

# RHCSA for Red Hat OpenStack - using OpenStack

## Managing Networks with Neutron

### Networks and subnets

---

List networks under the titan and demo projects

```
source demorc
openstack network list
source titanrc
openstack network list
```

The projects have different network lists, but they both have a *public* network. When you compare the UUIDs, you will find that the *public* networks are the same. The projects share a single network named *public*.

Explore properties of the *public* network. Compare its project ID with the *titan* project's ID.

```
openstack network show public
openstack project show titan
```

The project ID of the *public* network is different from *titan*'s ID. *public* is not owned by *titan*.

Now explore the subnet of *public*. Get the subnet ID from the `network show` command.

```
openstack subnet show SUBNET_ID
```

The command issues an error message "No Subnet found". This is misleading - the subnet does exist. Try again under the admin identity.

First, remove the `OS_PROJECT_ID` variable, which was set by `titanrc` but complicates things here.

```
unset OS_PROJECT_ID
openstack --os-username admin --os-project-name admin subnet show SUBNET_ID
```

You see that this subnet exists and even has a name, *public\_subnet*. The output tells us that its address range (*cidr*) corresponds to the range of floating IP addresses associated with *public*. In contrast to the tenant networks created so far, it doesn't feature a DHCP server.

Set the `OS_PROJECT_ID` variable again. Create a network and add a subnet to it.

```
source titanrc
openstack network create titan-nw
openstack subnet create --network titan-nw --subnet-range 10.100.100.0/24 titan-subnet
openstack network show titan-nw
```

Check that *titan-nw* has a *subnets* property whose value is the ID of *titan-subnet*.

### Ports

---

#### A DHCP port

```
openstack port list
```

You see a single port.

```
openstack port show PORT_ID
```

Have a look at the output.

- **network\_id**: A port is always associated with a network, and only one network.
- **mac\_address**: Networks are layer 2 resources, and ports always have a MAC address.
- **fixed\_ips**: If a subnet (layer 3) is associated with a port, the port also receives an IP address from the subnet range. Both the IP address and the subnet ID are shown in this property.
- **device\_owner** indicates what type of device is connected to the port - a DHCP server in this case.

Neutron created the DHCP server when you created the subnet. If you don't want to have a DHCP server for your subnet, use option `--no-dhcp`.

- **device\_id** identifies the device; in the case of DHCP servers, it contains the network ID.

### A port without IP address

A network can exist without a subnet. A port created on such a network has a MAC address, but not an IP address. Confirm this by creating such a network and port.

```
openstack network create no-subnet
openstack port create --network no-subnet port-no-subnet
openstack port show port-no-subnet
```

Such networks and ports are not very useful, at least in Newton: Nova refuses to launch an instance on a port that has no IP address.

```
openstack server create --image cirros --flavor 1 --nic port-id=PORT_ID server-without-ip
```

You don't need this network and port anymore. Remove them.

```
openstack port delete port-no-subnet
openstack network delete no-subnet
```

### An instance port

Launch a server on titan-nw.

```
openstack network list
openstack server create --image cirros --flavor 1 --nic net-id=ID_OF_TITAN_NW titan-server1
openstack port list
openstack server list
```

There is a new port in the list. Its IP address is identical to the instance's IP address. Display its details.

```
openstack port show PORT_ID
```

*device\_owner compute:None* shows that this is an instance port. *device\_id* is the instance ID.

You can also create a port, then launch an instance on it.

```
openstack port create --network titan-nw titan-port
```

Observe that this port has no *device\_owner* and no *device\_id*, since no device is connected to it. It does however have MAC and IP addresses. Launch a server on it.

```
openstack server create --image cirros --flavor 1 --nic port-id=PORT_ID titan-server2
openstack port show titan-port
```

Now it has *device\_owner* and *device\_id*.

You can list all ports of a certain type like this:

```
openstack port list --device-owner compute:none
openstack port list --device-owner network:dhcp
```

### Ports in the GUI

Log on to the GUI and set the project to **titan**. To see the ports, navigate to **Project->Network->Networks**, click on **titan-nw** and select the **Ports** tab.

### Ports can keep a network alive

Go back to the command line and ensure you have the *titan* identity.

Delete all servers running on *titan-nw*, then try to delete *titan-nw*. This fails with an error message *HttpException: Conflict*. This is not too useful. Use the debug option for more information.

```
openstack --debug network delete titan-nw
```

The `--debug` option outputs all API requests issued by the openstack client, and the corresponding responses. The output ends with a stack trace. Scroll back about 30 lines until you see the response to the last API request. It looks similar to

```
RESP BODY: {"NeutronError": {"message": "Unable to complete operation on network
82bd7771-1cbd-467a-b2f4-bba85258b852. There are one or more ports still in use on the
network.", "type": "NetworkInUse", "detail": ""}}
```

This message is explicit enough: The network still has ports. While the DHCP port will be removed automatically, since it was created automatically, Neutron won't automatically delete *titan-port*, since you created it manually. Consequently, you have to delete it manually.

```
openstack port delete titan-port
openstack network delete titan-nw
```

## Routers

---

In the titan project, re-create *titan-nw* and *titan-subnet*.

```
openstack network create titan-nw
openstack subnet create --network titan-nw --subnet-range 10.100.100.0/24 titan-subnet
openstack port list
```

Neutron has created a DHCP port again.

