Name: Chaitanya Pidugu.
Banner I'd: 800746834

Data Center Networks with Overlay Networks and Microsegmentation

In this part of the lab, we will create a few tenant VMs to illustrate how communication through the overlay/underlay network functions. We will also show how a set of security rules can be generated to map different rules to different VMs, based on their roles.

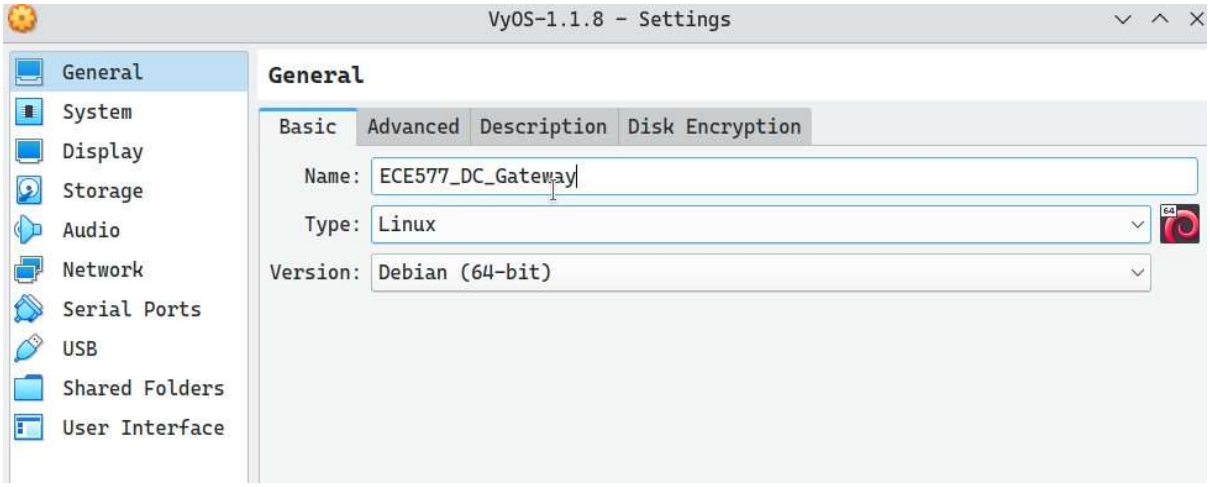
Step A - Install VirtualBox & Configure DC Gateway

1. Stop FRR from running so you don't potentially advertise a shorter route to the 192.168.42.0/24 network than your neighbor: eid@whateverphobia$ sudo systemctl disable --now frr.service
2. On your lab machines (or home machine, as long as you have 8 cores, >=32GB of RAM and 500+GB of solid-state disk space) install Virtual Box using apt: *eid@whateverphobia*$ sudo apt install virtualbox

```
cpidugu@pyrophobia:~$ sudo systemctl disable --now frr.service
[sudo] password for cpidugu:
Synchronizing state of frr.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable frr
cpidugu@pyrophobia:~$ sudo apt install virtualbox
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
virtualbox is already the newest version (6.1.38-dfsg-3~ubuntu1.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
cpidugu@pyrophobia:~$
```

3. You might want to reboot after install (I think it dynamically builds/loads some modules)
4. Download the VyOS VirtualBox appliance from the lab directory (vyos-1.1.8-amd64.ova)

5. Start VirtualBox and go to File → Import Appliance and select the VyOS .ova file (v 1.1.8 as of this writing). If you get a warning about a certificate, you can ignore it. On the Appliance Settings window, you should change the name from VyOS-1.1.8 to ECE577_DC_Gateway



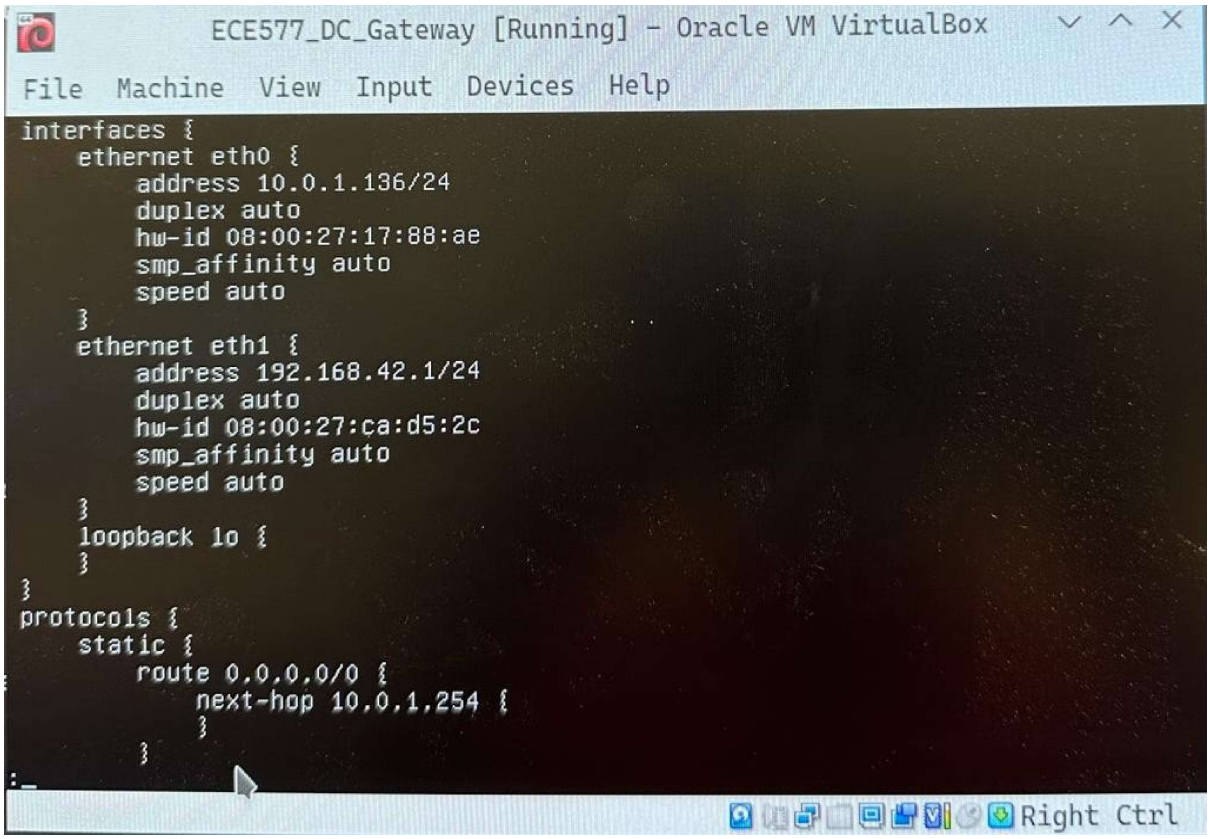
6. Wait for it to import and it should appear as a VM to run, so press the green Start arrow.
7. The image has two NICs, one that is bridged to your eno1 interface and the second on an internal network named “intnet”.

8. Set the IP address of the first interface (eth1) to be 10.0.1.1XX/24. Since this is bridged to your eno1 interface, it should be in the same subnet, and should be accessible from the same machine (or any other machine in EB3012 on the 10.0.1.0/24 subnet).
    a. Login using the username vyos and password vyos
    b. VyOS has a similar interface to FRR and our Baystack switches that use the command prompt, with a little difference. Like FRR/Baystack, typing “configure” is needed to modify the configuration of the device. (The prompt should change from vyos@vyos:~$ to vyos@vyos#). Most configuration is done using the “set” command, and you can always type ? to see the next options and press Tab to complete. For example, to do this step, you would type in config mode: vyos@vyos# set interfaces ethernet eth0 address 10.0.1.1XX/24
    c. Unlike FRR/Baystack, a successful set command doesn’t take effect until you also issue a “commit” command to actually make this change. This allows you to group commands together, instead of always having to make changes one at a time

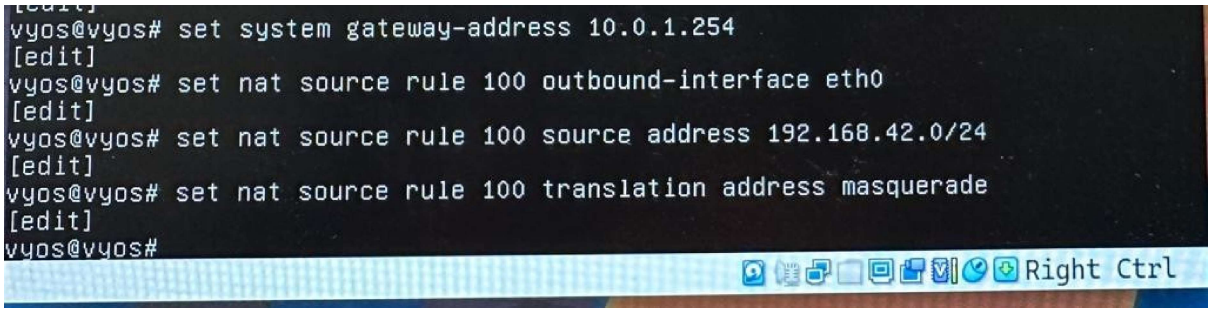
i. Typing “discard” undoes all uncommitted changes
ii. Typing “delete” followed by a command undoes that command (kind of like the “no” keyword in FRR/Baystack)

d. To save the current config, you just have to type the “save” command, which will write the current config to the file VyOS reads on startup.

9. Change the second NIC to have the address 192.168.42.1/24



10. Set the VyOS router’s default gateway to the 10.0.1.254 address
11. Figure out how to have the ECE577_DC_Gateway NAT all outbound traffic to the 10.0.1.1XX address



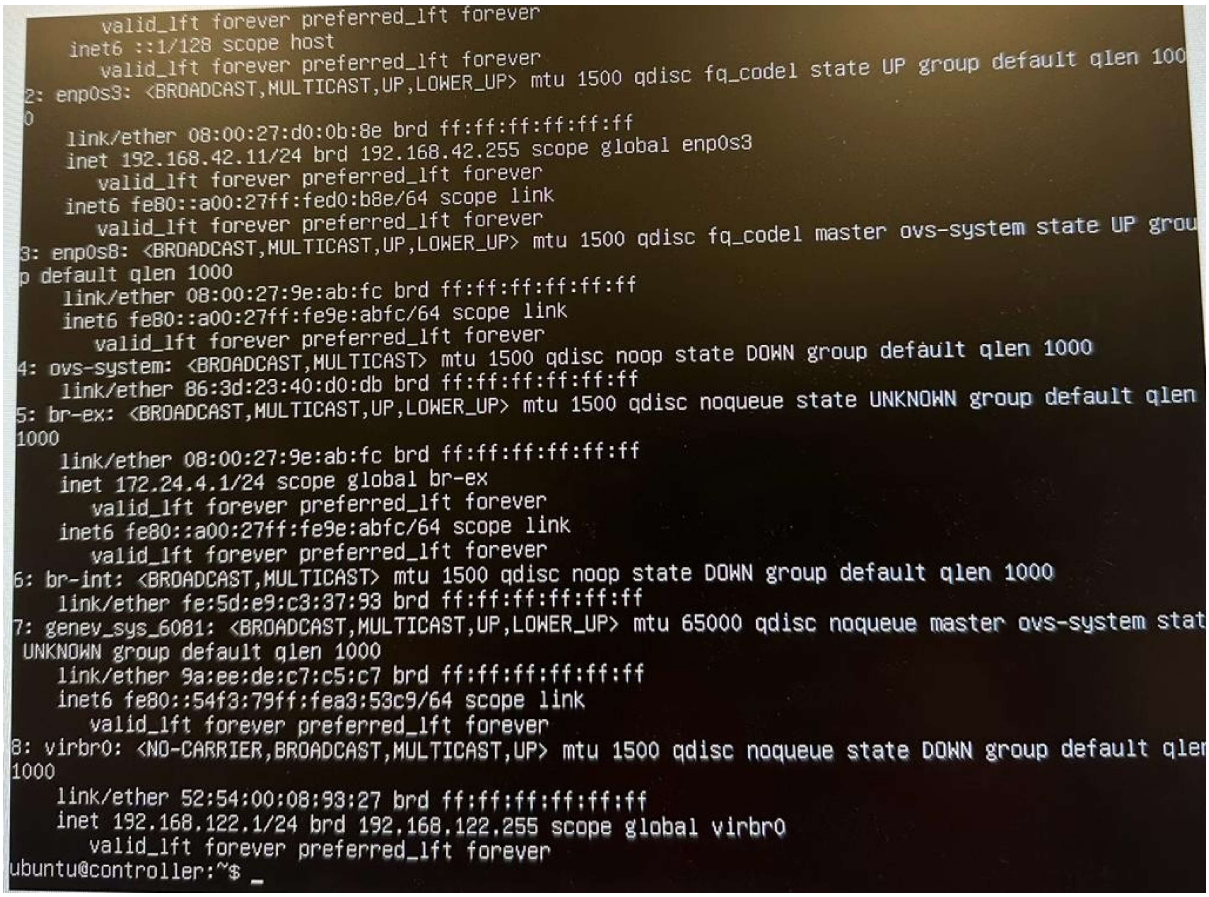
12. Set the DNS resolver of the ECE577_DC_Gateway to the 146.163.252.126 address

13. Configure the ECE577_DC_Gateway to forward DNS requests on the eth1 interface
    a. OPTIONAL: Turn on SSH on the ECE577_DC_Gateway so you can access it using a standard terminal instead of the VirtualBox window by typing:
    eid@whateverphobia$ ssh -oHostKeyAlgorithms=+ssh-rsa vyos@10.0.1.1XX
14. Once all of this is set up, your gateway should now allow anything on the intnet subnet (192.168.42.1) to access the internet by pointing to it (192.168.42.1) as their gateway and DNS resolver.

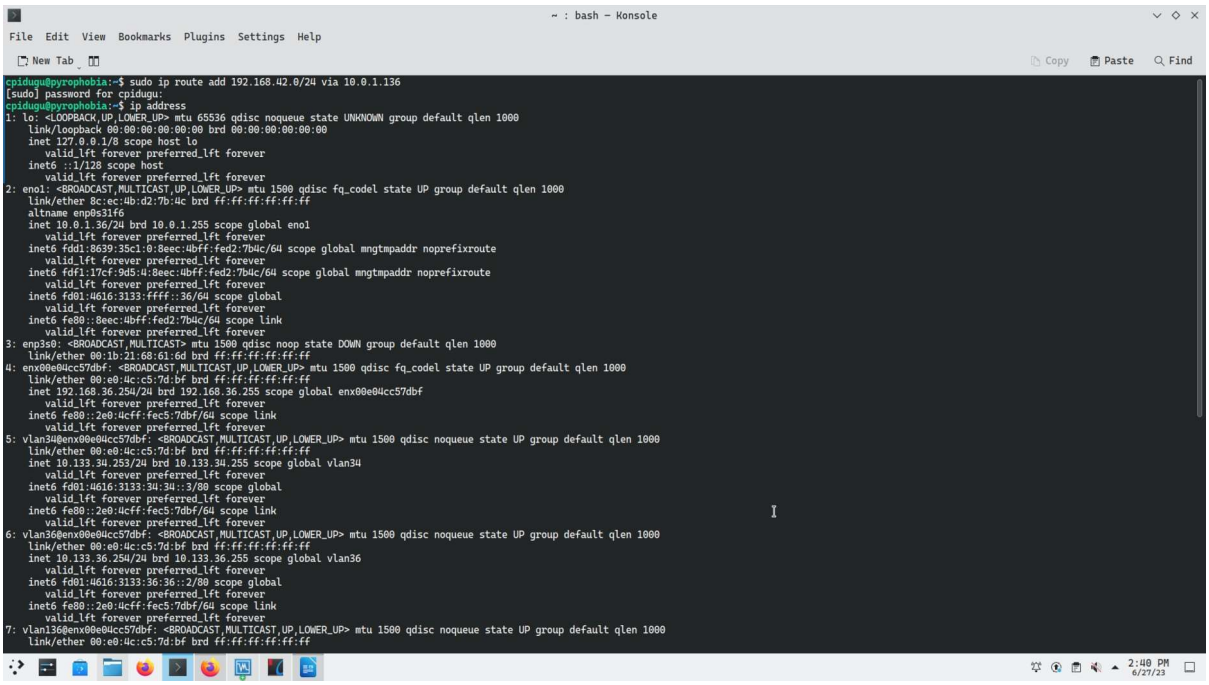
Step B - Getting the DC Running

1. Download the ECE577_SU23_Controller.ova image from here: ECE577_SU23_DC_Controller.ova. The shell login is ubuntu with password ubuntu.
2. In a few minutes, your new OpenStack controller should be ready. The DC Controller has 2 network interfaces, the one on the 192.168.42.0/24 network, which is used for communication to all OpenStack nodes, and the 2nd one (the "public" network) which is nominally used for direct communication with internal VMs. The "public" subnet in the 172.24.4.0/24 address space (Yes, I know this is slightly contradictory here, as a public network should be accessible from the outside with a real IP address. For the purposes of this lab, we'll pretend the "public" network would really be a publicly routable, IP network subnet assigned by your data center provider, and that this IP space is the 172.24.4.0/24 subnet, which we will also pretend is publicly routable). It creates a bridge interface called "br-ex" and should assign that bridge the 172.24.4.1 address to make it the gateway for the 172.24.4.0/24 subnet. As we will see in a few steps, this is the range of "publicly" accessible IPs that can be assigned to internal VMs.
   NOTE: IF YOU EVER REBOOT YOUR CONTROLLER, IT DOESN'T PROPERLY REASSIGN THE IP TO br-ex ON STARTUP!!! YOU MUST DO THIS:
   $ sudo ip addr add 172.24.4.1/24 dev br-ex
   $ sudo ip link set br-ex up

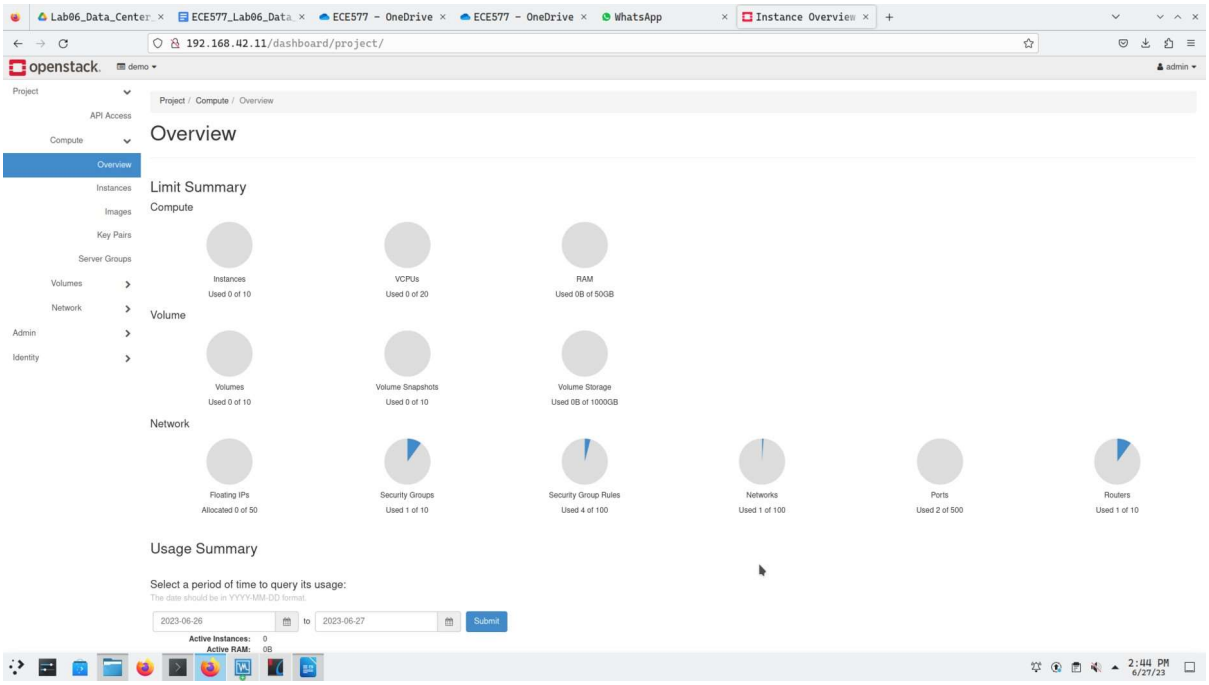
   (You might be able to get netplan to attempt this on reboots, but since netplan starts before OpenStack, it's probably not feasible without lots of additional scripting).



3. The Controller node is at IP address 192.168.42.11, but your lab machine doesn't know that network so, at this point, would forward that traffic to it's gateway (and be blocked!) So, tell your lab machine you know how to get to the 192.168.42.0 subnet via the ECE577_DC_Gateway at 10.0.1.1XX: eid@whateverphobia$ sudo ip route add 192.168.42.0/24 via 10.0.1.1XX



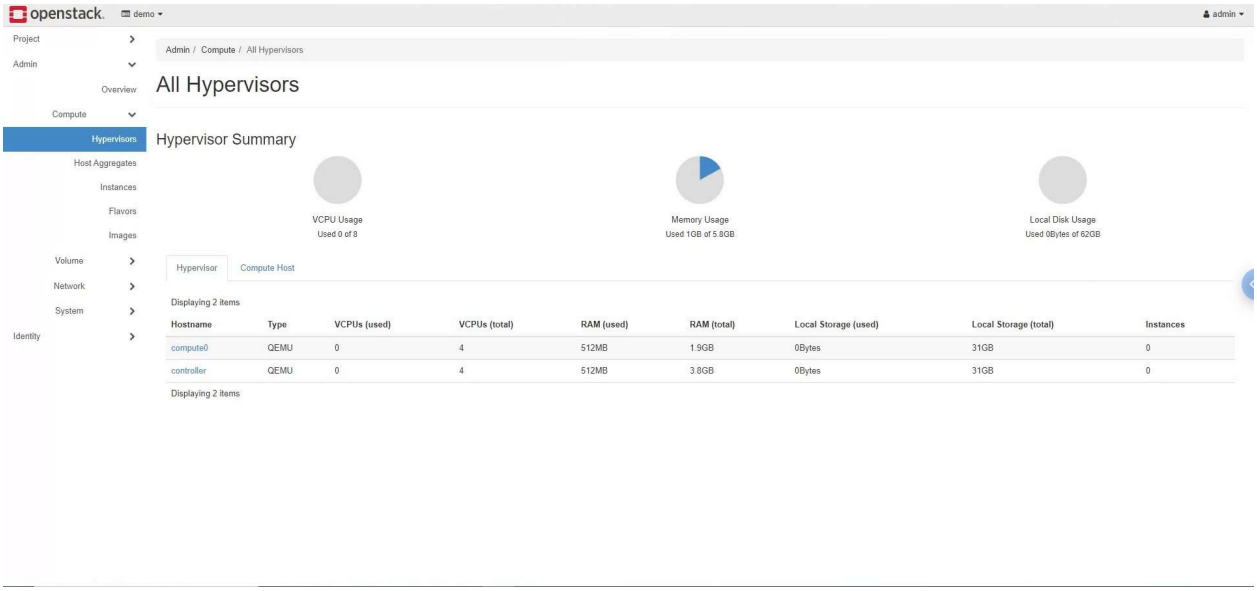
4. You should now be able to login to the admin console by pointing your lab machine's browser to 192.168.42.11, login is admin password is ece577

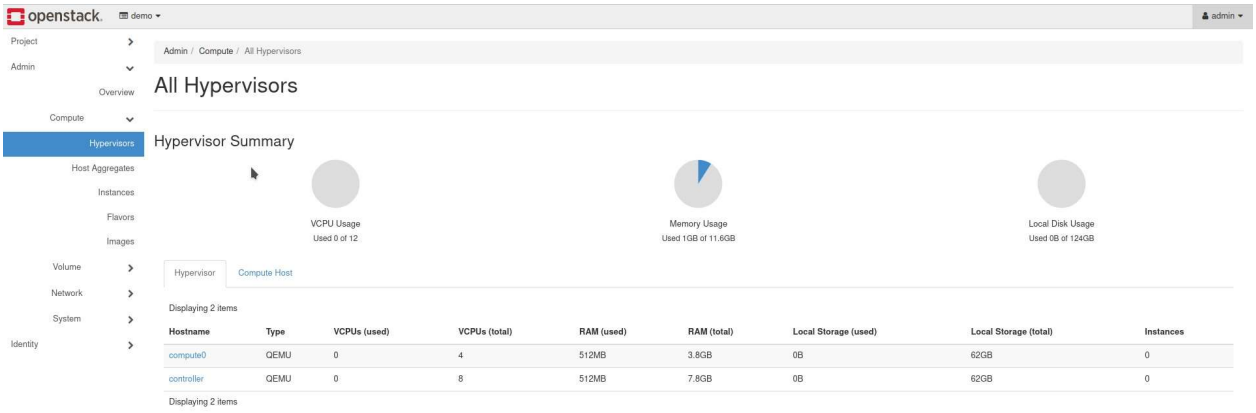


5. Download a Compute node image from here: [ECE577_SU23_Compute0.ova]

6. As before with the Controller, import it into VirtualBox and start it. The shell login is ubuntu and password is ubuntu.

