CSCI 5380

Network Virtualization and Orchestration

Lab 3

OpenStack: Service-Chaining

University of Colorado Boulder

Department of Computer Science

Network Engineering

Professor Levi Perigo, Ph.D.

# **PART 1: Service chaining**

# Summary:

In this part, you will manage service chaining in OpenStack.

# Section 1: In-network service chaining:

Create the service chaining in Figure 2 according to the lecture instructions.

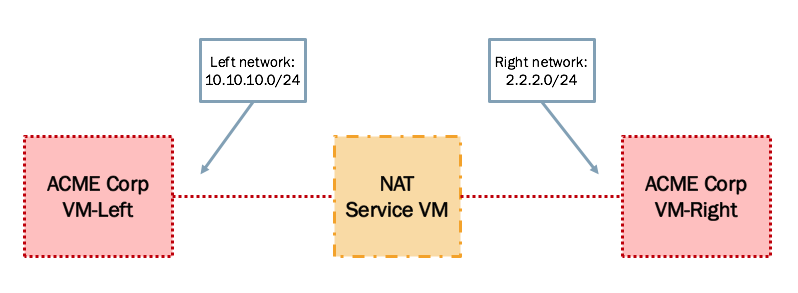


Figure 2. In-Network Service Chaining setup

You can either create your own NAT VM appliance or download and use one from another source such as: [pfSense](https://github.com/OverFlowJAMK/General/wiki/Installing-PFSense-into-OpenStack).

# Section 2: Transparent service chaining:

Add a Layer 2 firewall to the setup in Section 1. The firewall will block SSH protocol on port 22. The setup will look like Figure 3.

You may want to create your own Firewall VM or use the pfSense appliance mentioned in previous section.

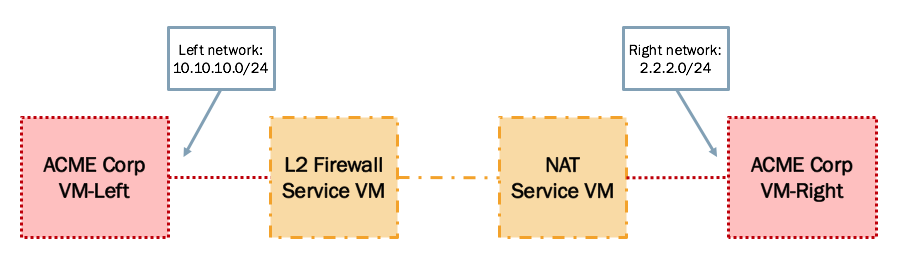


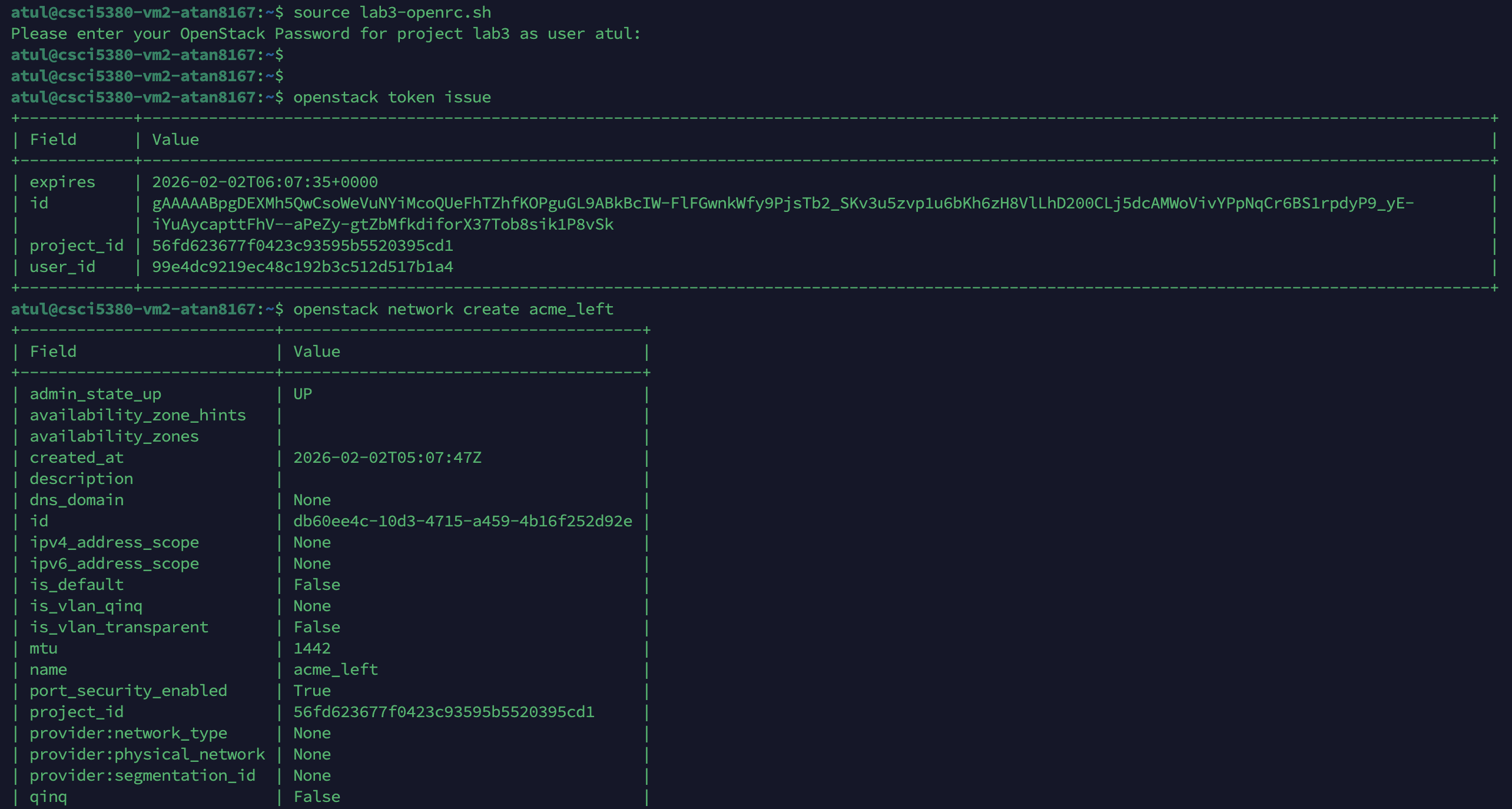
Figure 3. Service Chaining with L2 Firewall

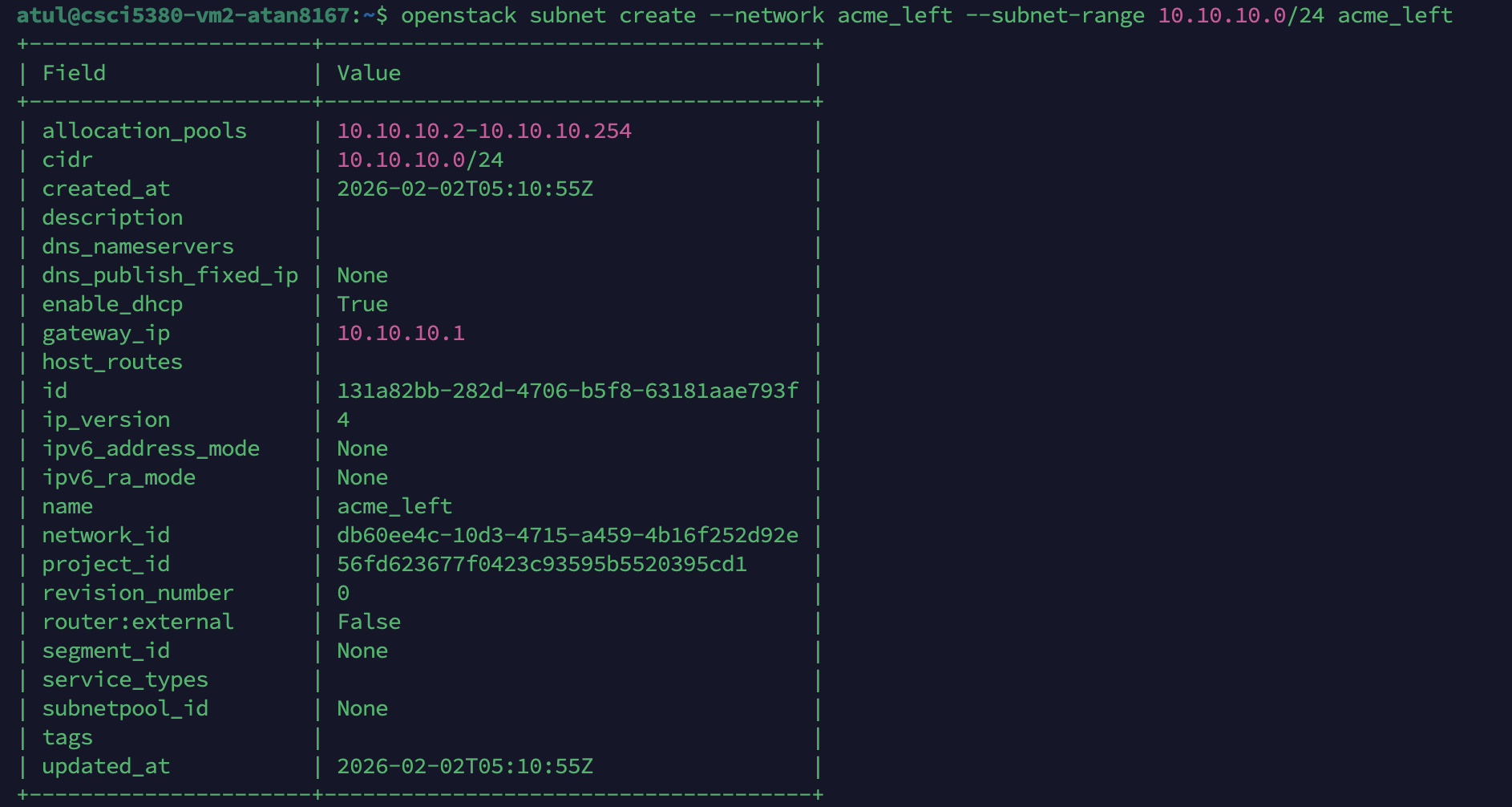
# Deliverable (100 points):

