MOC Openstack Tutorial

**Overview**

This document will guide the end-user to make use of the MOC Openstack platform and to

launch a VM and utilize the compute resources and install basic packages in order to run

sample nodejs hello-world example in a successful manner. Once the user has a

VM running they can set the development environment as per their needs by installing all the

required packages for the purpose .

**What is Openstack ?**

In general , OpenStack is a cloud operating system that controls large pools of compute,

storage, and networking resources throughout a datacenter, all managed and provisioned

through APIs with common authentication mechanisms.

A dashboard is also available, giving administrators control while empowering their users to

provision resources through a web interface.

Beyond standard infrastructure-as-a-service functionality, additional components provide

orchestration, fault management and service management amongst other services to ensure

high availability of user applications.

If I oversimplify it and say that with Openstack one can build their own private cloud, even at

your own personal computer but the compute resources will be limited based on your PC’s

configuration.

**Openstack Components**

Openstack is broken up into several components which offer you the services based on your

need. Some of the common components which are used for basic openstack configuration

are listed below :

* KEYSTONE- provides API for client authentication, service discovery and multi-tenant

authorization.

* GLANCE - provide services for discovering, registering, and retrieving virtual machine

images.

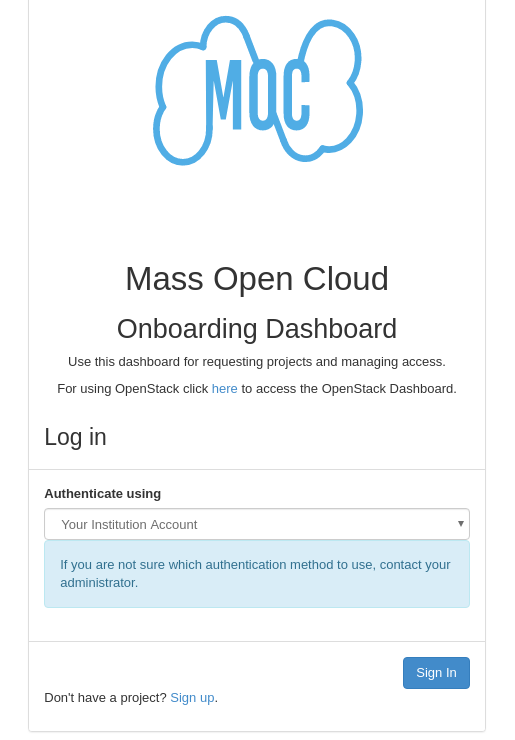
* NOVA - provides services for scaling and accessing compute resources on demand
* NEUTRON - provides networking-as-a-service (NaaS) in virtual compute

environments.

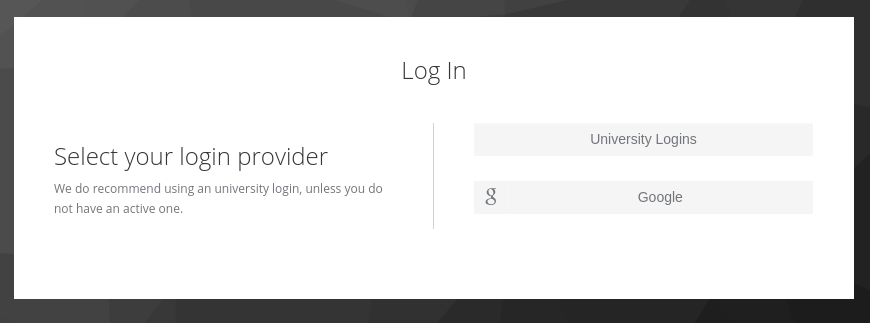
* SWIFT - provides highly available, distributed and consistent object/blob storage.
* CINDER - provides block storage service for openstack, virtualizes the management of block storage and consume the storage resource without the user having any knowledge where the actual storage is located and what type of storage is used.
* HORIZON - provides a web-based interface for openstack services to the end-user.

**Submit Onboarding Request**

Click on “Signup” link at the bottom if you are a first time user, else Authenticate using “Institution account”



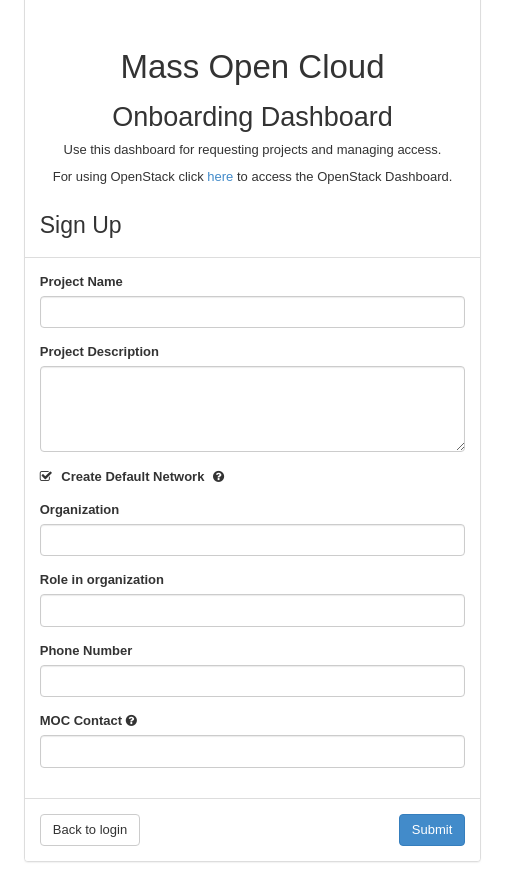
Select “University Login” and you'll be redirected to your university portal , login with your university credentials.



Enter the details as mentioned on the below screen and submit the form. Once your

onboarding request gets approved you can login to the MOC Dashboard as shown in the next

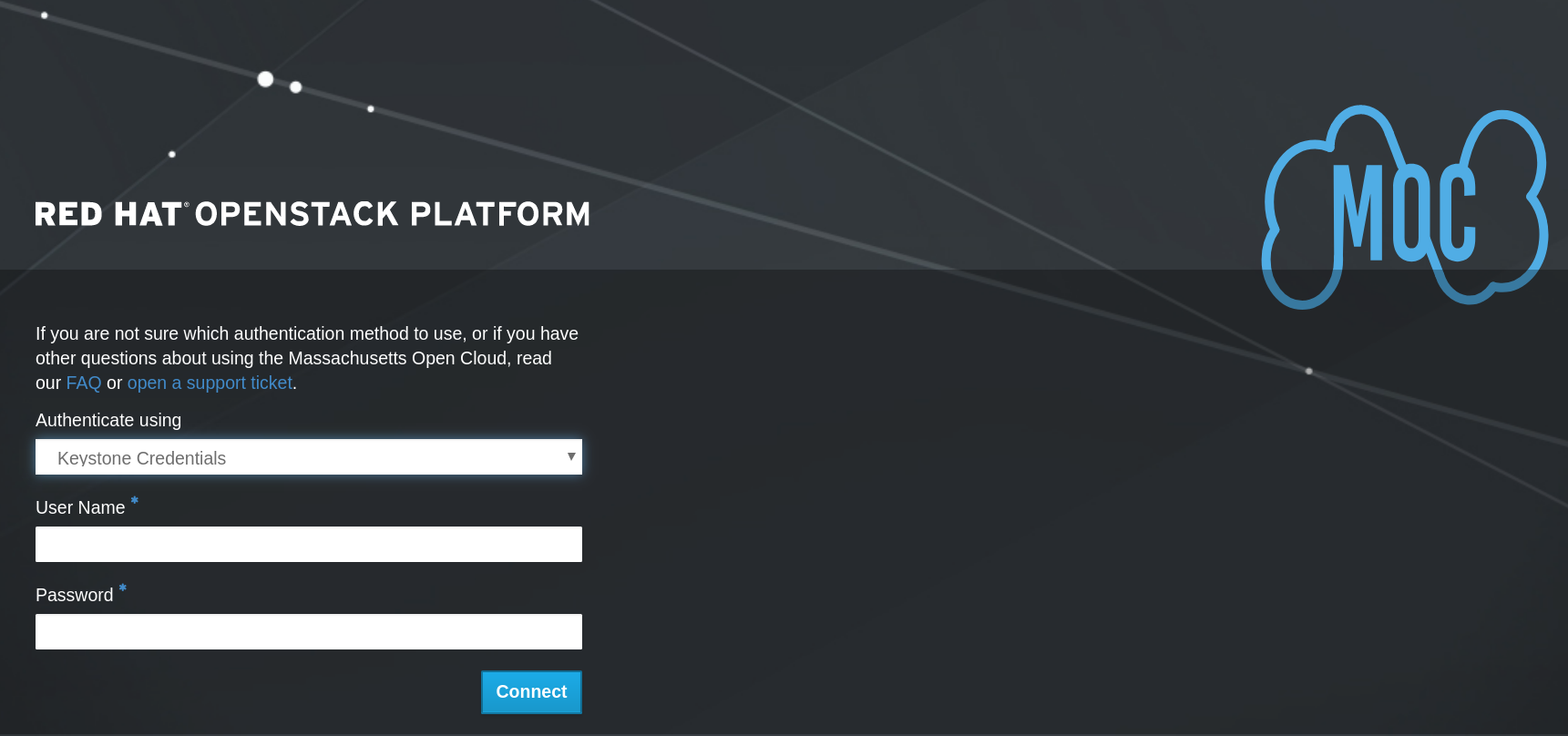
section.