7. Verify that the compute0 node is recognized by going to Admin → Compute → Hypervisors in the OpenStack admin console:





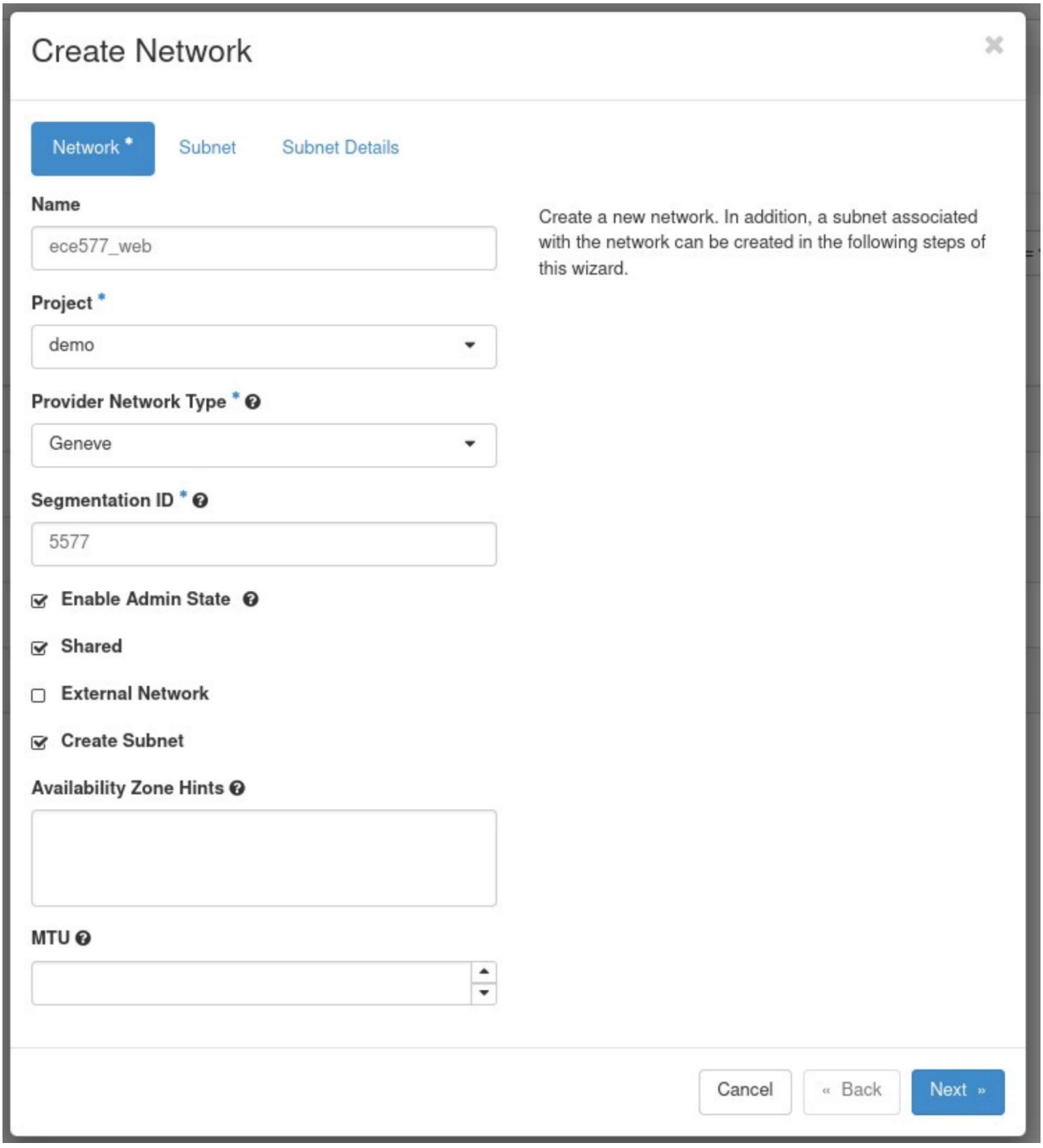
Step C - Creating the Virtual Networks

Now that you have added a Compute Node to the Data Center, it's time to start adding Virtual Machines. First, you will have to create a few new networks for these Virtual Machines.

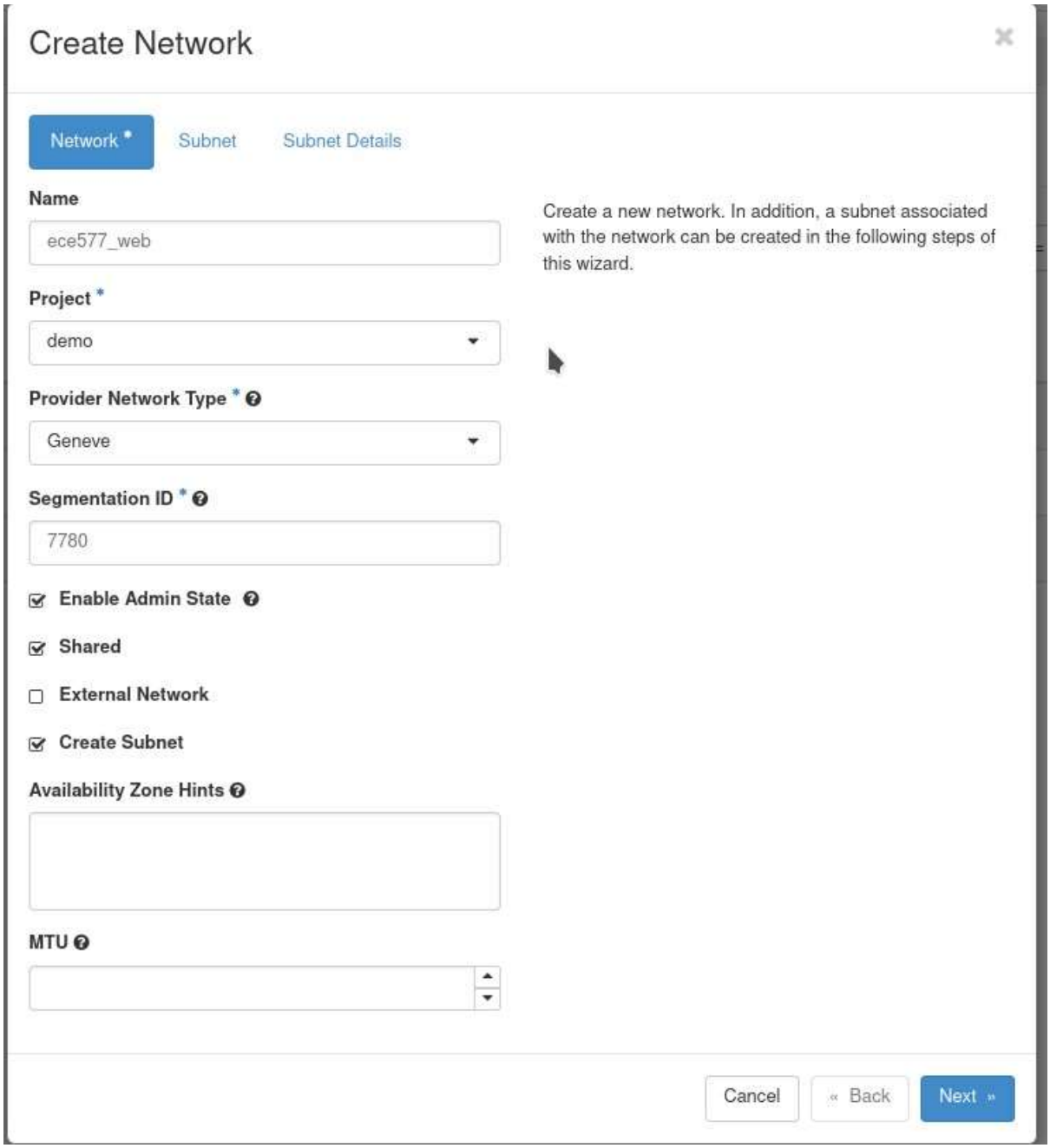
Network Creation:

1. Log back in to the OpenStack console. Click on Admin, then click on Network, then click on Networks. Click the +Create Network button
2. Name the network ece577_web. The project should be demo, the Provider network type should be Geneve, with Geneve Segment ID 7780. Enable Admin

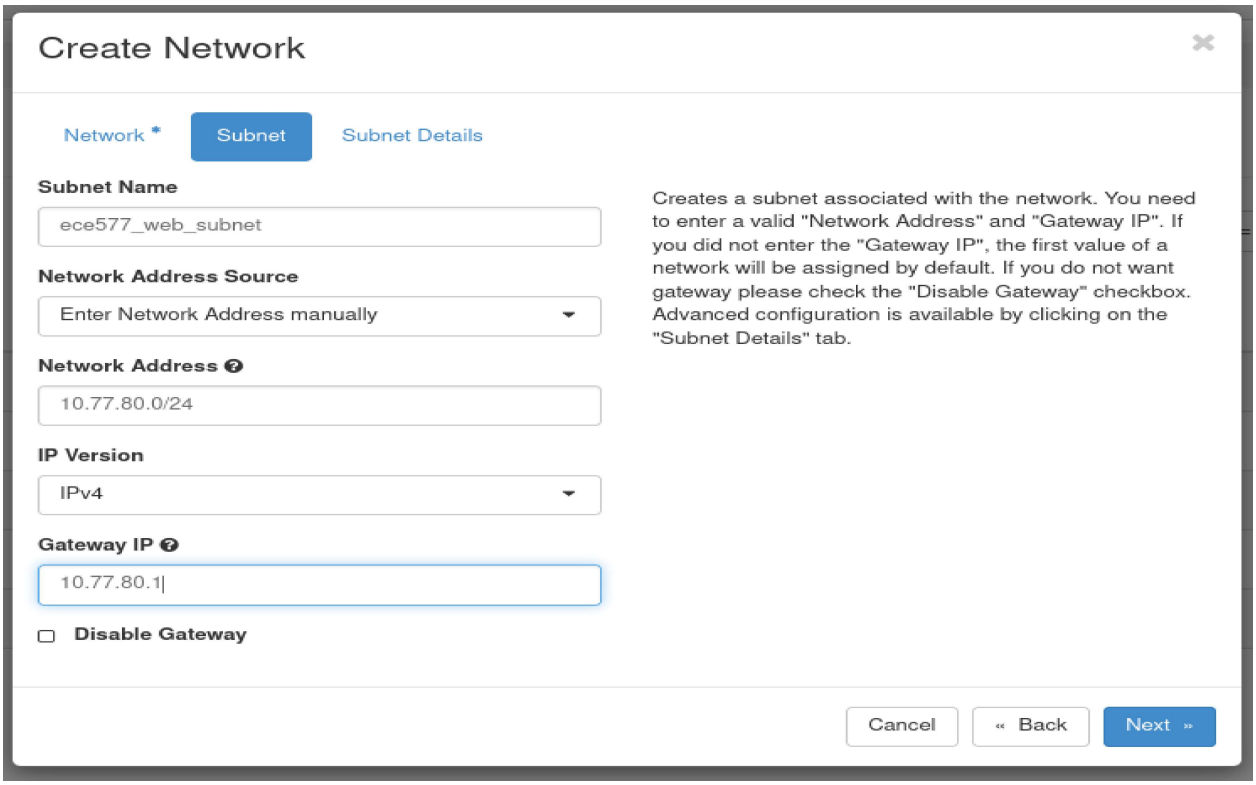
State, Shared, and Create Subnet should also be checked. Click Next to create the Subnet.

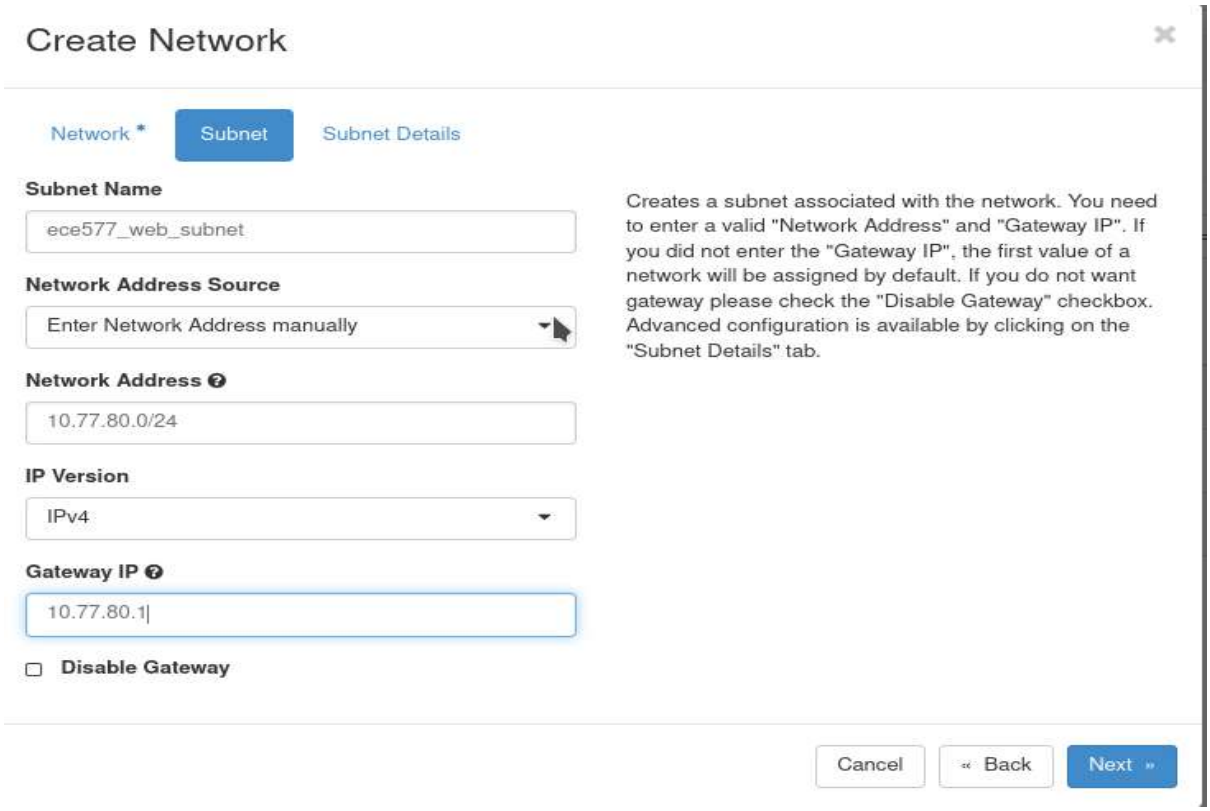


3. Name the subnet ece577_web_subnet. The address range should be

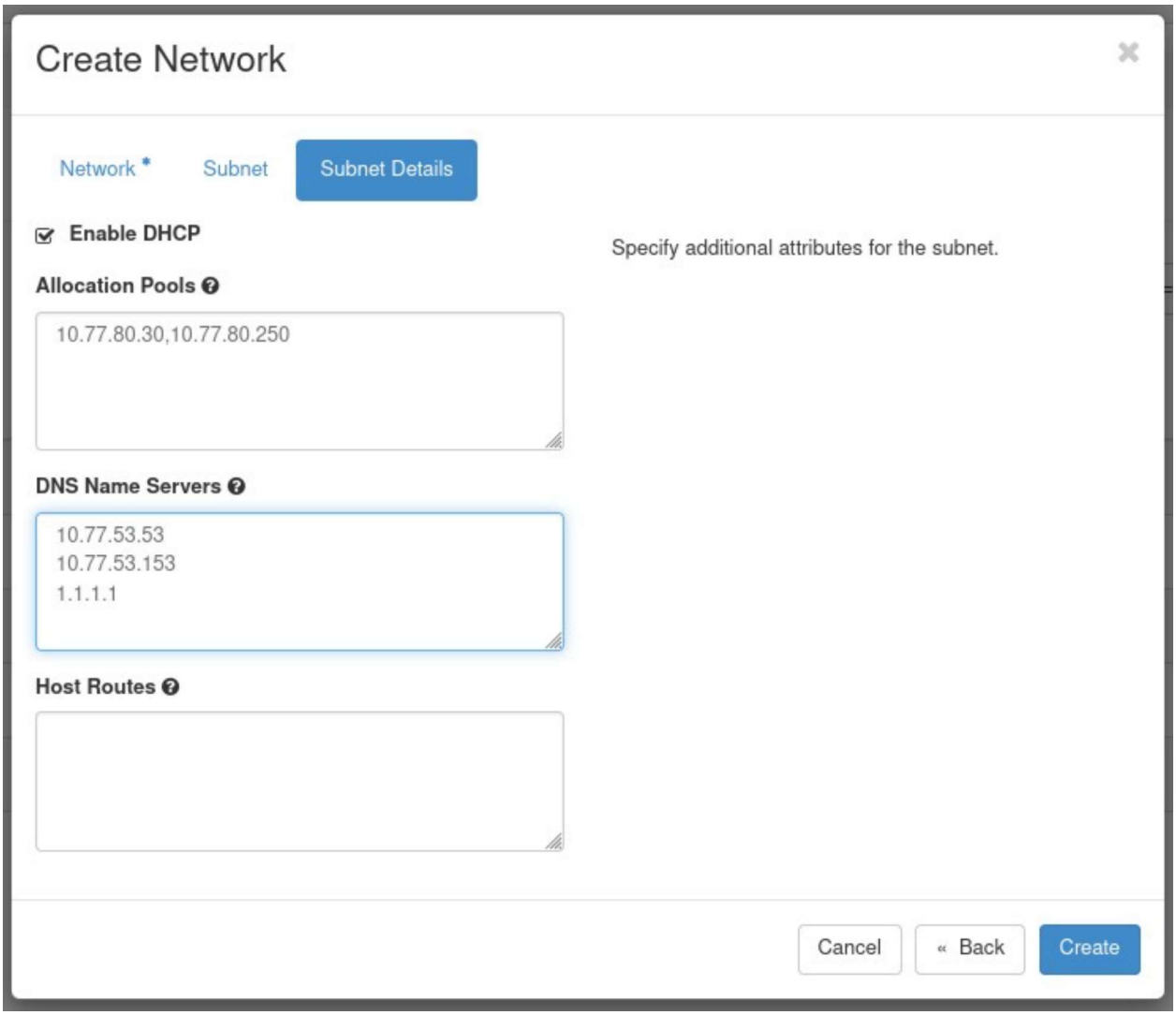


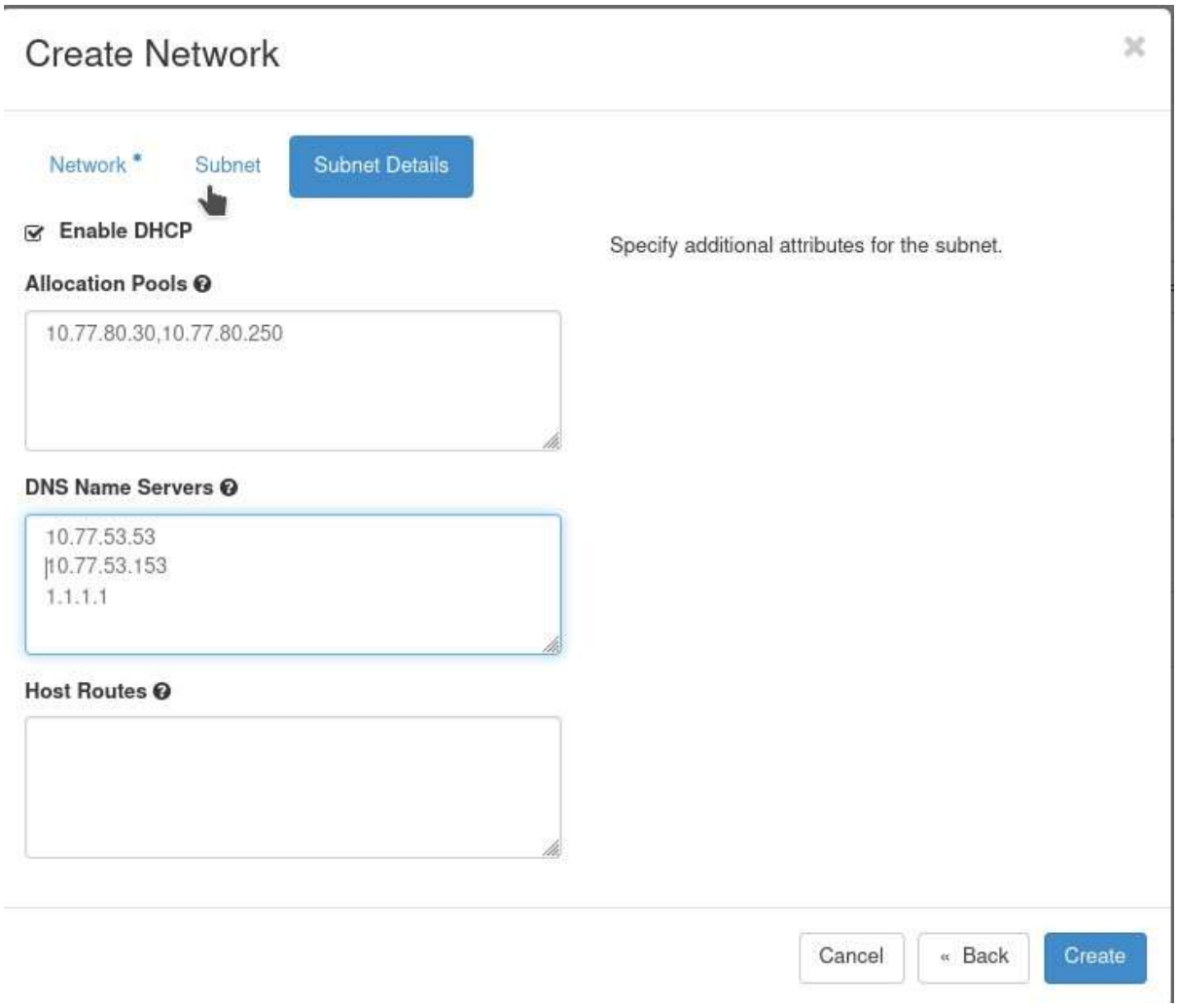
10.77.80.0/24. The Gateway IP (we'll get there in a minute) will be 10.77.80.1. Click Next.


Create Network
Network *
Subnet
Subnet Details
Subnet Name
ece577_web_subnet
Network Address Source
Enter Network Address manually
Network Address
10.77.80.0/24
IP Version
IPv4
Gateway IP
10.77.80.1
Disable Gateway
Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.
Cancel
« Back
Next »

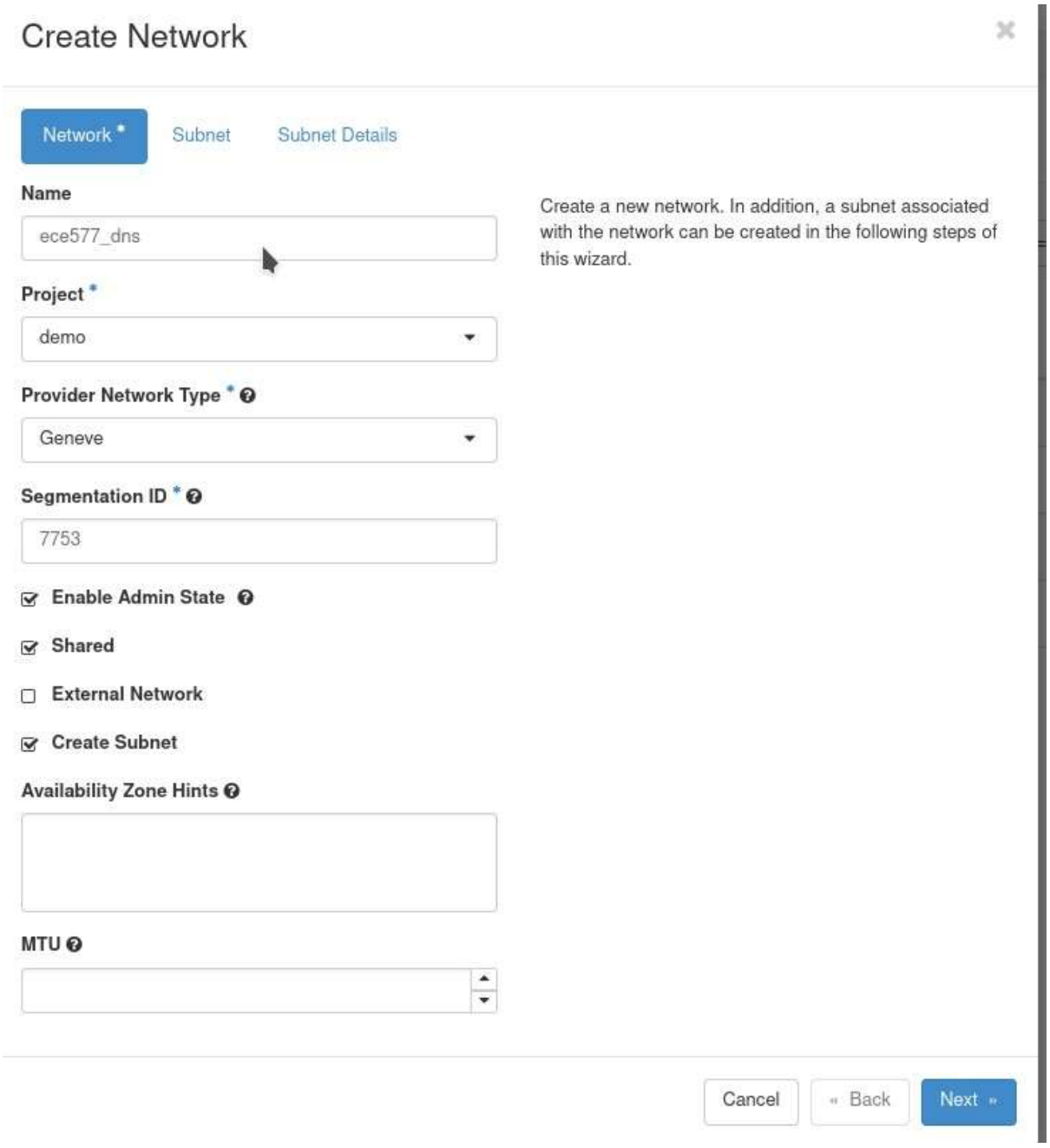


4. Under Subnet Details, add the subnet range 10.77.80.30,10.77.80.250 and name servers (one per line): 10.77.53.53, 10.77.53.153, 1.1.1.1. Click Create.