Create a tutorial document indicating the steps to achieve this lab.

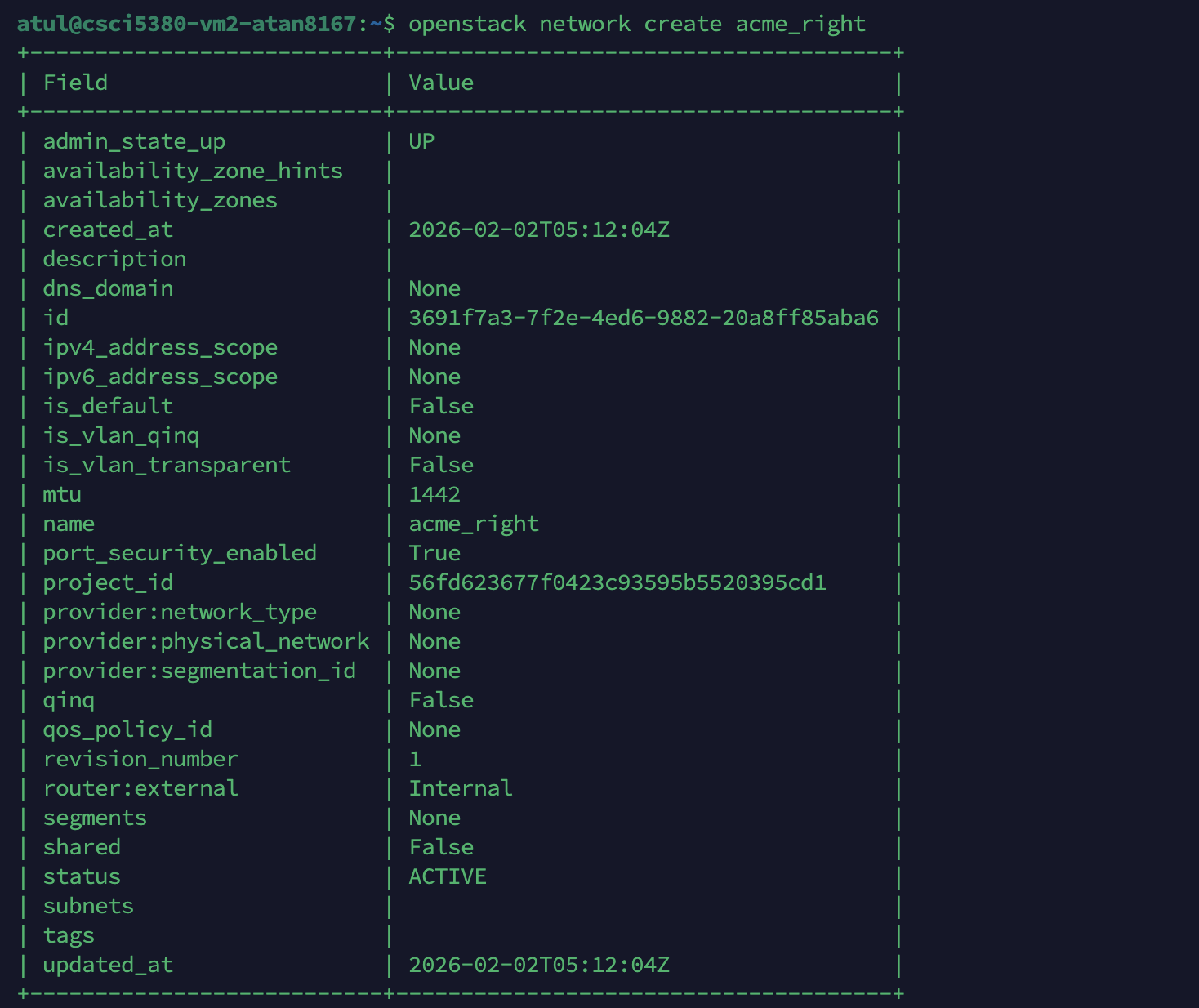
**Creating the Networks –** I have used the CLI to create the network for Left and Right networks –

