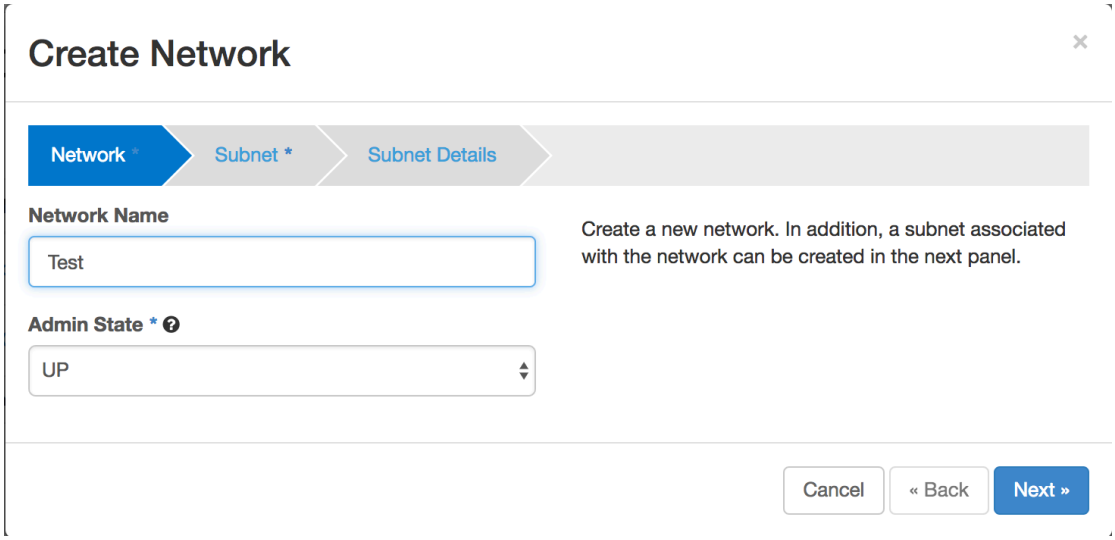

Fig 6. Creating network.

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.