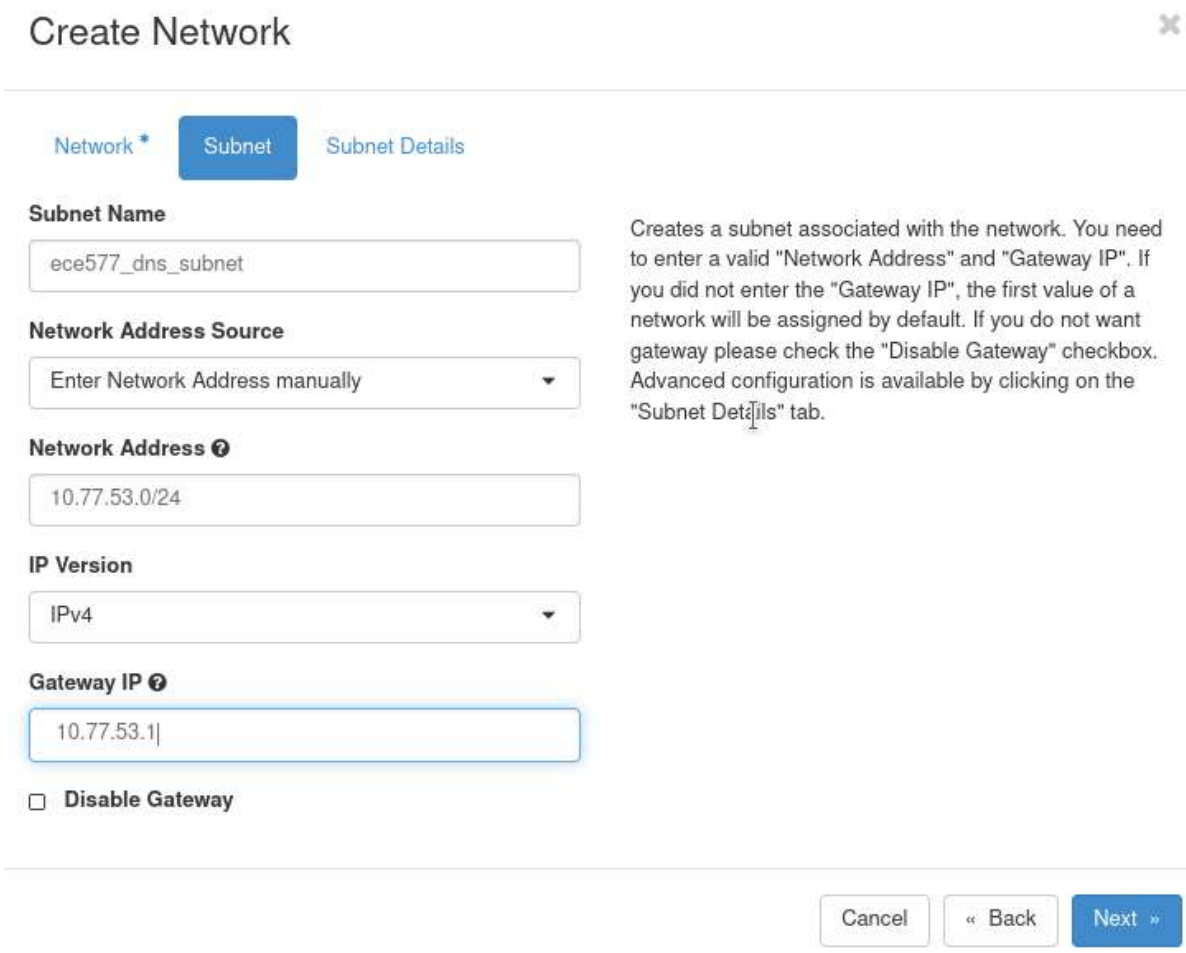

Create Network
Network *
Subnet
Subnet Details
Enable DHCP
Specify additional attributes for the subnet.
Allocation Pools
10.77.80.30,10.77.80.250
DNS Name Servers
10.77.53.53
10.77.53.153
1.1.1.1
Host Routes
Cancel
« Back
Create



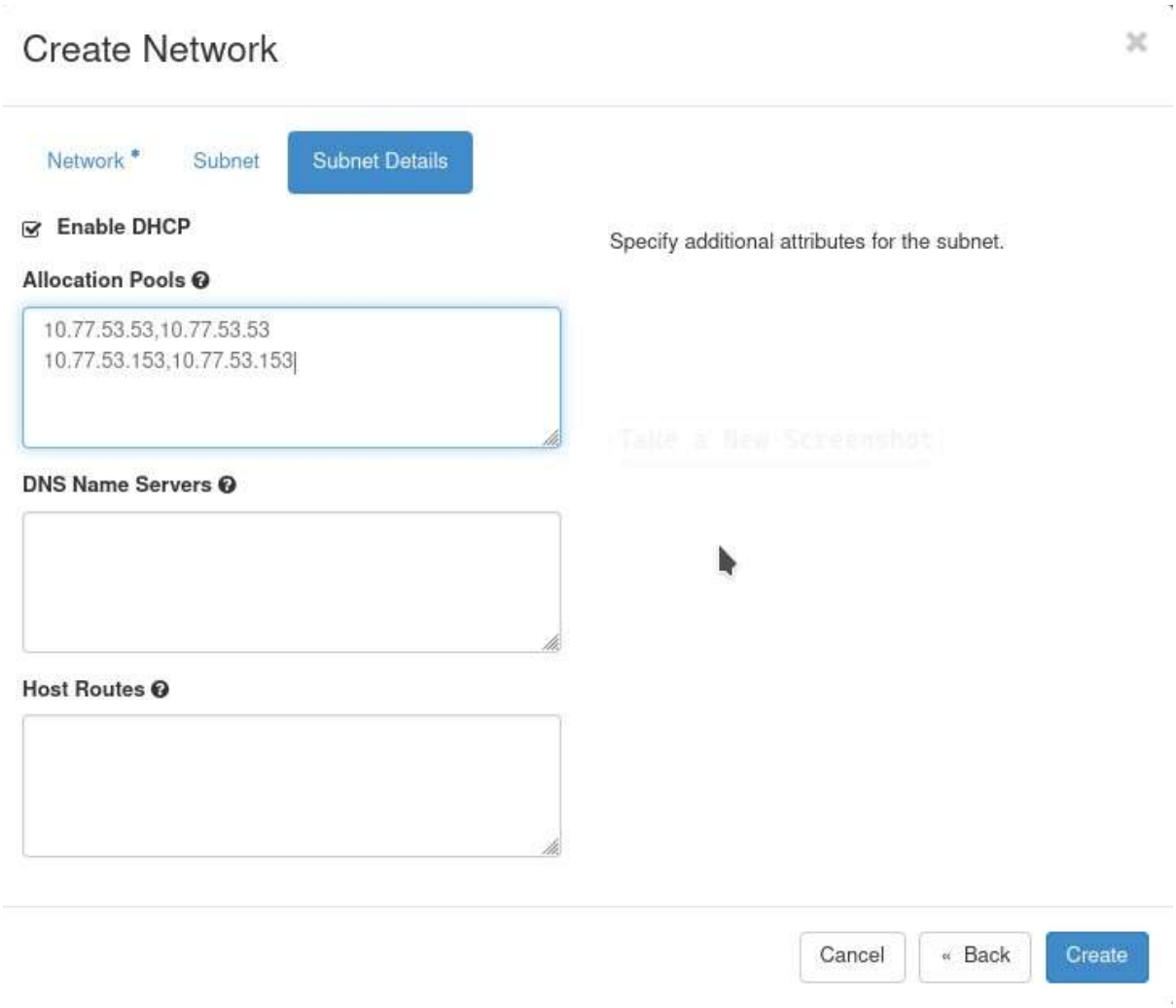
5. Repeat step 2, creating another network called ece577_dns. Provider network type should be Geneve, with Segment ID 7753. Project should be demo, Enable Admin State, Shared, and Create Subnet should also be checked. Click Next to create the Subnet.



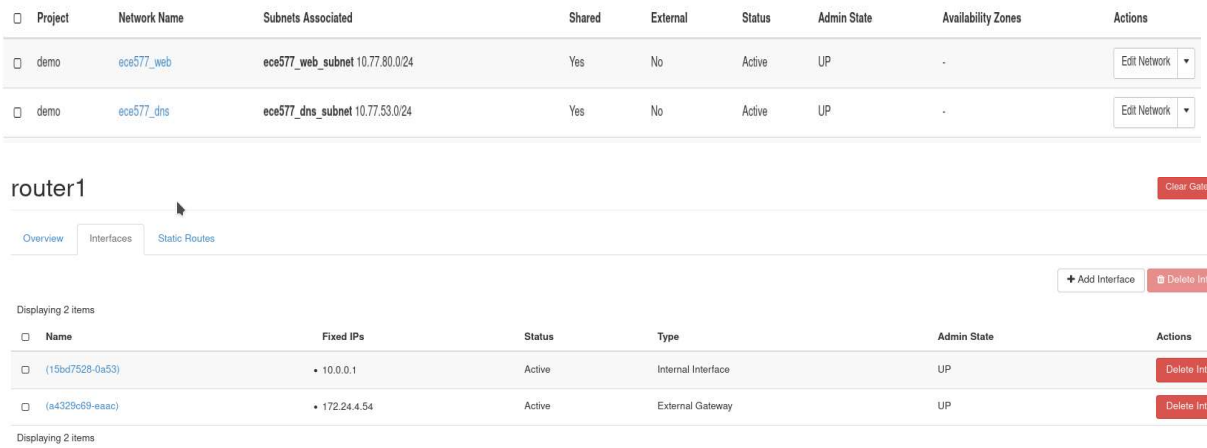
6. Repeat step 3, Name the subnet ece577_dns_subnet. The address range should be 10.77.53.0/24. The Gateway IP (we'll get there in a minute) will be 10.77.53.1. Click Next.

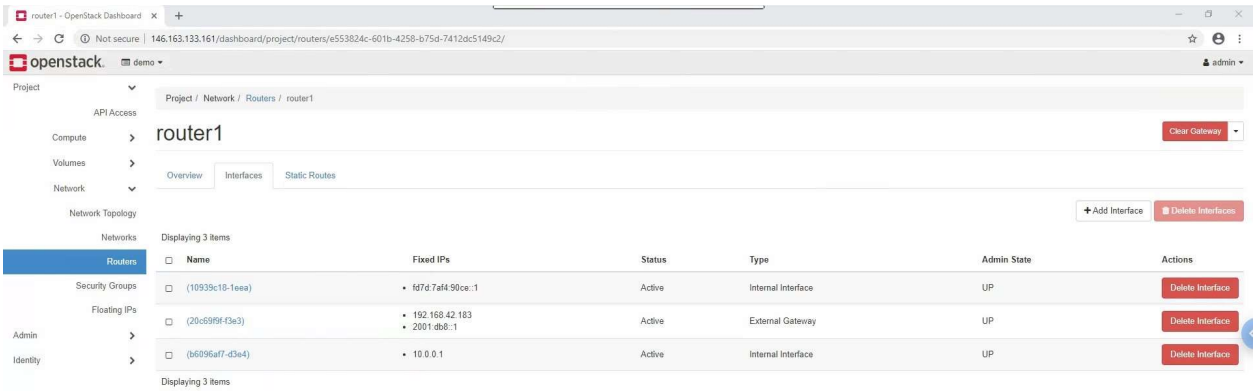


7. Configure the same subnet details as above, but with range 10.77.53.53,10.77.53.53 for the first line in the Allocation Pools box and 10.77.53.153,10.77.53.153 as the second line. This will ensure the generated VM will only have one of those two addresses.

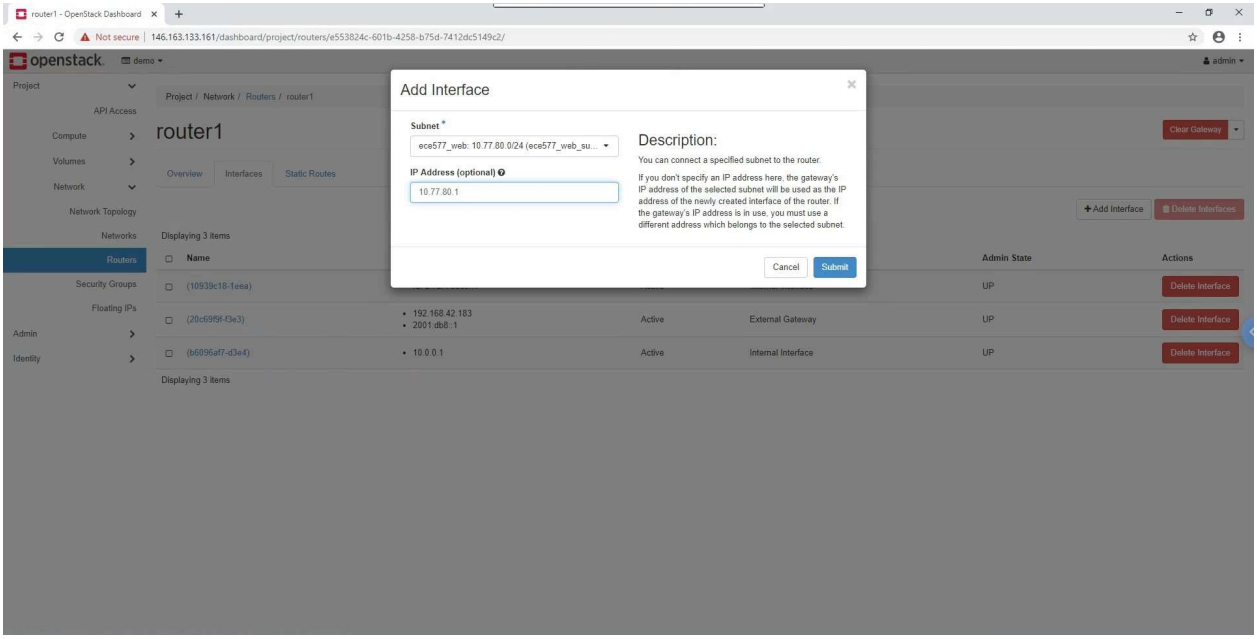


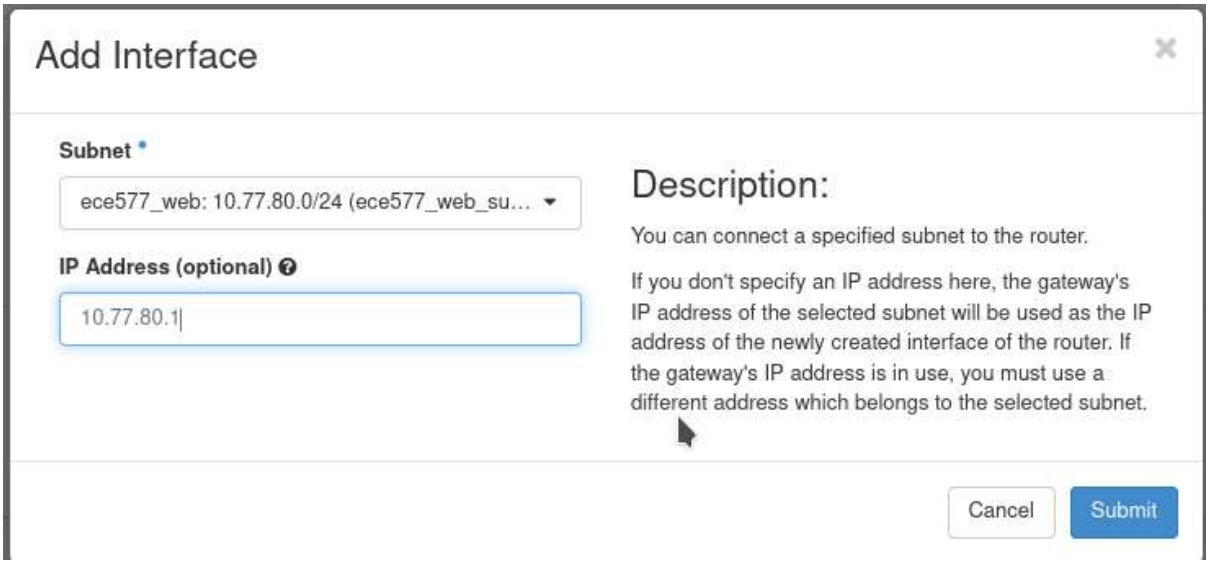
8. Now, you should now have 2 internal networks. We next need to create the gateways for these networks. Under Project → Networks → Routers, click router1. Then click the Interfaces tab. Click +AddInterface.



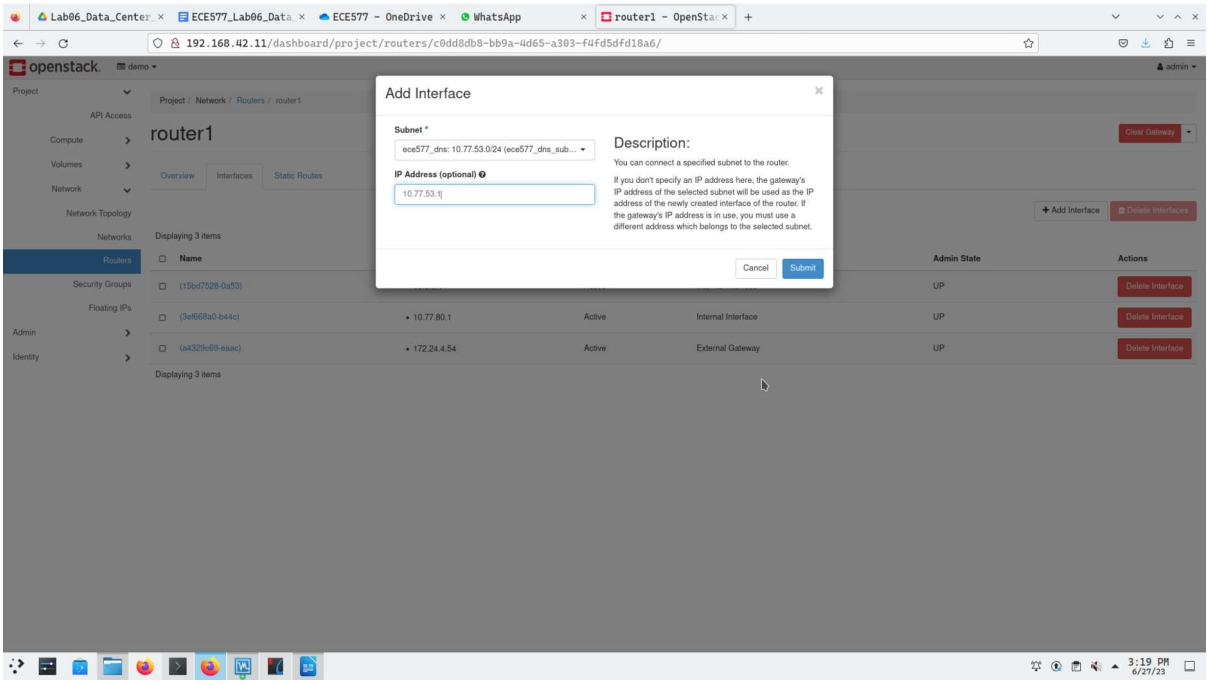


9. In the drop down, select your web subnet, and add the 10.77.80.1 address. Click Submit.





10. Repeat for your dns subnet with the 10.77.53.1 address. At this point, we now have two software defined networks on a software defined router. Next, we will need to add some Vms.



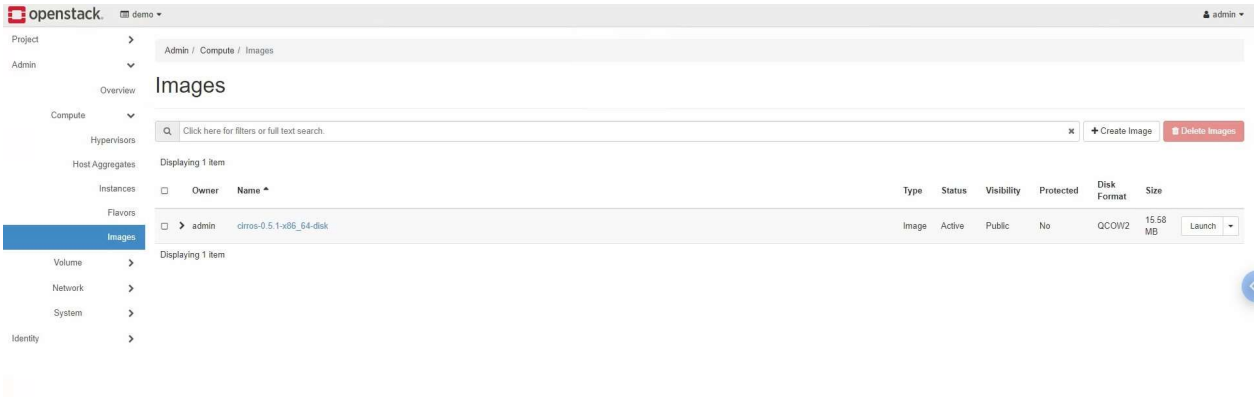
Step D - Creating VMs

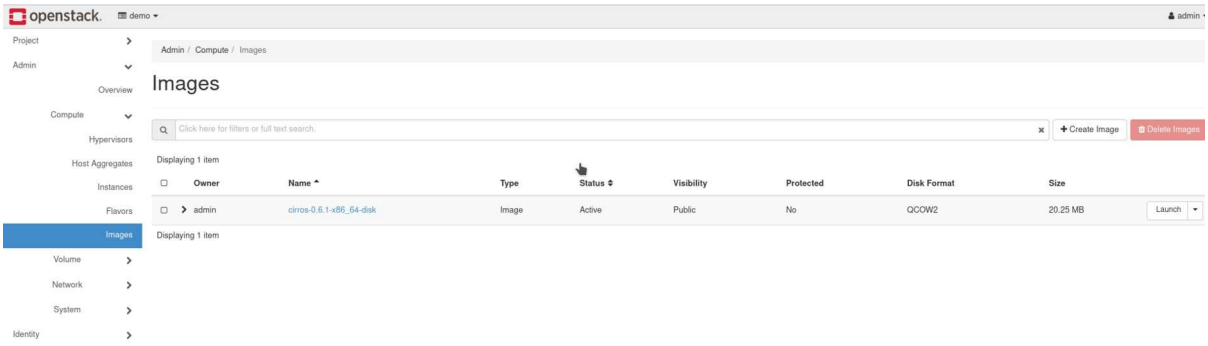
Downloading a Base Image

To create VMs, we first need to have a base OS image to build the VMs from. We will

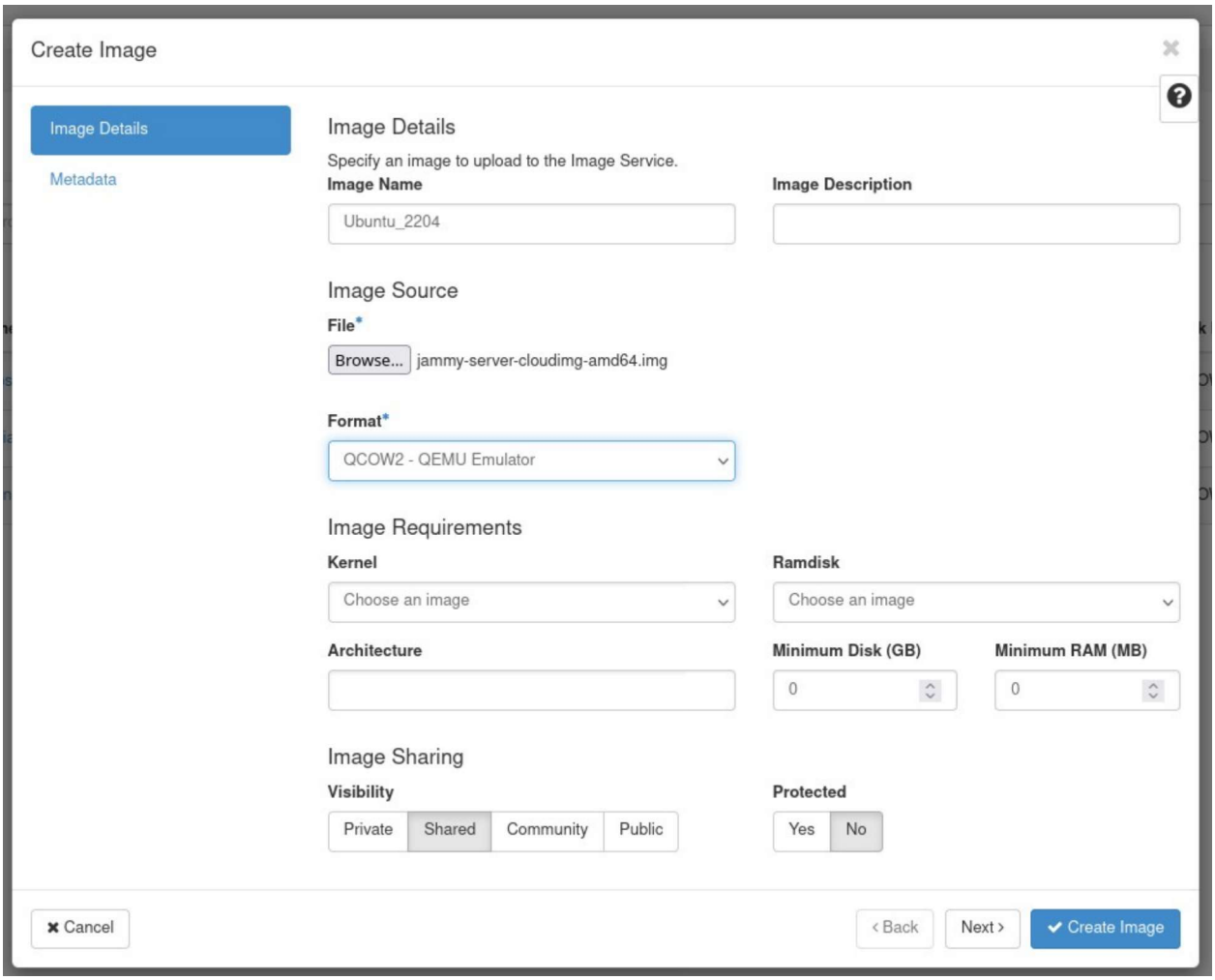
fetch the latest Ubuntu LTS cloud image.