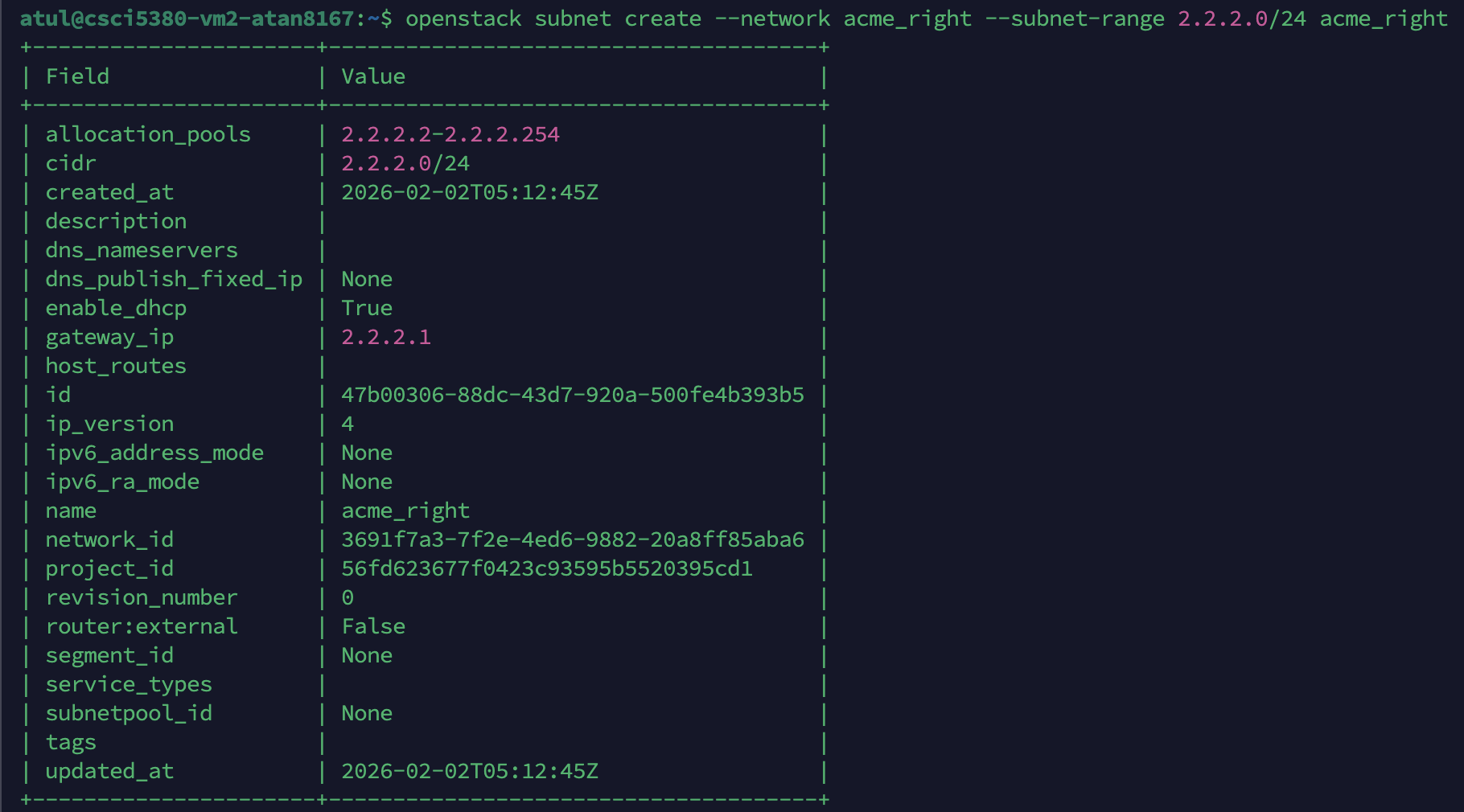
**Acme\_Left –**

****

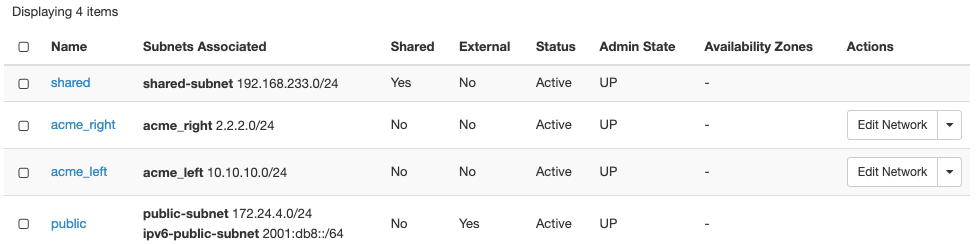
****

**Acme\_Right–**

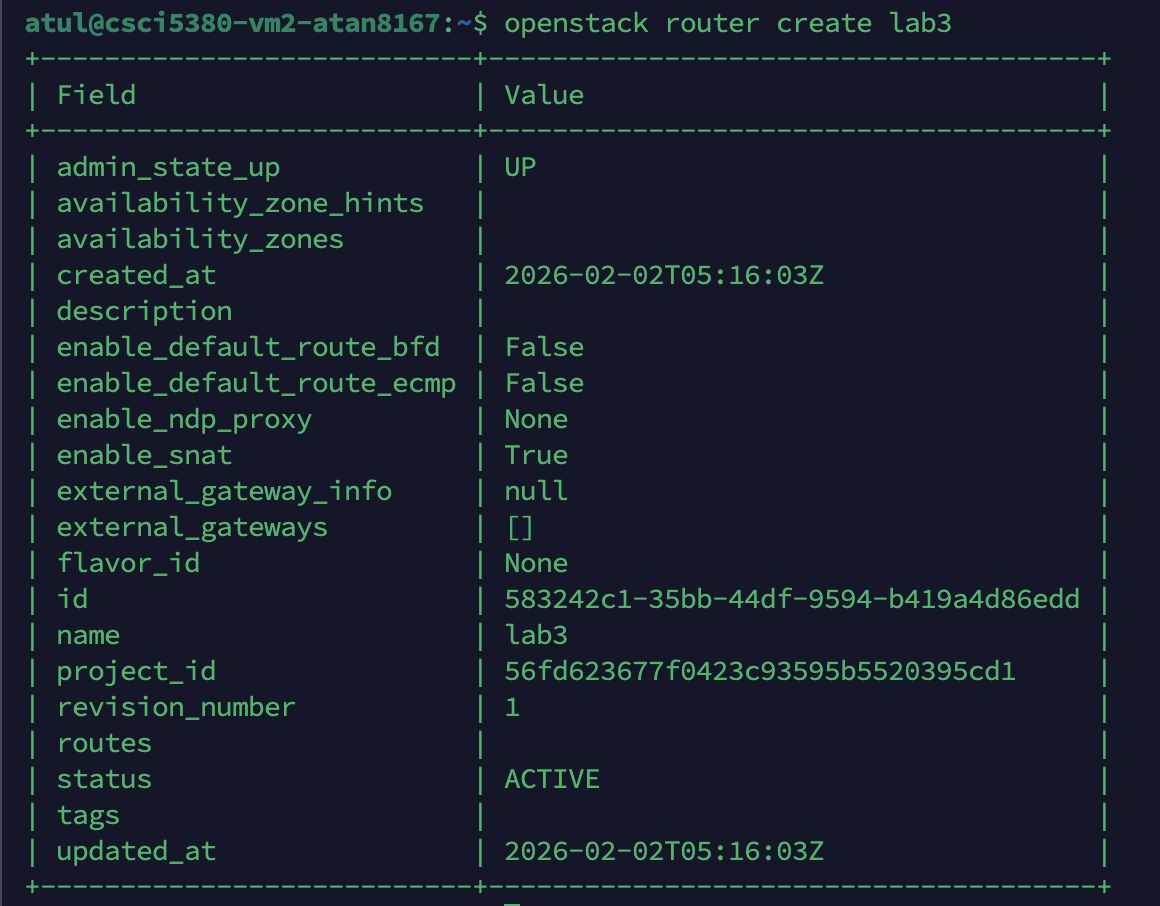
****

****

The same can be verified on UI as well.

