OpenStack Deployment with RDO Packstack

For our training we're going to provide 3 CentOS 7 servers with Core(TM) i5-2500 CPU @ 3.30GHz, 32 GB RAM and 500GB HDD

For this deployment we're using Packstack from RDO project, since this is the fastest and the best way to deploy a production ready OpenStack environment for development, testing and training.
For our 3 node deployment we're using 1 controller and 2 compute nodes.

Initial System Configurations Nodes

Configuring hosts names using hostnamectl

 # hostnamectl set-hostname hostname
 # hostnamectl status

Next, identify, stop, disable and remove NetworkManager and firewalld

 # systemctl disable firewalld
 # systemctl stop firewalld
 # systemctl disable NetworkManager
 # systemctl stop NetworkManager

Also edit **#vi /etc/sysconfig/selinux** file and set SELINUX line from enforcing to disabled as illustrated on the below screenshot. On all two nodes

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Configuring Passwordless Authentication from Controller node to Compute

 # ssh-keygen

 # ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.103.228
 # ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.103.234

Synchronize time with a NTP server

 # yum install ntp -y
 # timedatectl set-timezone Europe/Kiev
 # systemctl enable ntpd
 # systemctl start ntpd
 # ntpstat

```
[root@controller-node ~]# ntpstat
synchronised to NTP server (194.54.80.29) at stratum 4
   time correct to within 1026 ms
   polling server every 64 s
[root@controller-node ~]# date
Sun Jan  7 16:54:38 EET 2018
[root@controller-node ~]# timedatectl
      Local time: Sun 2018-01-07 16:55:09 EET
  Universal time: Sun 2018-01-07 14:55:09 UTC
        RTC time: Sun 2018-01-07 14:55:09
       Time zone: Europe/Kiev (EET, +0200)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: no
 Last DST change: DST ended at
                  Sun 2017-10-29 03:59:59 EEST
                  Sun 2017-10-29 03:00:00 EET
 Next DST change: DST begins (the clock jumps one hour forward) at
                  Sun 2018-03-25 02:59:59 EET
                  Sun 2018-03-25 04:00:00 EEST
[root@controller-node ~]#
```

Setup hosts file if DNS resolution is not configured

 # vi /etc/hosts

```
[root@controller-node ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.103.234 controller-node.training.local controller-node
192.168.103.228 compute-node.training.local compute-node
[root@controller-node ~]#
```

Controller Node Setup

On the controller node we need to update the server, install the latest RDO CentOS Ocata Release packages and the Packstack tool, generate the packstack answer file, enable root login and set the root password.

yum update -y (has been done already for you for the training)
reboot (has been done already for you for the training)

Run these commands on the controller during our training session:

 # yum install centos-release-openstack-ocata -y
 # yum update -y
 # yum install -y openstack-packstack
 # packstack --gen-answer-file=answer-node-provider-file

Adapt the Packstack Answer File

 # vi answer-node-provider-file

 # We want to use our own tenant / project and user, disable DEMO project / user
CONFIG_PROVISION_DEMO=n
 # SSL doesn't work for Horizon out of the box (is disabled by default)
CONFIG_HORIZON_SSL=y
CONFIG_CONTROLLER_HOST=192.168.103.234
 # Provide the IPs of your compute nodes
CONFIG_COMPUTE_HOSTS=192.168.103.228,192.168.103.180
 # We're using the controller as network node as well (don't change it)
CONFIG_NETWORK_HOSTS=192.168.103.234
 # Define NTP servers for time synchronization
CONFIG_NTP_SERVERS=0.pool.ntp.org,1.pool.ntp.org,2.pool.ntp.org,3.pool.ntp.org
 # Using provider network (was set to br-ex, which doesn't make any sense)
CONFIG_NEUTRON_L3_EXT_BRIDGE=provider
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=extnet:br-ex
CONFIG_NEUTRON_OVS_EXTERNAL_PHYSNET=extnet
 # Set the right interface (eth0) for Open vSwitch bridge on the controller / network node
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eno1
# Increase the Cinder Volume Size, was set to 20G
CONFIG_CINDER_VOLUMES_SIZE=100G

Run Packstack and start the deployment

Now we can run packstack with our answer-file as follow and start our multi-node deployment:

 # packstack --answer-file=nswer-node-provider-file

The deployment will take only 15 minutes, the complete output has been truncated here, at the end you shall get something like this:

```
Preparing OpenStack Network-related Nova entries     [ DONE ]
Preparing Nova Common entries                        [ DONE ]
Preparing Neutron LBaaS Agent entries                [ DONE ]
Preparing Neutron API entries                        [ DONE ]
Preparing Neutron L3 entries                         [ DONE ]
Preparing Neutron L2 Agent entries                   [ DONE ]
Preparing Neutron DHCP Agent entries                 [ DONE ]
Preparing Neutron Metering Agent entries             [ DONE ]
Checking if NetworkManager is enabled and running    [ DONE ]
Preparing OpenStack Client entries                   [ DONE ]
Preparing Horizon entries                            [ DONE ]
Preparing Swift builder entries                      [ DONE ]
Preparing Swift proxy entries                        [ DONE ]
Preparing Swift storage entries                      [ DONE ]
Preparing Gnocchi entries                            [ DONE ]
Preparing Redis entries                              [ DONE ]
Preparing Ceilometer entries                         [ DONE ]
Preparing Aodh entries                               [ DONE ]
Preparing Puppet manifests                           [ DONE ]
Copying Puppet modules and manifests                 [ DONE ]
Applying 192.168.103.234_controller.pp
192.168.103.234_controller.pp:                       [ DONE ]
Applying 192.168.103.212_network.pp
192.168.103.212_network.pp:                          [ DONE ]
Applying 192.168.103.228_compute.pp
192.168.103.228_compute.pp:                          [ DONE ]
Applying Puppet manifests                            [ DONE ]
Finalizing                                           [ DONE ]

 **** Installation completed successfully ******

Additional information:
 * NOTE : A selfsigned CA certificate was generated to be used for ssl, you should still change it do subordinate CA cert. In any case please save the contents
/packstackca/.
 * File /root/keystonerc_admin has been created on OpenStack client host 192.168.103.234. To use the command line tools you need to source the file.
 * NOTE : A certificate was generated to be used for ssl, You should change the ssl certificate configured in /etc/httpd/conf.d/ssl.conf on 192.168.103.234 to u
signed cert.
 * To access the OpenStack Dashboard browse to https://192.168.103.234/dashboard .
Please, find your login credentials stored in the keystonerc_admin in your home directory.
 * Because of the kernel update the host 192.168.103.212 requires reboot.
 * The installation log file is available at: /var/tmp/packstack/20180104-101305-yatSOT/openstack-setup.log
 * The generated manifests are available at: /var/tmp/packstack/20180104-101305-yatSOT/manifests
[root@controller-node ~]#
```

After the initial deployment it might be a good idea to reboot all servers (I didn't do it and I'm not sure if this is really needed).