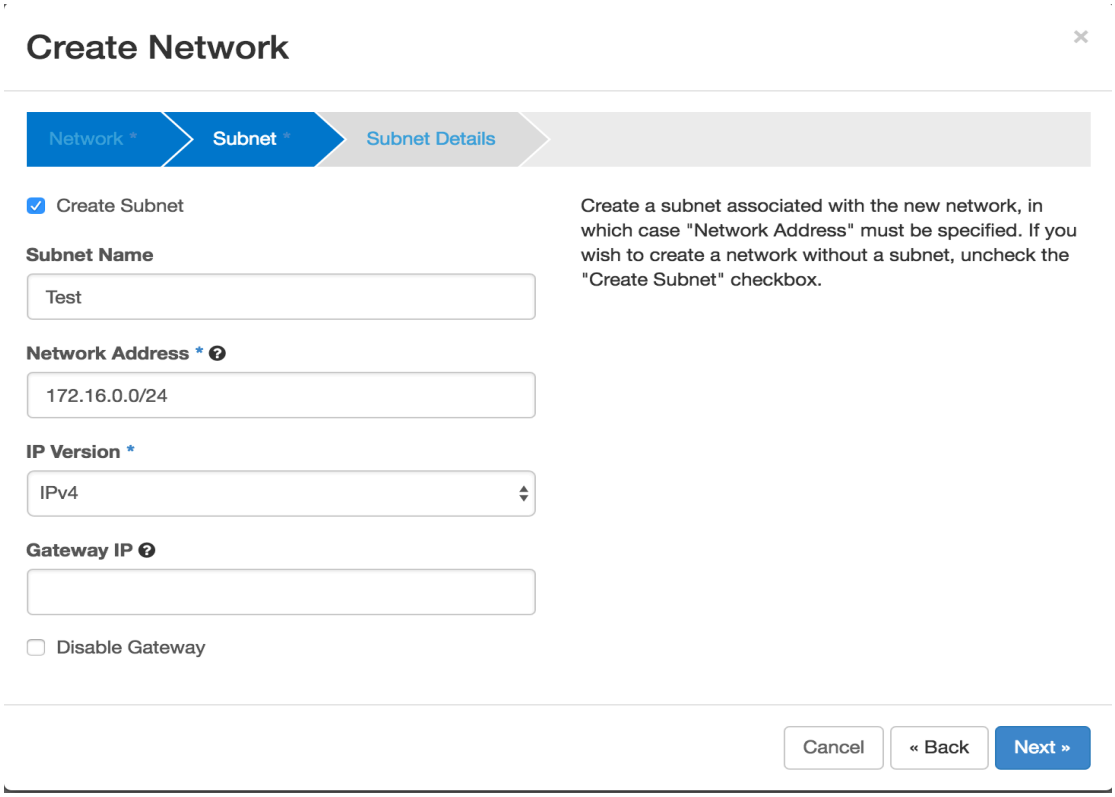

Fig 7: Creating subnet.

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

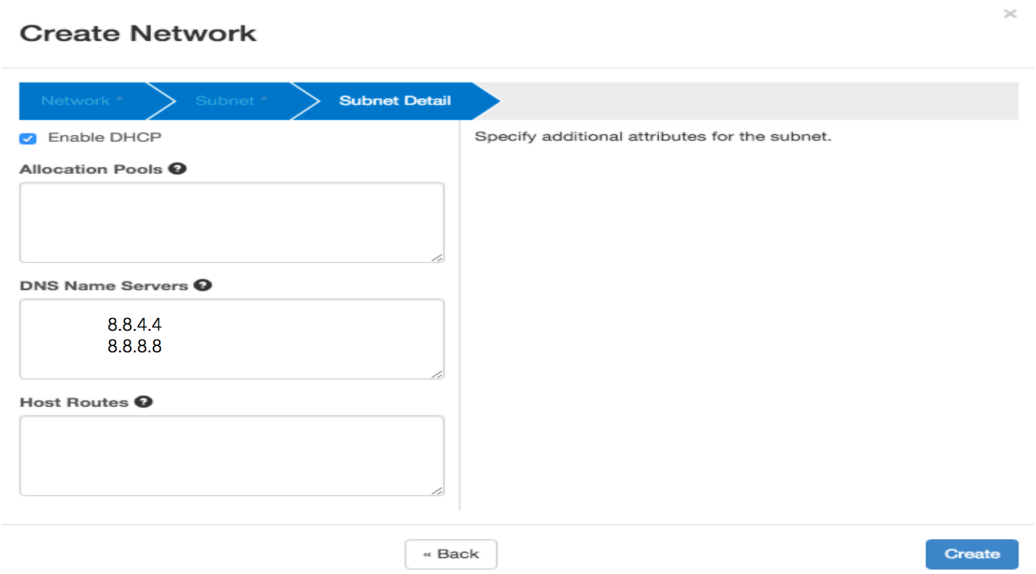

Fig 8: creating subnet final step.

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).

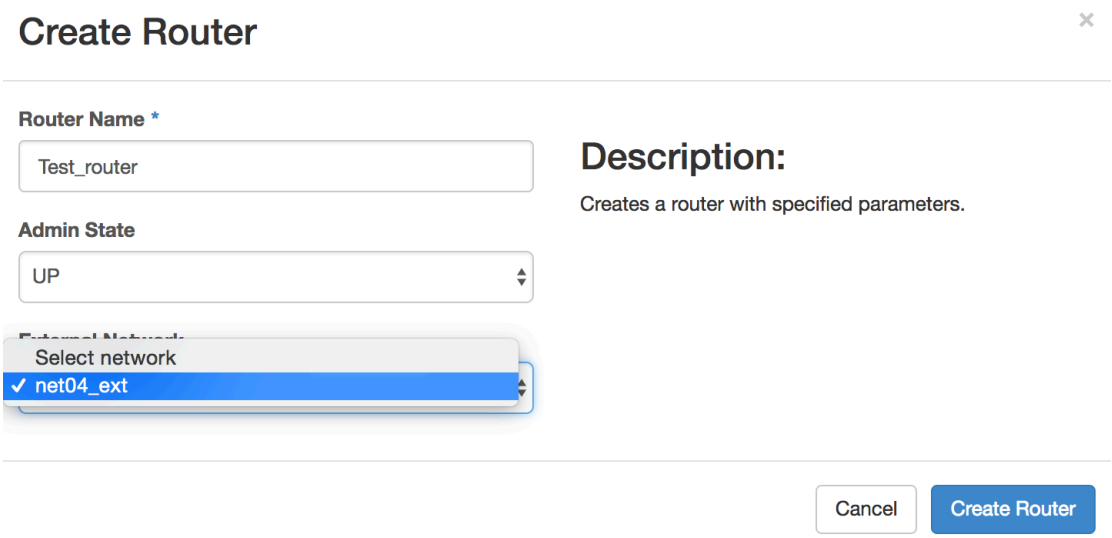

Fig 9: Creating router.

