

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

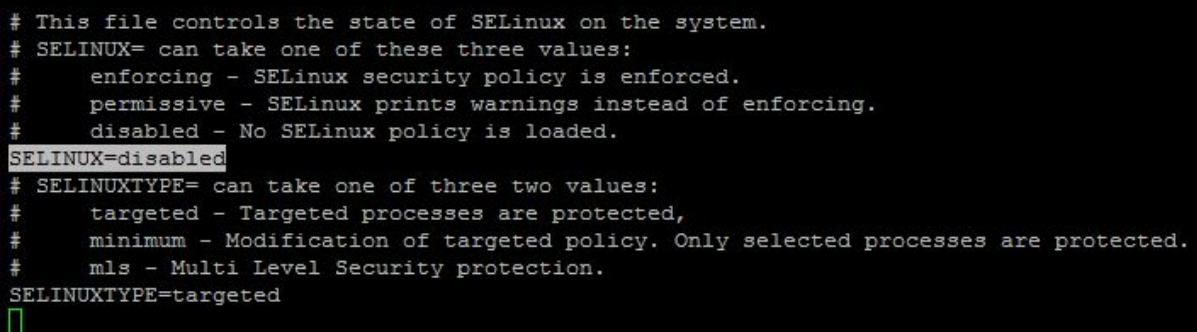
Configuring hosts names using hostnamectl

```
# 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=nsver-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 [ DONE ]
192.168.103.234_controller.pp: [ DONE ]
Applying 192.168.103.212_network.pp [ DONE ]
192.168.103.212_network.pp: [ DONE ]
Applying 192.168.103.228_compute.pp [ DONE ]
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 t
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 keystone_admin file:

cat keystone_admin

```
[root@controller-node ~]# cat keystone_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 keystone_admin file:

source keystone_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 CIDR 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
```

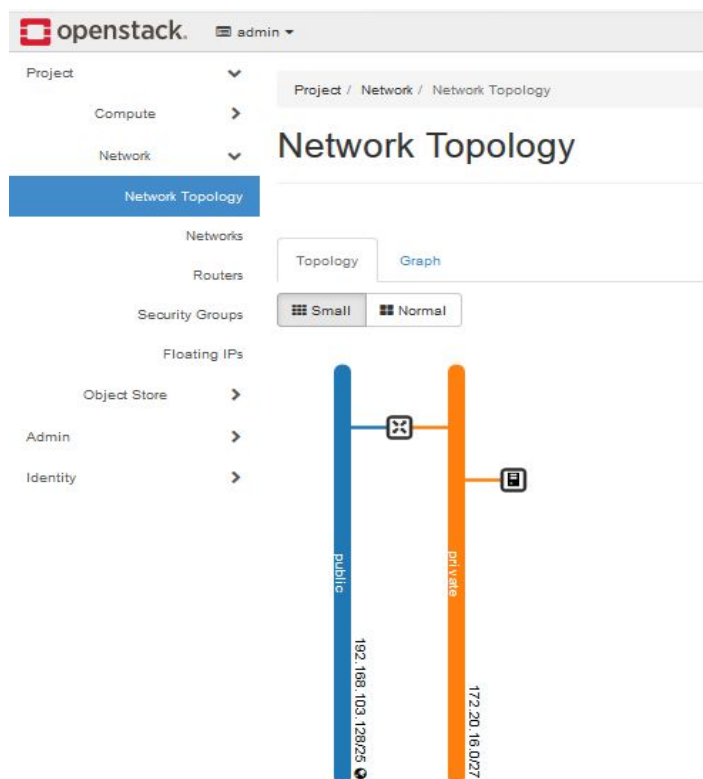
Set the gateway for this router to the public network:

```
# 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
```

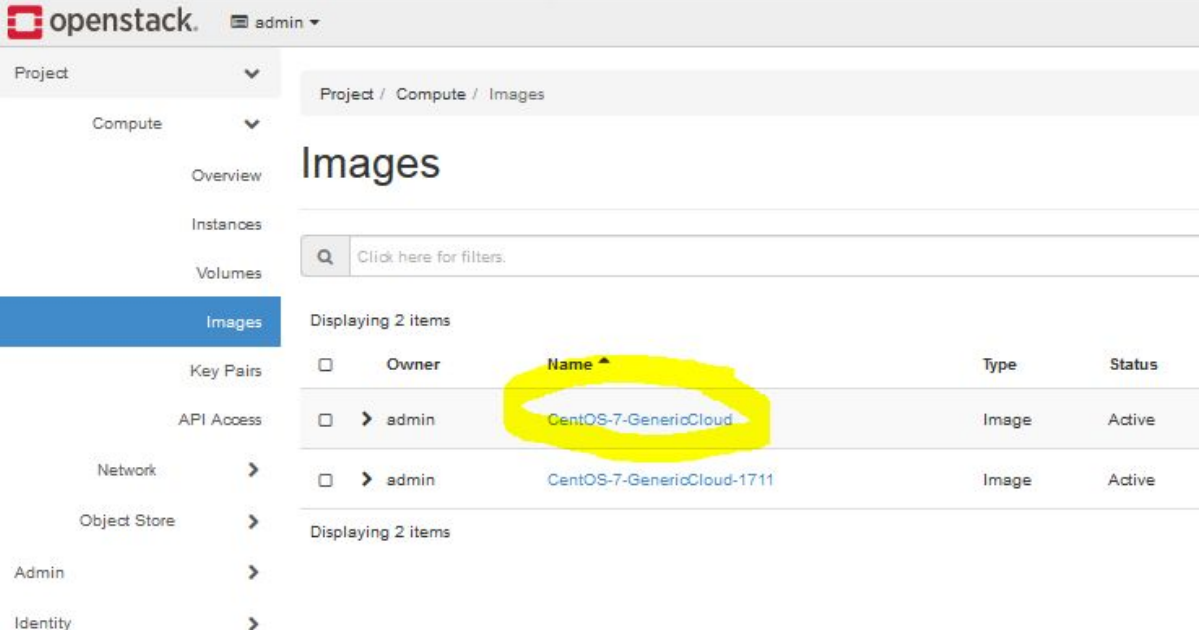
```
# xz -d CentOS-7-x86_64-GenericCloud.qcow2.xz
```

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:



The screenshot shows the OpenStack Horizon dashboard interface. The left sidebar contains navigation links: Project, Compute, Overview, Instances, Volumes, Images (highlighted), Key Pairs, API Access, Network, Object Store, Admin, and Identity. The main content area is titled 'Images' and shows a table of images. The table has columns for Owner, Name, Type, and Status. Two images are listed: 'CentOS-7-GenericCloud' and 'CentOS-7-GenericCloud-1711'. The first image is highlighted with a yellow circle.

Owner	Name	Type	Status
admin	CentOS-7-GenericCloud	Image	Active
admin	CentOS-7-GenericCloud-1711	Image	Active

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 Key Pair

(create keypair and download the private key):

Import Key Pair

Key Pair Name *

mykey

Public Key *

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDZI
7MPBW1z0i93B9XAJ7D5EOyQF7lgiBt2RAgeAY
wHbytQzj97udDNSoy7J6SRqdTP/g5w9wAT0rDO
Z4+0oP4MAEmfUaD9SvLf14k+U2Dhduv+WHPn
5AjmjpgEXI8FuoLDhsx98uiSX1PgtvbjGFOHU4x
be5VEYGPQ1BgHCumDXgY5TwGIVQCYc432n
qubmO
/sIVUf6W8qdnagDnZkUyDWOHorg3I6c092BdKS
2NBL4KPF9DtG6wV+f1ccGB
/waP9hkh3VZp+CJWjwMg4UDqdBmLqMCU43q

Key Pairs are how you login to your instance after it is launched.

Choose a key pair name you will recognise and paste your SSH public key into the space provided.

SSH key pairs can be generated with the ssh-keygen command:

```
ssh-keygen -t rsa -f cloud.key
```

This generates a pair of keys: a key you keep private (cloud.key) and a public key (cloud.key.pub). Paste the contents of the public key file here.

After launching an instance, you login using the private key (the username might be different depending on the image you launched):

```
ssh -i cloud.key <username>@<instance_ip>
```

Cancel

Import Key Pair

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

Projekt → Compute → Instanzen

And click on:

Launch Instance

And follow the next steps provided by the wizard:

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

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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

centos7-vm2

Availability Zone

nova

Count

1

Total Instances (10 Max)

20%

1 Current Usage

1 Added

8 Remaining

Cancel

Back

Next

Launch Instance

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:

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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	
> CentOS-7-GenericCloud	1/7/18 2:40 PM	836.13 MB	qcow2	Private	↓

Available 1

Select one

Q

Click here for filters.

Name	Updated	Size	Type	Visibility	
> CentOS-7-GenericCloud-1711	1/7/18 2:42 PM	836.13 MB	qcow2	Private	↑

Cancel

Back

Next

Launch Instance

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

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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.small	1	2 GB	20 GB	20 GB	0 GB	Yes

Available 4

Select one

Click here for filters.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
> m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
> m1.medium	2	4 GB	40 GB	40 GB	0 GB	Yes
> m1.large	4	8 GB	80 GB	80 GB	0 GB	Yes
> m1.xlarge	8	16 GB	160 GB	160 GB	0 GB	Yes

Cancel

< Back

Next >

Launch Instance

Assign the private network to the instance to use:

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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 > private	private_subnet	No	Up	Active

Available 1

Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
> public	public_subnet	No	Up	Active

Cancel

< Back

Next >

Launch Instance

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):

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Select the security groups to launch the instance in.

▼ Allocated 1

Name	Description
> default	Default security group

▼ Available 0

Select one or more

Q

Click here for filters.

×

Name	Description
No available items	

✕ Cancel

< Back

Next >

Launch Instance

Our key will be selected by default as well:

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

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.

+ Create Key Pair

Import Key Pair

Allocated

Displaying 1 item

Name	Fingerprint
> mykey	f7:93:4a:c6:9d:77:a6:66:a4:af:f7:94:d4:89:99:66

Available 1

Select one

Q

Click here for filters.

×

Displaying 1 item

Name	Fingerprint
> vk	74:25:5b:77:84:a9:8a:79:fd:d8:3a:15:08:ca:c6:dd

✕ Cancel

< Back

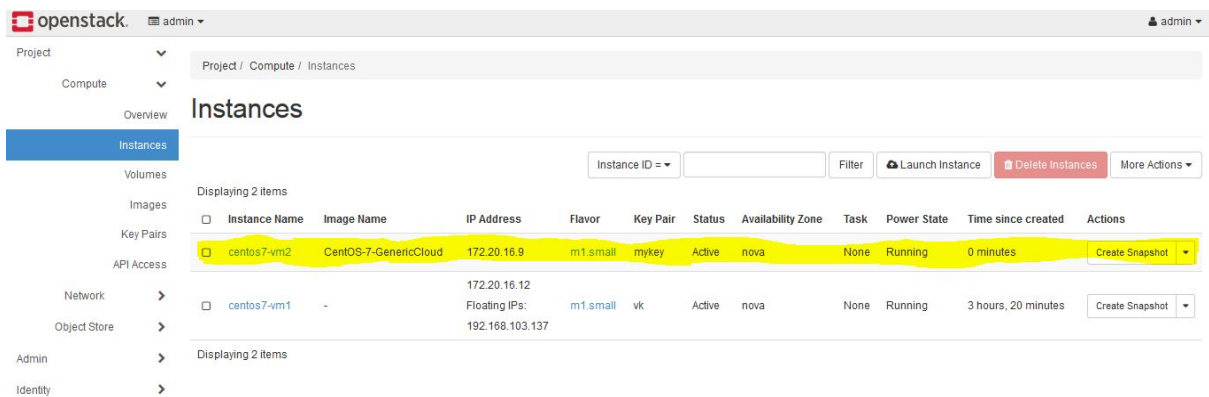
Next >

Launch Instance

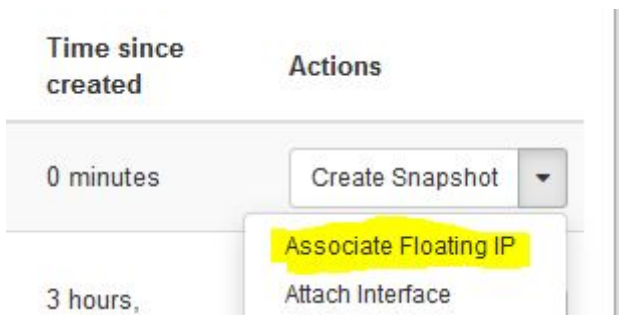
We can jump over the next steps and launch our instance by clicking on:



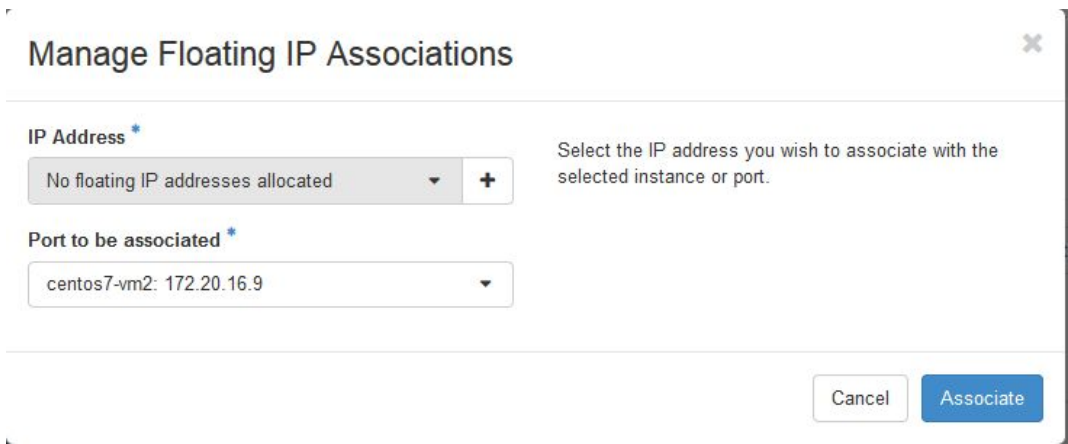
After few seconds we'll see that our instance is running:



Now we can click on “Associate Floating IP”



Click on the  icon:



Click on “Allocate IP”

Allocate Floating IP

Pool *

public

Description:

Allocate a floating IP from a given floating IP pool.

Project Quotas

Floating IP

1 of 49 Used

Cancel

Allocate IP

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

Manage Floating IP Associations

IP Address *

192.168.103.140

+

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

Port to be associated *

centos7-vm2: 172.20.16.9

Cancel

Associate

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

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	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
<input type="checkbox"/>	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):

Project / Network / Security Groups / Manage Security Group Rule...

And add a new rule as follow:

Add Rule

Rule *

Custom TCP Rule

Direction

Ingress

Open Port *

Port

Port ?

22

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.

Cancel

Add

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/IM82QPWlUz0Vlu3KeNA/rtRWId4WTNX8L2dtlQ.

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 ~]$
```



```

[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:eT1a/IM82QPW1U20Viu3KeNA/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 ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP qlen 1000
    link/ether fa:16:3e:57:bd:24 brd ff:ff:ff:ff:ff:ff
    inet 172.20.16.9/27 brd 172.20.16.31 scope global dynamic eth0
        valid_lft 84191sec preferred_lft 84191sec
    inet6 fe80::f816:3eff:fe57:bd24/64 scope link
        valid_lft forever preferred_lft forever
[centos@centos7-vm2 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=45 time=48.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=45 time=47.8 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 47.831/48.300/48.769/0.469 ms
[centos@centos7-vm2 ~]$

```

Thanks!
Victor Kalinichenko