1. In your host (or wherever you are accessing the web console from) go to: https://cloud-images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img
2. Once the image file is downloaded, go to Admin → Compute → Images in the OpenStack web console:

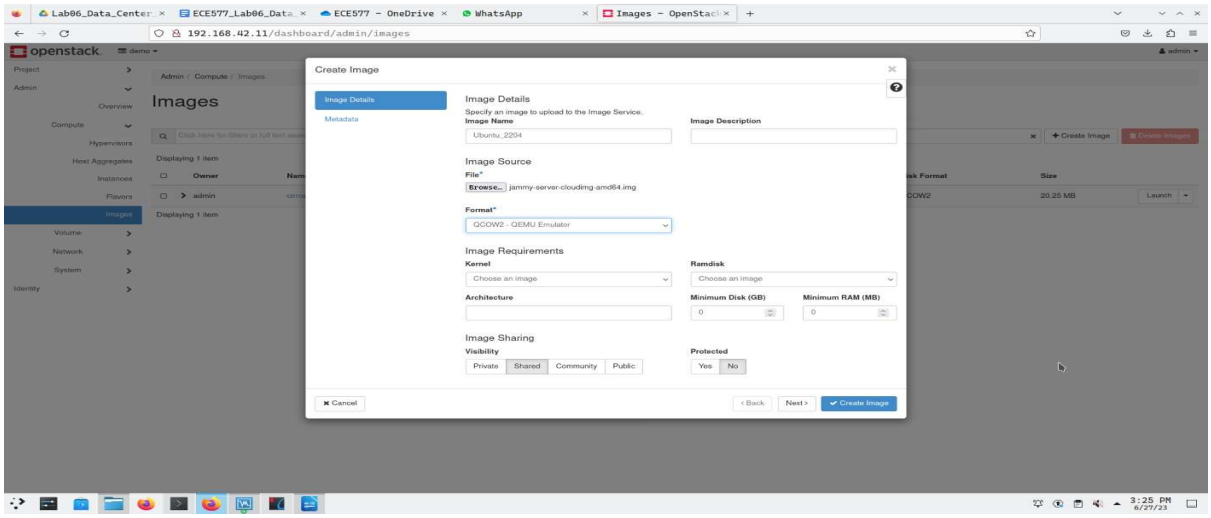




3. Click on Create Image. The Name should be Ubuntu_2204, the Image Source should be the jammy-server-cloudimg-amd64.img file you just downloaded, and the format is QCOW2 - QEMU Emulator in the dropdown.

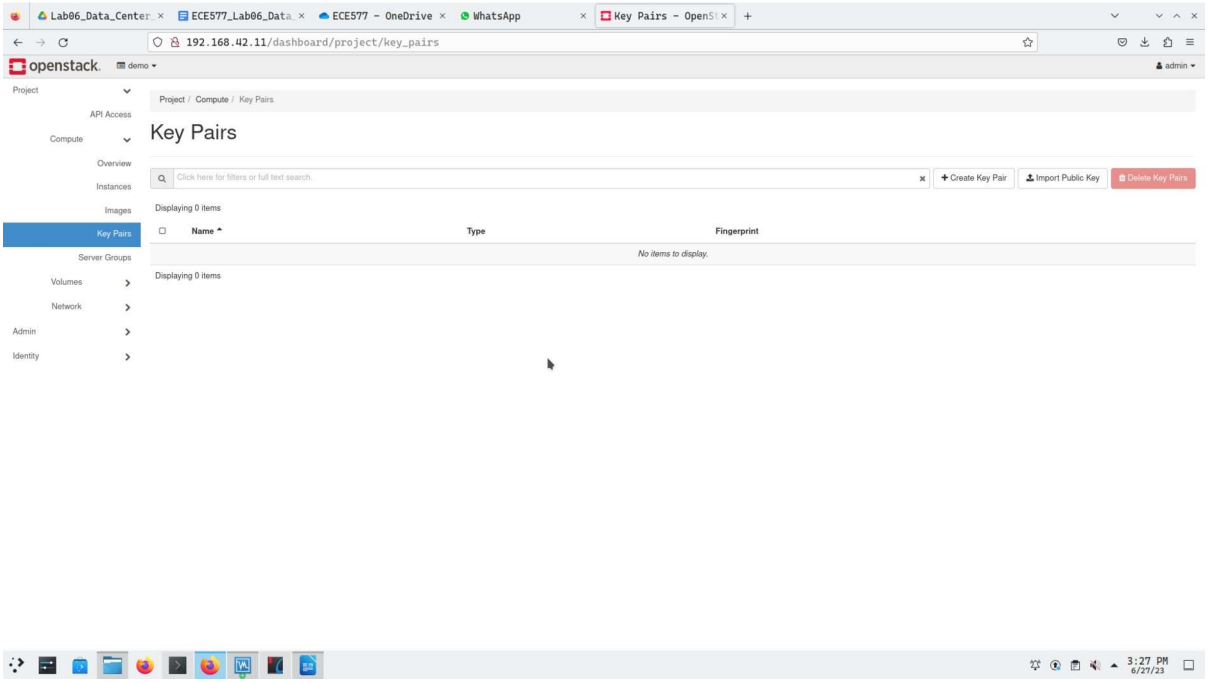


Click “Create Image”

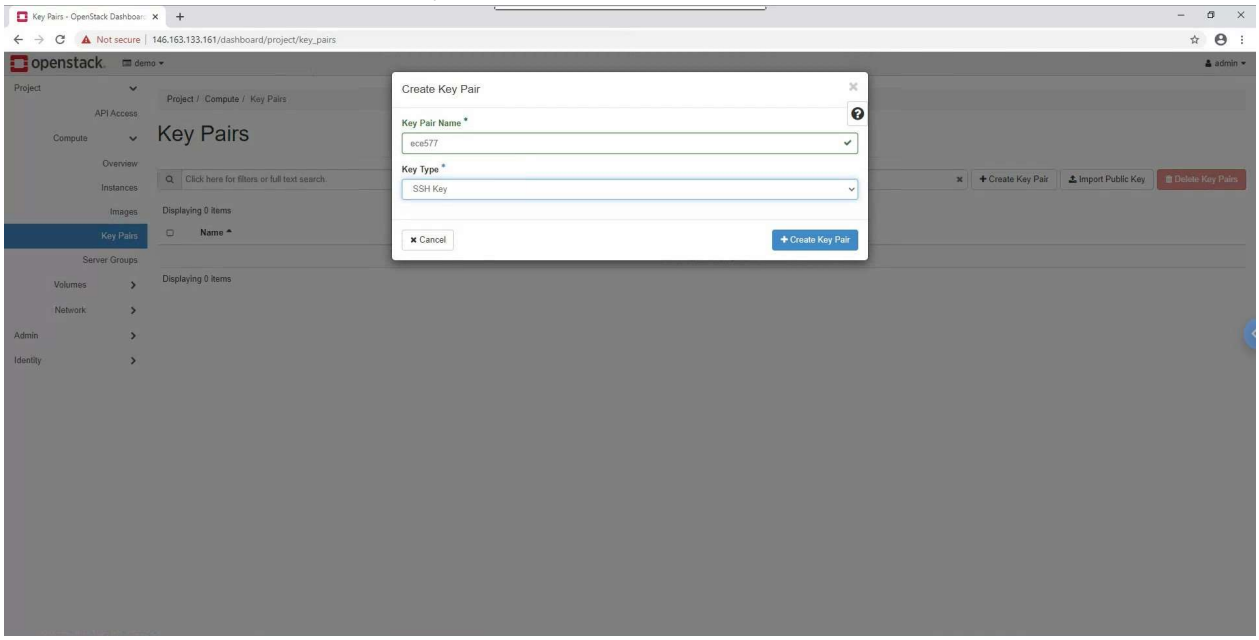


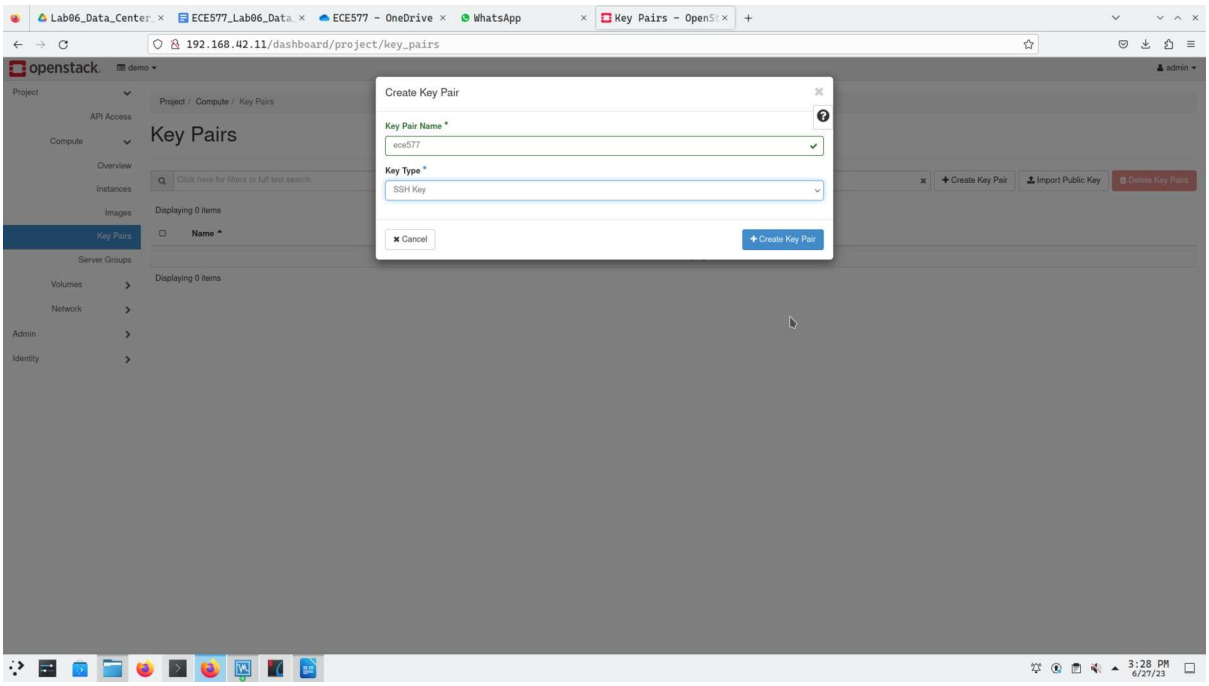
Creating a Key File

4. Before we create VMs, we need to create an encryption key pair for SSH into the machines (by default there is no login password for security purposes!). Log back in to the OpenStack console. Click on Project → Compute → Key Pairs



5. Click +Create Key Pair. For the Key Pair name, use ece577. For Key Pair type, make is SSH. Click Create Key Pair.



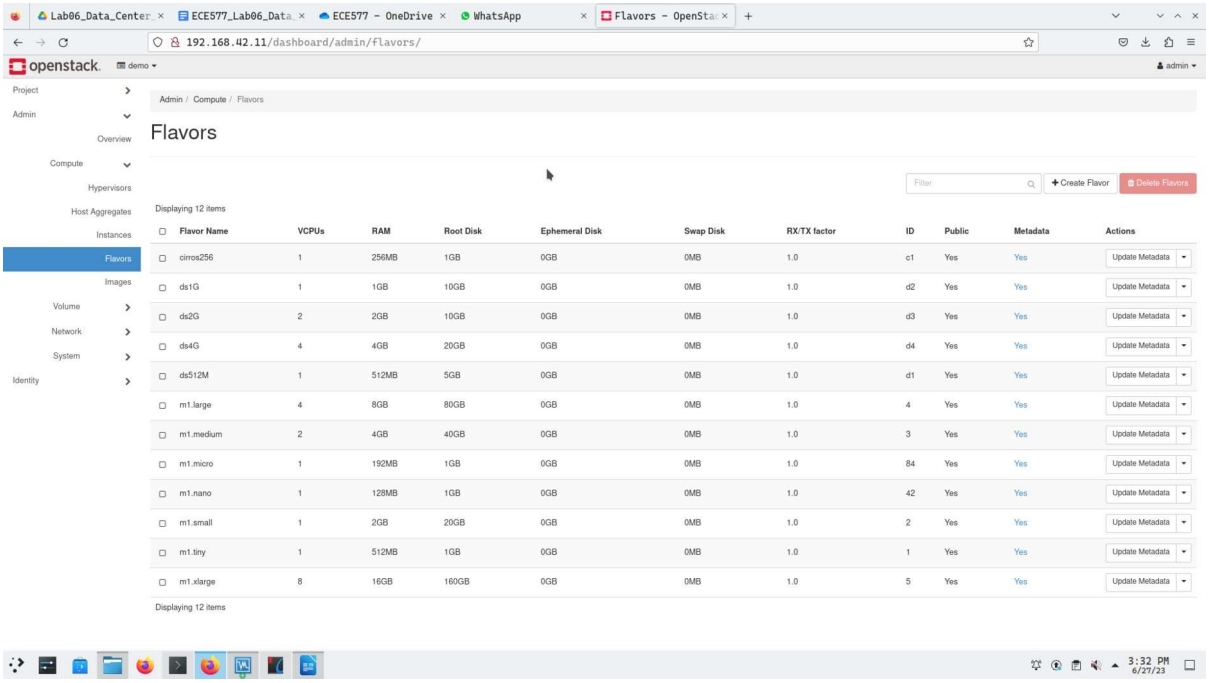


6. This should generate a ece577.pem file for you to download. It will likely end up in your Downloads folder. DON'T LOSE THIS FILE!!! IT'S THE ONLY WAY TO LOG IN!!!

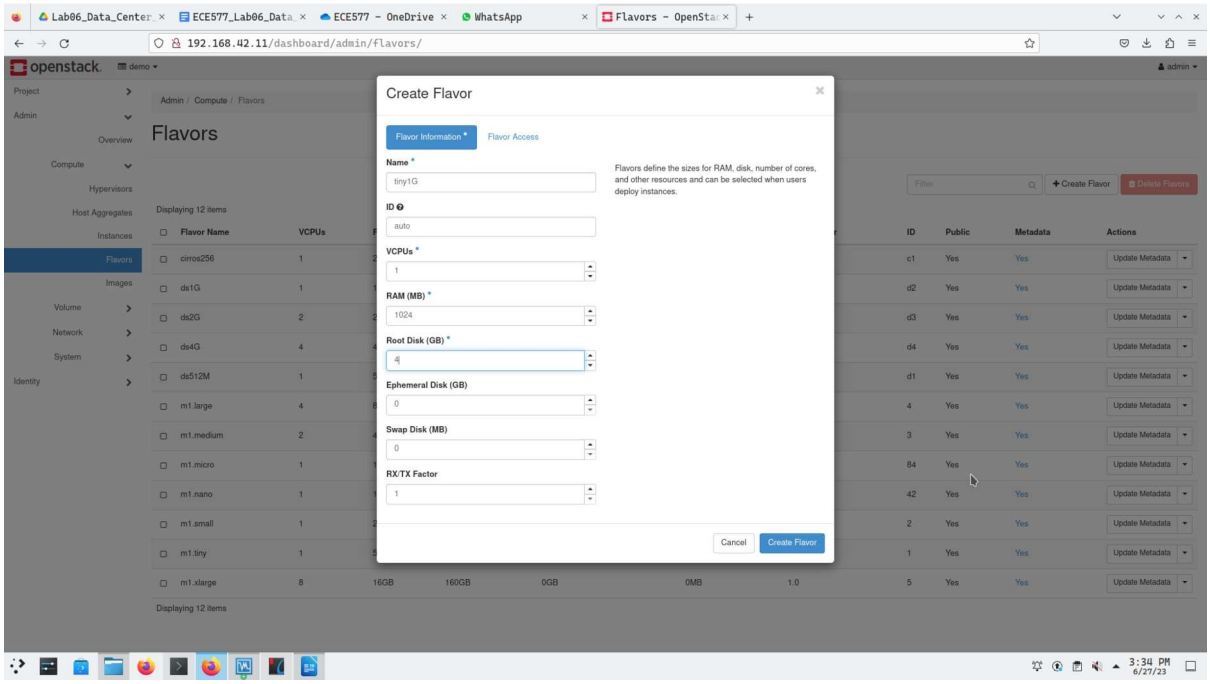
Create VM Flavor

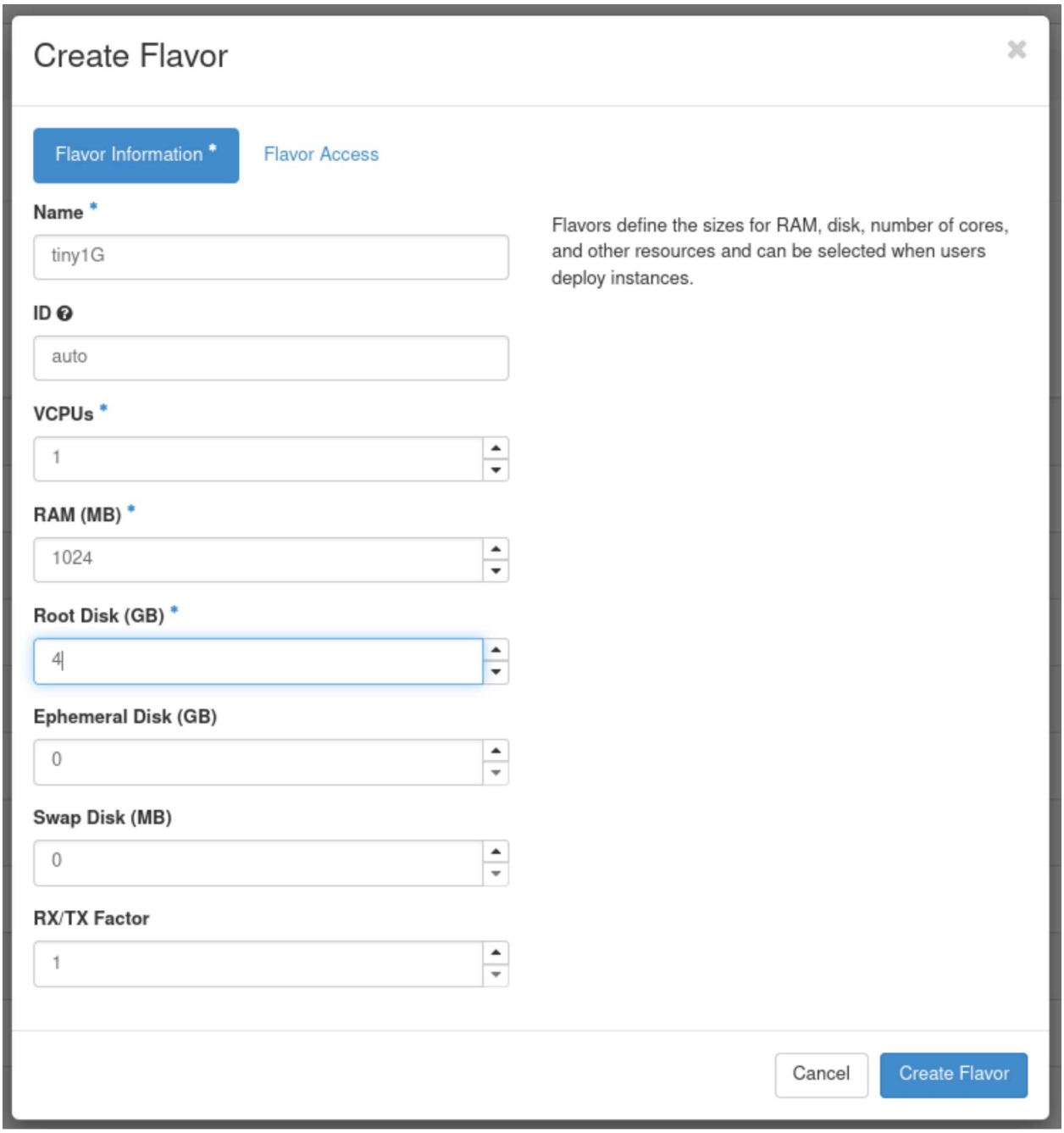
A VM flavor is a predefined configuration of vCPUs, RAM, and disk space. Our Ubuntu_2204 image needs 2G of RAM and 4G of disk space, however there isn't a pre-defined flavor that has this so we will make one.

7. Go to Admin → Compute → Flavors and click "Create Flavor". Name it tiny1G, set vCPUs to 1, RAM to 1024, and Root Disk to 4.