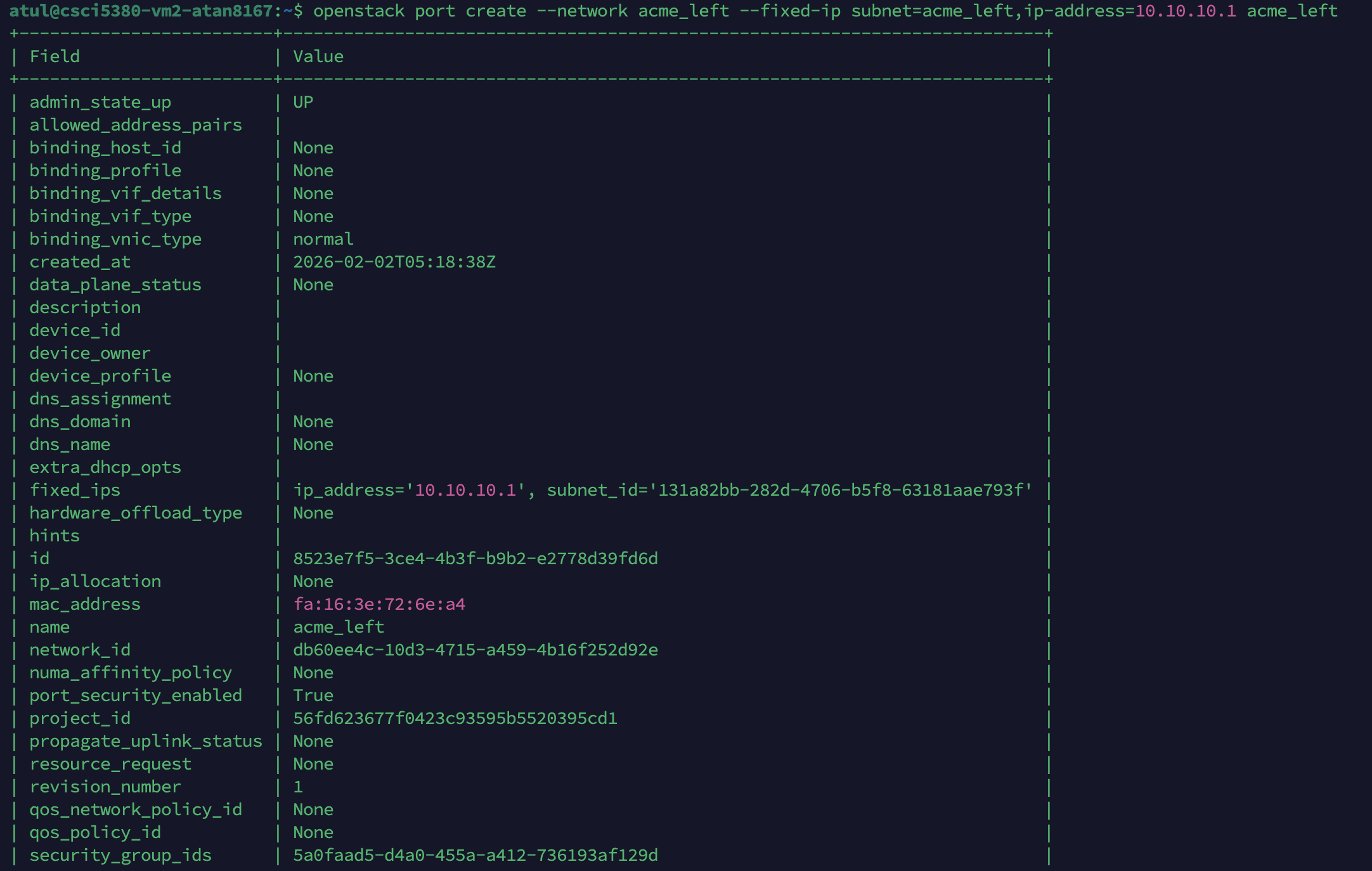
****

**Now, I will create the router with the ports for left and right ACME VM’s-**

****

**The ports for left and right of the router are created as follows –**

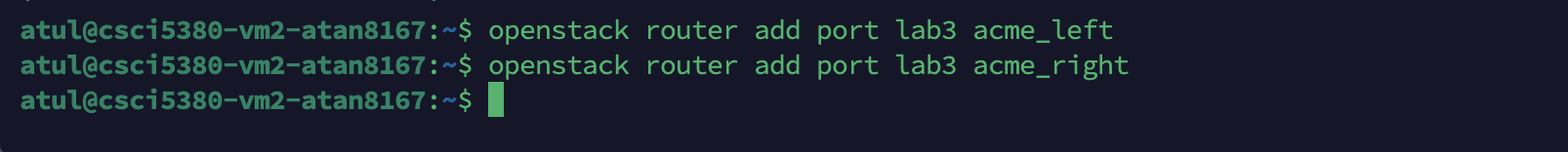
**Acme\_Left Port -**

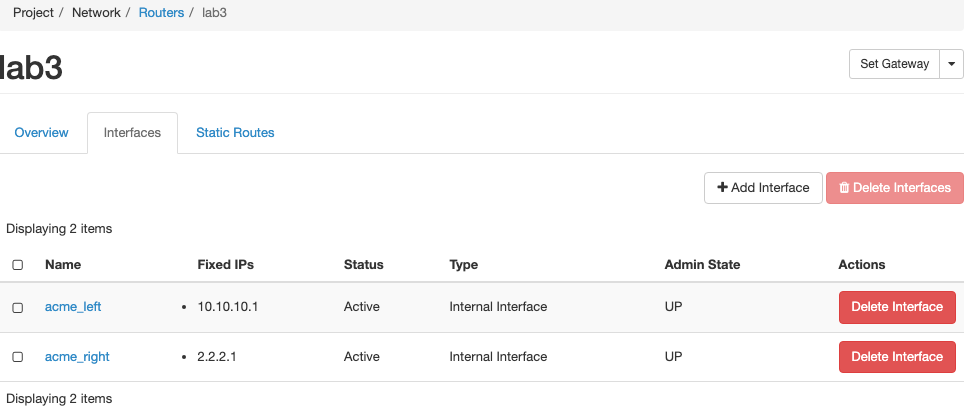


**Acme\_Right Port –**



Ports are added to the router –



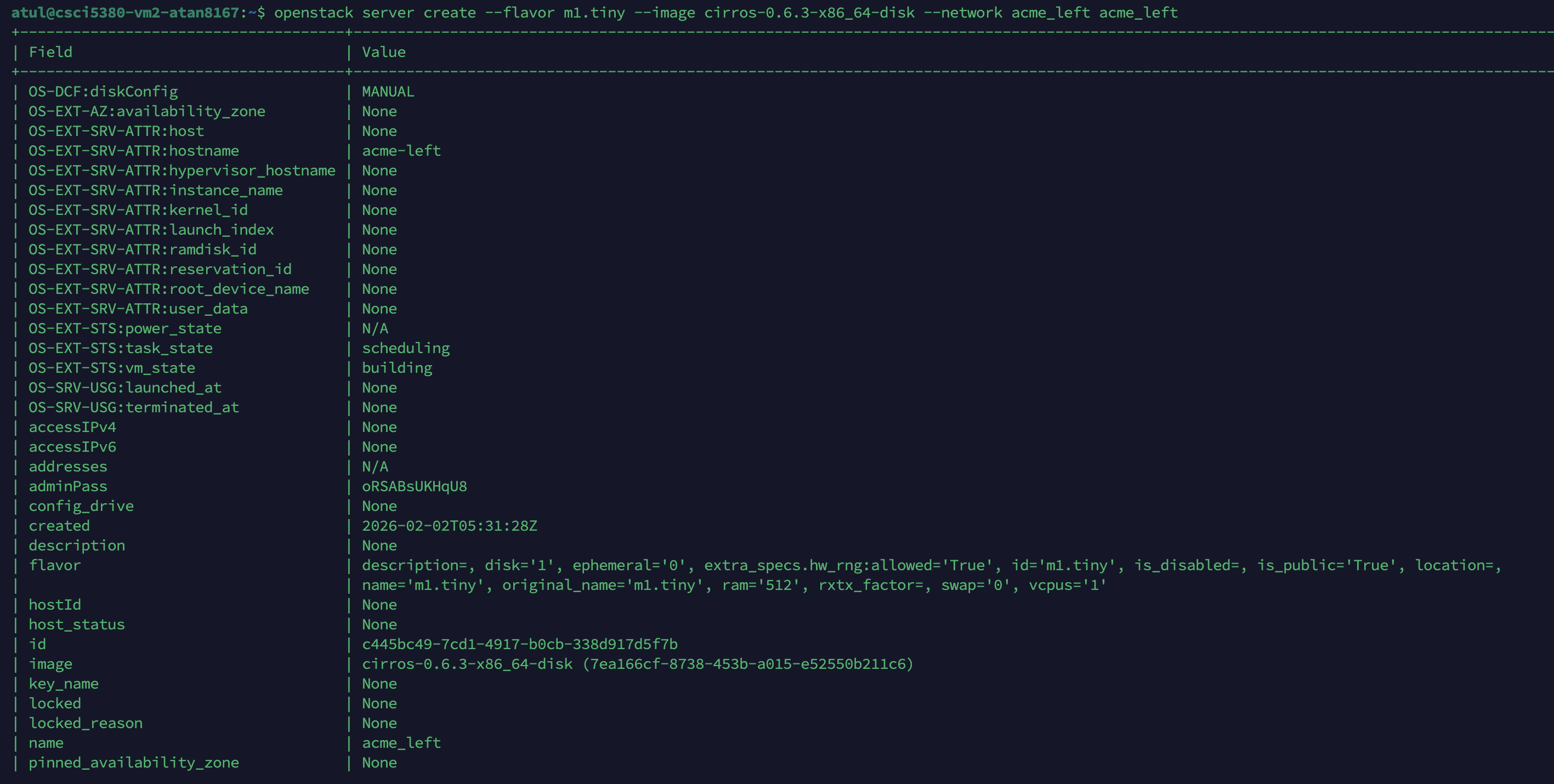


**External Port are set on router –**

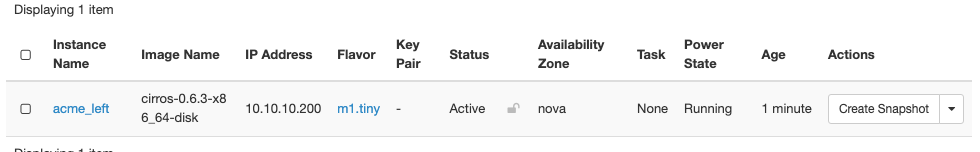
Used UI to map the router lab3 to add to public network and enabled the SNAT –



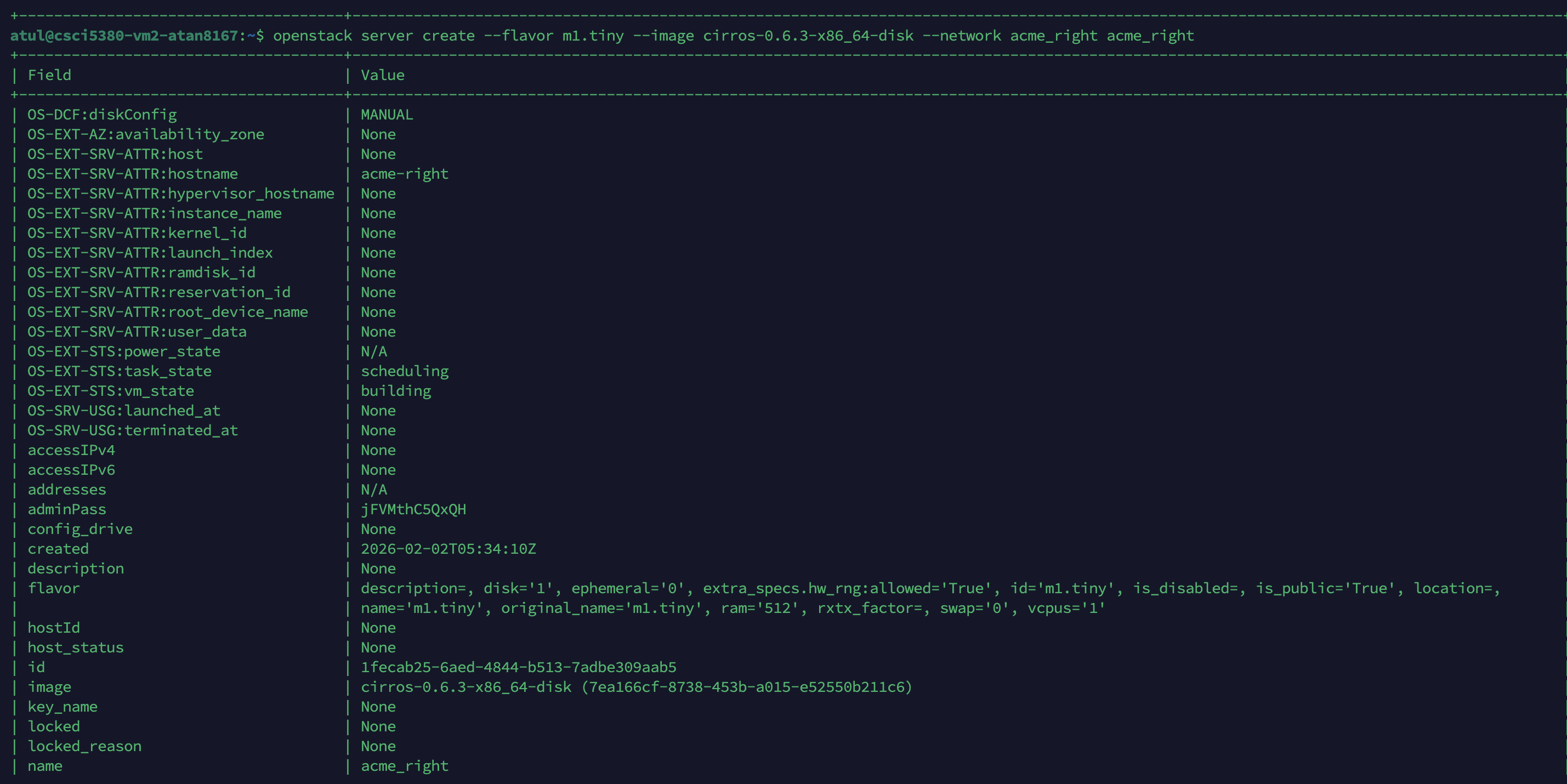
Created the VM using Cirros instance for Acme Left and passed the network flag to allow nova for auto creation of the port –