To be able to log into the Horizon dashboard, you'd need to grab the admin password from the keystonerc_admin file:

 # cat keystonerc_admin

```
[root@controller-node ~]# cat keystonerc_admin
unset OS_SERVICE_TOKEN
    export OS_USERNAME=admin
    export OS_PASSWORD=526c9ad3fe5c4ab1
    export OS_AUTH_URL=http://192.168.103.234:5000/v3
    export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
    [root@controller-node ~]#
```

Log into Horizon Dashboard as the admin user with the password provided above for your installation (replace the IP with the IP of your controller):
https://192.168.103.234/dashboard

Create Networks

Before creating public and private networks, we need to source our keystonerc_admin file:

 # source keystonerc_admin

Create the public network

 # neutron net-create public --provider:network_type flat --provider:physical_network extnet --router:external

Create the public subnet

To create the public subnet, one needs to request an IP block from training.local, but for our example here we assume we can use a floating IP range from 192.168.103.130 to 192.168.103.254 with the network 192.168.103.128/25

 # neutron subnet-create --gateway 192.168.103.129 --allocation-pool start=192.168.103.130,end=192.168.103.254 --disable-dhcp --name public_subnet public 192.168.103.128/25

Create a private network with a subnet

First we'll create a private network named private with the subnet private_subnet and the CIRD 172.20.16.0/27 and enable DHCP and define the name servers

 # neutron net-create private

 # neutron subnet-create private 172.20.16.0/27 --name private_subnet --enable-dhcp=True --dns-nameserver 8.8.8.8 --dns-nameserver 8.8.4.4

And create a router named router1:

 # neutron router-create router1

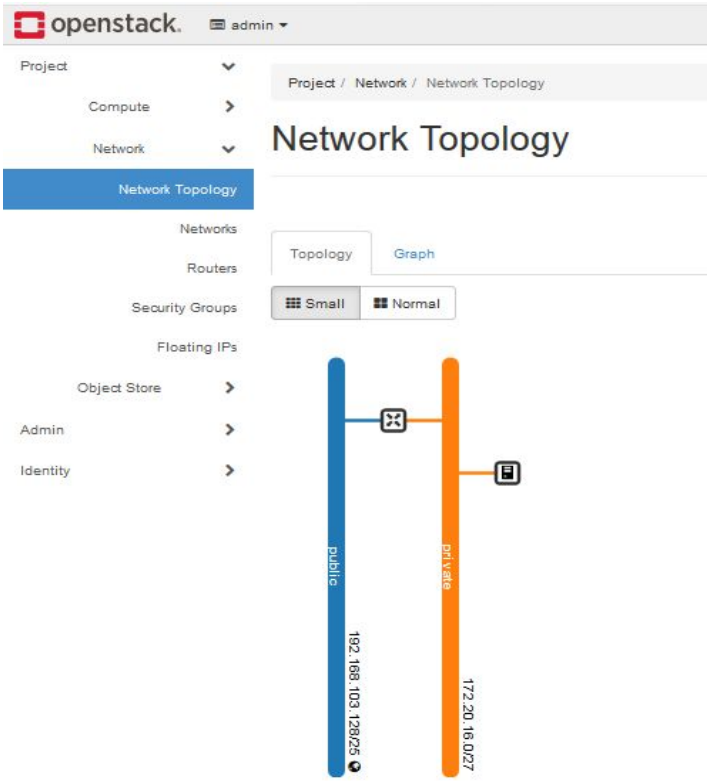
Set the gateway for this router to the public network:

 # neutron router-gateway-set router1 public

And finally set the router interface to the private subnet:

 # neutron router-interface-add router1 private_subnet

In Horizon we shall see our new private subnet and the router:



Add an Image to glance

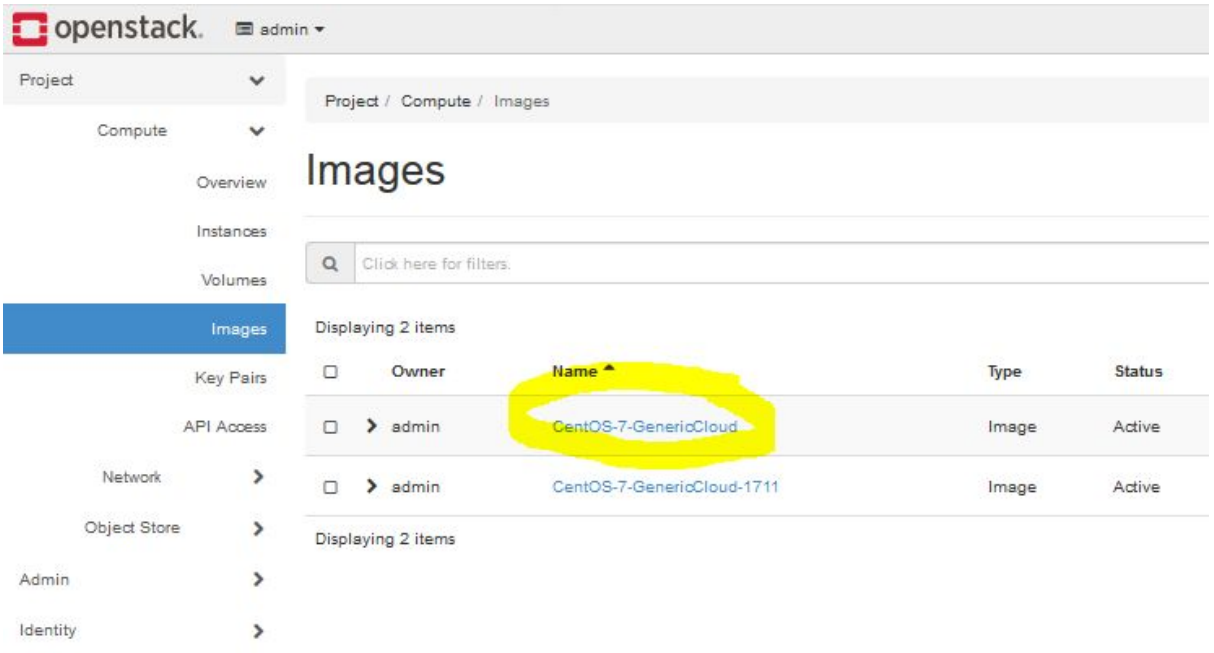
Download the latest CentOS QCOW2 Image and extract the qcow2 image:

 # wget http://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud.qcow2.xz
 # xz -d CentOS-7-x86_64-GenericCloud.qcow2.xz

Add the image to glance:

 # glance image-create --container-format=bare --disk-format=qcow2 --name=CentOS-7-GenericCloud < CentOS-7-x86_64-GenericCloud.qcow2

In Horizon navigate to Compute → Images and verify if your image is listed there:



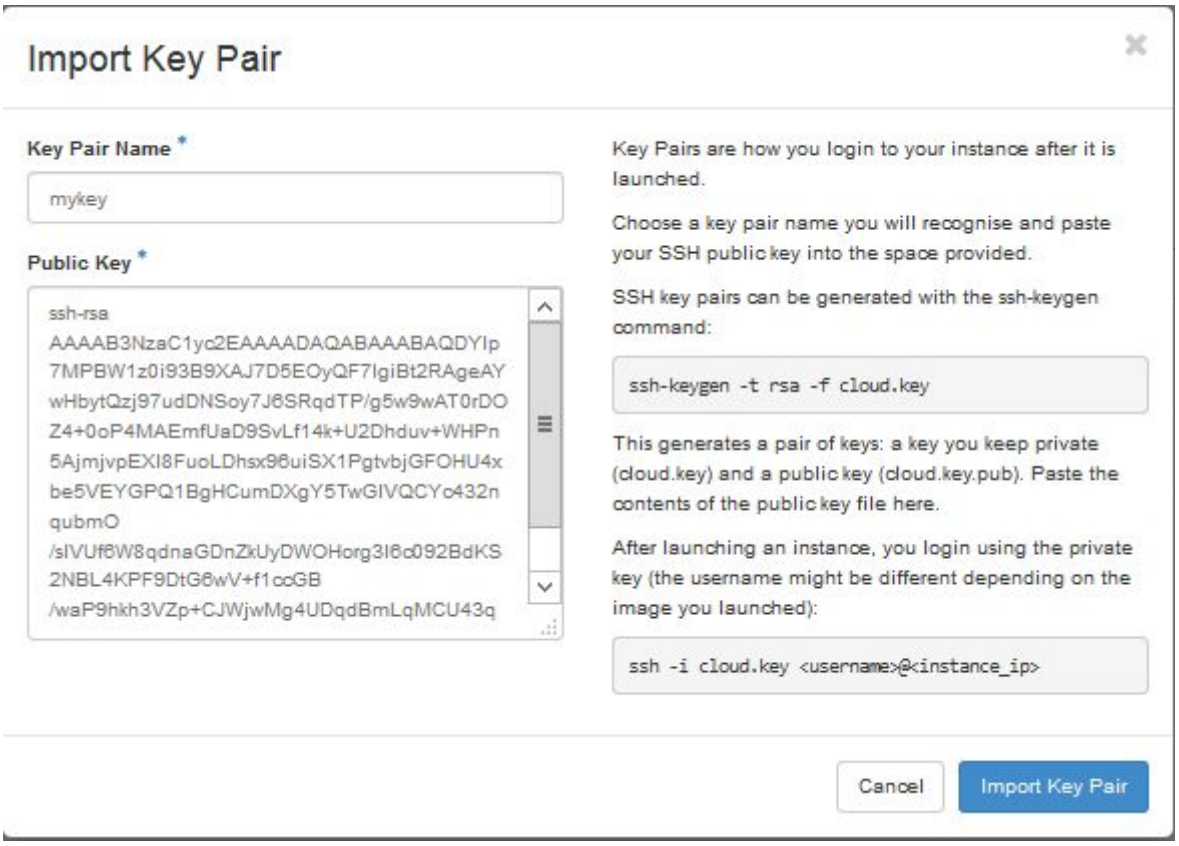
Run an instance through Horizon Dashboard

Before running an instance we need to create a key pair, or if we have already one key pair, we can import our existing public key through the dashboard:

To do so, navigate in Horizon to Projekt → Compute → Keypair and click on “import keypair” and provide a name “mykey” and your public key.

Alternatively you can create a new keypair by clicking on:

(create keypair and download the private key):



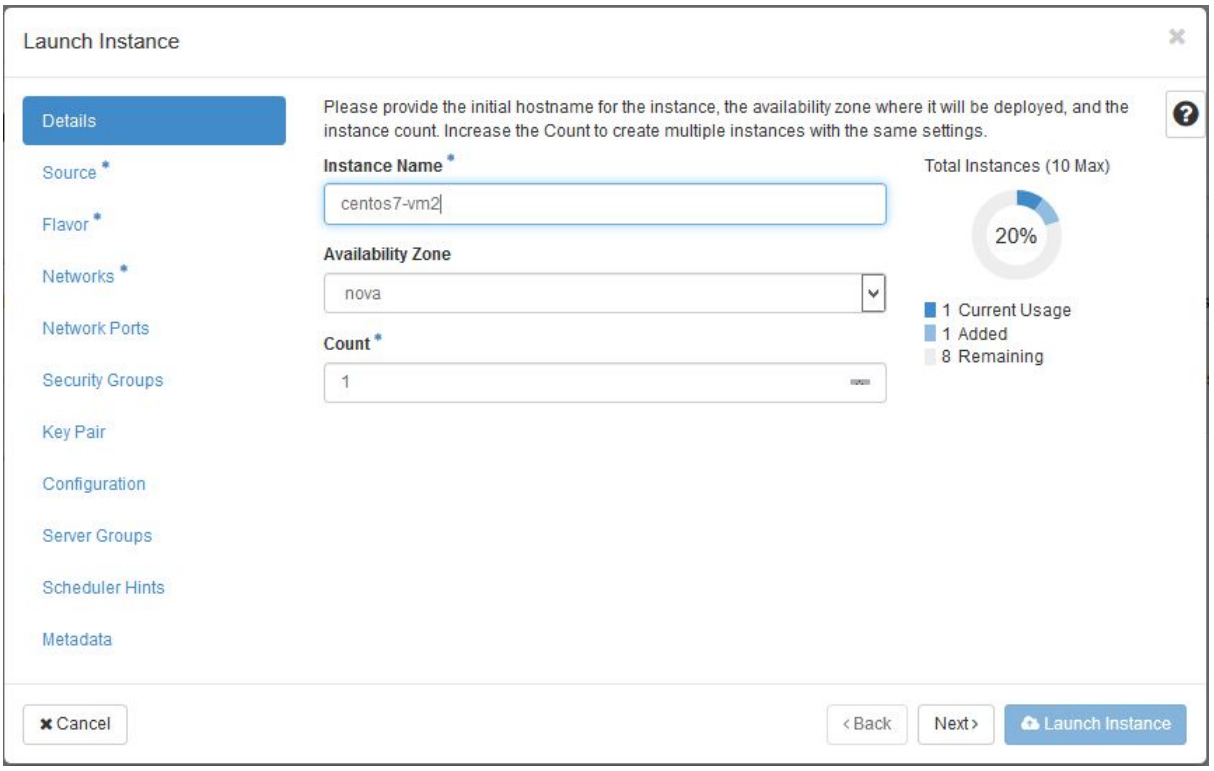
Finally we can start a new instance from our CentOS QCOW2 Image by navigating to:

`Projekt → Compute → Instanzen`

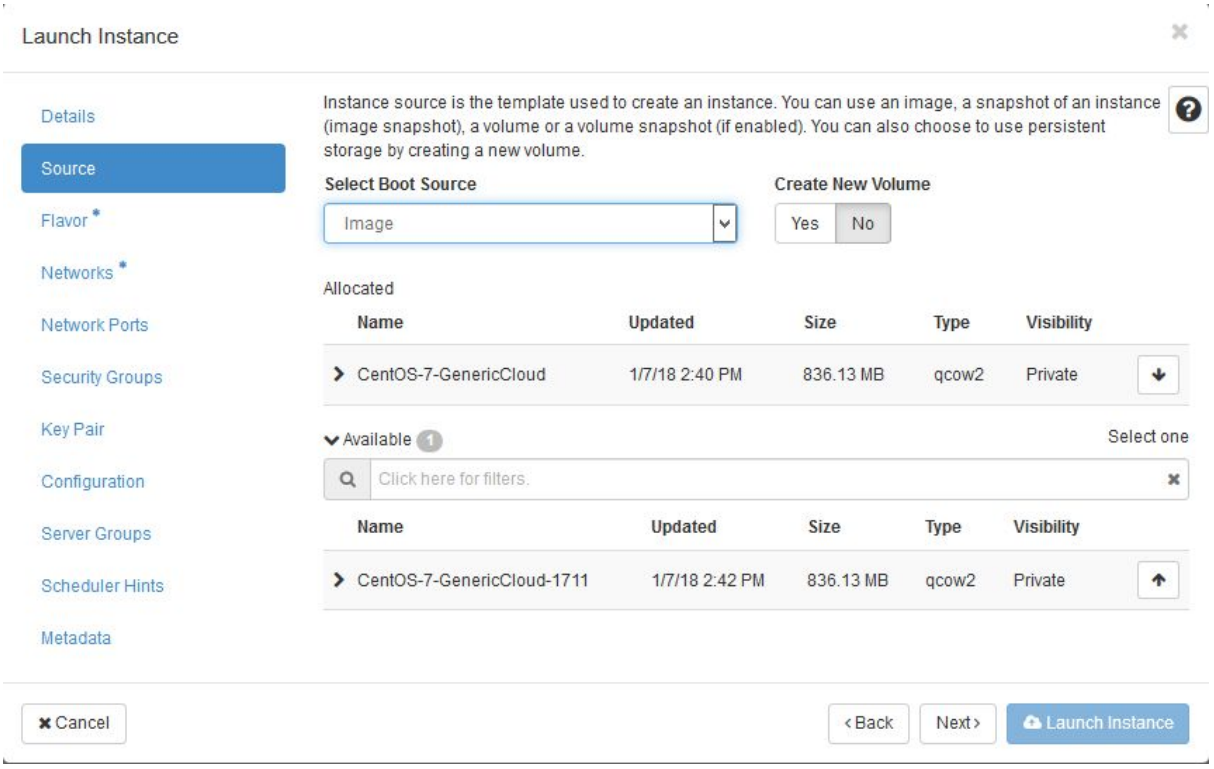
And click on:

And follow the next steps provided by the wizard:

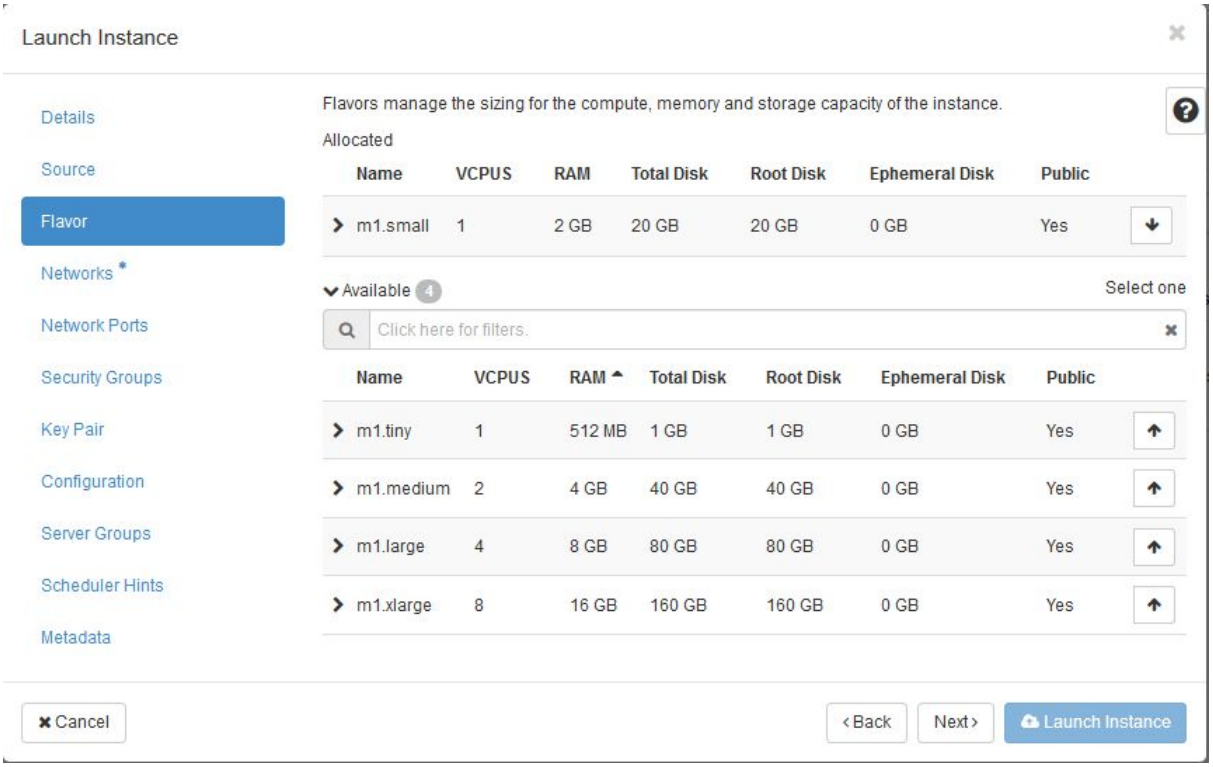
Provide a name for the instance (e.g. centos7-vm2)