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.

b. On the Router Details page, click the Interfaces tab, and then click Add Interface.

c. In the Add Interface dialog box, select the Subnet you created above (see Fig 10).

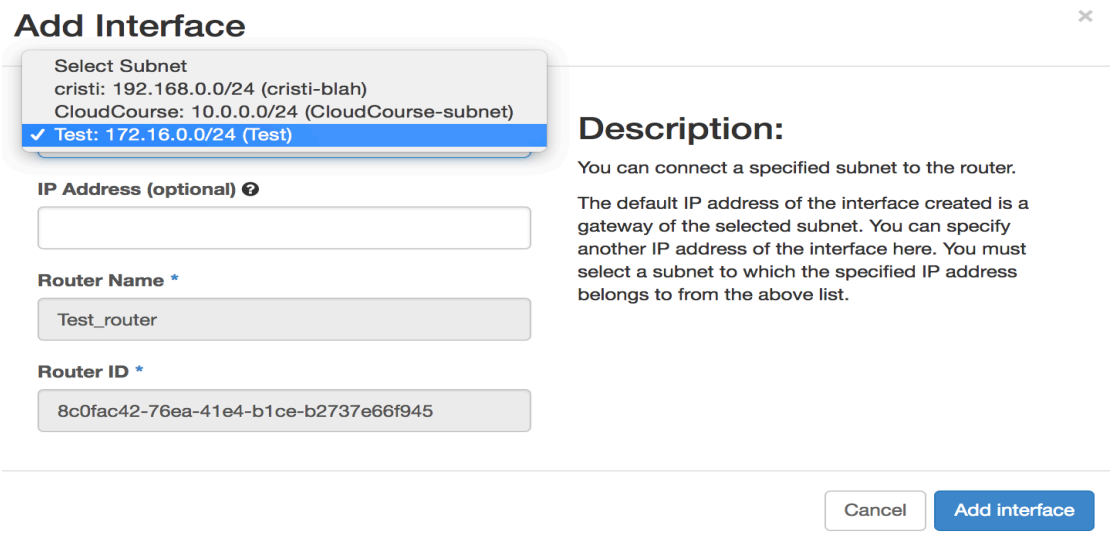

Fig 10: Connecting private network with router.

Setting up a VM

It is now time to create a VM and play with it!!

1) 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'.

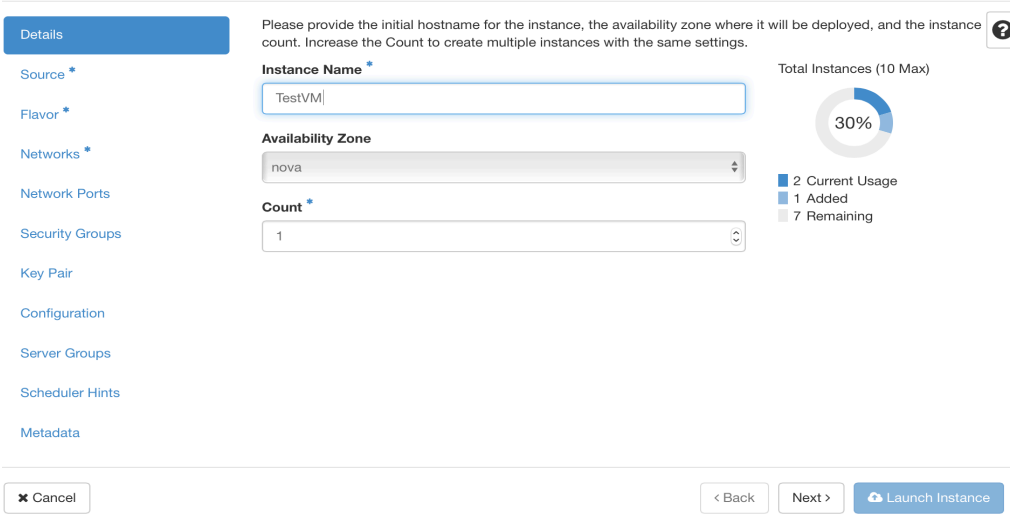

Fig 11: Creating VM.