Lab06_Data_Center
ECE577_Lab06_Data
ECE577 - OneDrive
WhatsApp
Flavors - OpenStack
192.168.42.11/dashboard/admin/flavors/
openstack.
demo
admin
Project
Admin
Overview
Compute
Hypervisors
Host Aggregates
Instances
Flavors
Images
Volume
Network
System
Identity
Admin / Compute / Flavors
Flavors
Filter
+ Create Flavor
Delete Flavors
Displaying 12 items
Flavor Name | VCPUs | RAM | Root Disk | Ephemeral Disk | Swap Disk | RX/TX factor | ID | Public | Metadata | Actions
cirros256 | 1 | 256MB | 1GB | 0GB | 0MB | 1.0 | c1 | Yes | Yes | Update Metadata
ds1G | 1 | 1GB | 10GB | 0GB | 0MB | 1.0 | d2 | Yes | Yes | Update Metadata
ds2G | 2 | 2GB | 10GB | 0GB | 0MB | 1.0 | d3 | Yes | Yes | Update Metadata
ds4G | 4 | 4GB | 20GB | 0GB | 0MB | 1.0 | d4 | Yes | Yes | Update Metadata
ds512M | 1 | 512MB | 5GB | 0GB | 0MB | 1.0 | d1 | Yes | Yes | Update Metadata
m1.large | 4 | 8GB | 80GB | 0GB | 0MB | 1.0 | 4 | Yes | Yes | Update Metadata
m1.medium | 2 | 4GB | 40GB | 0GB | 0MB | 1.0 | 3 | Yes | Yes | Update Metadata
m1.micro | 1 | 192MB | 1GB | 0GB | 0MB | 1.0 | 84 | Yes | Yes | Update Metadata
m1.nano | 1 | 128MB | 1GB | 0GB | 0MB | 1.0 | 42 | Yes | Yes | Update Metadata
m1.small | 1 | 2GB | 20GB | 0GB | 0MB | 1.0 | 2 | Yes | Yes | Update Metadata
m1.tiny | 1 | 512MB | 1GB | 0GB | 0MB | 1.0 | 1 | Yes | Yes | Update Metadata
m1.xlarge | 8 | 16GB | 160GB | 0GB | 0MB | 1.0 | 5 | Yes | Yes | Update Metadata
Displaying 12 items
3:32 PM
6/27/23

8.

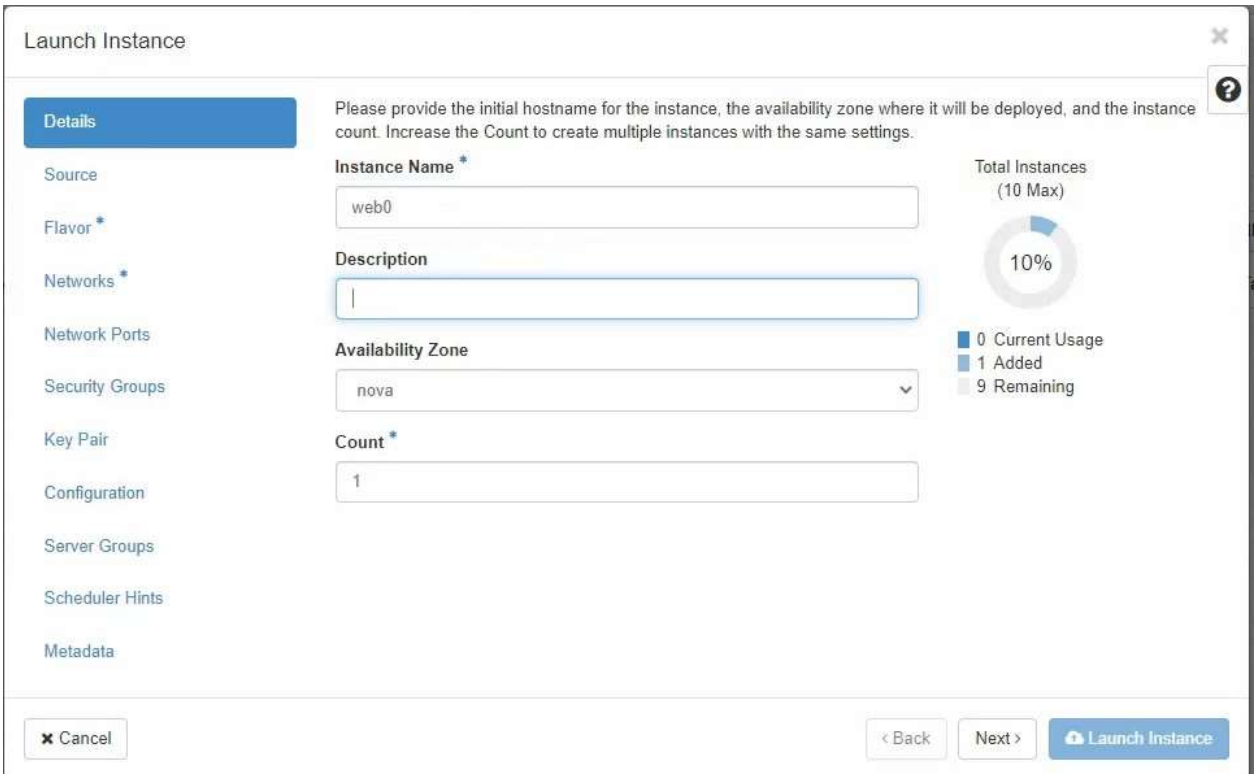

Lab06_Data_Center
ECE577_Lab06_Data
ECE577 - OneDrive
WhatsApp
Flavors - OpenStack
192.168.42.11/dashboard/admin/flavors/
openstack.
demo
admin
Admin / Compute / Flavors
Flavors
Create Flavor
Flavor Information *
Flavor Access
Name *
tiny1G
ID
auto
VCPUs *
1
RAM (MB) *
1024
Root Disk (GB) *
4
Ephemeral Disk (GB)
0
Swap Disk (MB)
0
RX/TX Factor
1
Flavors define the sizes for RAM, disk, number of cores, and other resources and can be selected when users deploy instances.
Cancel
Create Flavor
Displaying 12 items
3:34 PM
6/27/23

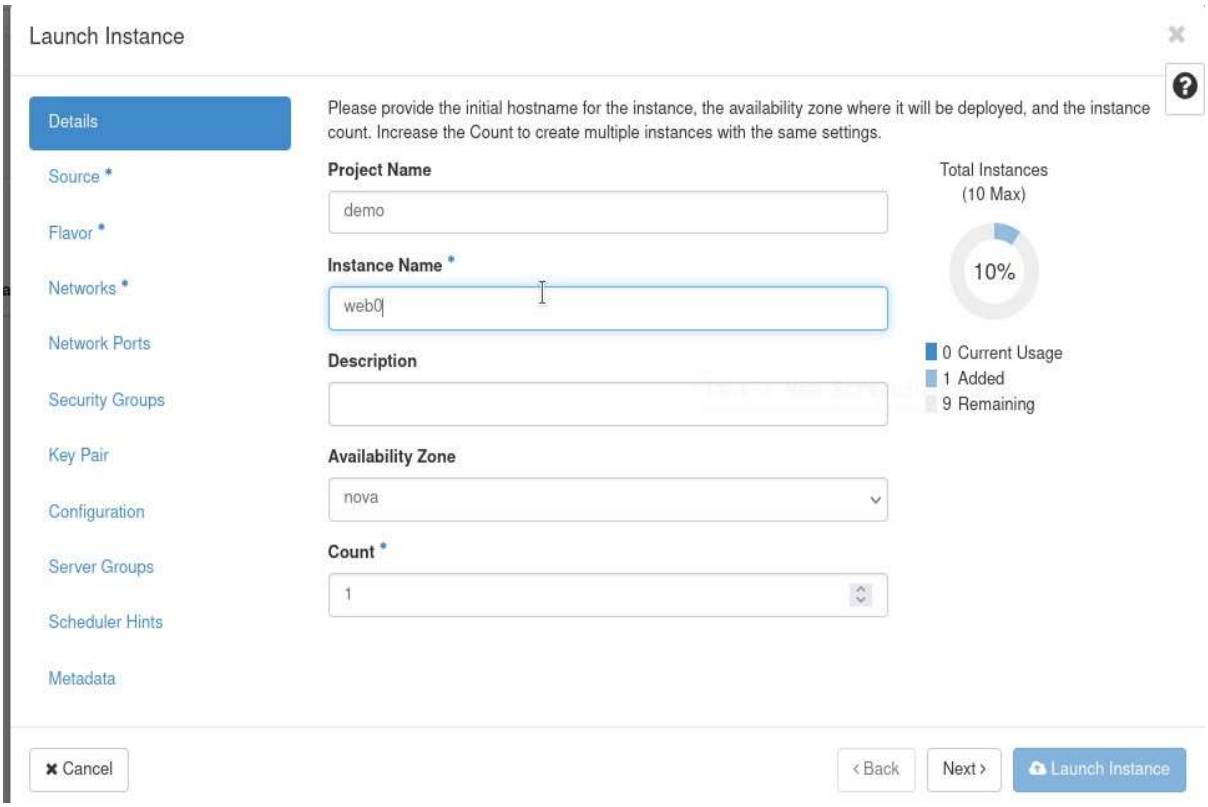


Click Create Flavor.

Creating VMs Through Web Console

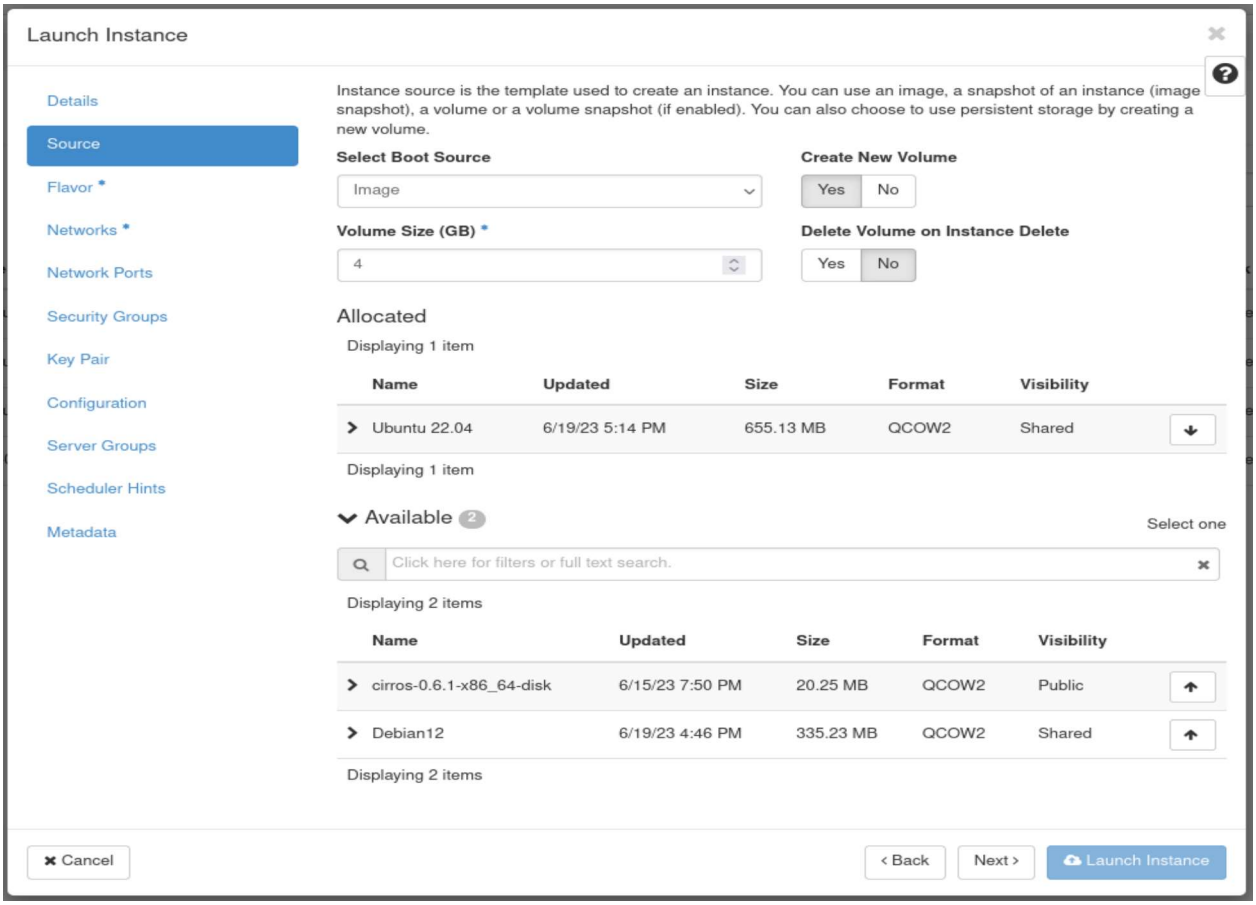
9. Go to Project → Compute → Instances. Click “Launch Instance”. Create an instance named “web0” and click Next

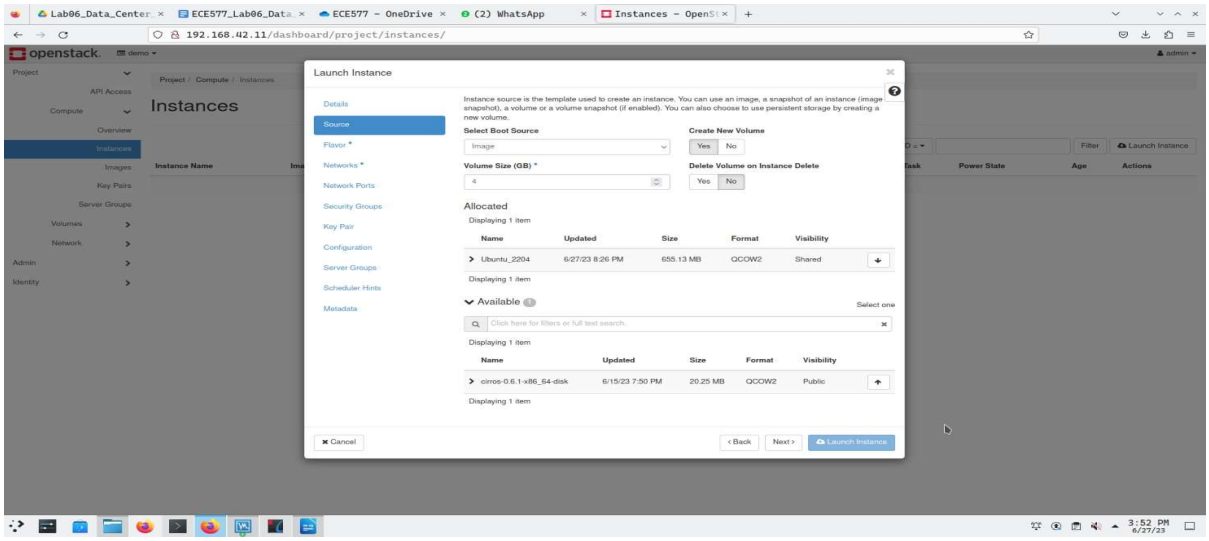




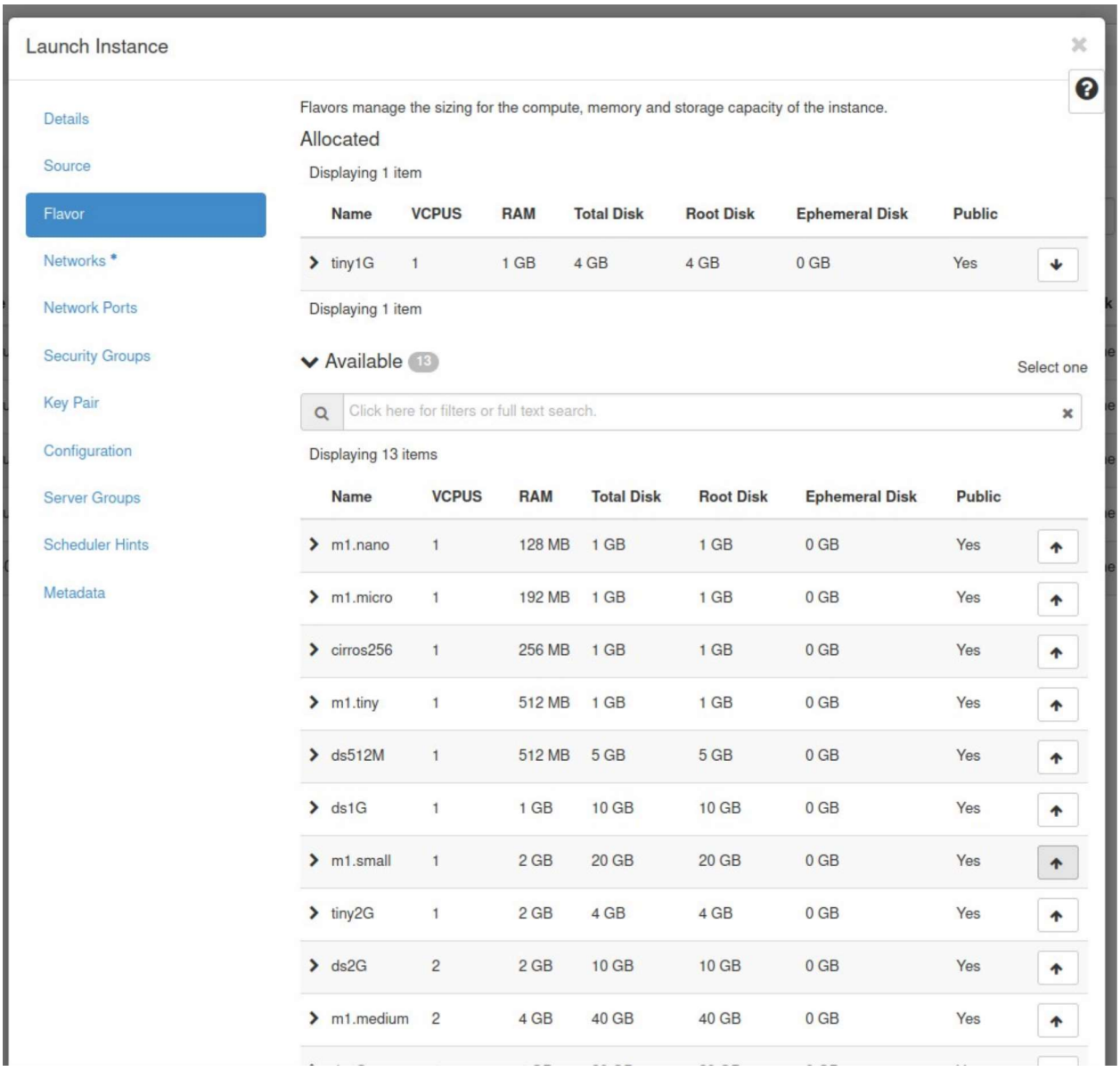
10. Select “Image” as Boot Source from dropdown, make sure Create a New Volume is checked, set Volume Size to 4 and make sure Delete Volume on Instance

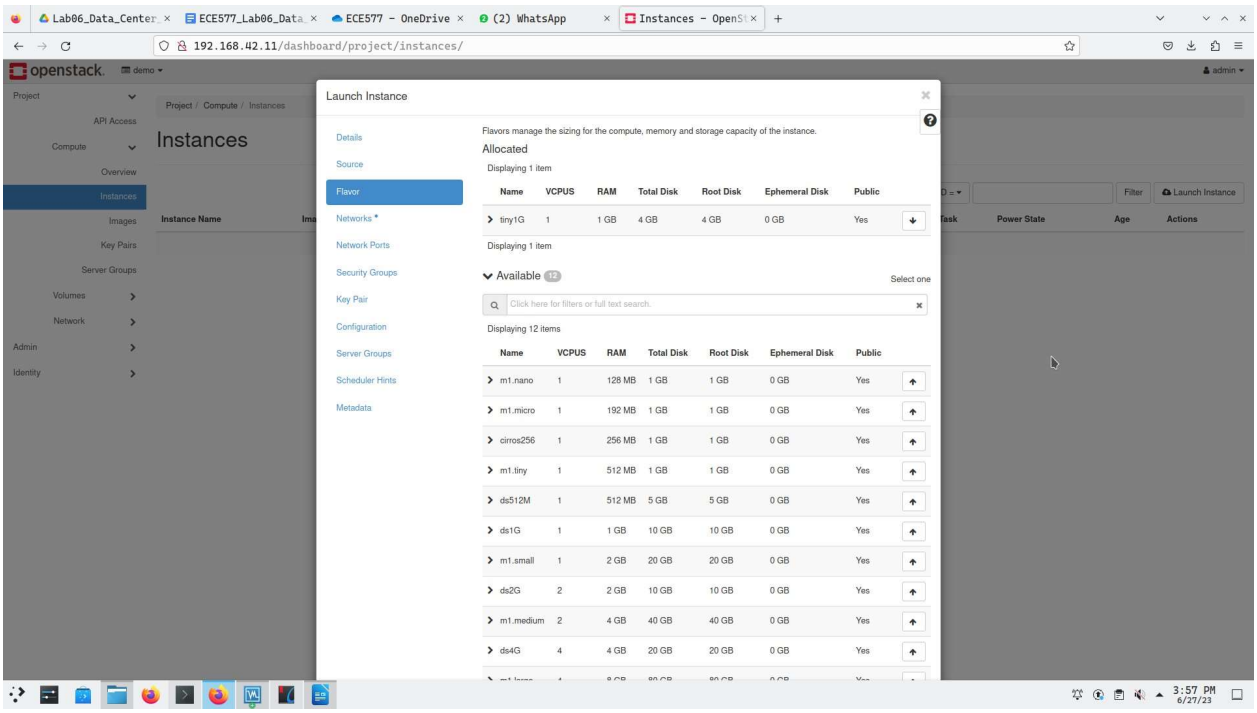
Delete is checked. Select the Ubuntu_2204 image and click Next.





11. select tiny1G as the Flavor and click Next.





Step B - Live Migrating The Machine

Make sure the web0 machine is assigned a floating IP and that you can ping that floating IP from the host.

Start a continuous ping on the floating IP ($ping -t in Windows). Keep an eye on this.

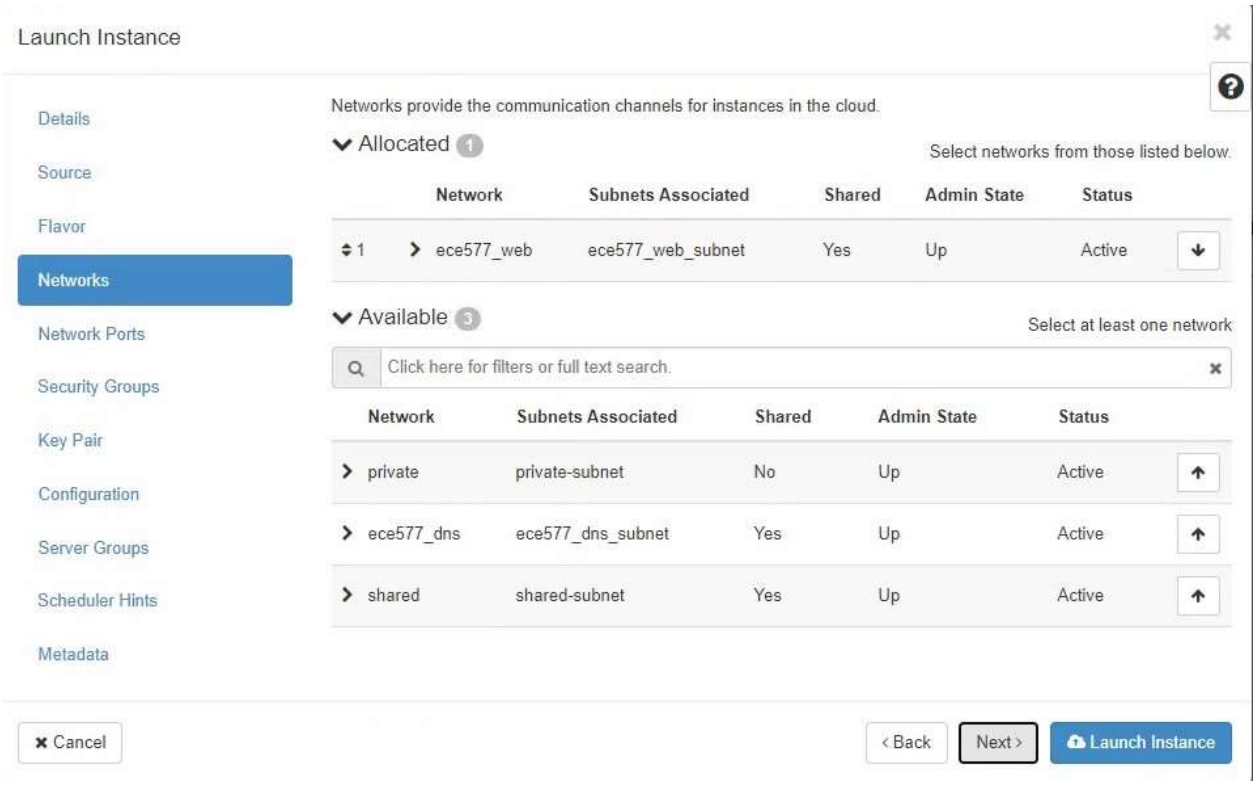
Go to Admin → Compute → Instances and select “Live Migration” from the drop-down next to web0:

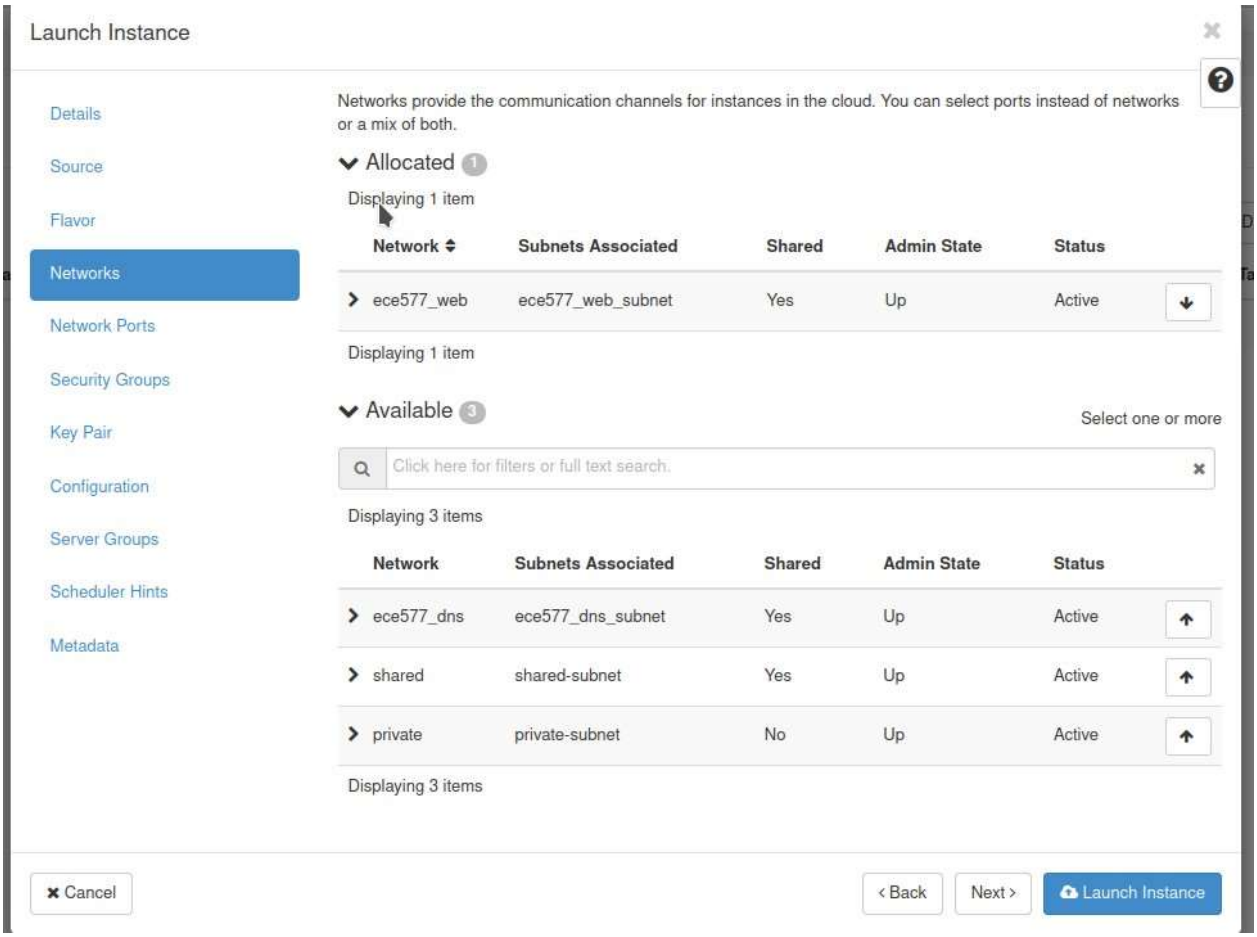
You will be prompted to select the new host node, so select compute0

Verify that web0 is now running on compute0

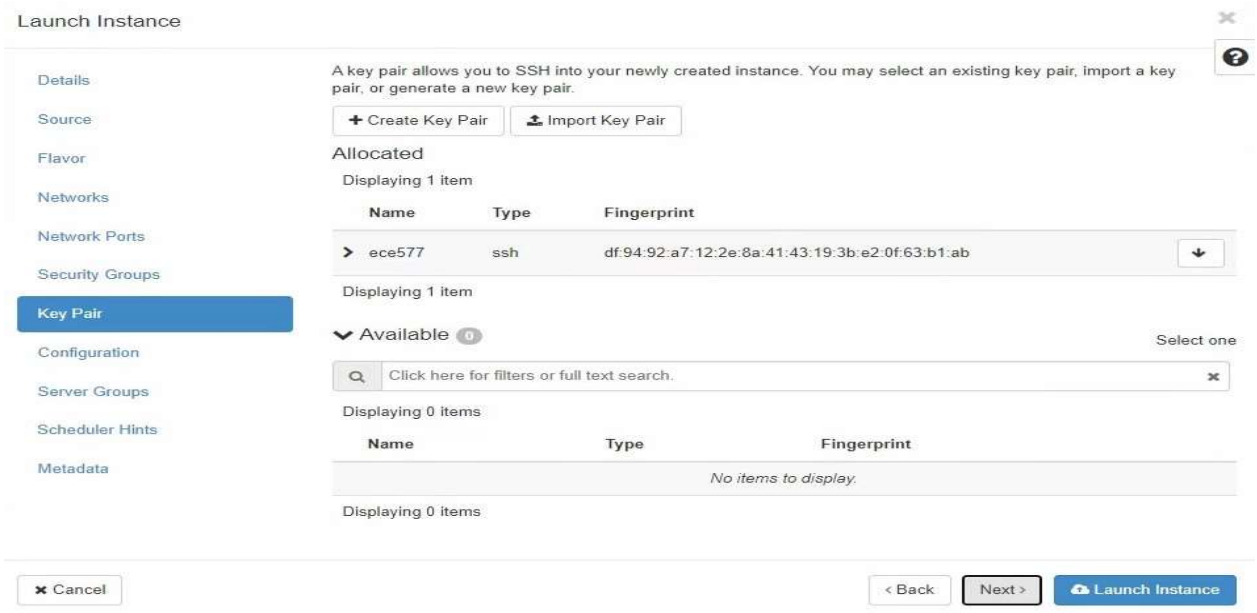
Go back to your host Terminal and see how many (if any) pings were lost during the migration

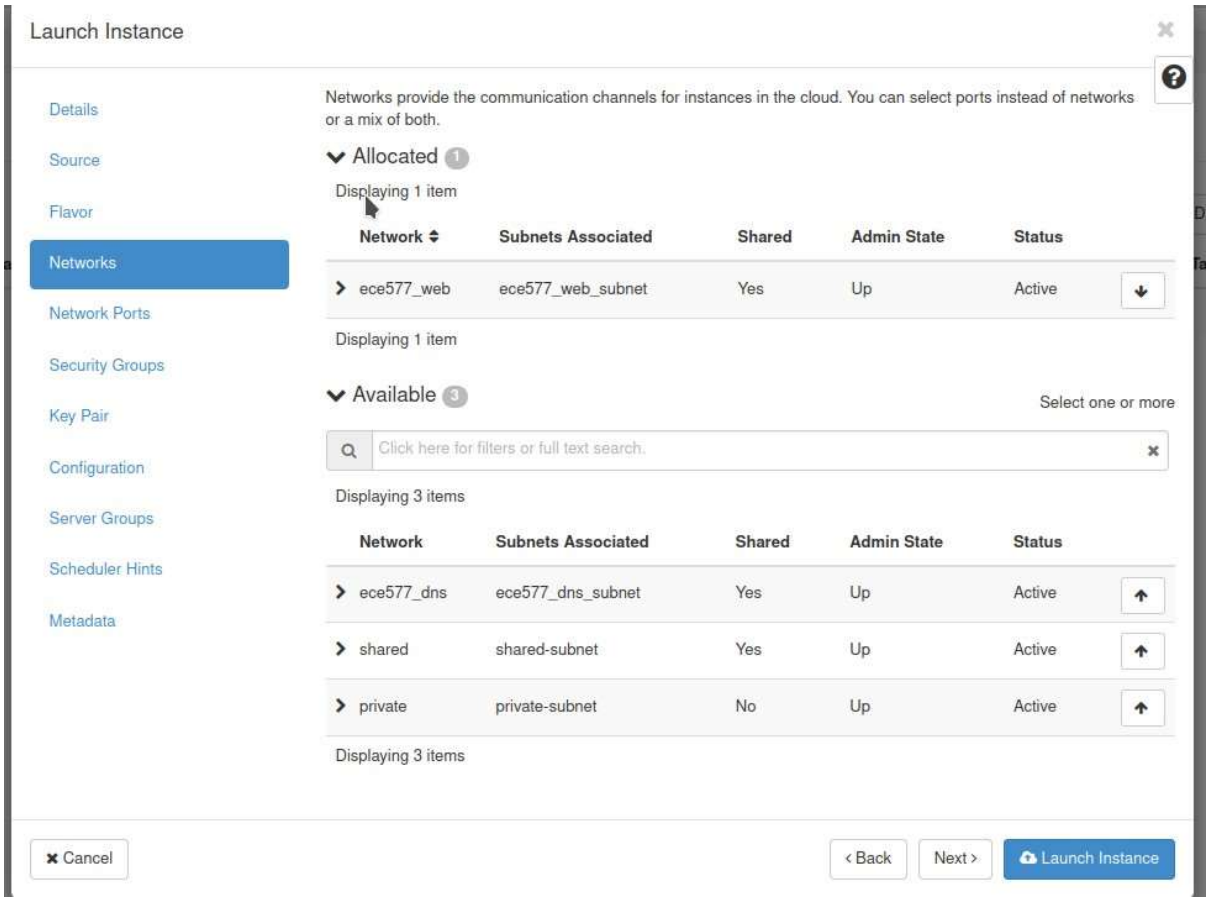
11. Select the ece577_web Network and click Next.


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
ece577_web
ece577_web_subnet
Yes
Up
Active
Available 3
Select at least one network
Click here for filters or full text search.
Network
Subnets Associated
Shared
Admin State
Status
private
private-subnet
No
Up
Active
ece577_dns
ece577_dns_subnet
Yes
Up
Active
shared
shared-subnet
Yes
Up
Active
Cancel
Back
Next
Launch Instance

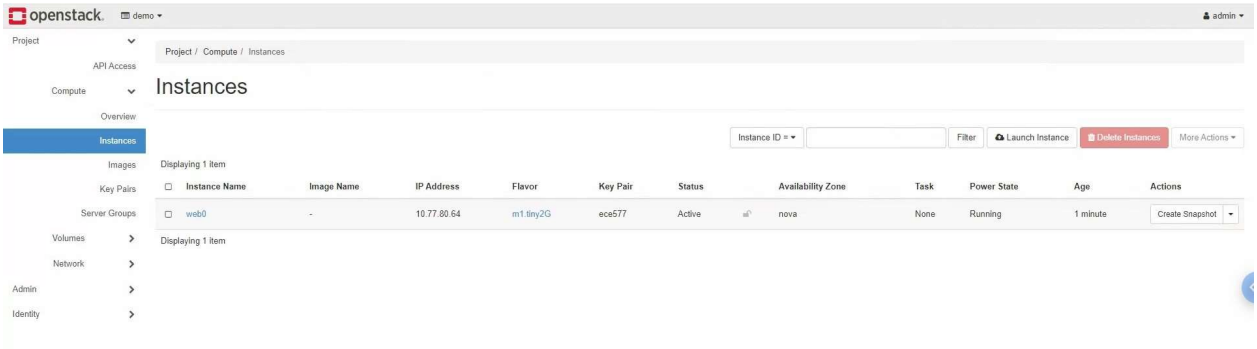


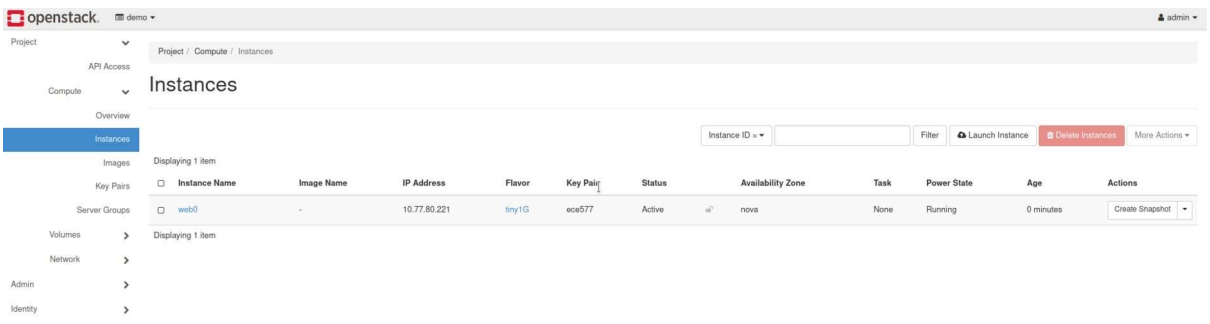
12. Click Next on Network Ports, Next on Security Groups, and make sure the ece577 Key Pair is selected. Click Launch Instance



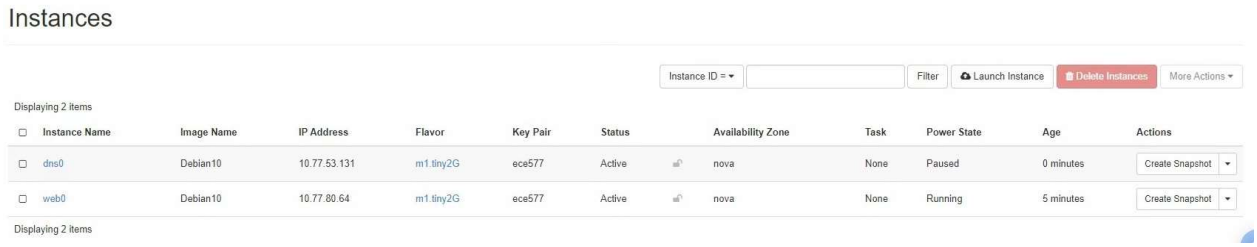


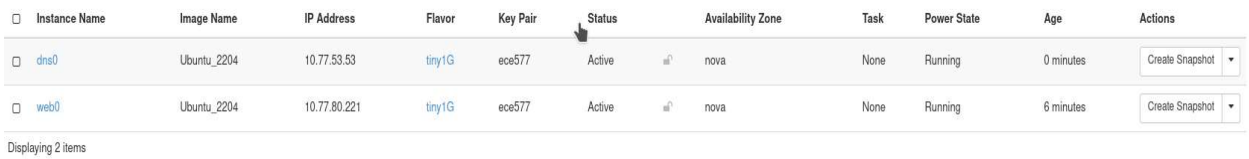
13. It may take a few minutes (it is, after all, a nested VM, running on a loopback disk image mounted on a virtual disk, so we probably couldn't be less efficient if we tried!) but if everything is successful, it should be running!





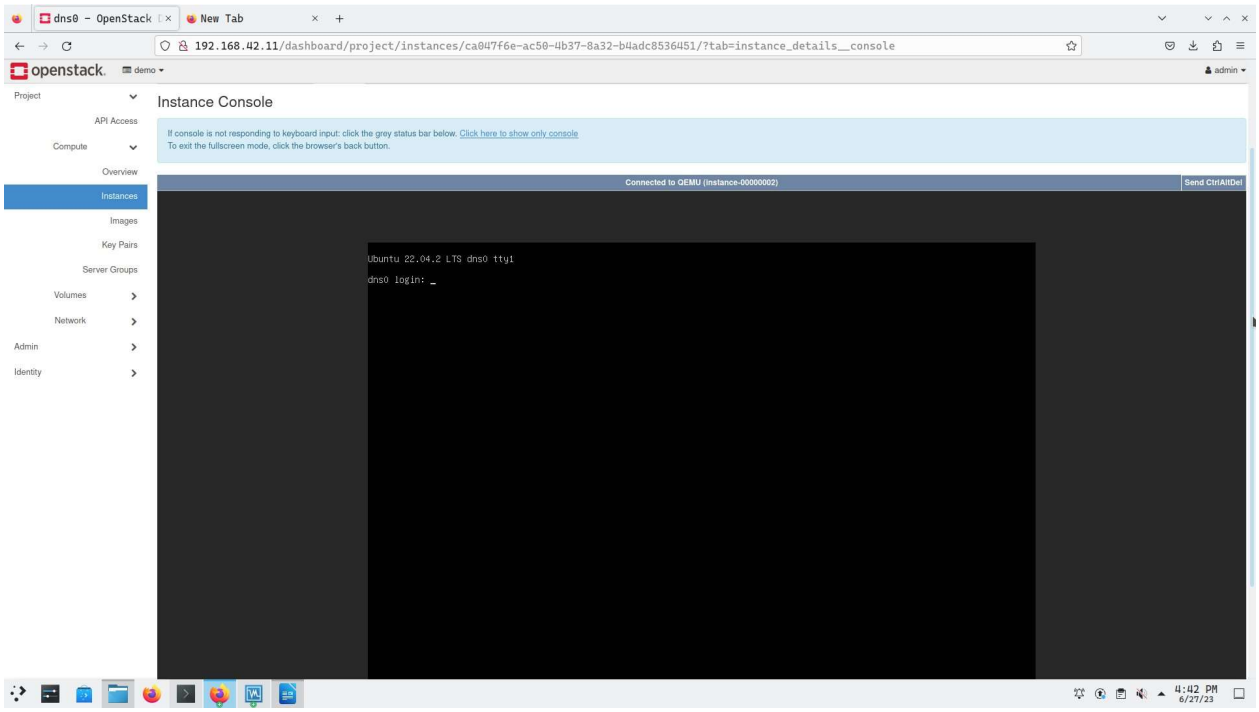
14. Repeat the steps to create another VM, but this time name it dns0 and place it on the ece577_dns Network. (Your IPs will vary as they are assigned by the OpenStack DHCP server)





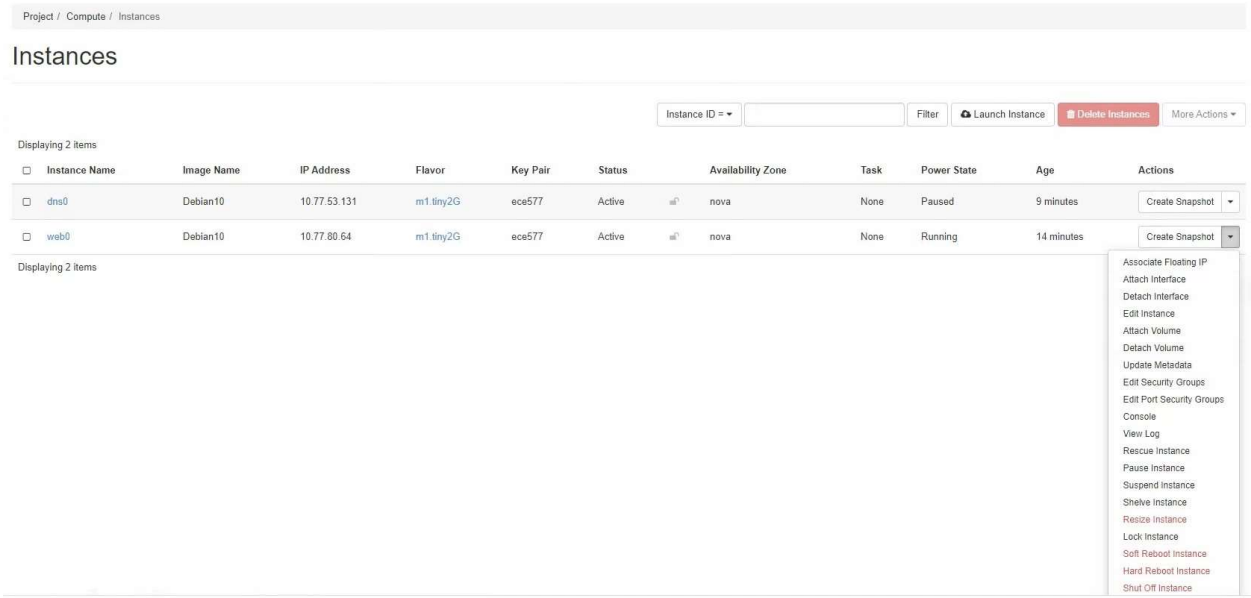
Accessing the VMs

There isn't much to do at this point, as the machine, while running, is not practically accessible due to the fact that the root/user accounts are locked, and there is no default password! (There is also NO route from your compute machine to the internal network yet either!) You can click on the VM name under Instances and click on the Console tab to see the VM's console, but at this point, it won't do you much good. The only way to log in to your machine is to access it via SSH. However this requires three things, the SSH Key (which you should have in your ece577.pem file), a route to the machine (which you currently don't as the ece577_web and ece577_dns networks are internal, tenant networks), and a security rule that allows SSH (The default rule blocks all inbound traffic!)

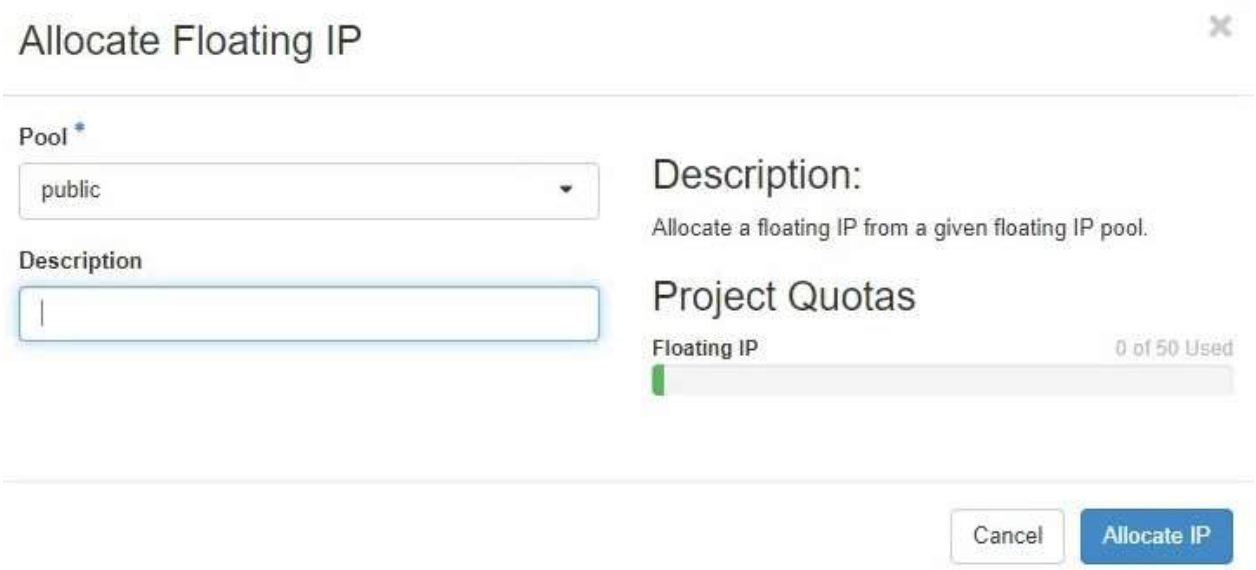


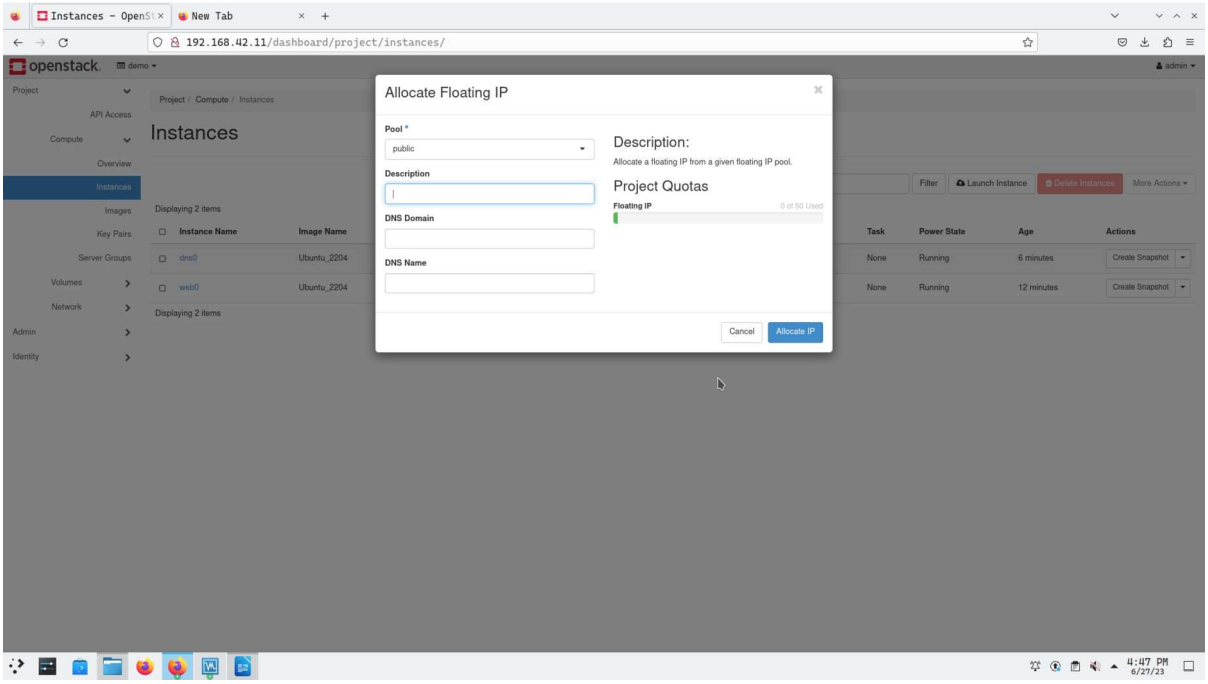
Assigning a Public Floating IP

15. Under Project → Compute → Instances there is a drop-down menu with options for your VMs. Pick the web0 VM first and select the “Associate Floating IP” option