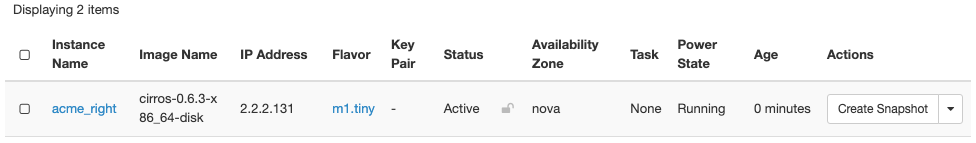
**Verify using UI and I can see IP is picked up from correct Network -**



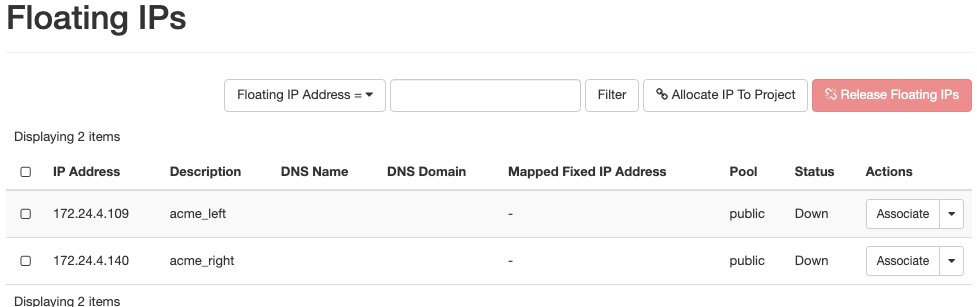
Similarly, Now I will create the VM for the Acme Right network –



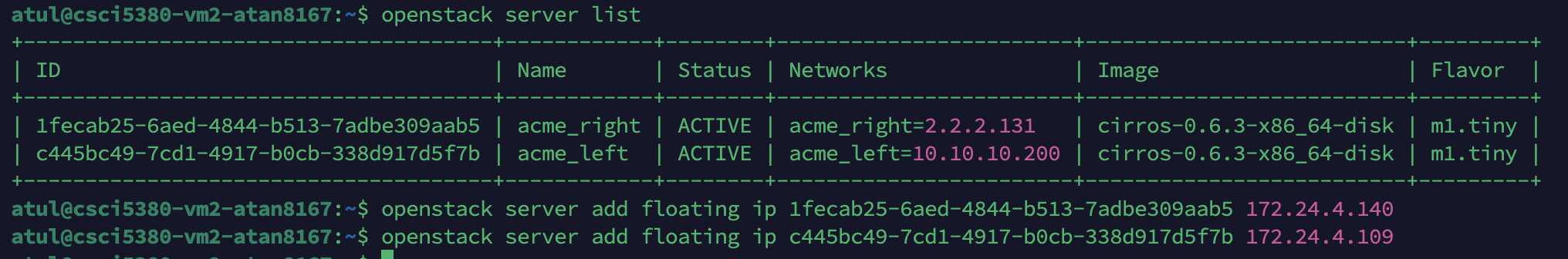
Verified using UI –



Assigning the floating IP address to the VM’s –

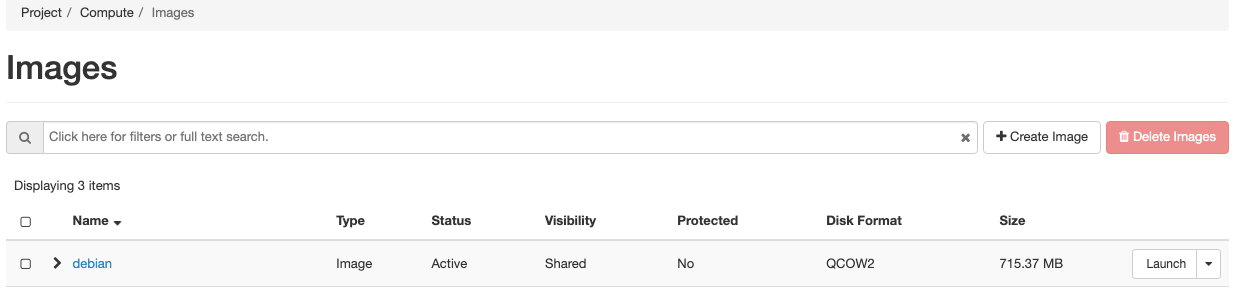


Assigned it using CLI (Since this lab does not require outside connectivity – there is no need for floating IP address) -