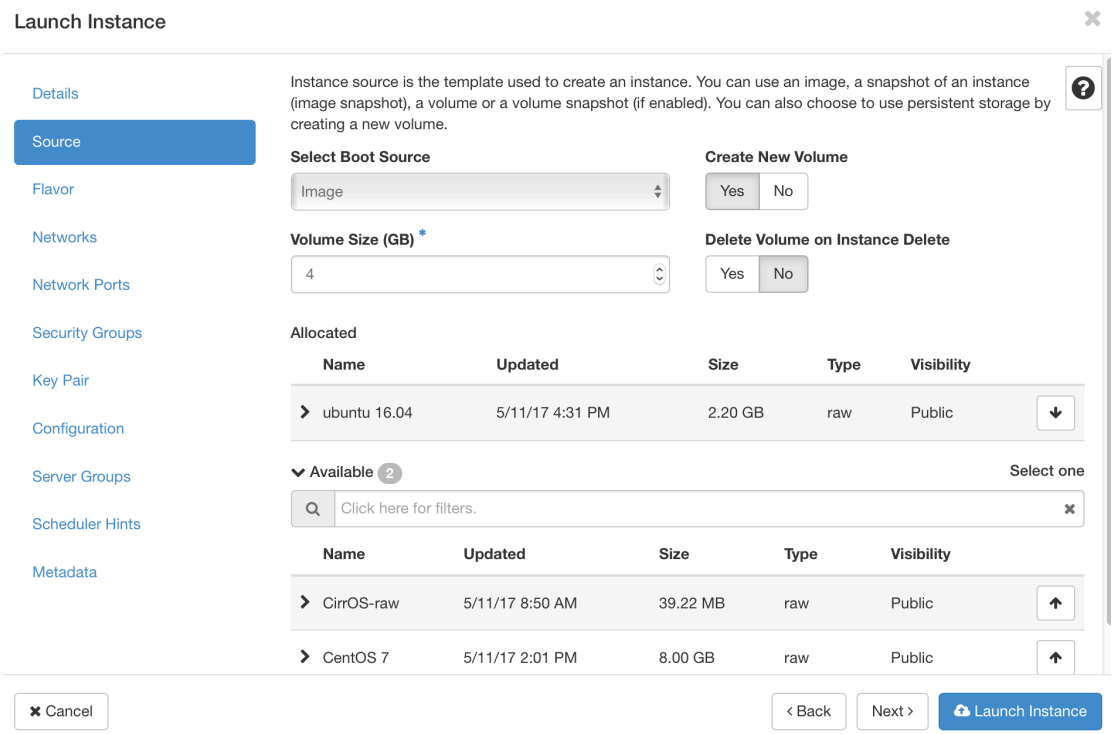

Fig 12. Selecting Source.