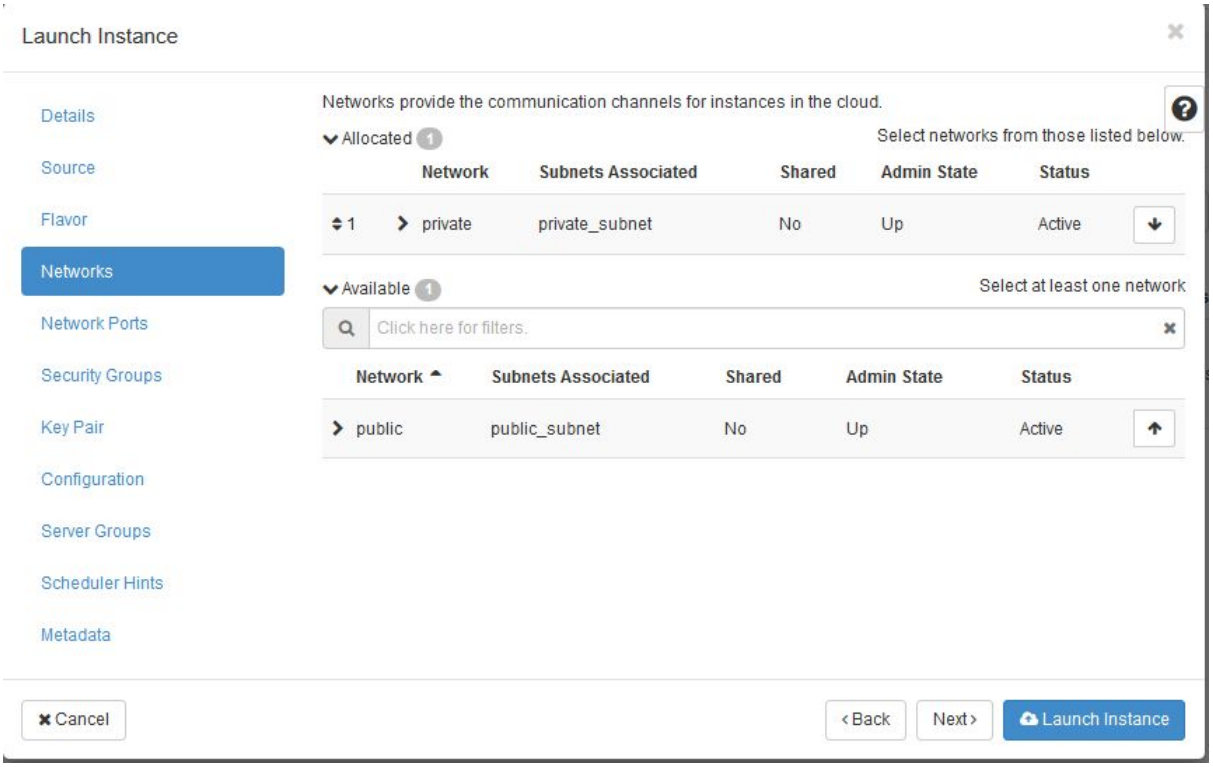
Select the image and select “No” for (create new volume), unless the instance creation will fail! Add the image with the Up arrow under the available images:



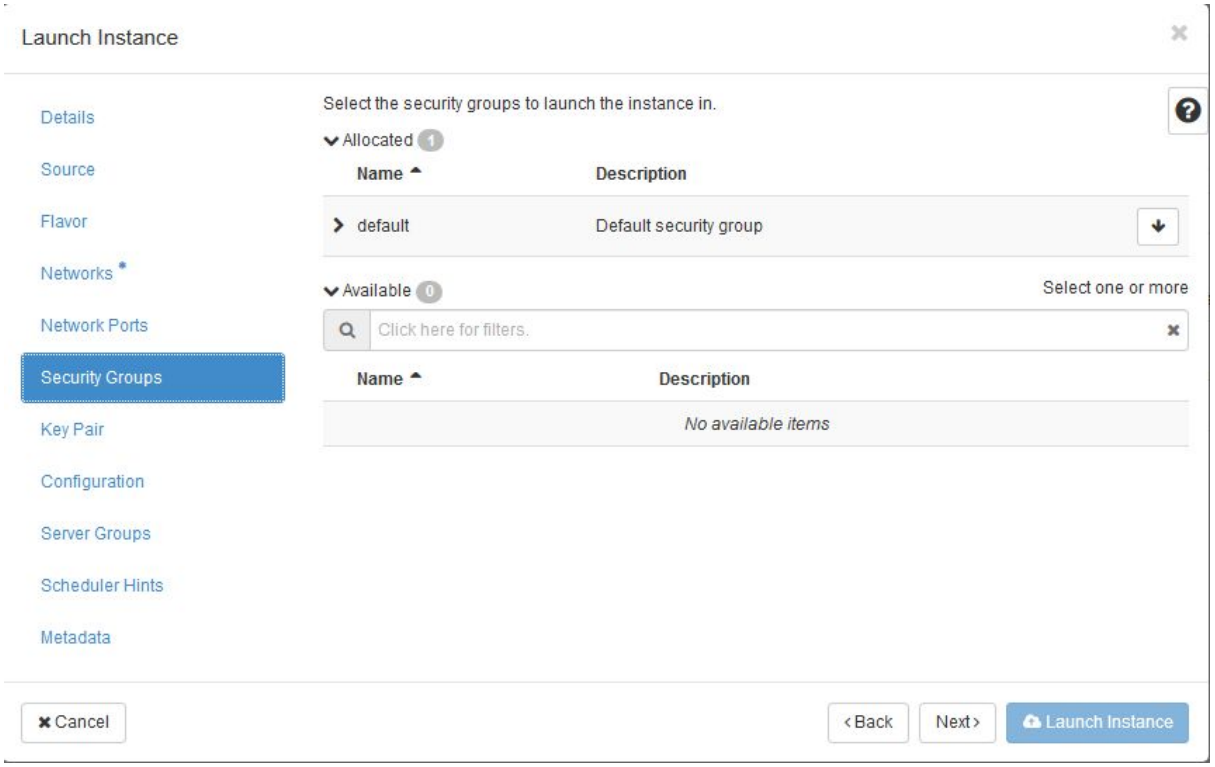
Select the m1.small flavor type, use the up arrow:



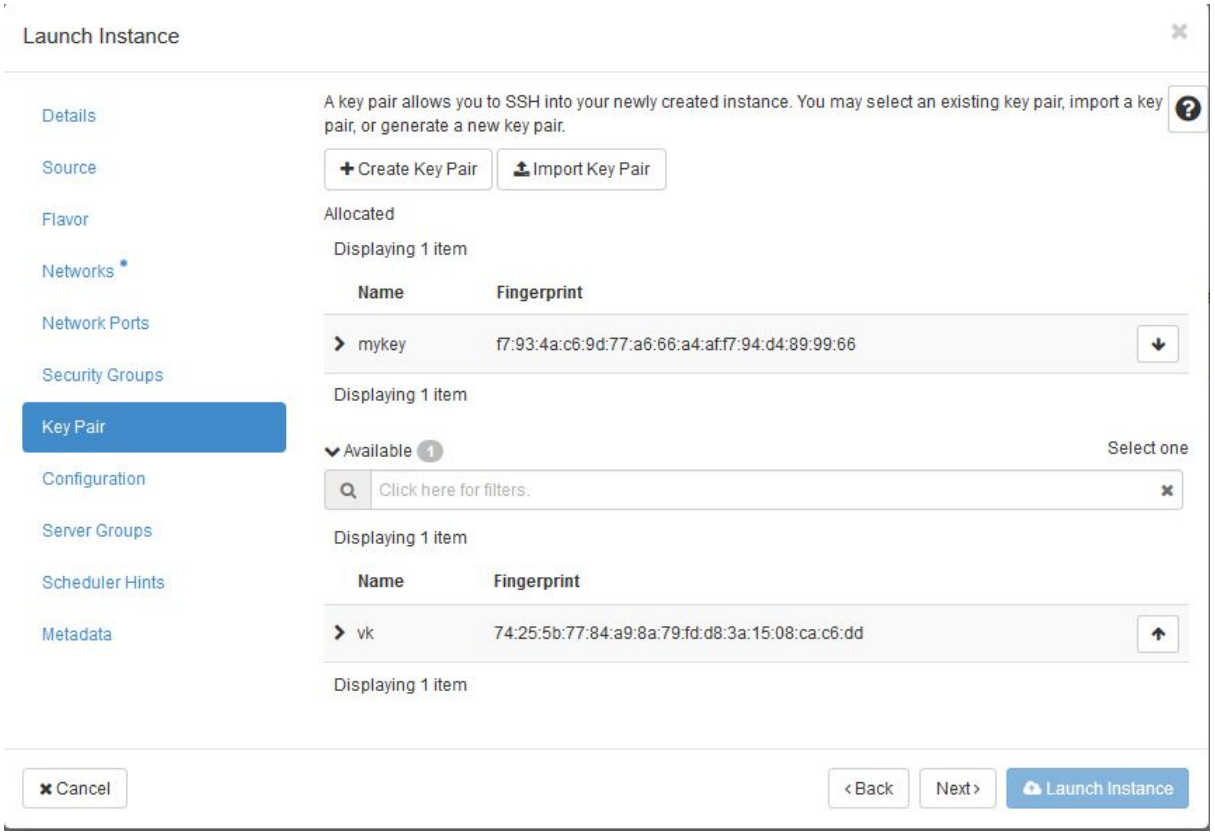
Assign the private network to the instance to use:



Jump over Network ports to the Security Group, the Default Security Group is selected by default (since this is the only security group which was created during the installation):

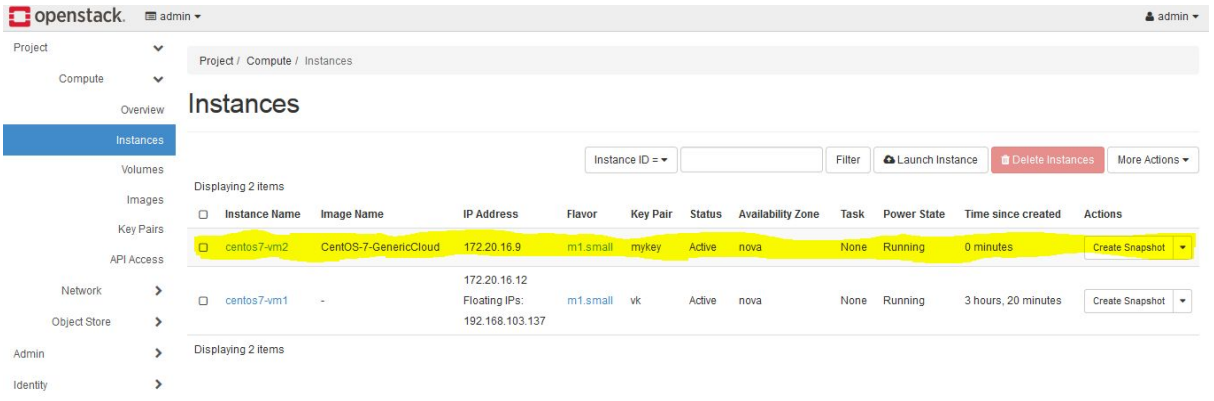


Our key will be selected by default as well:

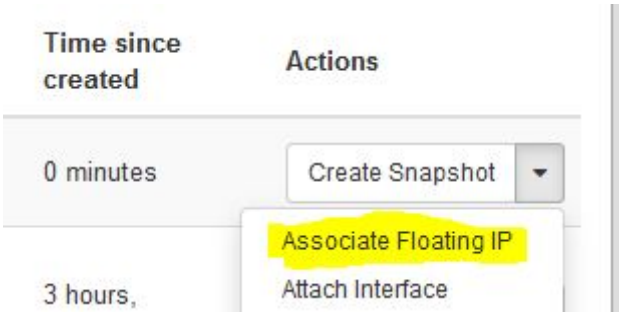


We can jump over the next steps and launch our instance by clicking on:

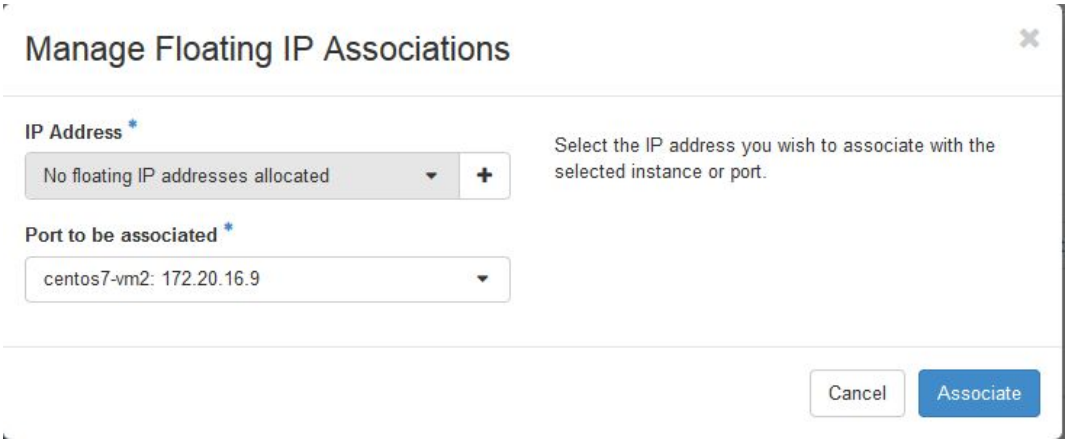
After few seconds we'll see that our instance is running:



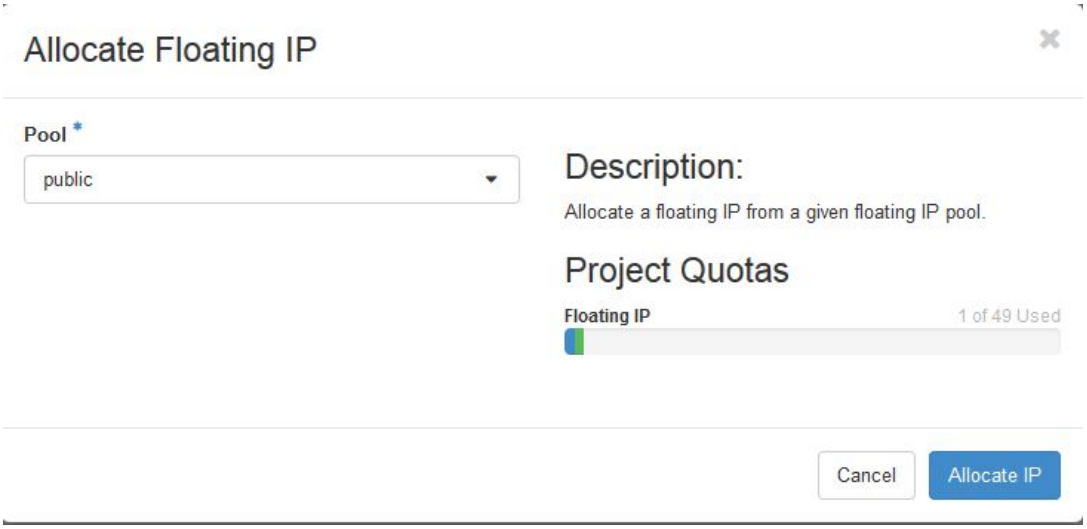
Now we can click on "Associate Floating IP"



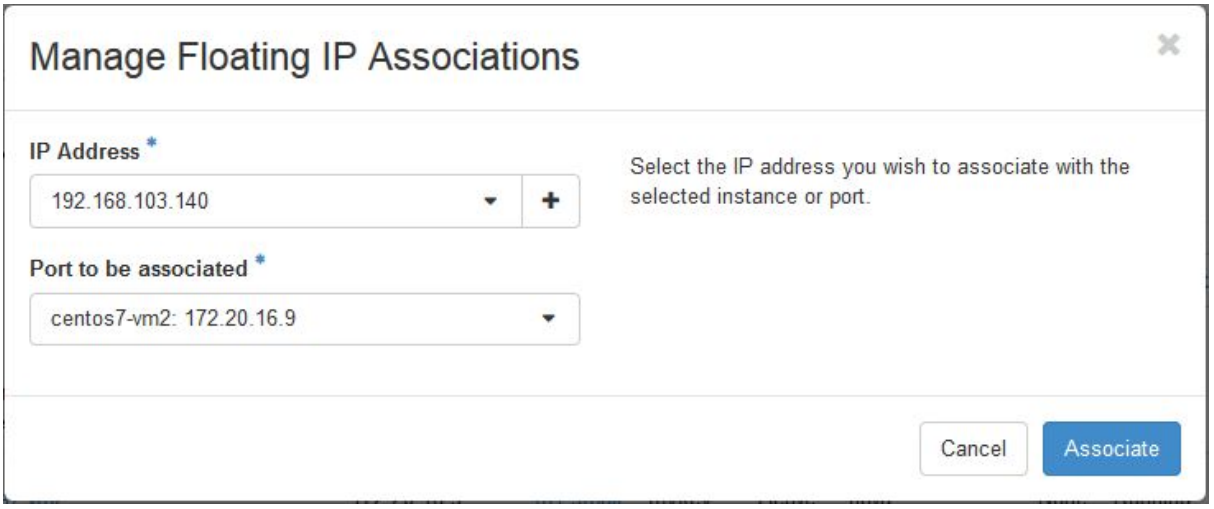
Click on the  icon:



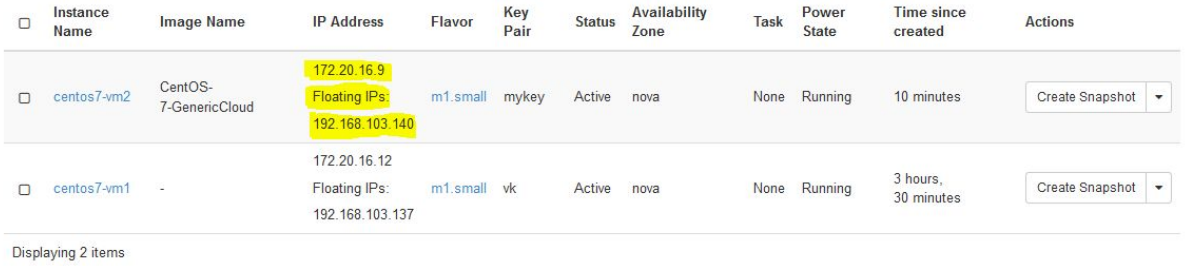
Click on “Allocate IP”