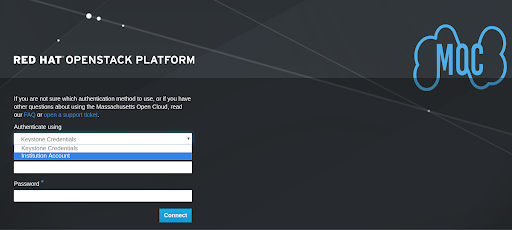
**Login To MOC Openstack Dashboard**

Login URL for MOC Openstack Dashboard(Horizon)– <https://kaizen.massopen.cloud/>

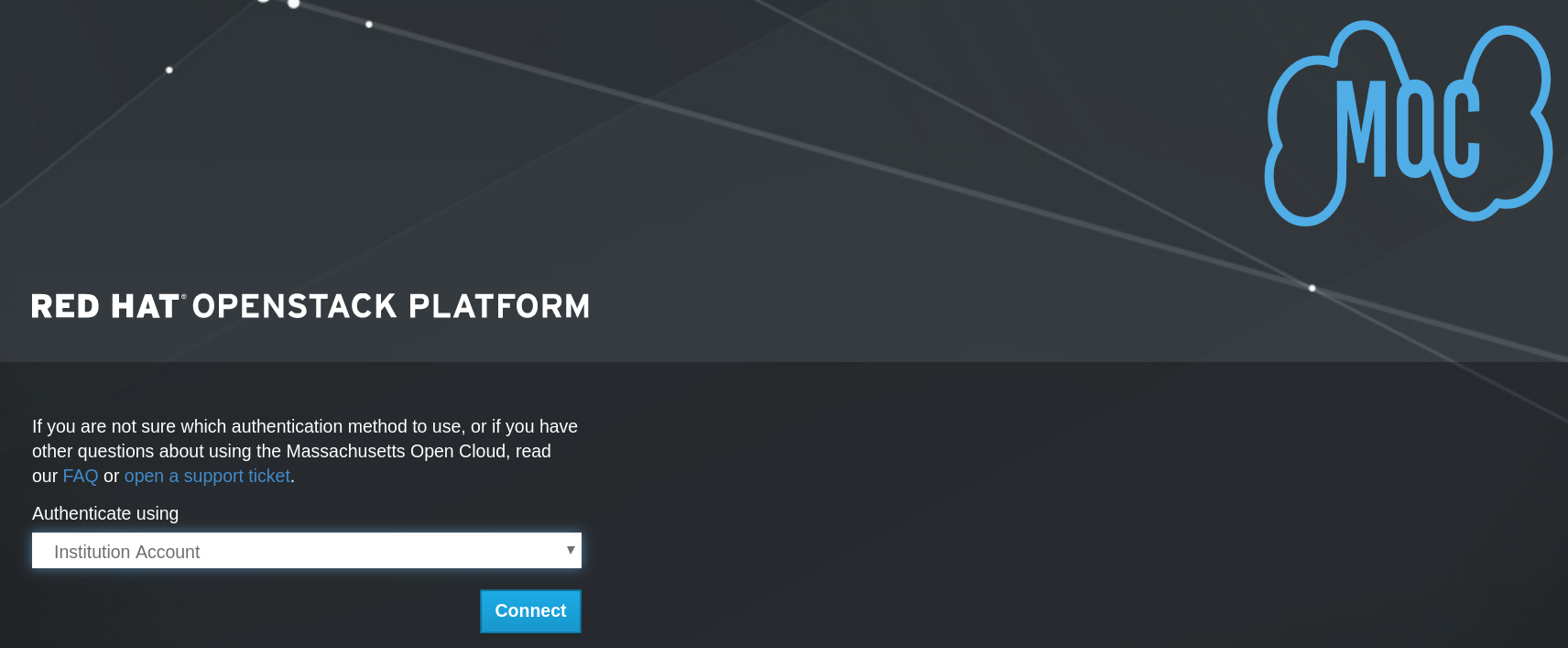
Once you are on the Login page , it will look like as below :



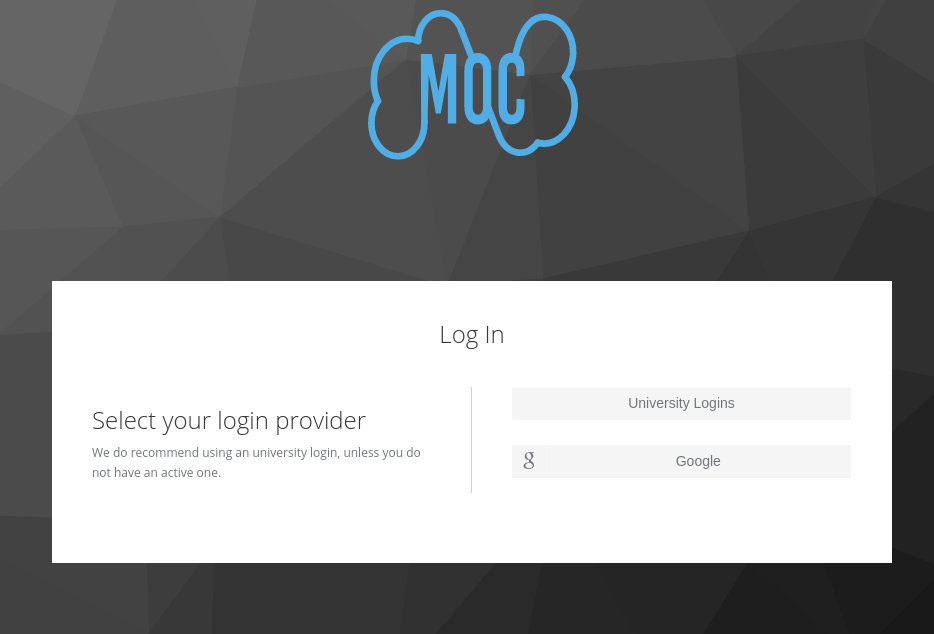
Now Click on the drop-down field “Authenticate Using” and select “Institution Account”



As soon as you select the “Institution account” as the preferred method for authentication, you will see the following screen



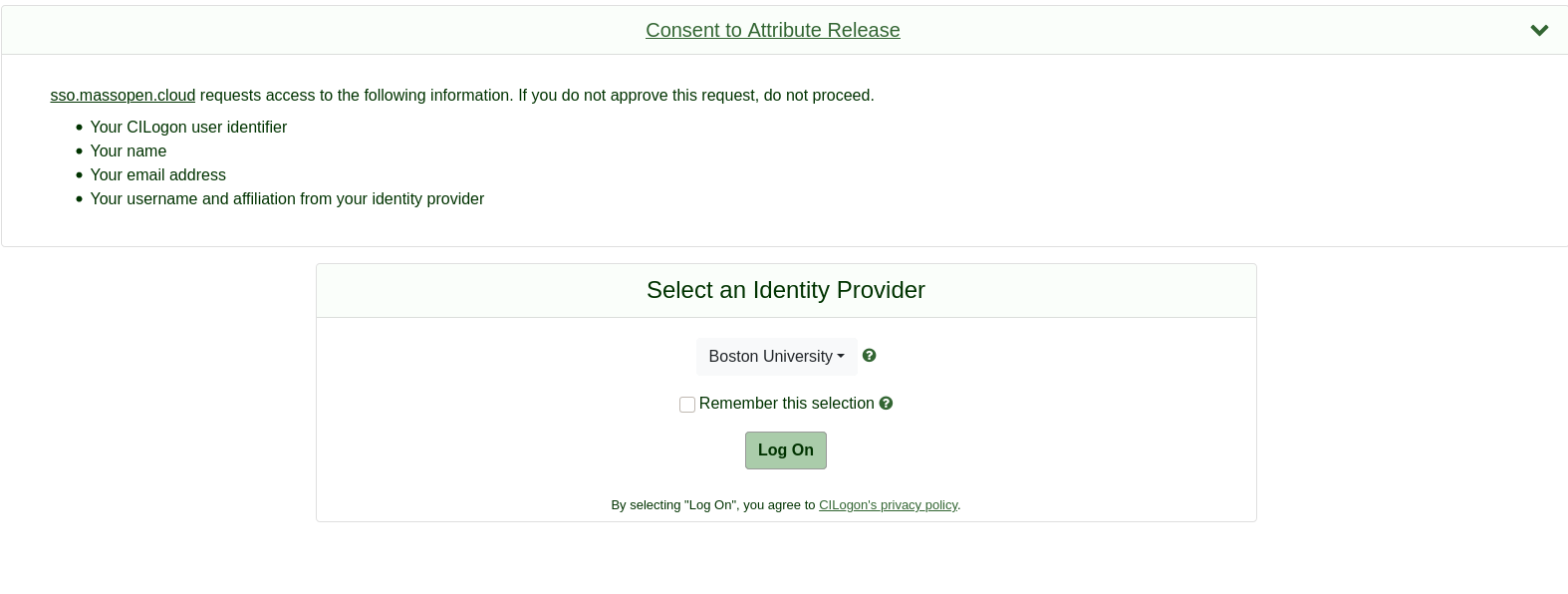
Click the “Connect” button to move on to the next screen which is as below:



Once you have the above screen on your browser , select the “University Logins” option,

which is the recommended way for login. You will be redirected to the CILogon screen as

soon as you select the “university logins” option :



Now, Select your university from the list “Select An Identity Provide”or type it in the “Search”

text field. After selecting the university/institution hit the “Log On” button and you will

redirected to your university/institution login webpage.

Now, enter your university/institution “User-name” and “Password” for authentication

purposes.

The MOC does not see or have access to your institutional account credentials. To reset

“Remember this selection”, visit [cilogon.org/me](https://cilogon.org/me) and select “Delete Browser Cookies”.