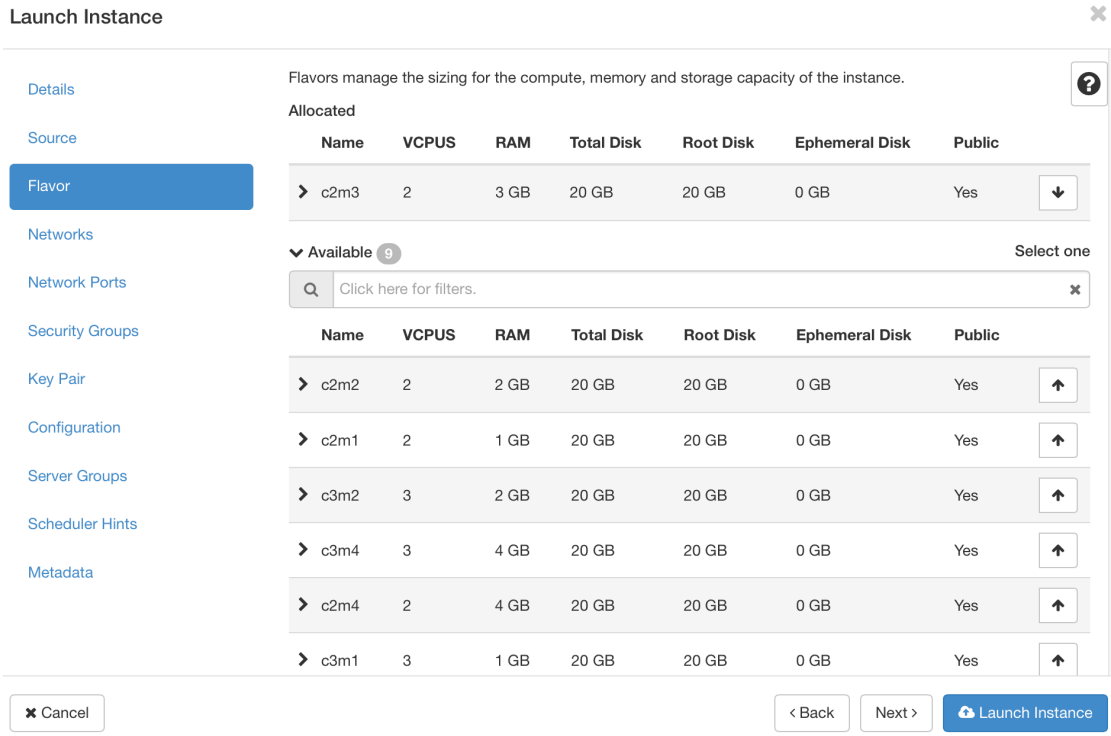

Fig 13. Selecting Flavor.

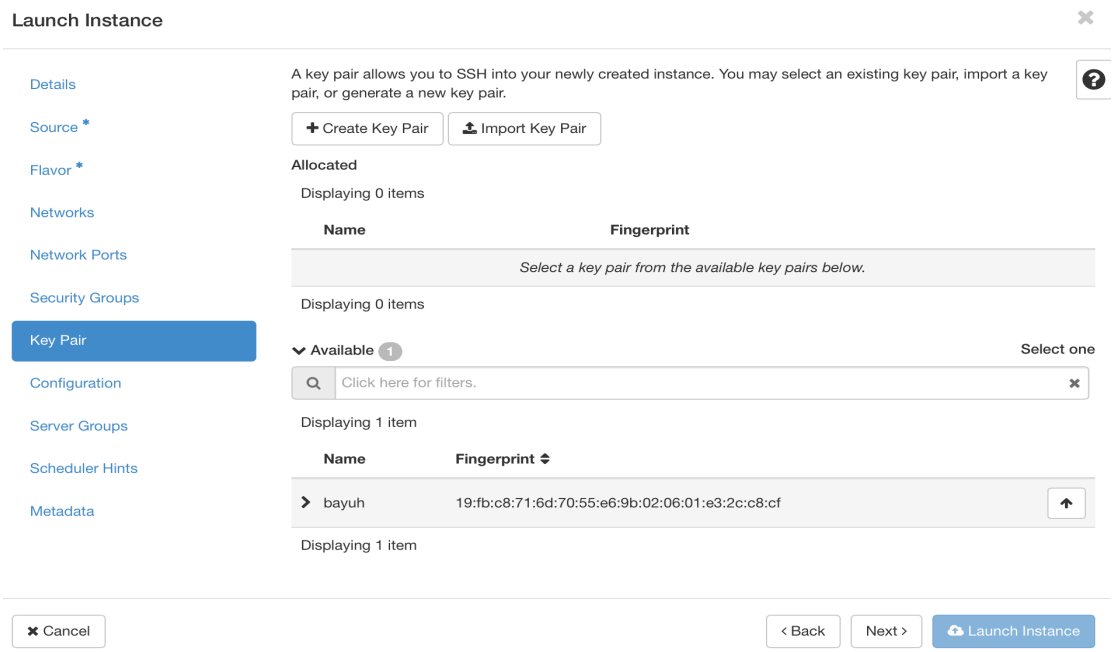

Fig 14: Select the key pair if you have more than one (Choose the key pair you created above).