A floating IP will be created from the pool which we defined during the public net creation. And finally click on “Associate”



Now we can see that the floating IP is assigned to our instance:

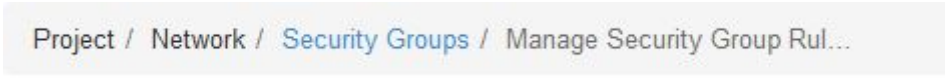


Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
centos7-vm2	CentOS-7-GenericCloud	172.20.16.9 Floating IPs: 192.168.103.140	m1.small	mykey	Active	nova	None	Running	10 minutes	Create Snapshot
centos7-vm1	-	172.20.16.12 Floating IPs: 192.168.103.137	m1.small	vk	Active	nova	None	Running	3 hours, 30 minutes	Create Snapshot

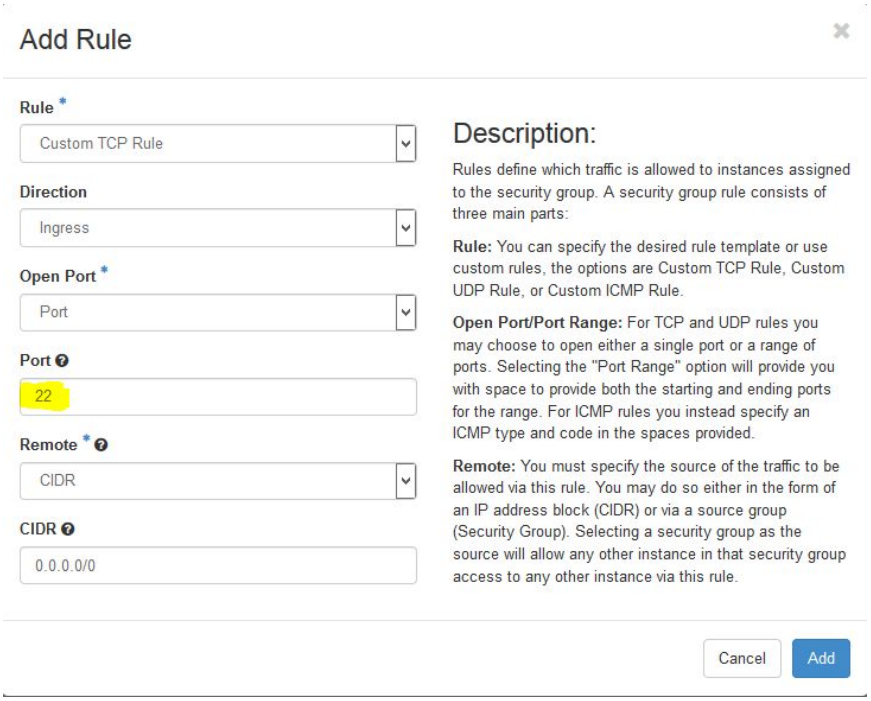
Displaying 2 items

Note: If you do not have floating IP addresses for our training in tracing.local, we can not use the external address for ssh! In our example, we using the private IP address of our instance, as shown below.

First we need to allow SSH through port 22 by defining the ingress rule in our default security group by navigating to (Project → Network → Security Groups):



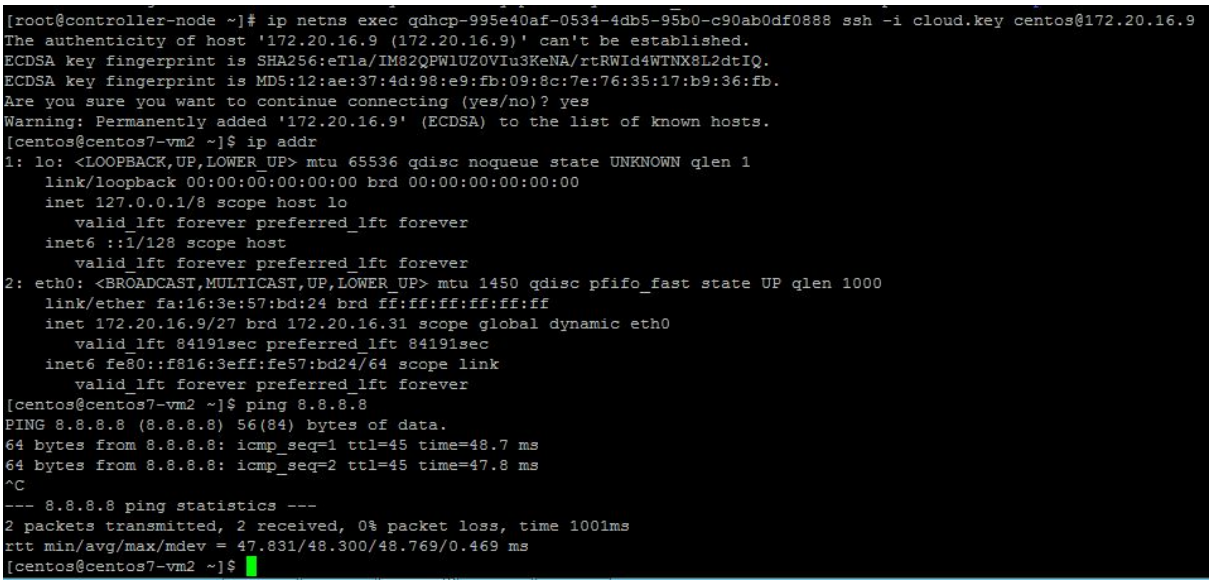
And add a new rule as follow:



Now on the controller we can examine with “ip netns” the dhcp namespace:

[root@controller-node ~]**# ip netns**
qrouter-ac6fa1b8-d1f3-4b9b-8702-cb66596ee416
qdhcp-995e40af-0534-4db5-95b0-c90ab0df0888

And ssh into the instance with our private key “cloud.key” and the default user “centos”:
[root@controller-node ~]**# ip netns exec qdhcp-995e40af-0534-4db5-95b0-c90ab0df0888 ssh -i cloud.key centos@172.20.16.9**
The authenticity of host '172.20.16.9 (172.20.16.9)' can't be established.
ECDSA key fingerprint is SHA256:eTla/IM82QPWlUZ0VIu3KeNA/rtRWId4WTNX8L2dtIQ.
ECDSA key fingerprint is MD5:12:ae:37:4d:98:e9:fb:09:8c:7e:76:35:17:b9:36:fb.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.20.16.9' (ECDSA) to the list of known hosts.
[centos@centos7-vm2 ~]$



Thanks!
Victor Kalinichenko