Once you are logged in you’ll be redirected to the Openstack Dashboard(Horizon)

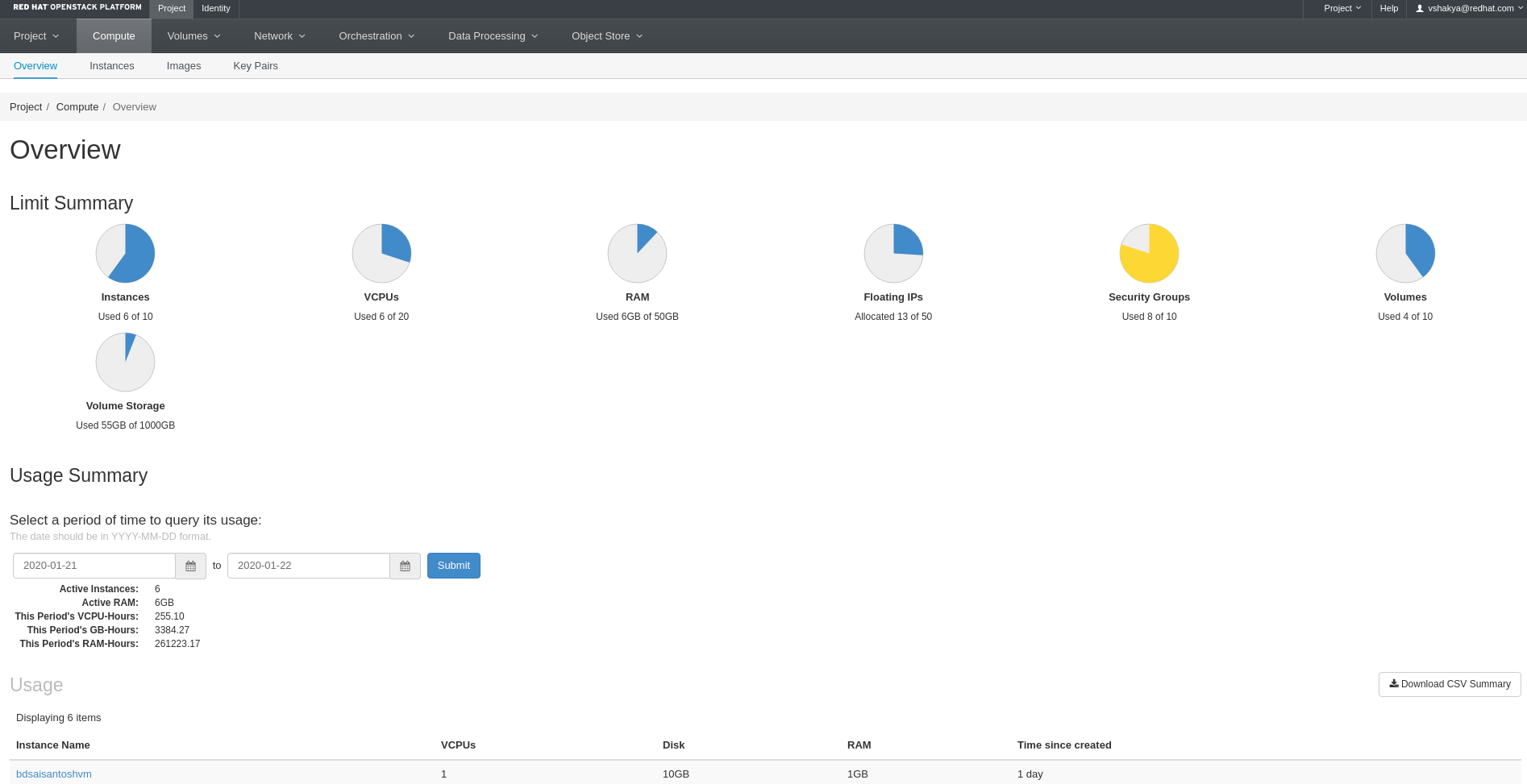
You are currently on “Compute” tab under “Projects” tab

As you can see , this tab displays the overview of the project you have selected.

It displays the Usage summary of the resources such as VM instances , VCPUs, RAM ,

floating IP’s,security groups and volume storage.

You can also see the list of instances currently created under the active project.



**Launch VMs**

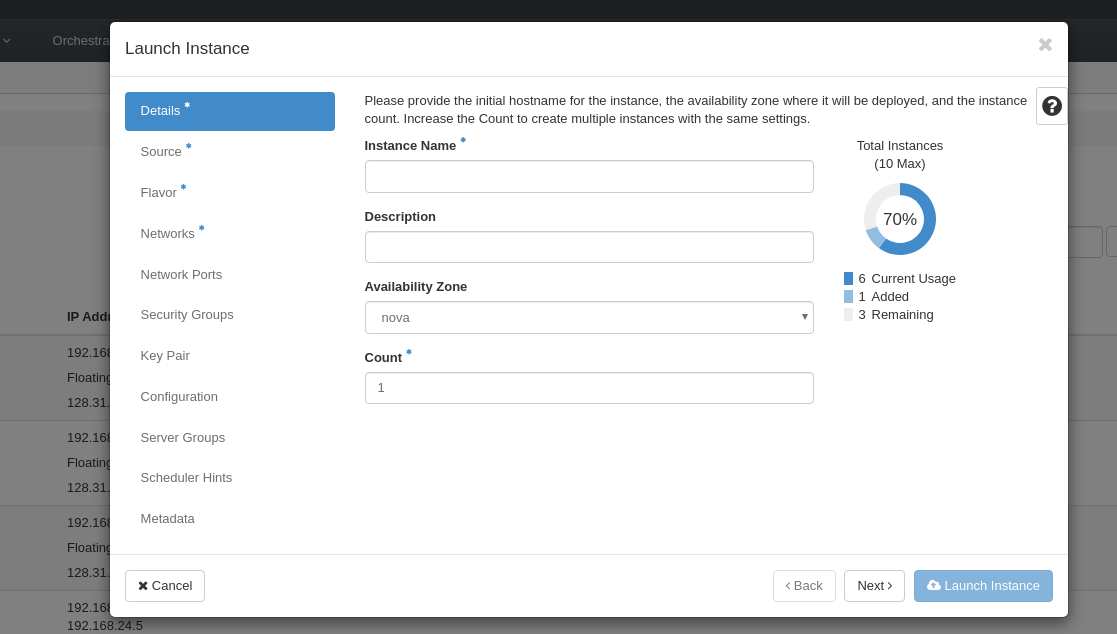


In order to Launch a new VM , switch to “Instances” tab and click on the

button.

Now a pop-up will appear on your screen asking for instance details, flavors, networks and all

the essential parameters required to launch the instance successfully.



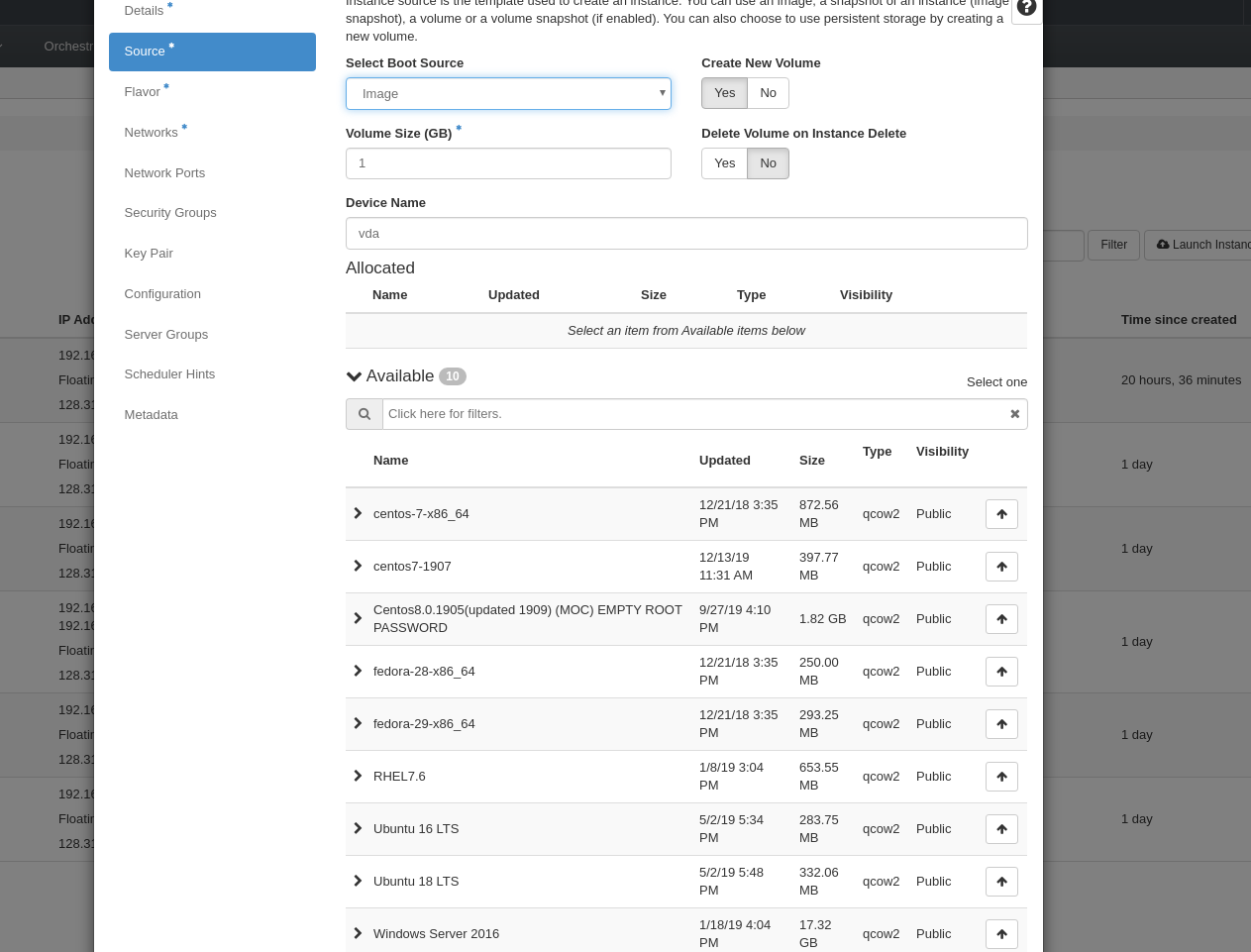
Provide the required(\*) details and click “Next” , Instance name can be any name that you

would like.

In order to launch the VM successfully you need an OS image , MOC openstack provides

images that are already available for selection, select any one of them as per your

requirement.

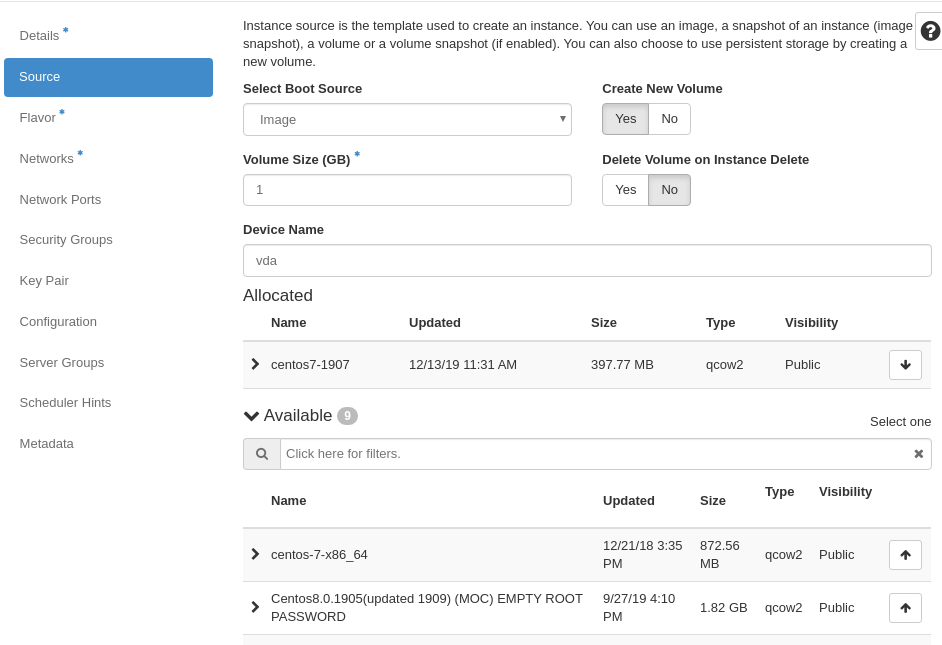




Click on the button to select an image and will be added under the “Allocated” section.