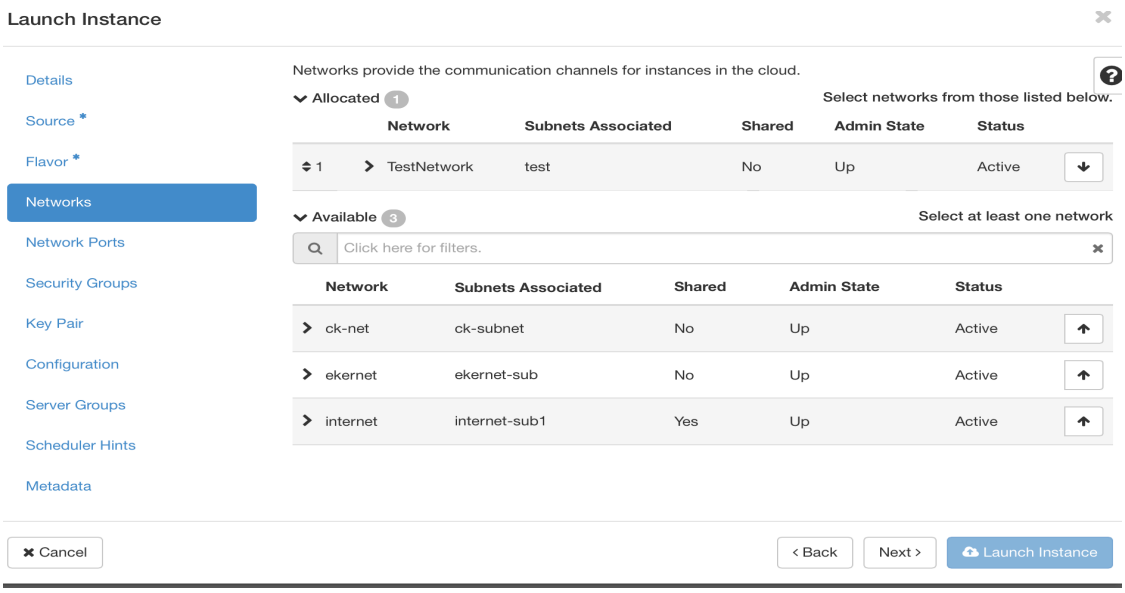

Fig 15: Network setup (choose the network you created above)

2) 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 and click Associate (see fig 16). Sometimes, you may not see

any Floating IPs, in which case you need to click the "+" button to the right of the floating IP. Your VM is now accessible from anywhere!! Please remember the IP, you will need it soon!!

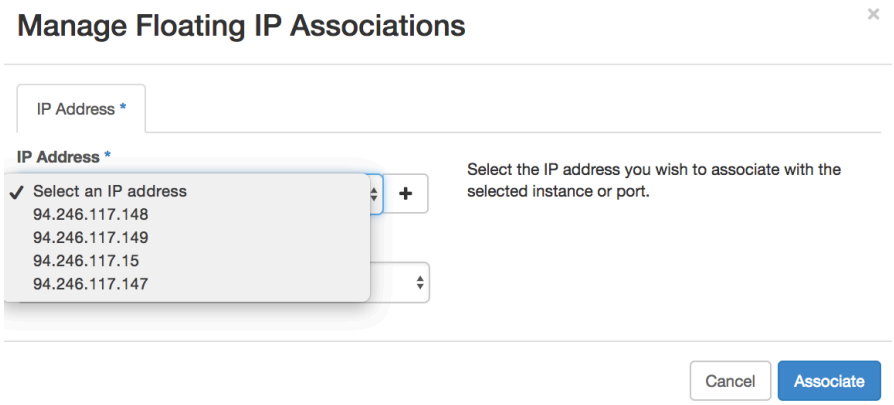

Fig 16: Associating floating IP.

It is now time to play with the VM. Let us login to the VM. Remember the key pair file that you downloaded sometime ago. It is time to use it now to connect to the vm.

Use the **ssh** command to make a secure connection to the instance as shown below.

```
$ ssh -i MyKey.pem ubuntu@floating_ip
```

II. Python Script

The operations that you did using the dashboard can also be done using python. To do that we use the VM that you created above as a devVM. Please login to the selected devVM. Once you are there, you need to install the following OpenStack libraries (This tutorial uses python-novaclient 7.1.0).

```
$ sudo apt-get update
$ sudo apt install python-dev python-pip
$ export LC_ALL=C
$ sudo pip install python-novaclient==7.1.0
$ sudo pip install python-swiftclient
```