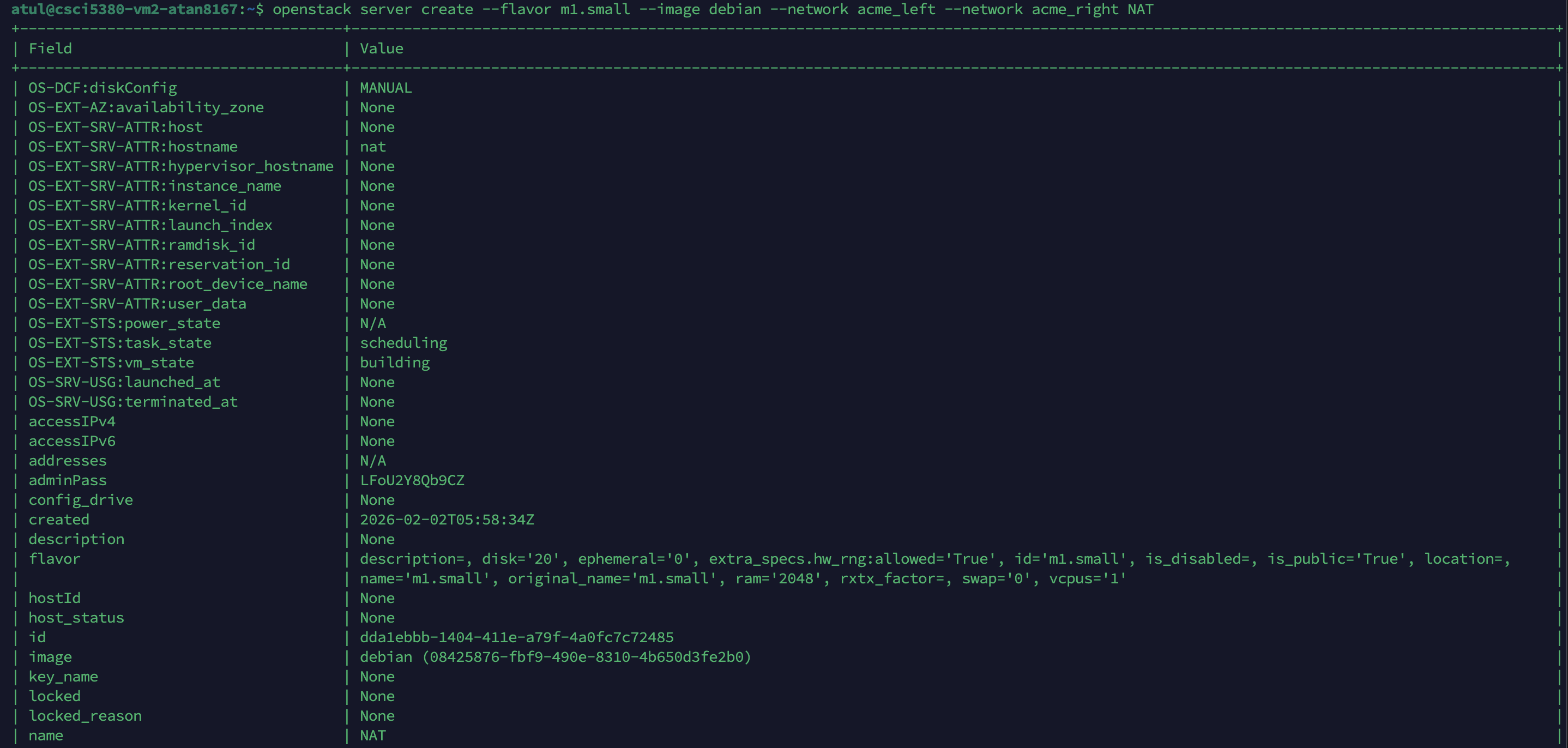


For NAT VM – I will be using the Debian Image which will allow me to create the IP Tables rules for the NAT for left and right VM’s –

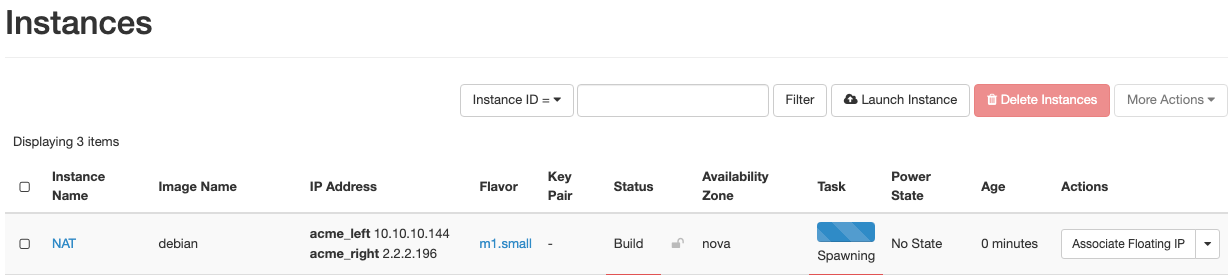
I downloaded the image from the internet and uploaded under projects >> Images –



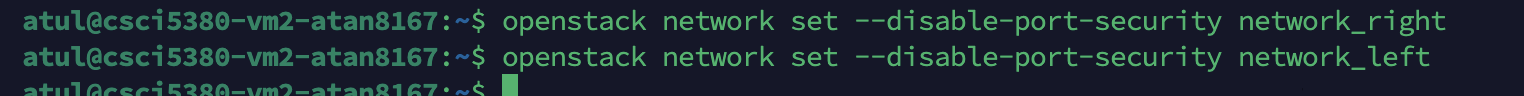
Created an NAT instance form the image by using CLI –



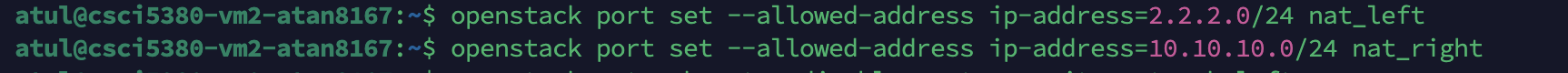
**I can see by using Network Flag the ports IP are auto selected –**



I edited the names to avoid confusion for networks and ports – disabled the port security on both the subnets –



Since by default, Neutron only allows the IP assigned on the interface to be able to send and receive traffic, I am allowing the IP to ensure NAT works as expected –

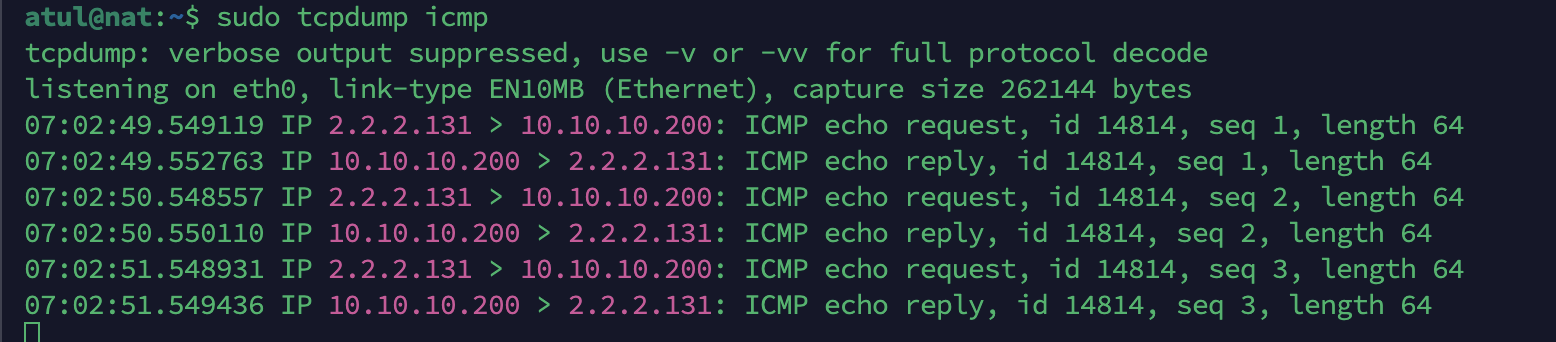


I set the default gateway on each of the VM to the interface IP of NAT instance where I enabled IP forwarding and I could see both VM from left to right can ping each other –

**Packet capture on vm\_left**

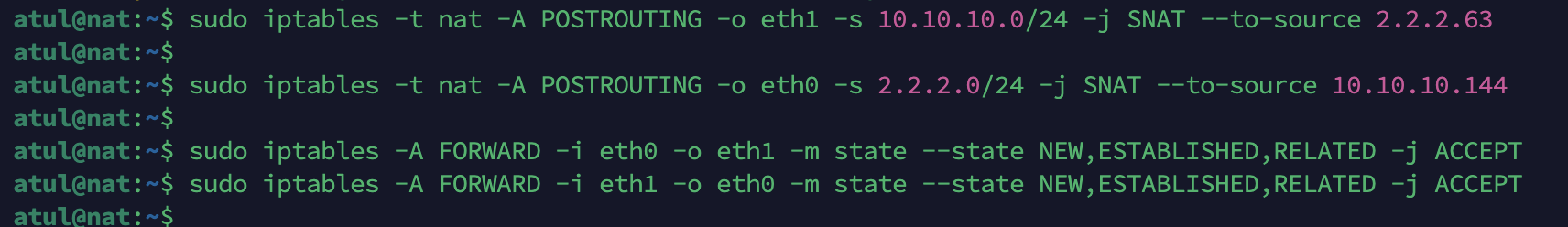


**Packet capture on NAT instance showing packet traversing the packets via its interfaces –**



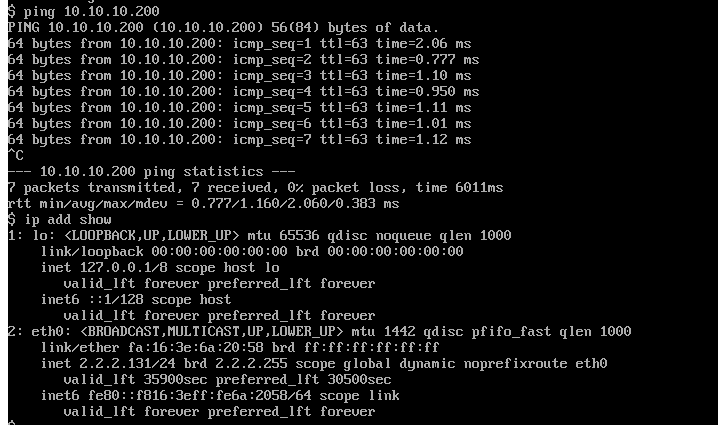
I would enable now the Nat rules using IP tables on the NAT instance which would ensure these packets are using NAT to ping from left to right.