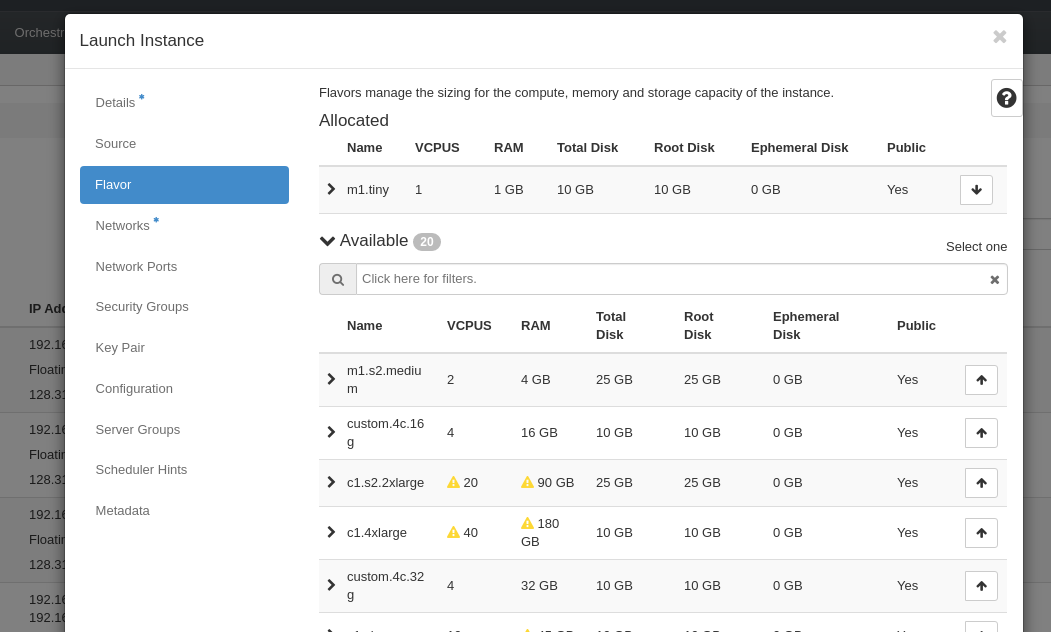
Select “Create New Volume” as “No” if you want to create the volume as per the flavors you’ll

be selecting.



Click “Next” to select the flavors , select “m1.tiny” from the list of flavors in order to create a

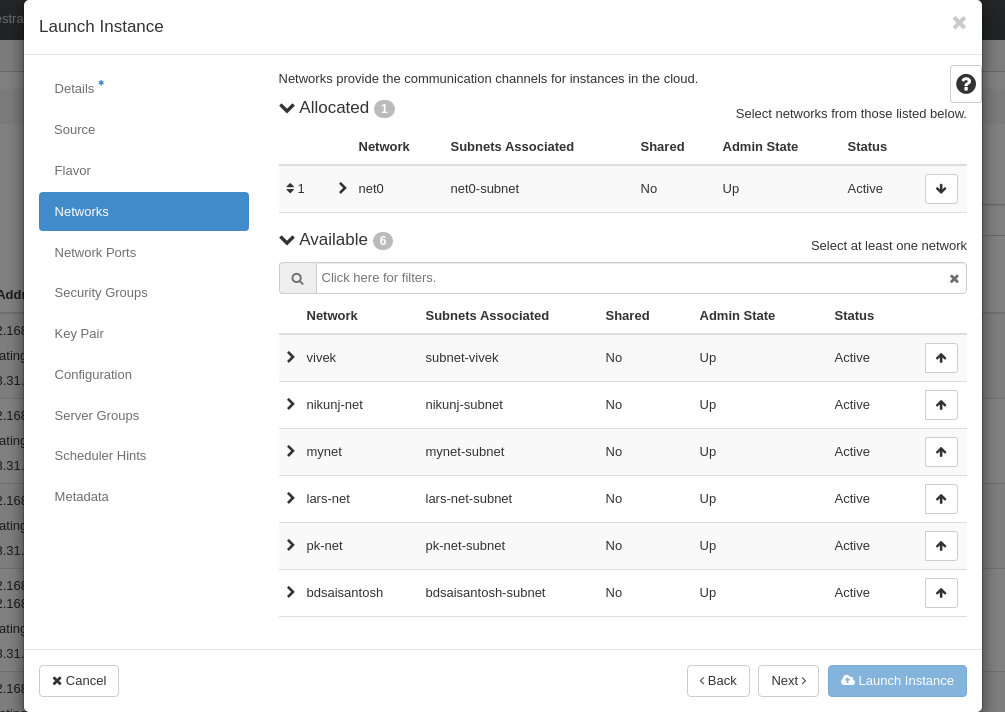
VM with 1 GB RAM, 10 GB Total Disk space.



Select the private network that you have created, if not then create a private network from the



Networks tab. Click on to make sure the network is moved to the “Allocated” section.



**Security Groups**

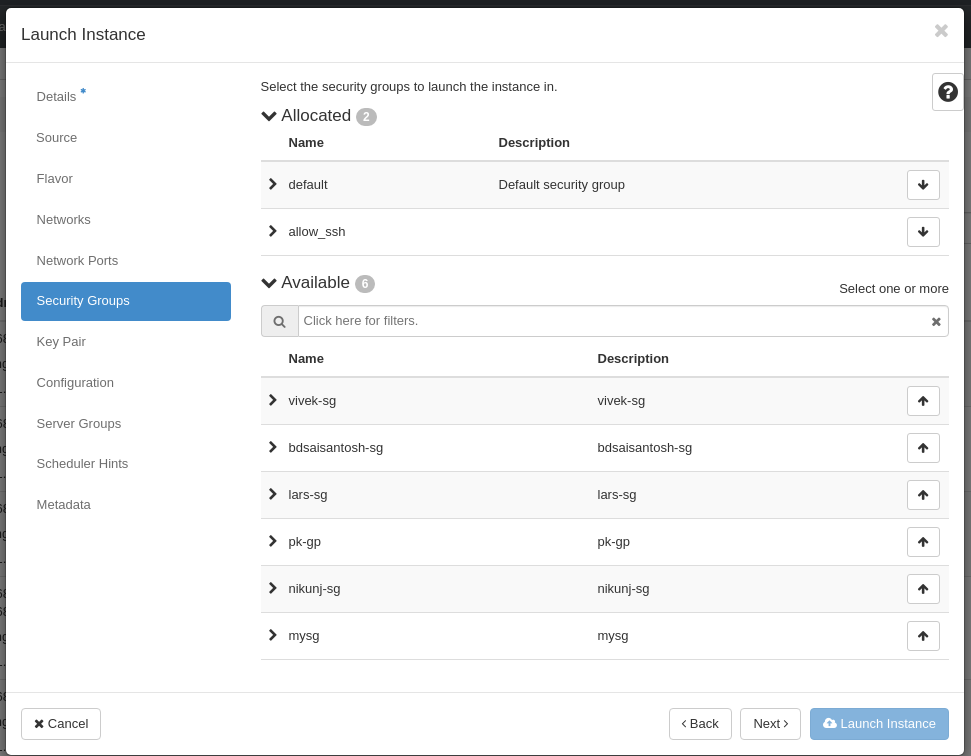
In order to successfully SSH to your VM, you need to add security groups while creating a VM

You can also add security groups later after your VM is launched.



Click on to make sure you have added the security group to the “Allocated” section

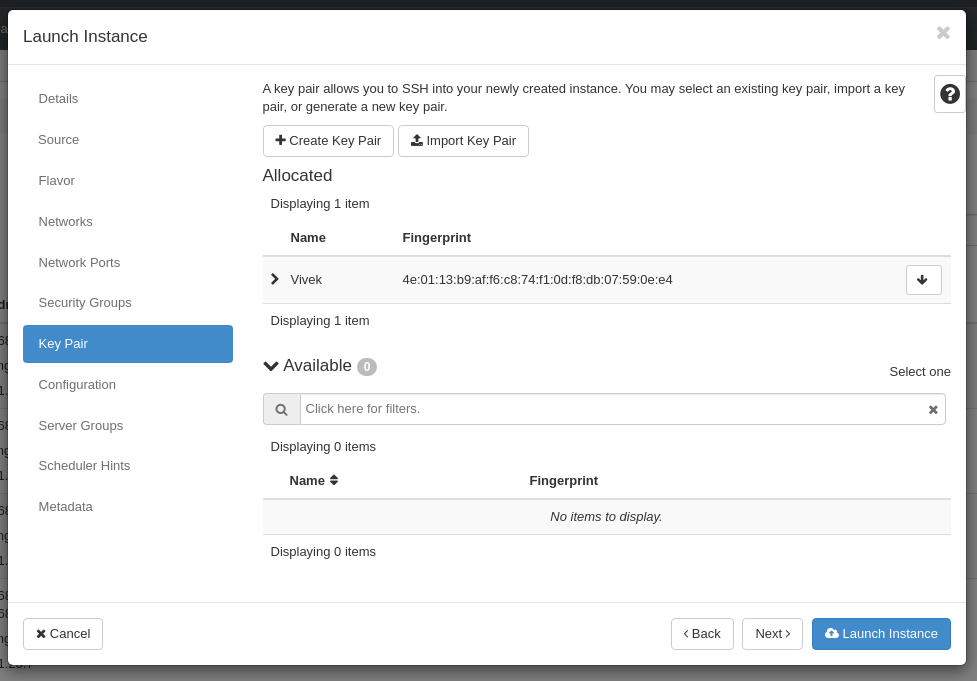
which enables you to SSH to your VM.