VM-related Operations

We have provided a python script for some of the operations. Please clone https://github.com/ewnetu/WASP.git and check **vm-operations.py** and **vm-init.sh** scripts (the scripts are the basis for the micro services section below). The script is based on python-novaclient 7.1.0 and some of the operations may not work if you use a different version (for more: https://docs.openstack.org/developer/python-novaclient/ref/v2/).

```
$ git clone https://github.com/ewnetu/WASP.git
```

In order to run the script you need to specify the right information in **config.properties** file. Below is a sample example (Note that you only need to change the values for the username, password,projectName, keyName and netId properties). **Don't forget to put your key file in the same directory as vm-operations.py** since it will be used during VM instantiation (check createVM() method)

```
[user]
username:your  username
password: your pass
[openstack]
projectName:your project name
user_domain_name:xerces
project_domain_name:xerces
project_domain_id: to get project id login to the dashboard  select identity tab then projects
authUrl:https://xerces.ericsson.net:5000/v3
keyName:your keyname
netId:your network name
```

Add the following line in ***/etc/hosts*** file.

```
129.192.68.4  xerces.ericsson.net
```

The examples below show how to create a vm called 'WASP" as well as list all VMs for the tenant specified in the property file. Go through the source and try out the different operations implemented in the script and see their effect.

```
$ python vm-operations.py -o create -n WASP
$ python vm-operations.py -o listVM
```

Micro services

We have also added a small micro service inside WASP vm. Get the IP address of the WASP VM either from the dashboard or using python as shown below.

```
$ python vm-operations.py  -o VMIP -n WASP
```

Then you can access the service as follows (Make sure to enable port 5000) (it is based on Flask rest API, for more http://flask.pocoo.org/docs/0.12/quickstart/):

```
$ curl -i http://10.0.0.37:5000/v1/hello
```

Please note that the VM might take as much as 5 minutes to download the necessary packages. If the above command does not work the first time, retry a few minutes later or check the VM log for errors.

We will now introduce a simple master-worker service with three VMs (Frontend, RabbitMQ and Backend) each having different roles. The Frontend posts messages to RabbitMQ while the Backend pulls messages from RabbitMQ and prints them. For more information on RabbitMQ visit https://www.rabbitmq.com/tutorials/tutorial-one-python.html.

```
$ git clone https://github.com/muyiibidun/WASP.git
```

In order to deploy the three VMs, please create *credentials.txt* file and put the following info (and don't forget to replace the values with your credentials and project name):

```
[auth]
username:your  username
password: your pass
tenant_name: your project name
user_domain_name:xerces
project_domain_name:xerces
project_domain_id: to get project id login to the dashboard  select identity tab then projects
auth_url: https://xerces.ericsson.net:5000/v3
net_id: net_id
```

```
pkey_id:pkey_id
```

Now you can deploy the three VMs as:

```
$ ./deploy-waspmq.sh
```

Get the IP addresses of the Frontend, RabbitMQ, and the Backend. ssh to the Frontend and Backend VMs and perform the following operations. You must copy the SSH key (PEM file) generated at the start of the tutorial on the devVM. You need to open 3 terminals, one with the devVM, one with the frontend VM (via the devVM), and one with the backend VM (via the devVM).

a) Frontend VM
Once you are inside the Frontend VM, perform the following operations.

```
$ cd /var/www/WASP/waspmq
```

Then edit credentials.txt file and insert the IP address of rabbitmq server. Once you are done editing, run the service.

```
$ python frontend.py -c credentials.txt
```

b) Backend VM

Once you are inside the Backend VM, perform the following operations.

```
$ cd /usr/local/WASP/waspmq
```

Then edit credentials.txt file and insert the IP address of rabbitmq server. Once you are done editing, run the service.

```
$ python backend.py -c credentials.txt
```

Now everything is ready, go back to your devVM and test the services.

```
$ curl -i http://[frontend-ip]:5000/v1/waspmq/hej
```

```
$ curl -i http://[frontend-ip]:5000/v1/waspmq/welcome+to+umea
```

Notice how the backend VM is picking up the 'work' submitted through the frontend VM. Thanks to their decoupling through a message queue (here RabbitMQ), you can scale up the application by adding several frontend VMs and backend VMs.

Exercise

How can the above manual operations (setting IP address of rabbitMQ to frontend and backend config files) be automated? (Hint: Start the rabbitMQ service first, get its IP and set the IP to credentials.txt. This should be done after the cloning but before starting the services.)