IP tables to configure the SNAT -

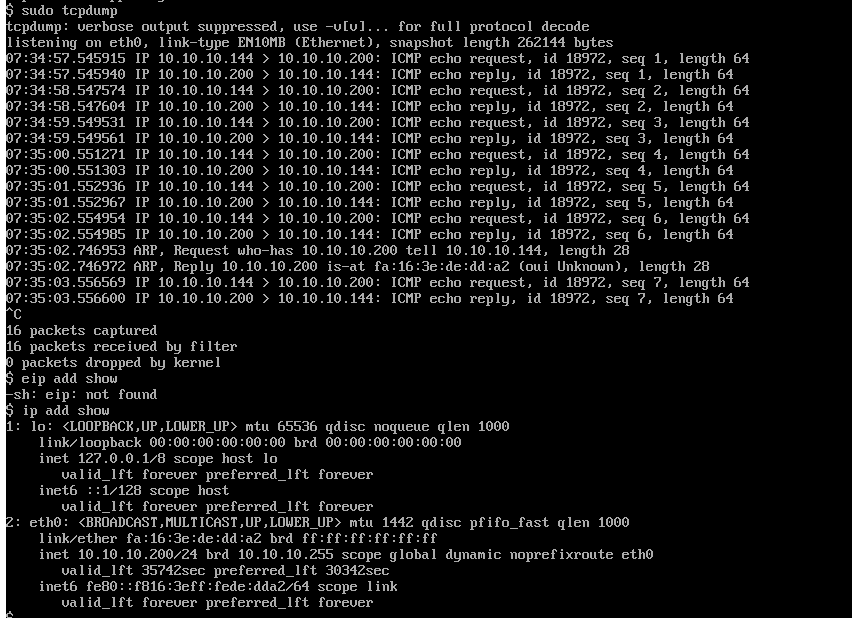


Once the NAT table rules are created and Port security is disabled on the VM instances –

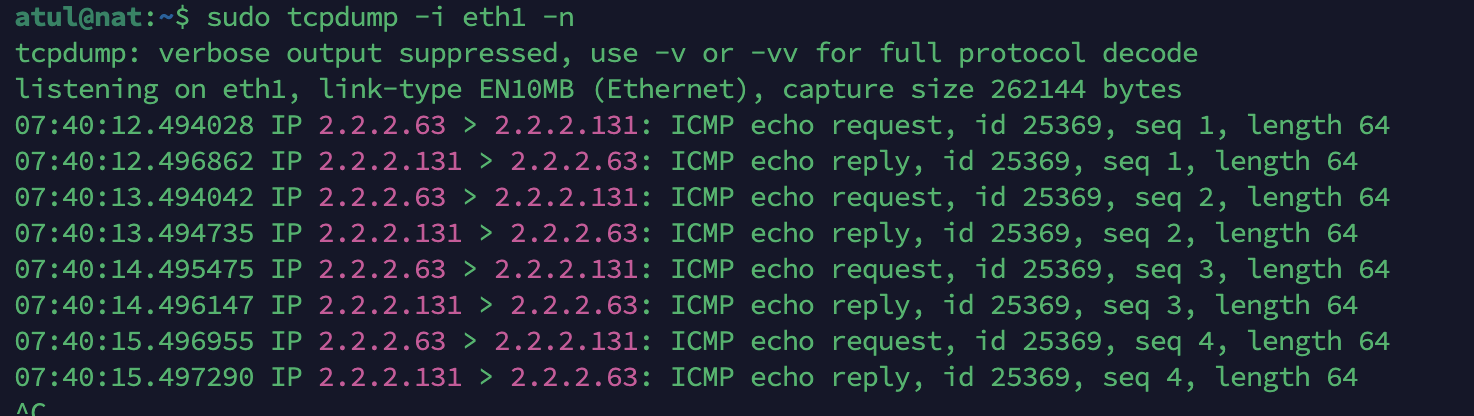
Ping started from VM\_right from 2.2.2.131 to 10 network –



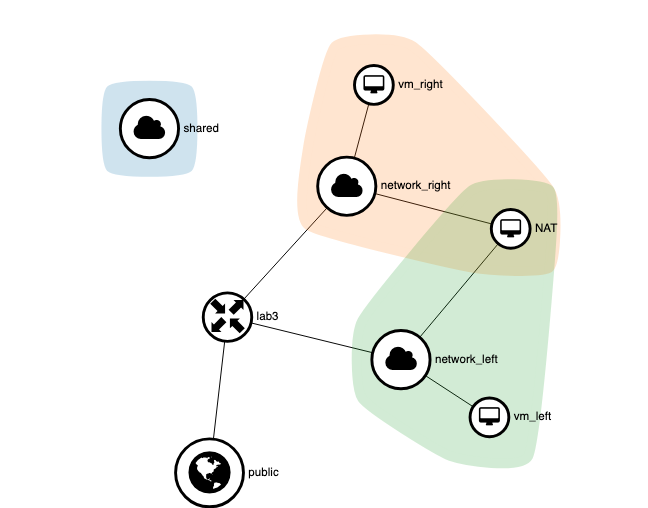
**Packet capture on the VM\_left – This shows the packet is coming from the NAT device IP which is in the same network**