You will now set up a router to connect *titan-nw* to public.

Use `openstack router list` to confirm that you have no router. Then `openstack router create titan-router` to create one. Check the result in the GUI's network topology.

Create a router interface connected to *titan-subnet*.

```
openstack router add subnet titan-router titan-subnet
openstack port list
```

Display the details of the new port with `openstack port show PORT_ID`. Notice *device\_owner* and *device\_ID*. This is a router port, and the *device\_ID* is the router's ID.

The router also needs a gateway to *public*. In Newton, the openstack client is unable to create a gateway.

```
neutron router-gateway-set titan-router public
openstack port list
```

The port list hasn't changed. Only an administrator can see a port connected to the external network. The option `--router titan-router` filters the output and only shows ports associated with this router.

```
unset OS_PROJECT_ID
openstack --os-username admin --os-project-name admin port list --router titan-router
openstack --os-username admin --os-project-name admin port show PORT_FOR_GATEWAY
```

The port list command lists two ports, one with an IP address on *titan-nw*, and one with an external IP address. The latter is the gateway port. Notice its *device\_owner*. The *device\_id* is the router.

Reset the environment for the titan project and try to delete the router.

```
source titanrc
openstack router delete titan-router
```

This fails, again because of a "conflict". The `--debug` option clarifies that here again, there are still ports. You need to remove the interface; the gateway will be removed automatically.

```
openstack router remove subnet titan-router titan-subnet'
openstack router delete titan-router
```

## Specifying subnets and IP addresses

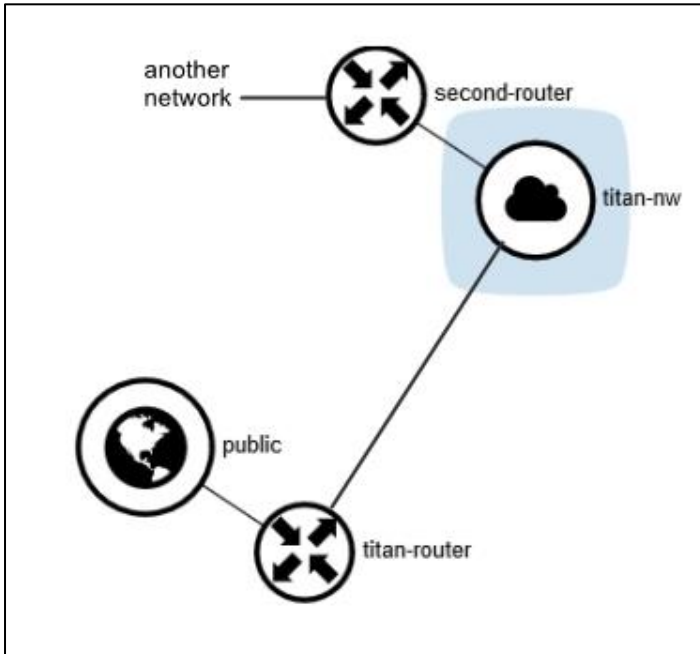
---

Create second subnet for titan-nw.

```
openstack subnet create --network titan-nw --subnet-range 10.200.200.0/24 subnet-200
openstack port list
```

Note that the DHCP port is now associated with both subnets and has two IP addresses - one per subnet.

Imagine that this subnet needs to be connected to the outside world and also to another network in the background. These two connections require two routers. For example like in this diagram:



Therefore, create the two routers and connect them to the titan-200 subnet.

```
openstack router create second-router
openstack router add subnet second-router titan-200
openstack router create titan-router
openstack router add subnet titan-router titan-200
```

The last command fails. Again, `--debug` helps understanding the problem: "IP address 10.200.200.1 already allocated in subnet ed6a6f6d-14e2-4b48-b238-1db9ae4ffefc"

titan-200 has contributed address 10.200.200.1 to second-router, and Neutron doesn't automatically use another address for titan-router. You have to do that explicitly by creating a port on titan-200 and connect titan-router to the port instead of the subnet.

```
openstack port create titan-router-port --network titan-nw --fixed-ip subnet=titan-200
```

Since titan-nw has two subnets, you have to tell Neutron which subnet you want. Now you can connect the router.

```
openstack router add port titan-router titan-router-port
```

Check the result on the network topology screen.

When you want to launch an instance on a specific subnet, you can do the same thing - first create a port on that subnet, then launch the instance on the port.

Alternatively, you can ask Neutron to give the instance a specific IP address. Neutron then creates a port with this address launches the instance on this port. Try this out.

```
openstack server create --image cirros --flavor 1 --nic net-id=5877730c-d8f0-498d-8238-a4be3a129cce, fixed-ip=10.200.200.123 myserver
```

The disadvantage of this approach is that you have to find a free IP address. Unfortunately, the openstack

server create command doesn't have a subnet option.

This concludes the Neutron exercises.