**Key Pair**

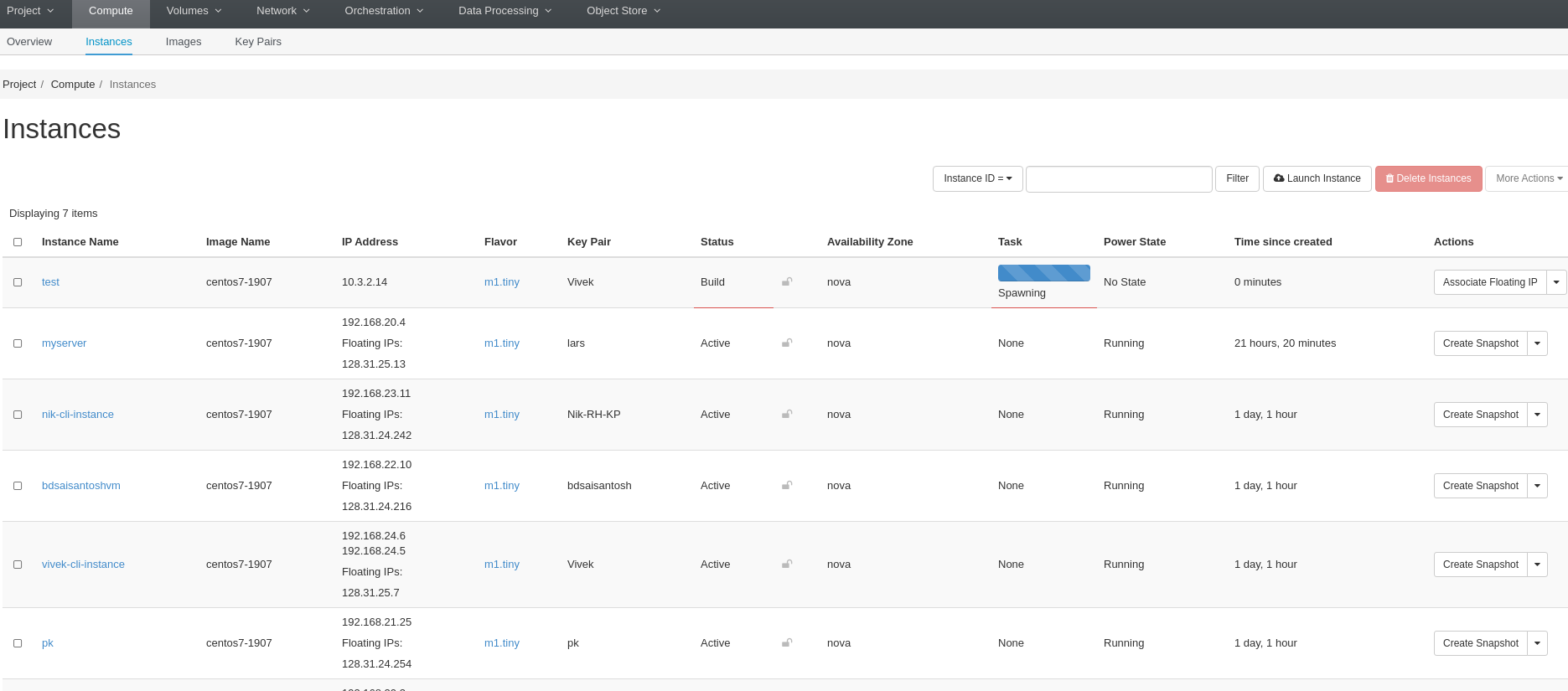
Make sure you add the Key pair you want to use added to the “Allocated” section. If you

haven’t added a key pair yet then go to the “Key Pair” tab add the new one.

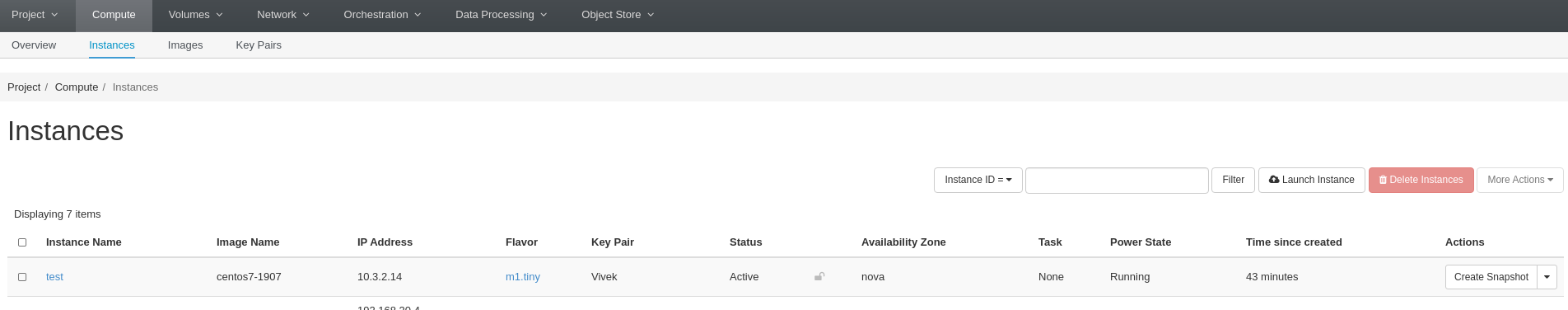


Click on “Launch Instance” button to build and launch your instance. Your VM will be in

spawning state until it is launched successfully.

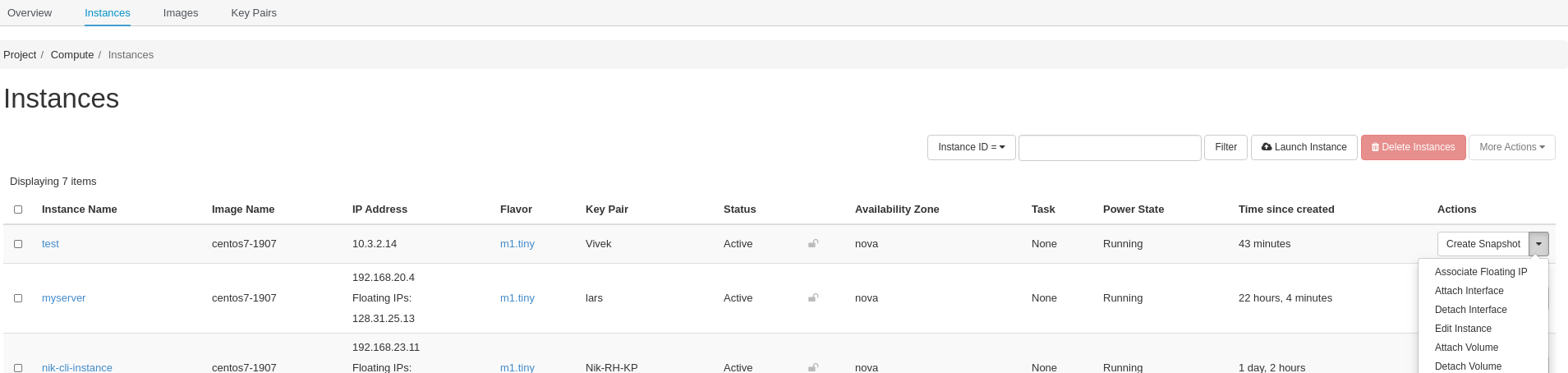


Once the the VM is spawned successfully, you will see the “Power State” is changed from “No

State” to “ Running”

**Associate Floating IP’s**

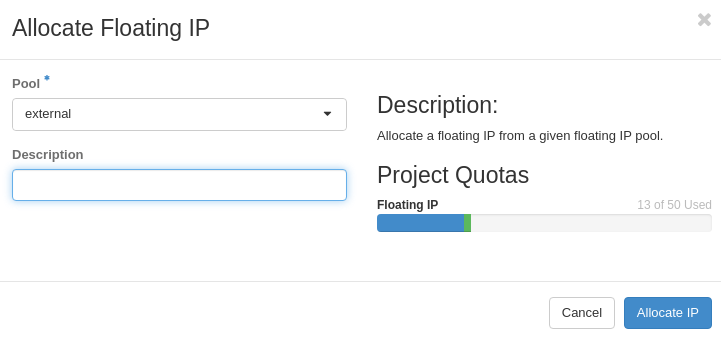
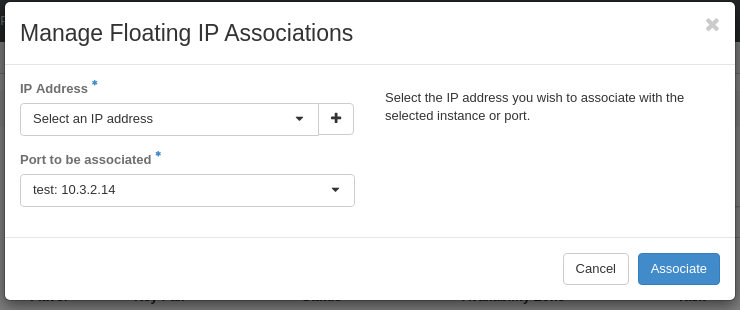
In order to SSH to your VM instance you need to attach floating IP’s to it.