NAT –



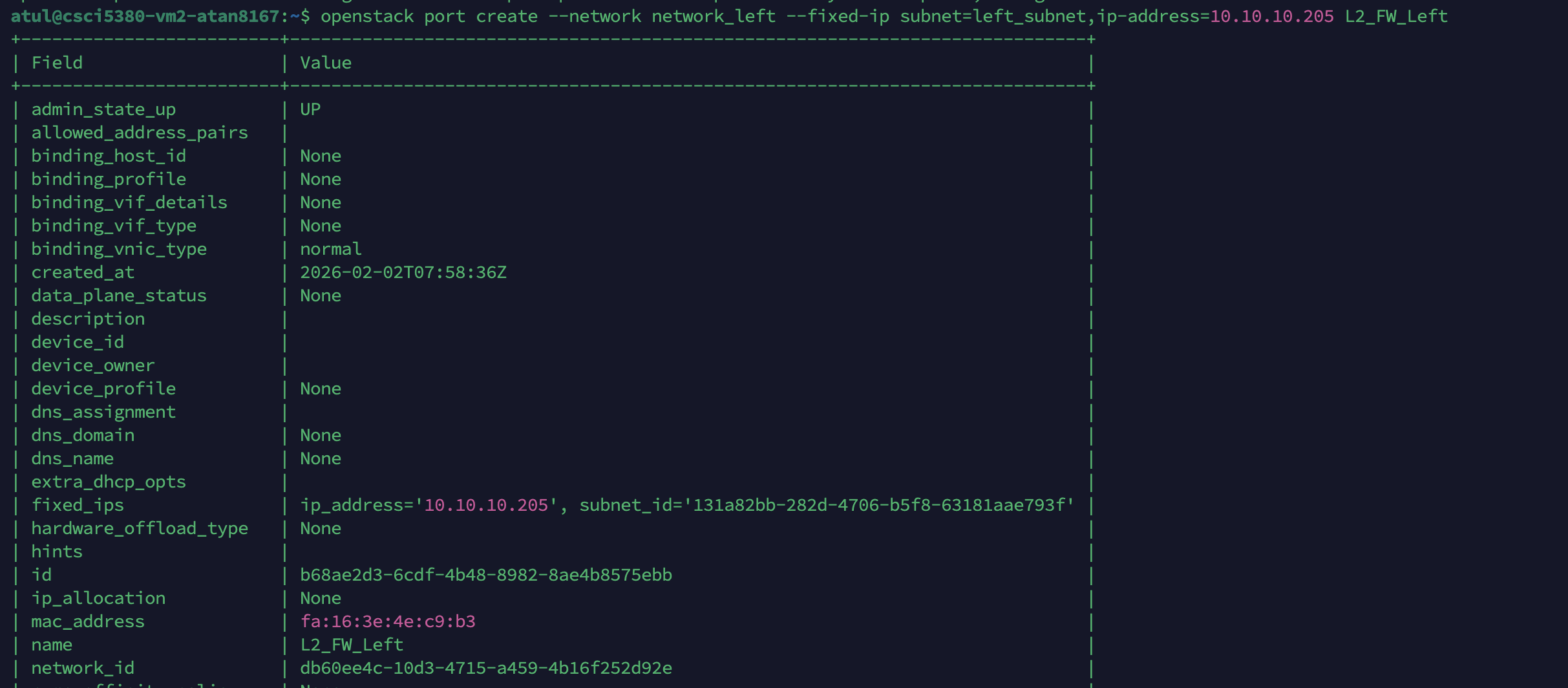
**Network Diagram –**



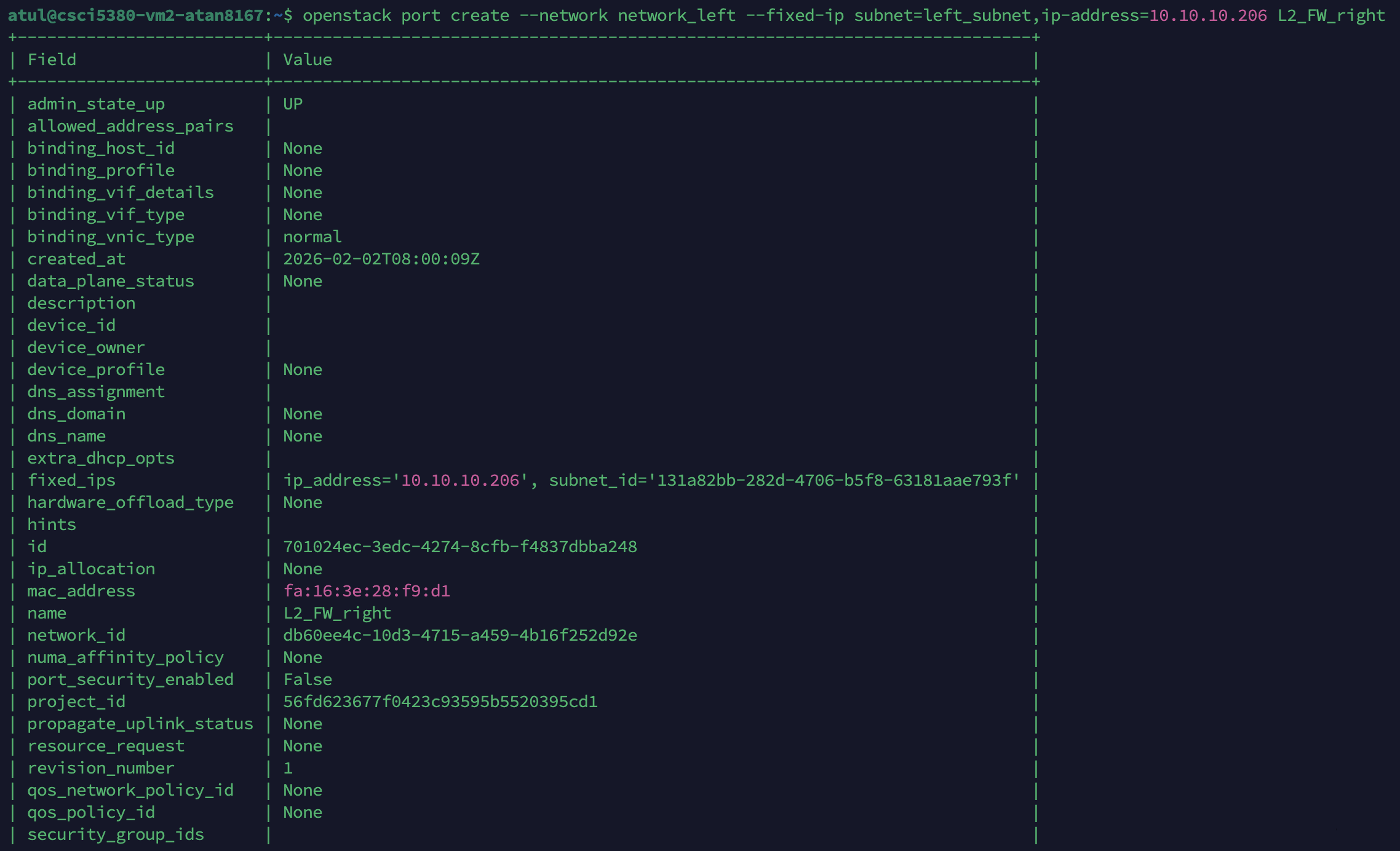
**For Sections 2:**

This only requirement is to add a VM and create port pairs. So, we will create an L2 VM appliance and we will chain the VM host ports to this L2 VM appliance and apply the IP tables rules for dropping the ssh port traffic from the other VM

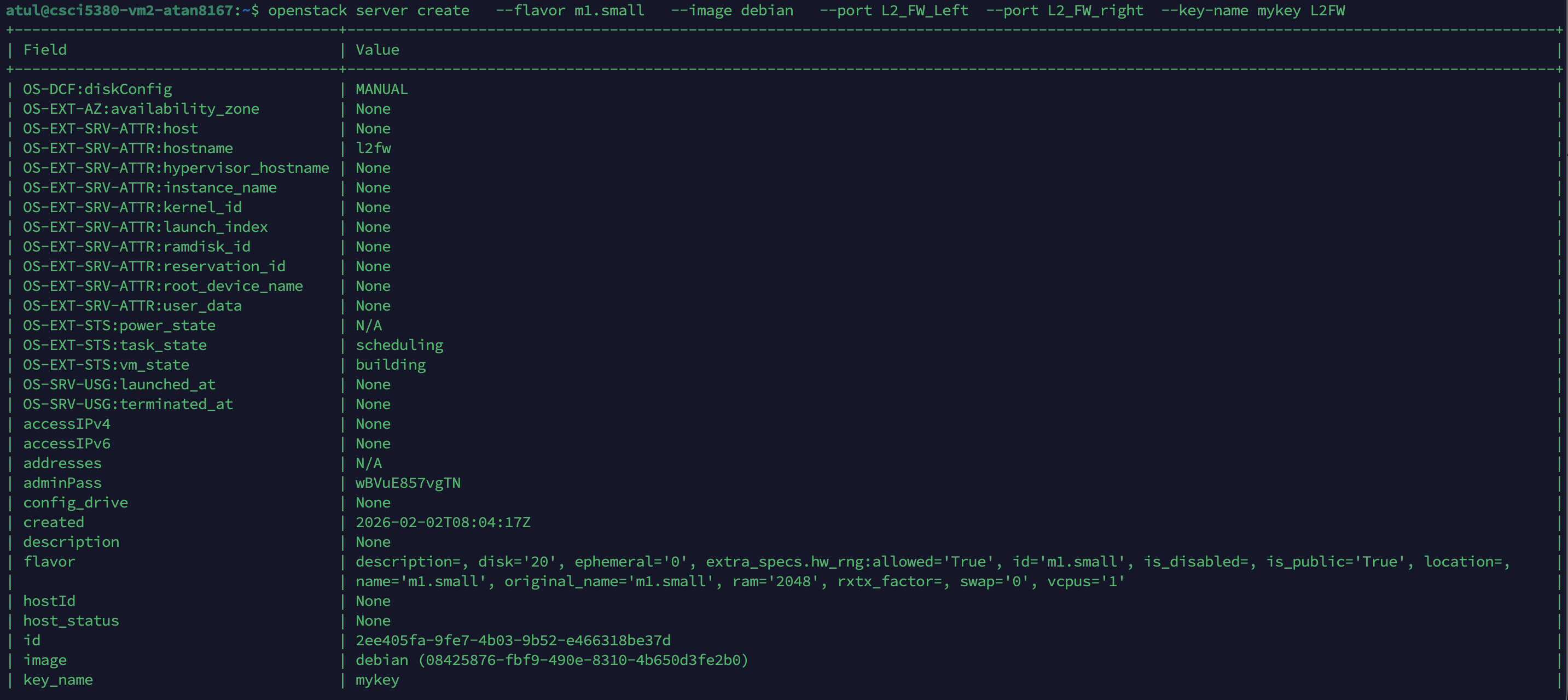
I created the L2\_FW Left port –



L2\_FW Right port

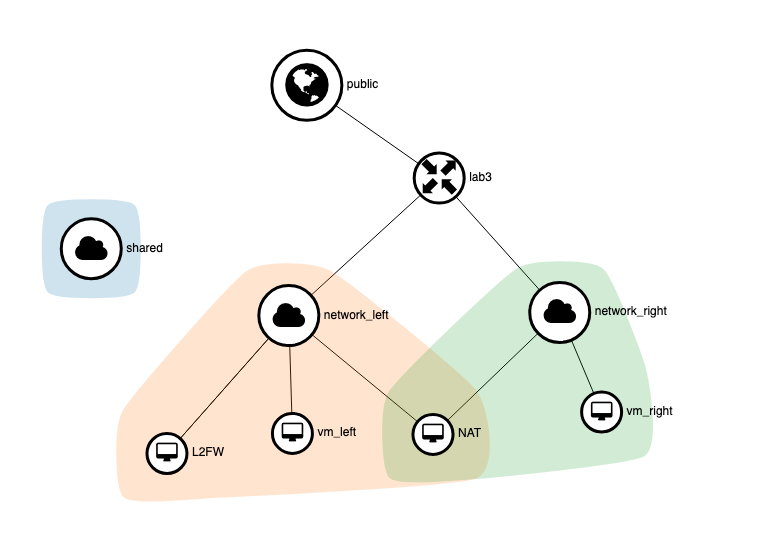


**I will create an L2\_FW instance and associate these ports with that instance –**



L2FW created and Now I will associate this before the Left VM -





I changed the Gateway IP on the VM\_left to the L2FW and setup Iptable forwarding rules on the L2FW which allowed me to use Iptables to deny SSH traffic but other setup as above did not require any change.

Commands for Iptables –

**Sudo Iptables -A FORWARD -s <ip\_add> -d <ip\_add> -p tcp –dport 22 -j drop**