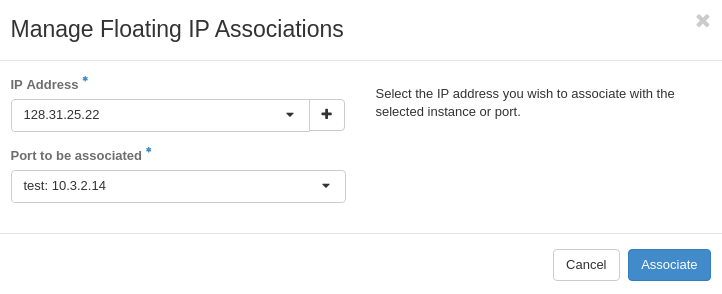


Click on “Associate Floating IP” → Click on “+” button to generate a new IP → Select

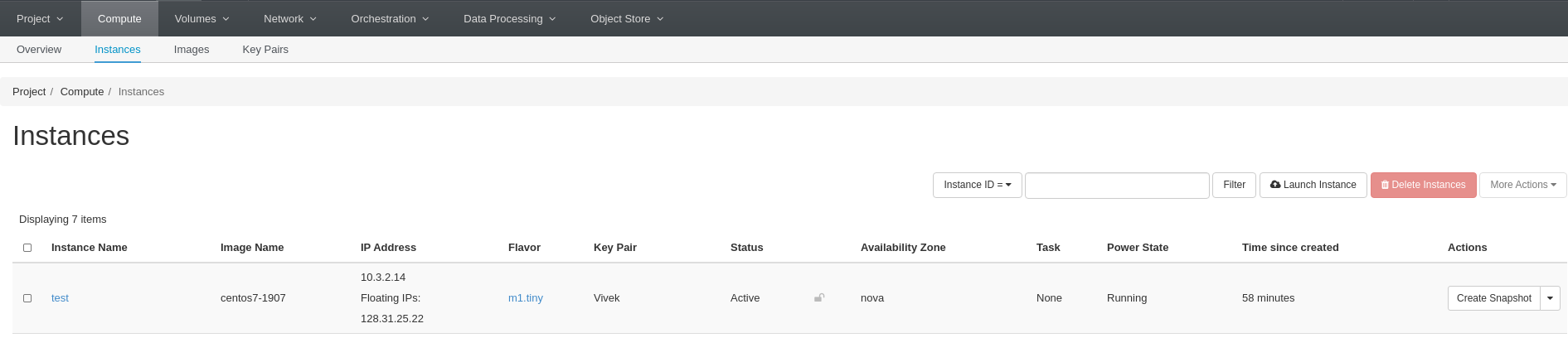
“external” pool which connects to the external networks - > Click “Allocate IP” button - > Click

on “Associate” button to successfully attach the IP to the VM.





You can verify the IP by going to the “Instances” under “Compute” tab



**SSH to VM**

In order to successfully Ssh to your VM you need to make sure that you have done the

following steps in a correct manner :

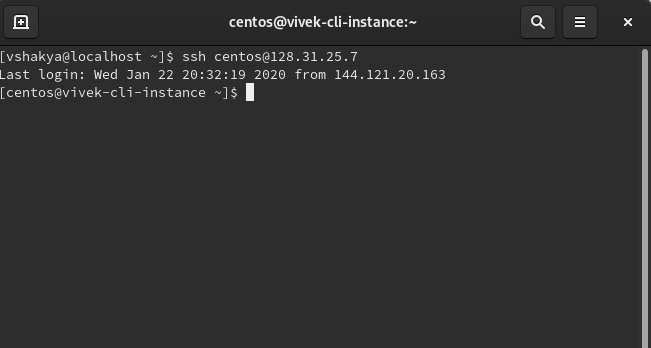
* configure the Security Group which enables you to SSH
* added the Key Pair to setup the public SSH key
* Associate the Floating IP

Default User name for Base Images are :

* CentOS images – centos
* Ubuntu images - ubuntu
* RHEL images – cloud-user

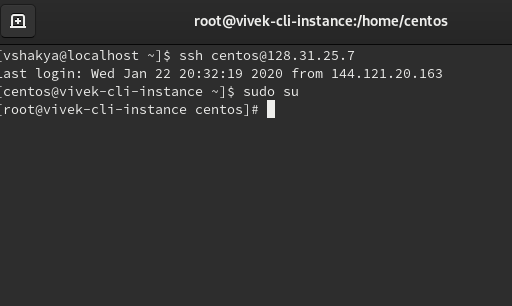
Open the “Terminal” window in your local machine and type the following command

# ssh centos@your\_floating\_IP



In order to become the root user type the following command

#sudo su



**Build and Run a Nodejs hello-world web-app**

Install Nodejs on your VM

Execute the following command on the Terminal to enable nodejs repository

curl -sL https://rpm.nodesource.com/setup\_10.x | sudo bash -

Install Nodejs , execute the following command :

sudo yum install nodejs

Verify the node version and npm version , execute the following commands

node –version cd

npm --version

Create a project repository , for example nodejs\_project

mkdir nodejs\_project

Create the application file “ app.js” which will define the routes , ports and base

directory for your app. Copy paste the following code :

var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello World!');

}).listen(8080)

save it the file app.js, run the following command to verify if nodejs is working :

node app.js

You’ll see “Hello World !” as output on your browser or if anybody access your computer on

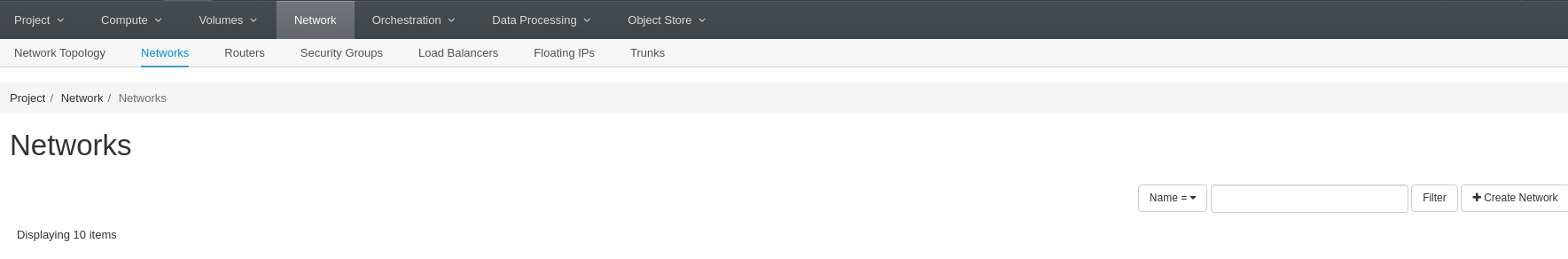
port 80.

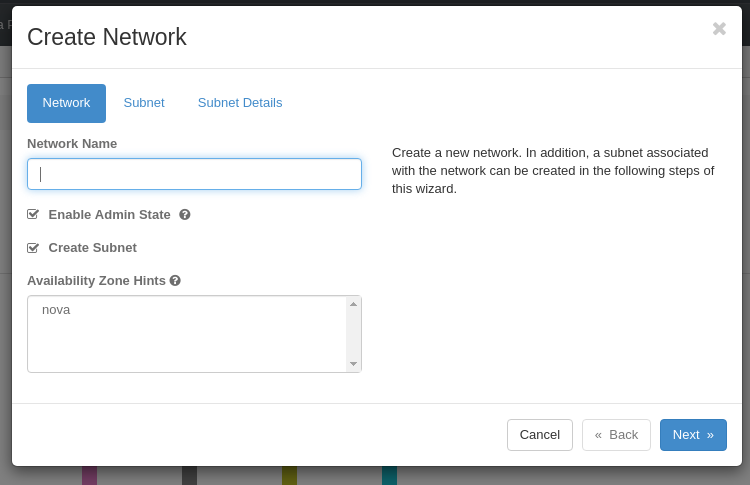
**Create a Network**

Go to the the “Networks” -> “Network Topology” ,here you can see the “Create Network”

button. In order to create your private network click it and a pop-up will appear. Enter the

Network name and leave the default settings as it is. Do not check/uncheck any checkbox.



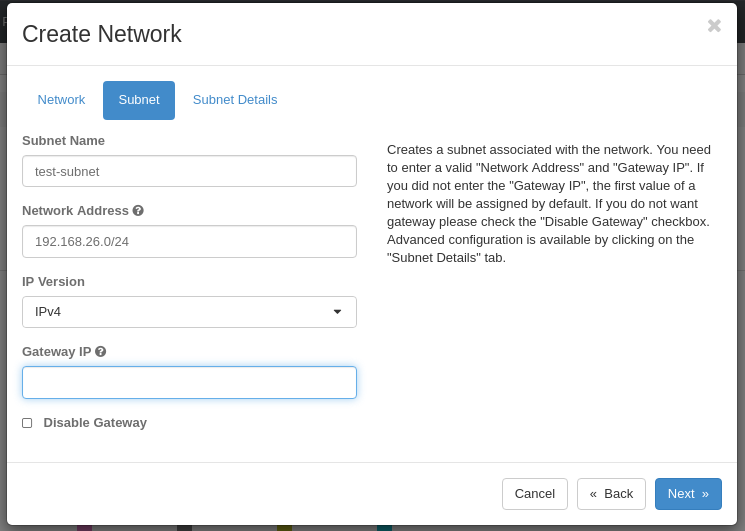


Click “Next” to setup the subnet for your private network.

A **subnet** is a logical partition of an IP network into multiple, smaller network segments. It is typically used to subdivide large networks into smaller, more efficient subnetworks.

Your Private IP addresses should range from the following subnets, as these are reserved IP’s for private networks:

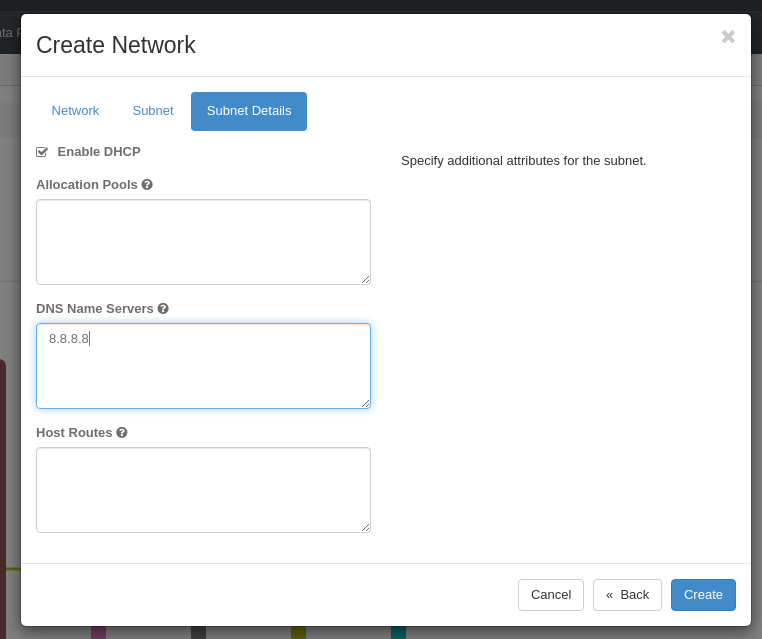
* Range from 10.0.0.0 to 10.255.255.255 — a 10.0.0.0 network with a 255.0.0.0 or an /8 (8-bit) mask
* Range from 172.16.0.0 to 172.31.255.255 — a 172.16.0.0 network with a 255.240.0.0 (or a 12-bit) mask
* A 192.168.0.0 to 192.168.255.255 range, which is a 192.168.0.0 network masked by 255.255.0.0 or /16



Click “Next” to enter the “Subnet Details”

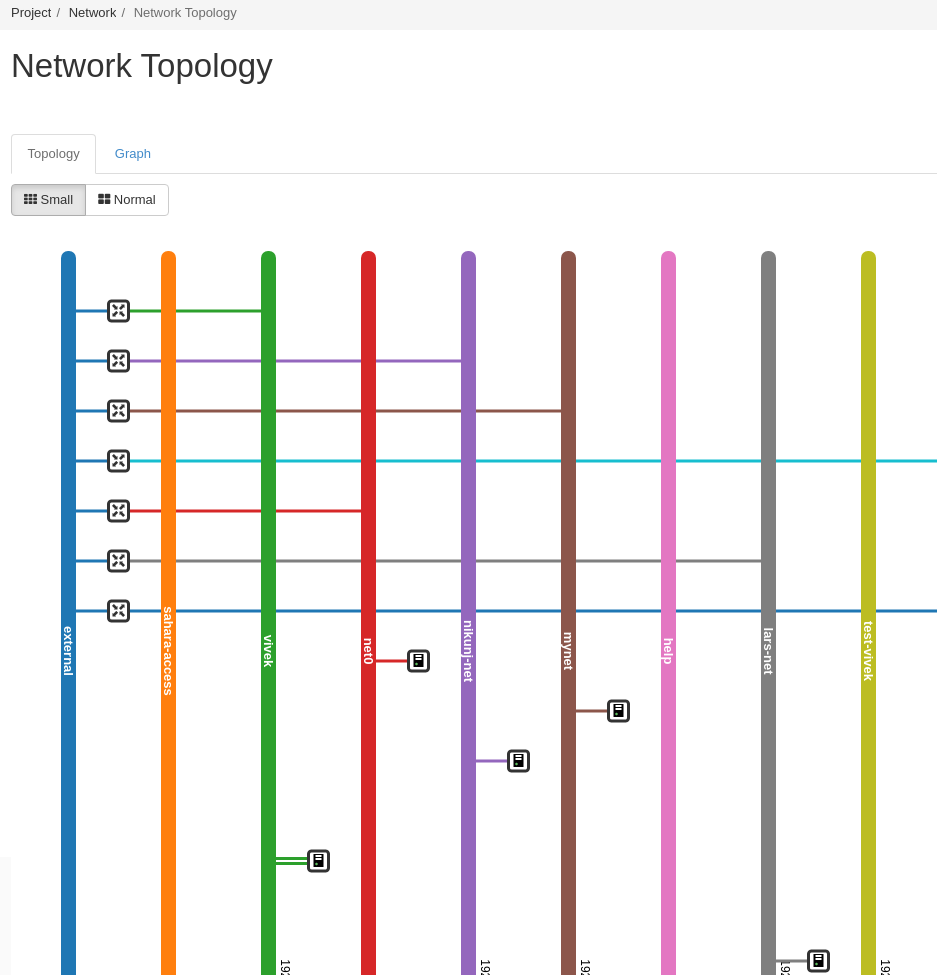
For now leave allocation pool and Host Routes as blank, enter “8.8.8.8” as DNS Name

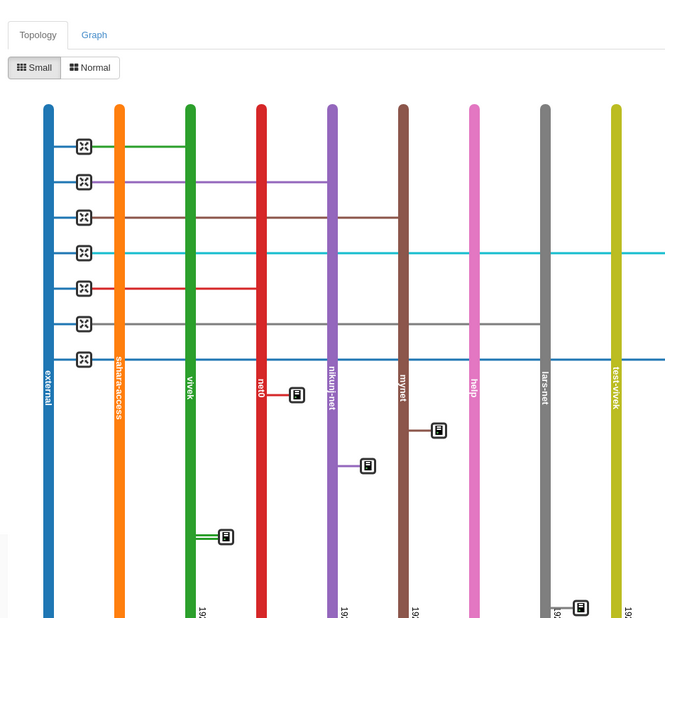
Servers(Google’s public name server) and click on the “Create” button.

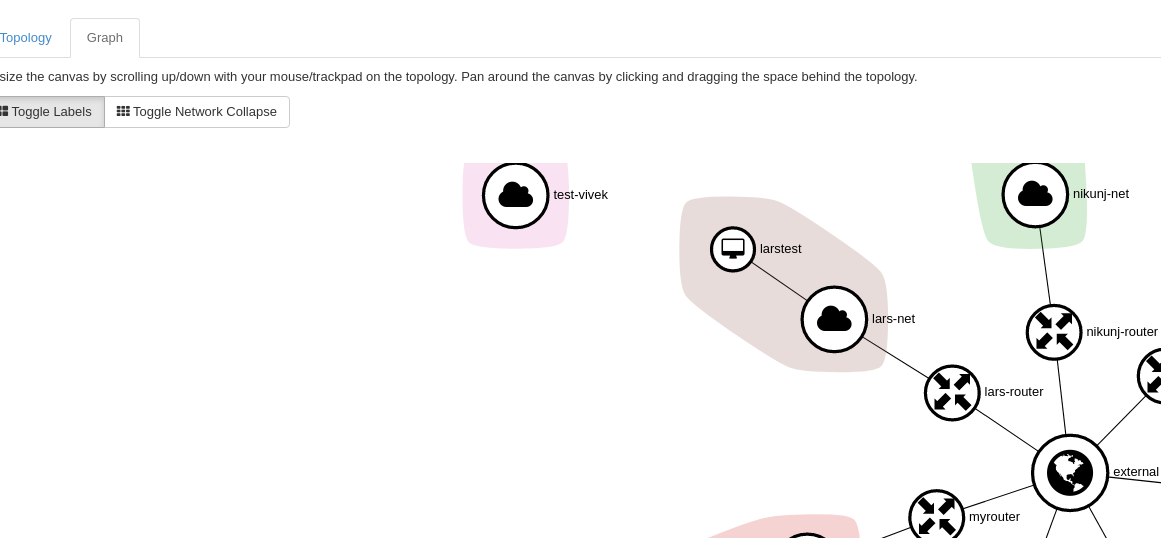


You’ll be now redirected to “Network Topology ” screen and can see your virtual private

network(VPN) next to the public network.







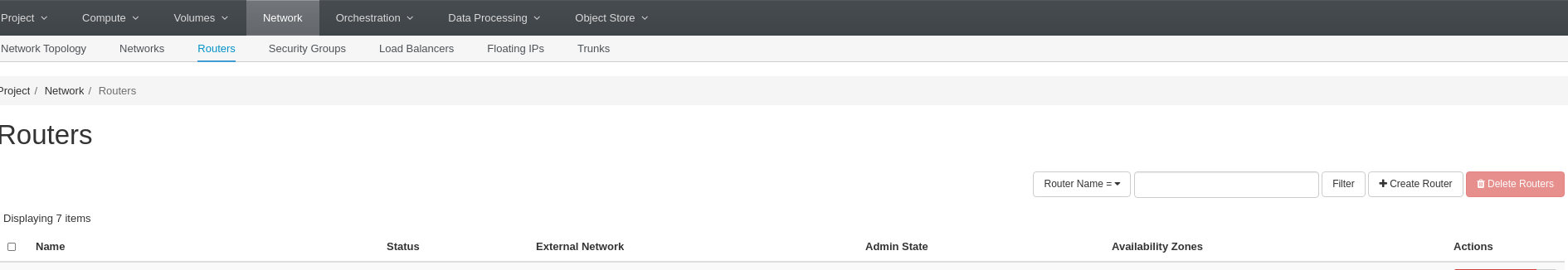
**Create A Router**

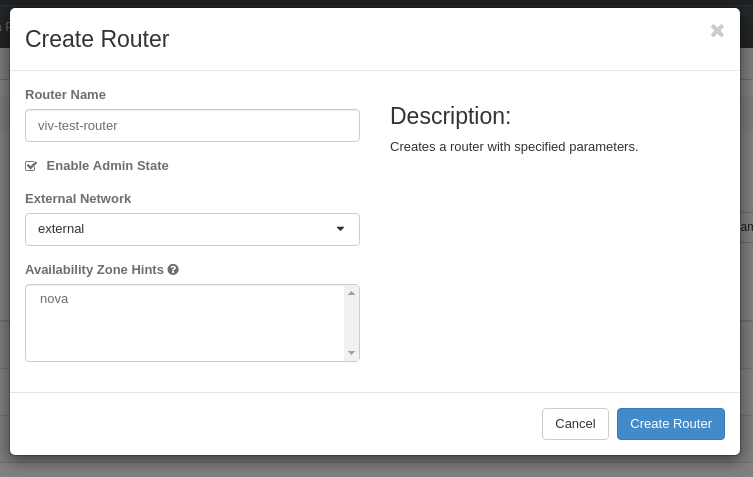
Now that you have create your private network, in order to connect it to public network you

need to create a “Router”. A router connects you to the external gateway. As you can see in

the above screenshot that your network is not connected to the external network.

Click on “Routers” tab under the “Networks” tab , you will notice a “Create Router” button.

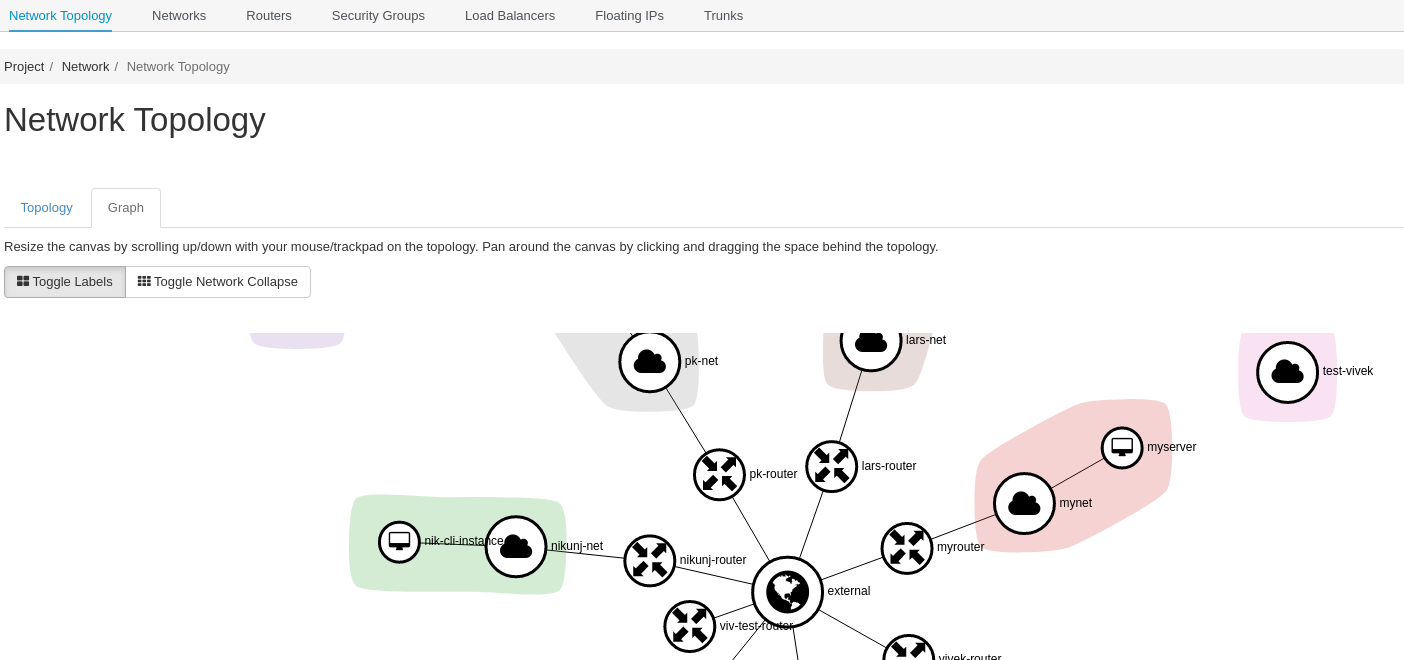




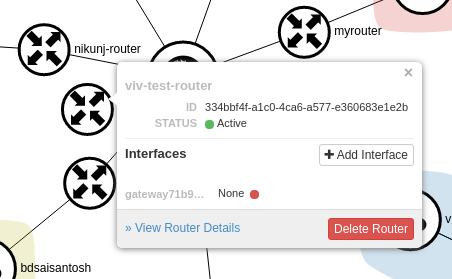
Enter the router name , select the network you want to connect to and click “Create Router”

Now if you go to “Networks” -> “Network Topology”->”Graph” , you’ll see that your router is

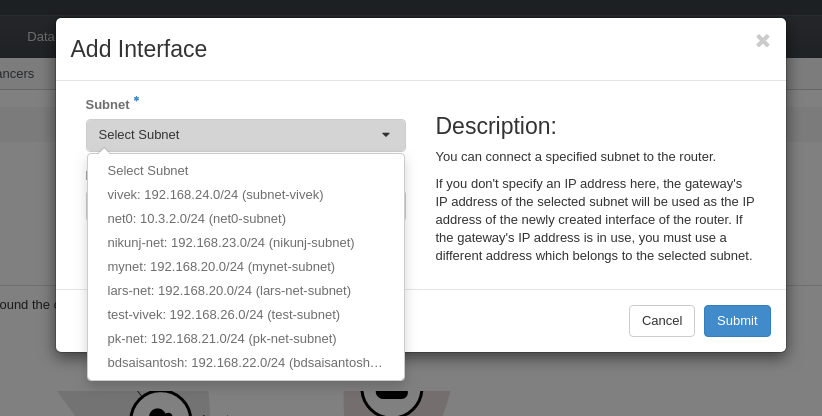
connected to the external network but not to your private network.



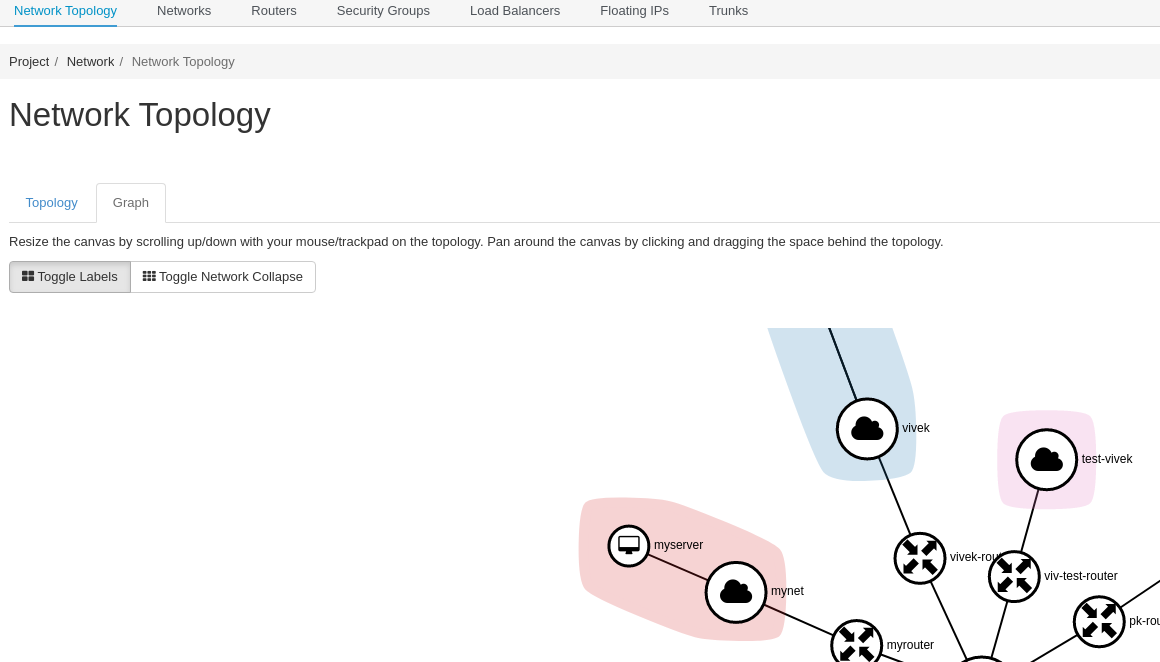
In order to connect your VPN to the router you created, Click on your router and click on “Add Interface” button



Select your VPN from the list on networks available and “Submit” your selection.



Now can see that your VPN is connected to the public network via the router you just created



**Openstack Command Line**

All the steps that we have followed till now, we can perform those from the command line as

well.

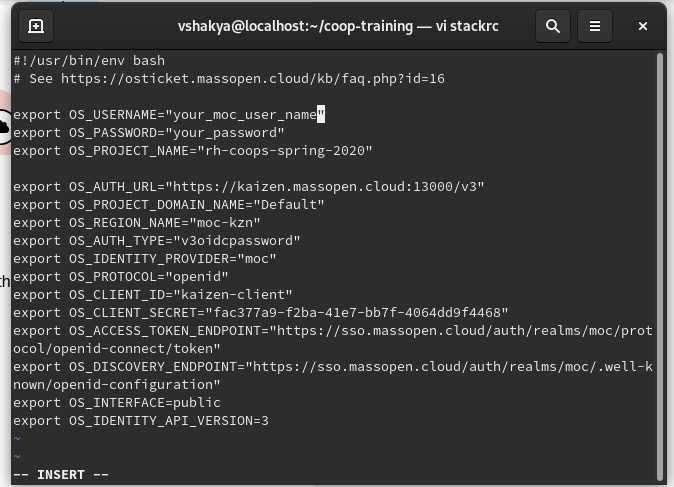
1. Install openstack command line

# sudo yum install python-openstackclient

1. Create the stackrc file

vi stackrc

1. Your stackrc file should look like the below one



1. execute the following command

#.stackrc

1. Create a network

# openstack network create your\_network\_name

1. Create the subnet

#openstack subnet create --network your\_network\_name your\_subnet\_name --subnet-range 192.168.24.0/24

1. Create Router

#openstack router create your\_router\_name

1. Connect router to external gateway

#openstack router set --external-gateway external your\_router\_name

1. Add subnet to the router

#openstack router add subnet you\_router\_name your\_subnet\_name

1. Create security group to allow SSH

#openstack security group rule create your\_security\_group\_name --protocol tcp --dst-port 22 --description 'Allow ssh'

1. Create Floating IP

#openstack floating ip create external

1. Commands to list OS images, flavors and keypair available

#openstack image list

#openstack flavor list | grep tiny

#openstack keypair list

1. Create the VM with flavor “m1.tiny” and “centos7” image

#openstack server create --flavor m1.tiny --image centos7-1907 --key-name

your\_key\_pair\_name --nic net-id=your\_network\_name your\_instance\_name

1. Associate the floating IP

#openstack server add floating ip your\_instance\_name IP\_Addr\_you\_created\_in\_step\_11

1. Add the security group

#openstack server add security group your\_instance\_name your\_security\_group

1. SSH to your VM

#ssh centos@128.31.25.7

Note: Replace centos with “fedora” if it’s fedora VM and “clouduser” for RHEL VM

**REFERENCES**

1. <https://www.openstack.org/software/>
2. <https://docs.massopen.cloud/en/latest/openstack/OpenStack-Tutorial-Index.html>