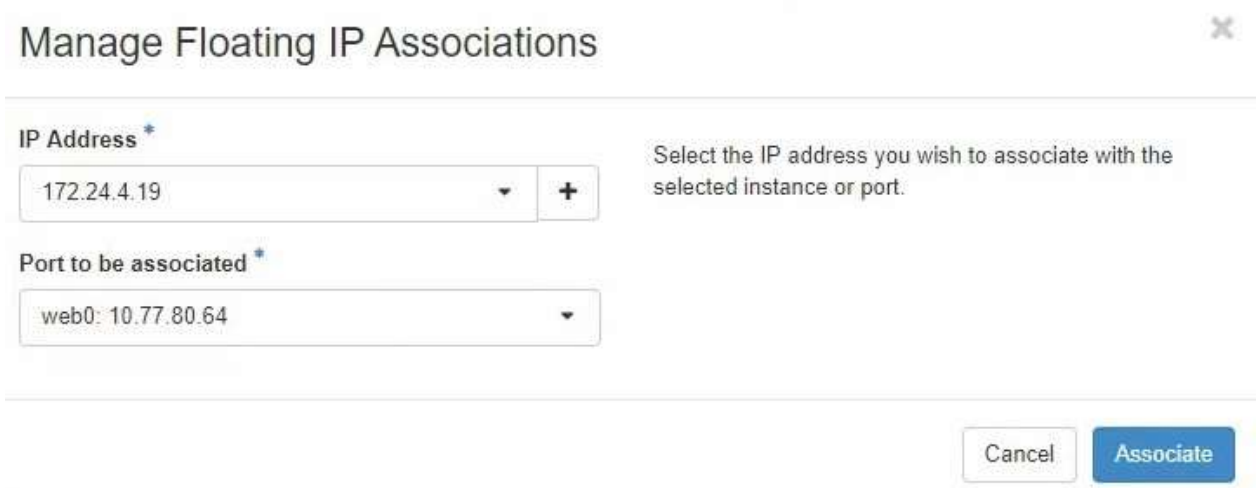


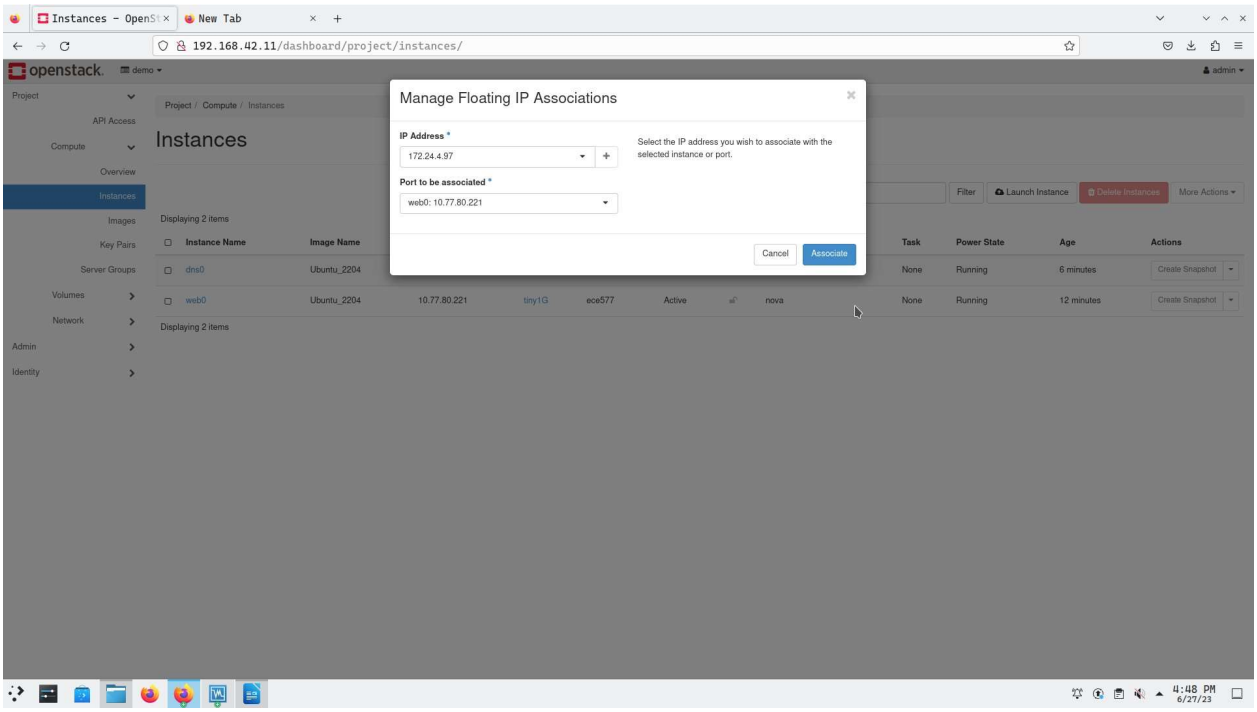
16. Click the + Next to IP Address and when the IP selection screen pops up, click Allocate IP





17. That will bring you back to the original screen and you should then see an IP in the 172.24.4.0/24 network. Click Associate.



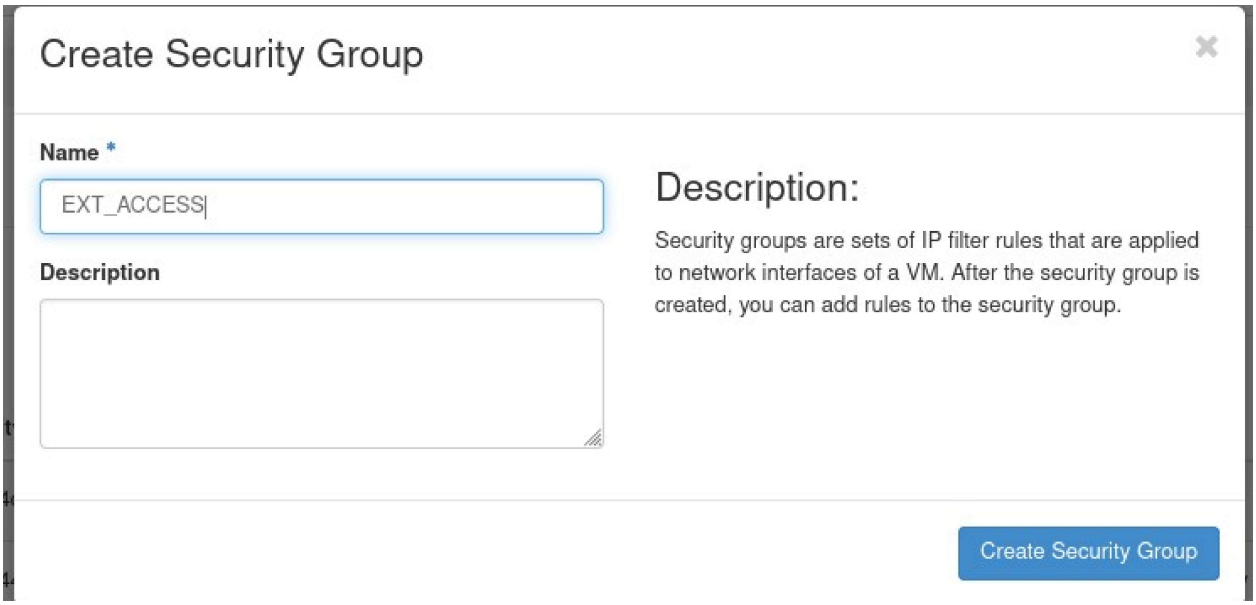
18. 

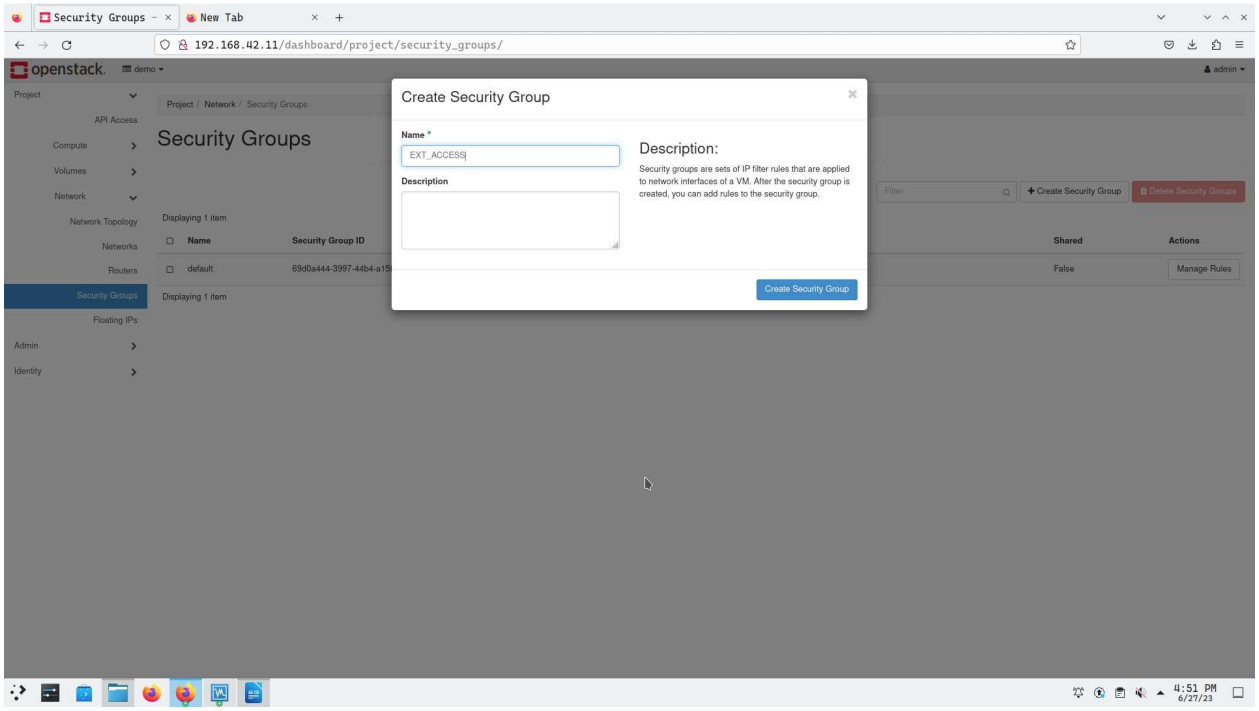
Your machine is now externally accessible on the 172.24.4.0/24 network! Now, let's fix the SSH security rule

Network Security Groups

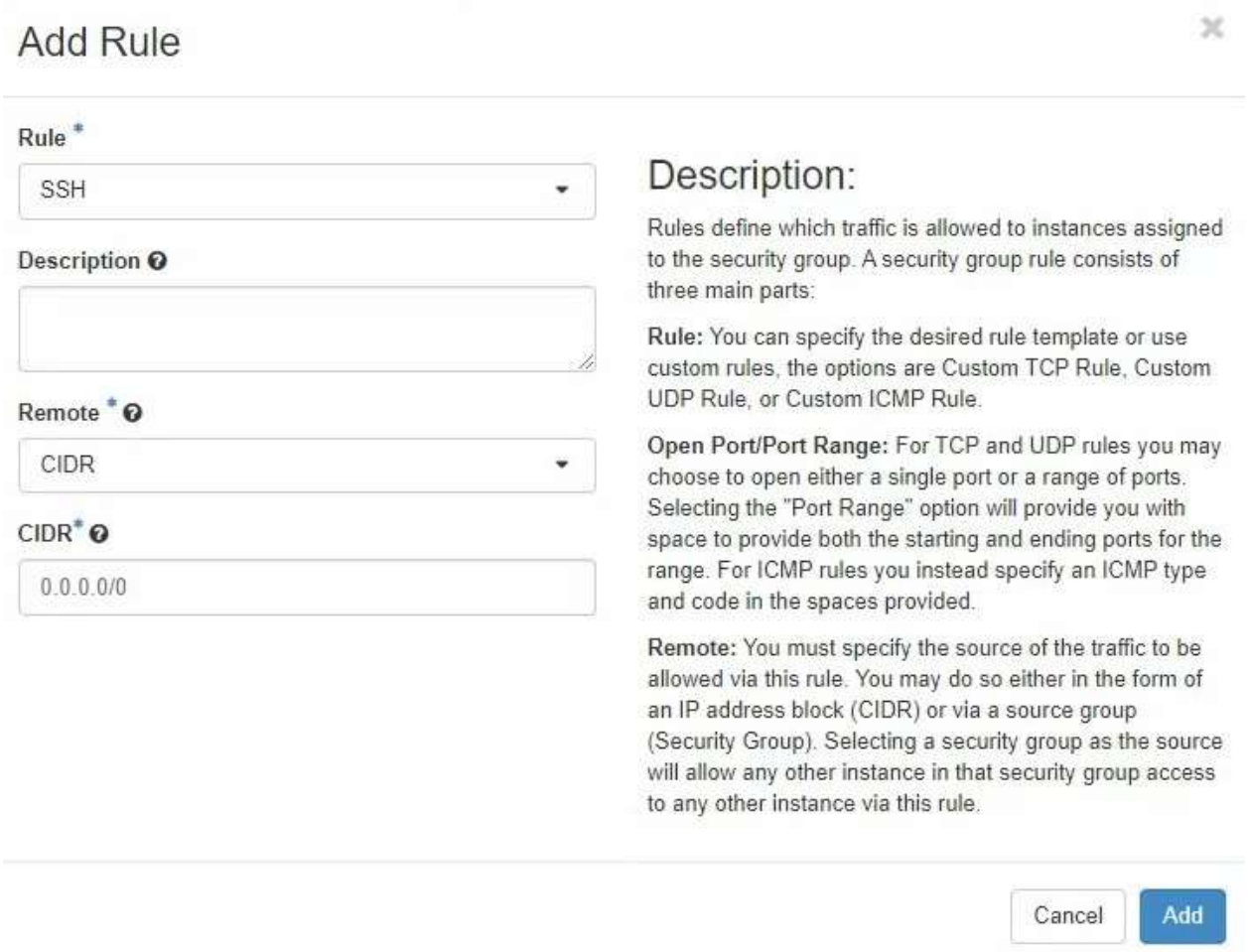
Security Groups are a set of firewall rules that can be applied selectively to individual VM interfaces. The default group allows all outbound traffic but no inbound traffic. While good for a default policy, we would like to access our newly created VMs. Let's create a rule that allows us to do so

19. Click In Project → Network → Security Groups. Click "Create Security Group"
20. Name it "EXT_ACCESS" and click "Create Security Group"

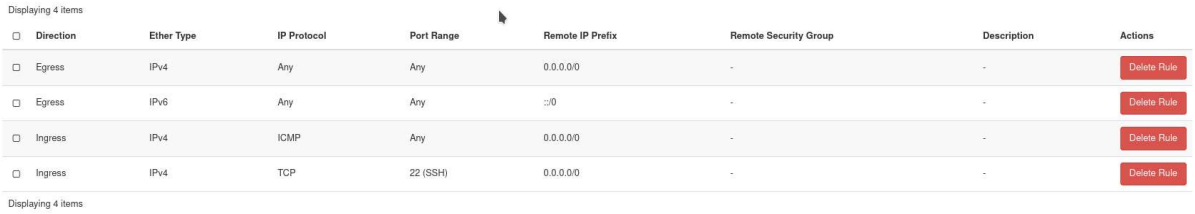


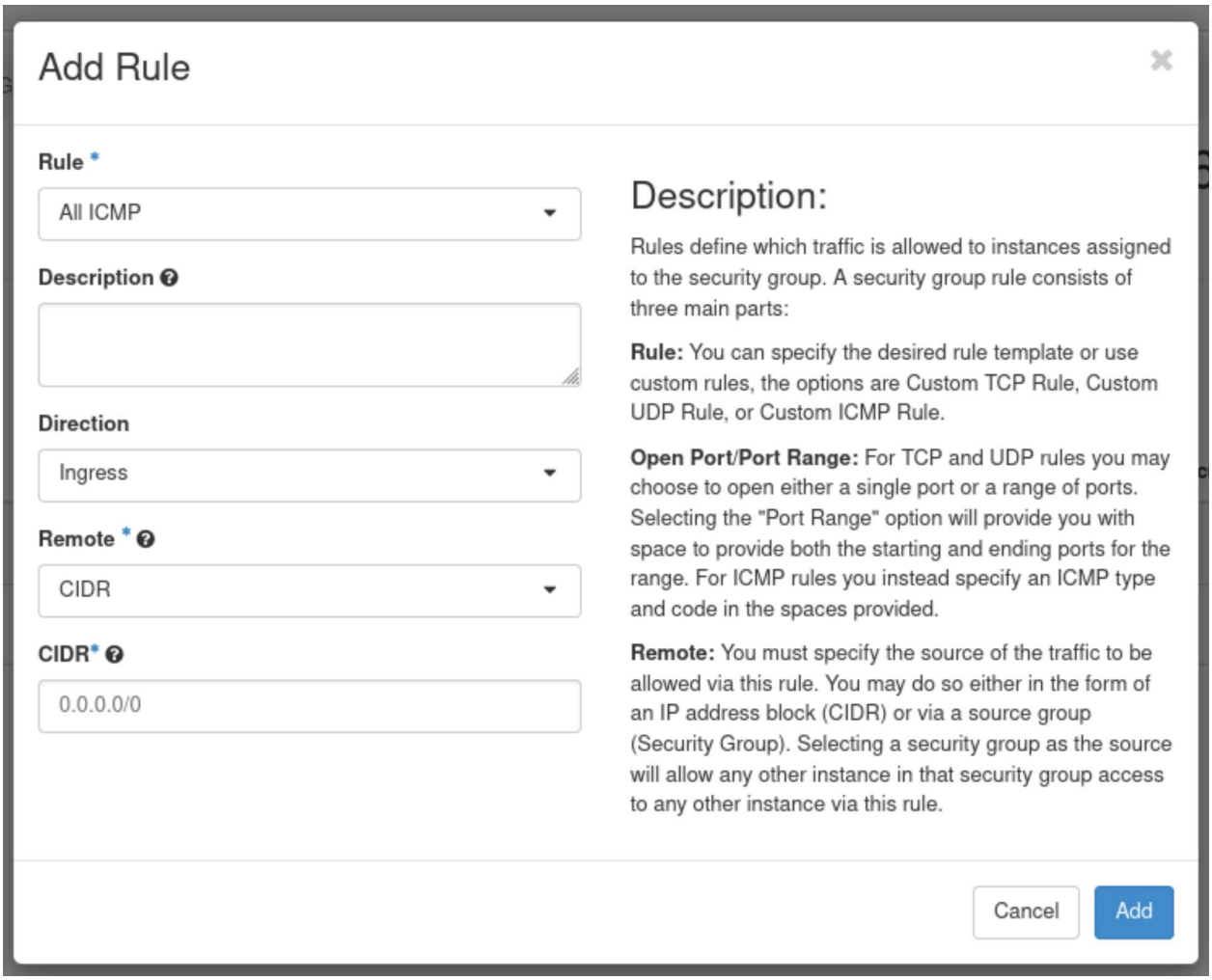


20. Click “Add Rule”, select “SSH” under the “Rule” drop down and click “Add”

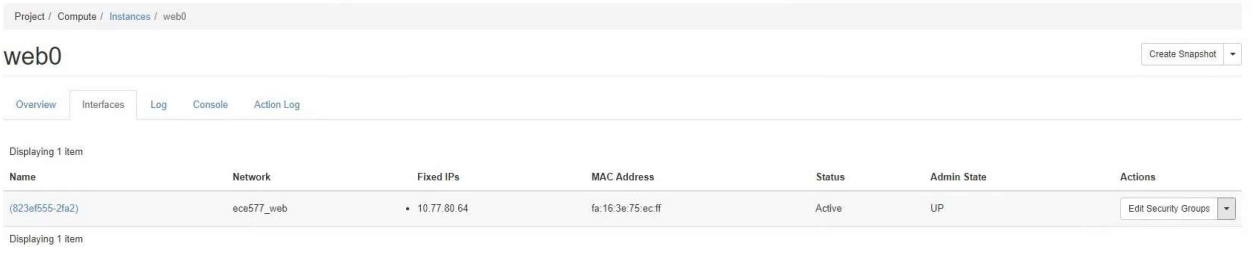


22. Click “Add Rule”, select “All ICMP” under the “Rule” drop down and click “Add”:

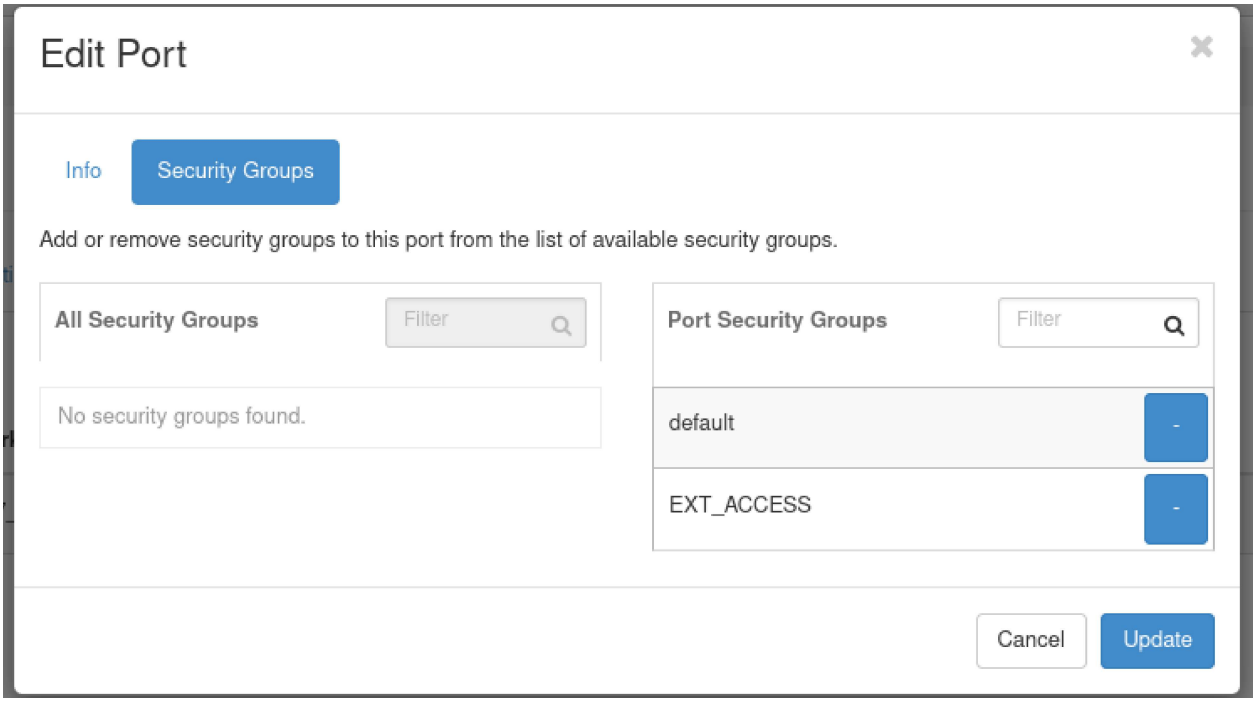




23. Go back to Project → Compute → Instances and click on the web0 VM. Click on the Interfaces tab and click the "Edit Security Groups" option under Actions:



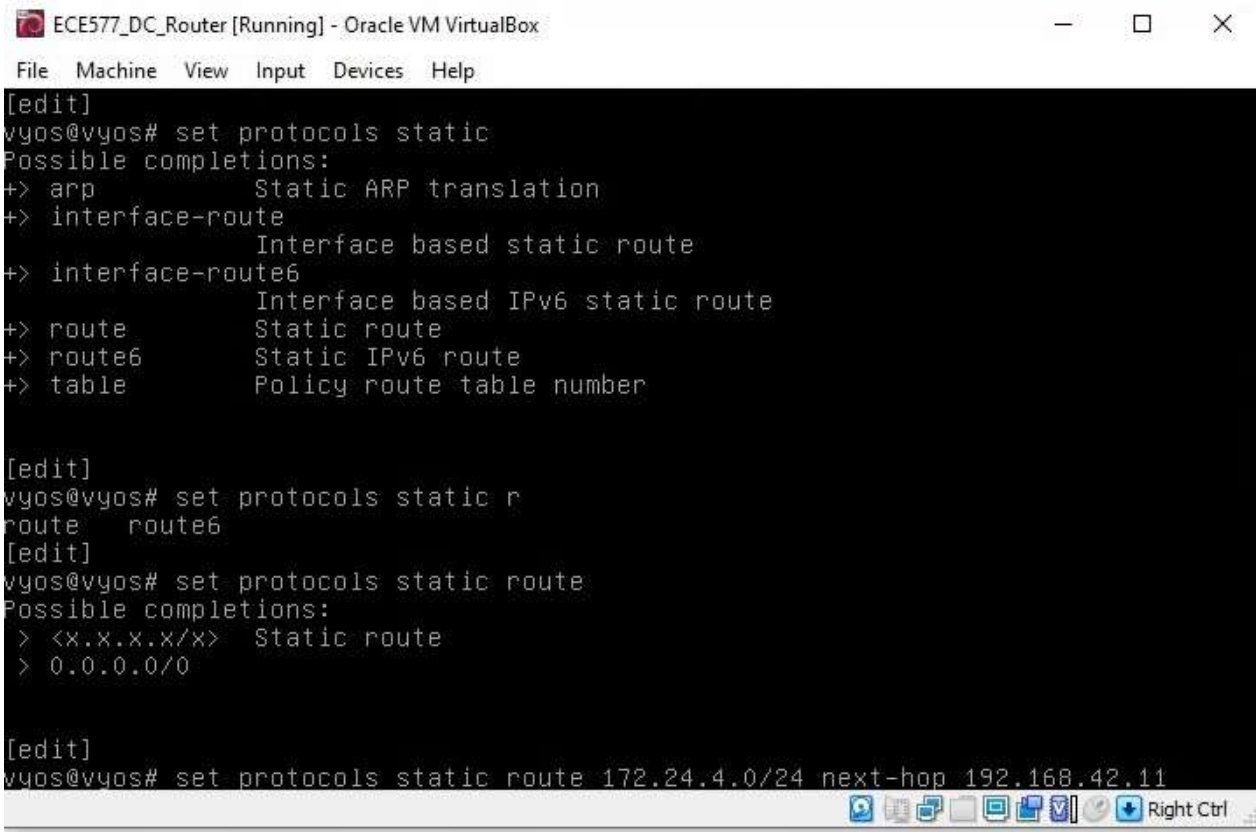
12. 24. Click the + next to the EXT_ACCESS group and click “Update”



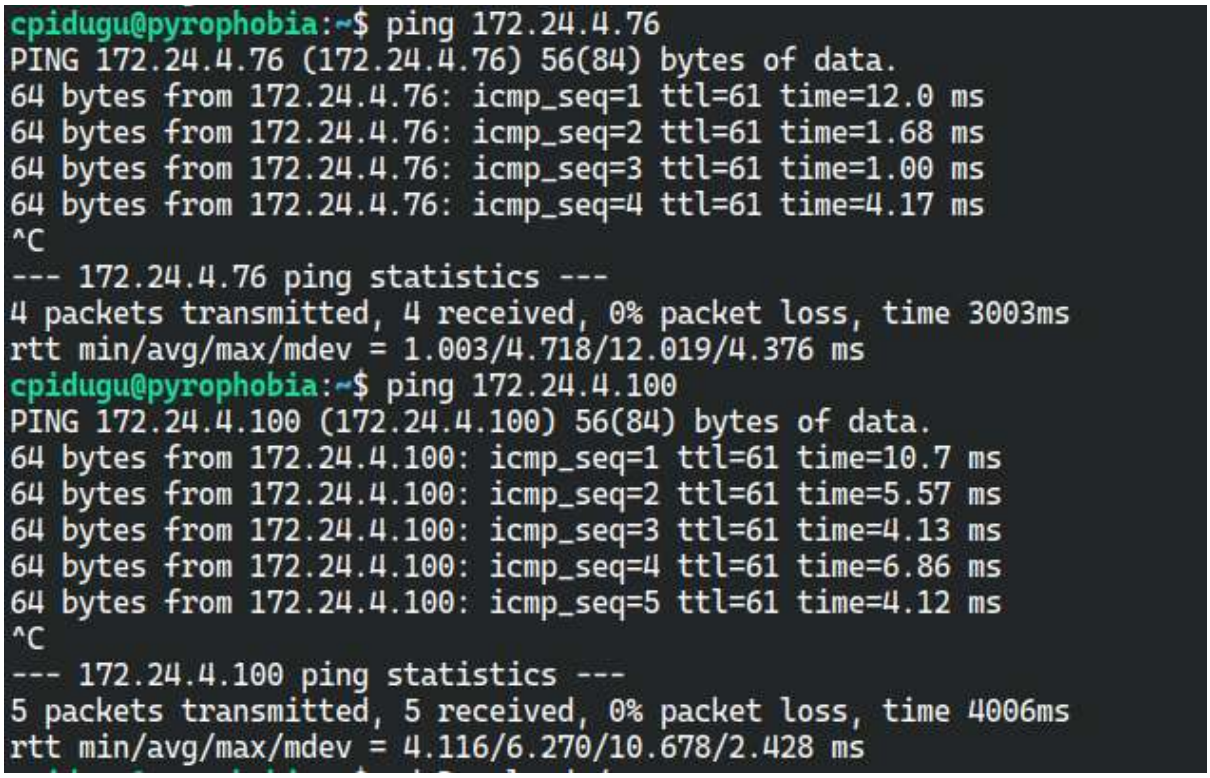
Route To Public Subnet

We can now at least SSH into our VM, BUT at this point, only the controller node has access to the 172.24.4.0/24 network. So, to fix this and to allow external devices to SSH into it (like what would happen in a real data center), we need to add routes to this network.

25. Go to your VyOS VM and add a static route to the 172.24.4.0/24 subnet through your Controller’s 192.168.42.11 interface (YES, I KNOW THIS SHOULD BE ACCESSIBLE DIRECTLY ON THE PUBLIC NETWORK, BUT BEAR WITH ME HERE)



26. Now that your VyOS machine knows how to get to 172.24.4.0/24, inform your host
    a. eid@whateverphobia$ sudo ip route add 172.24.4.0/24 via 10.0.1.1XX
27. Your host should now have access to the machine. You should be able to ping it at this point, which, if it were in a real datacenter, would have a real, public facing IP address and global reachability!



SSH Login

28. You can only log in with your key.
    b. You should just be able to eid@whateverphobia$ ssh -i ece577.pem ubuntu@172.24.4.ZZZ (whatever floating IP was assigned) from a terminal to get access.

```
cpidugu@pyrophobia:~/Downloads$ sudo chmod 600 ece577.pem
[sudo] password for cpidugu:
cpidugu@pyrophobia:~/Downloads$ ssh -i ece577.pem ubuntu@172.24.4.76
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-75-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jun 28 17:09:43 UTC 2023

  System load:  0.15380859375     Processes:             85
  Usage of /:   38.3% of 3.70GB   Users logged in:       0
  Memory usage: 18%               IPv4 address for ens3: 10.77.80.50
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@web0:~$
```

29. If you don't want to have to do all this again in the future, you can give your ubuntu user an actual password for logging in through the web console, so go ahead and do that now ($ sudo passwd ubuntu), but this is actually pretty uncommon to do in a real data center environment, as you want only tenants with the key to have direct access to machines for security reasons.

```
ubuntu@web0:~$ sudo passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
ubuntu@web0:~$
```

30. Repeat this process for the dns0 VM.

```
cpidugu@pyrophobia:~/Downloads$ ssh -i ece577.pem ubuntu@172.24.4.100
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-75-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jun 28 17:16:01 UTC 2023

  System load:  0.26025390625     Processes:             86
  Usage of /:   38.3% of 3.70GB   Users logged in:       0
  Memory usage: 18%               IPv4 address for ens3: 10.77.53.53
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@dns0:~$ sudo passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
ubuntu@dns0:~$
```

ASSIGNMENT PART 1:

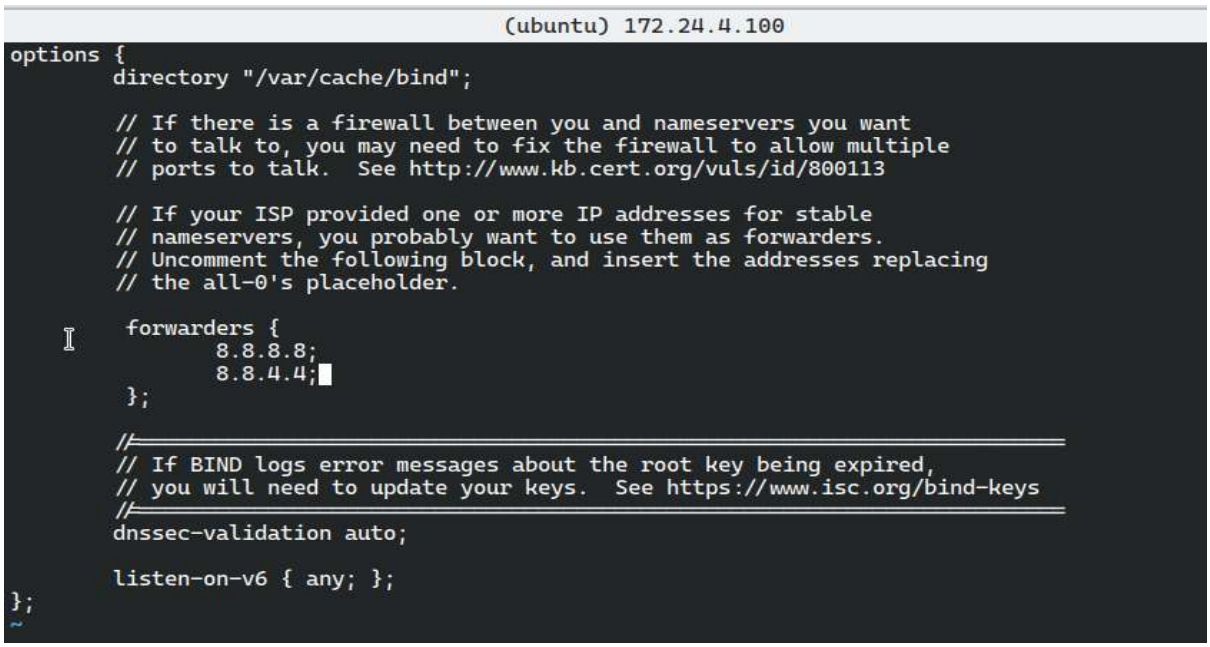
Your VMs are fully operational, you can assign them reachable IPs, but at this point they can't actually access the Internet!

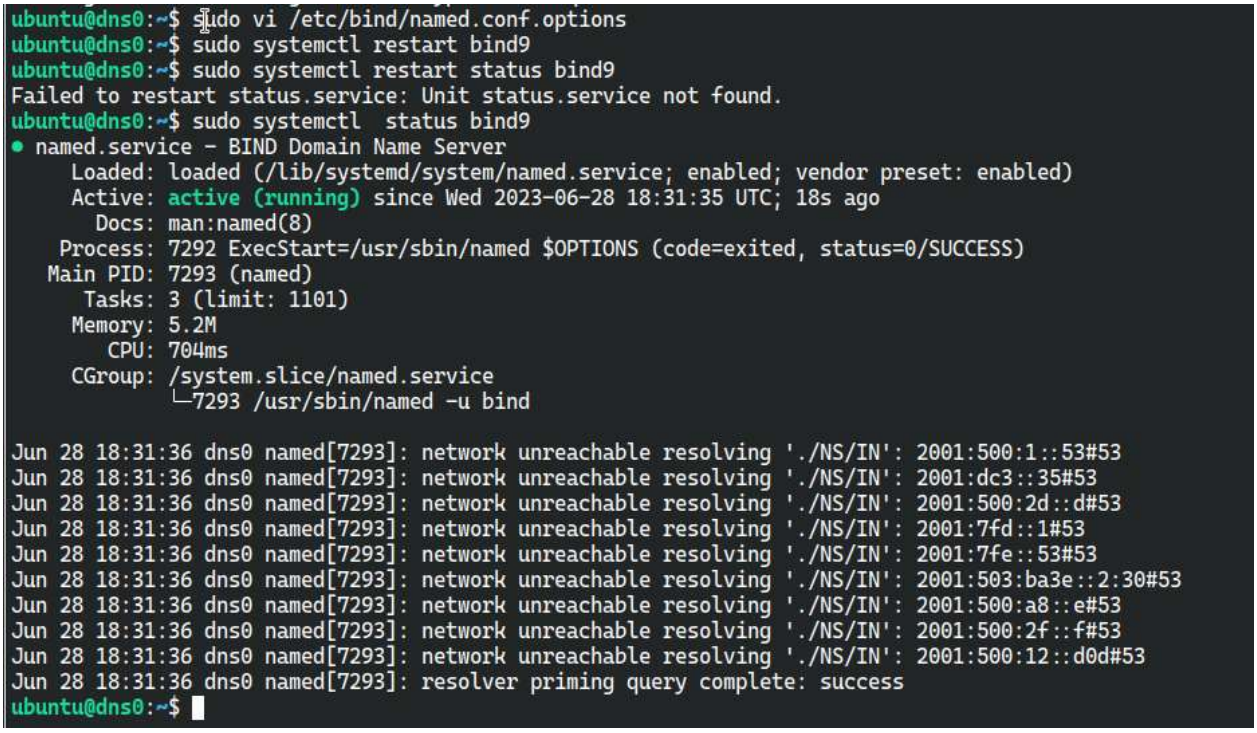
1) Your job here in the assignment is to figure out how to get your VMs to be able to access the Internet and it should work whether they have a floating IP assigned or not. Document and explain this in your report. It might work already through

NAT on the ECE577_DC_Gateway, but verify it is.

2) Once the Internet is working, install a Web Server on the web0 VM (any one, I don't really care, you could just do $ python -m SimpleHttpServer in a directory with an index.html file). Make a security rule that allows only HTTP & HTTPS into your web server

3) Install BIND on your dns0 VM. Listen for DNS requests through this VM like back in Lab 2. Don't forget to allow DNS with a security rule here. Reboot any VMs for this take effect if needed.



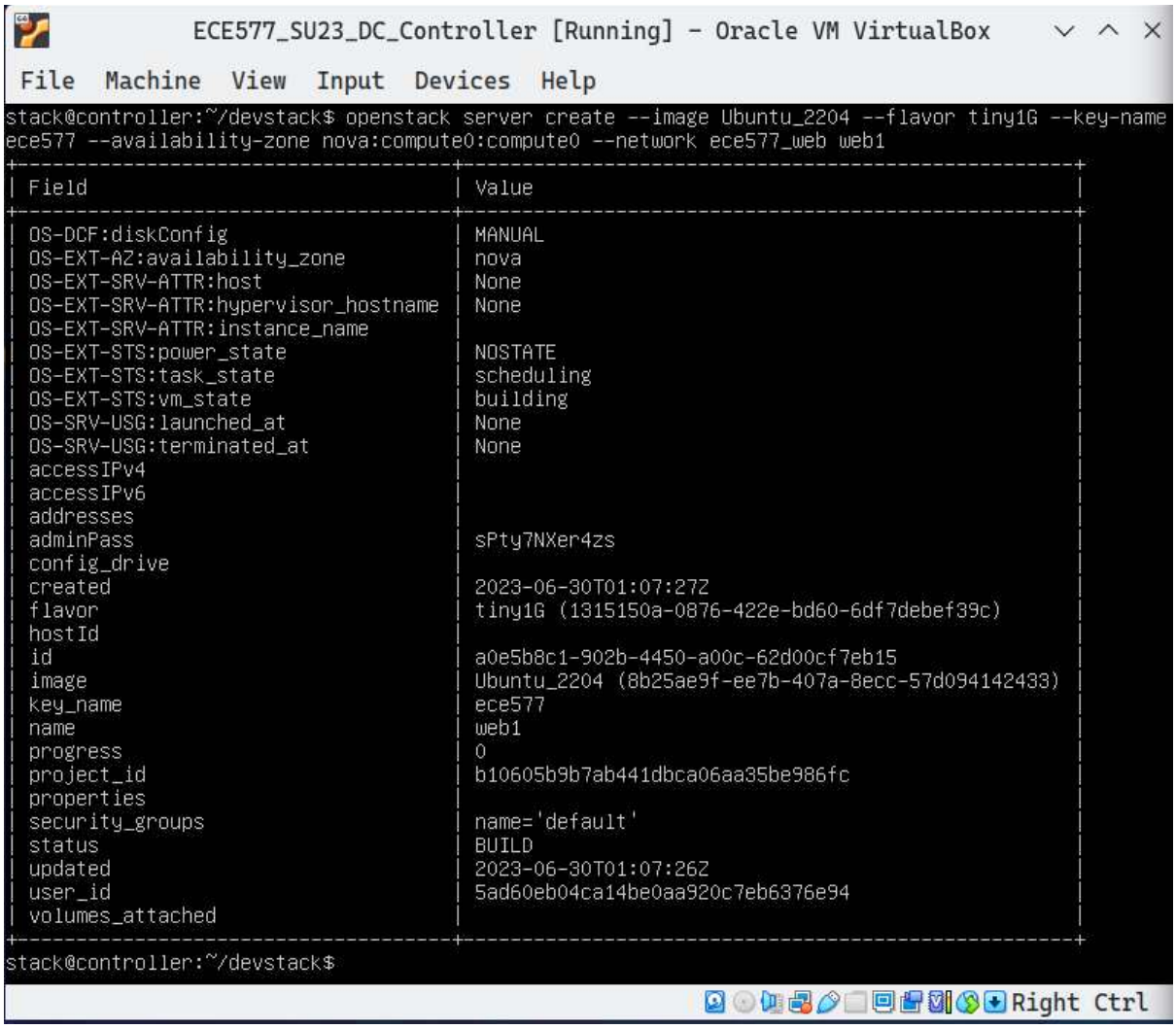


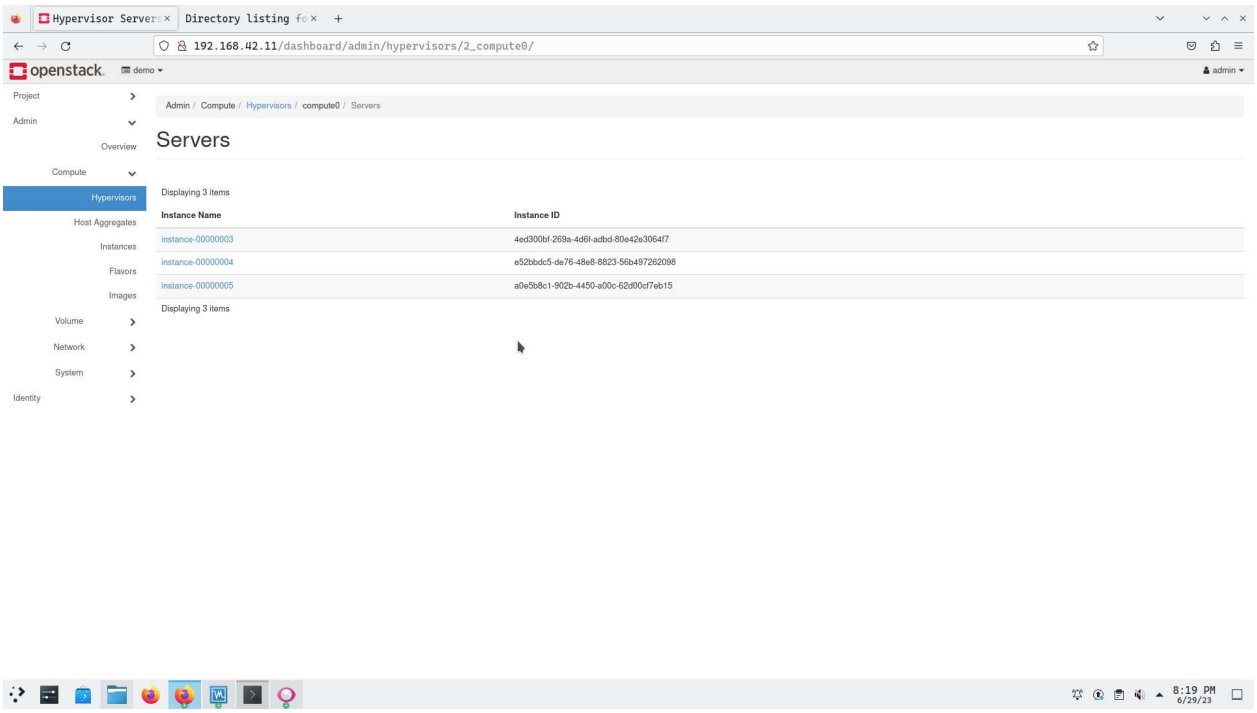
STEP D - VMs On ANY Host

The VMs created in the previous part are most likely running on the Controller node since that is where you launched them (and also it's the first node in the pool of

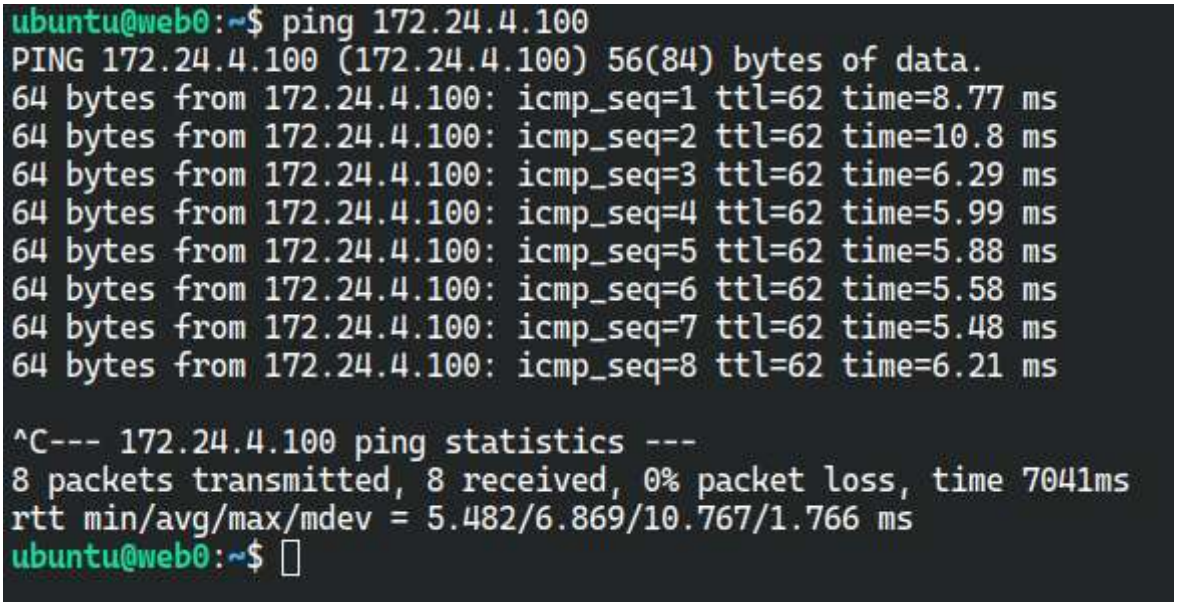
hypervisors). You can check which nodes are running them by going to Admin → Compute→ Hypervisors and looking. We are going to launch two new VMs explicitly on the compute0 node now and investigate how the overlay network is functioning to both.

1. Go to the terminal of your controller VM. As the stack user, you also have access to command line equivalents of all of the web interface stuff we just did. We are going to use this to create two new VMs.
2. Log in to your Controller node as ubuntu. Change into the stack user using the following command:
   ubuntu@controller$ sudo su - stack
3. Type the following command as stack:
   stack@controller:~/devstack$. openrc admin demo
4. This give you admin access to all openstack commands and places you in the demo project
5. Type the following to launch a new instance:
   stack@controller:~/devstack$ openstack server create --image Ubuntu_2204 --flavor tiny2G --key-name ece577 --availability-zone nova:compute0:compute0 --network ece577_web web1
6. Verify that it is running on compute0 by looking at the Hypervisors again
7. Create a Security Rule that allows ICMP if you haven't already. Apply that rule to both web0 and web1.
8. Attempt to ping from web0 to web1. It should be successful


ECE577_SU23_DC_Controller [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
stack@controller:~/devstack$ openstack server create --image Ubuntu_2204 --flavor tiny1G --key-name ece577 --availability-zone nova:compute0:compute0 --network ece577_web web1
Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	None
OS-EXT-SRV-ATTR:hypervisor_hostname	None
OS-EXT-SRV-ATTR:instance_name	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	sPty7NXer4zs
config_drive	
created	2023-06-30T01:07:27Z
flavor	tiny1G (1315150a-0876-422e-bd60-6df7debef39c)
hostId	
id	a0e5b8c1-902b-4450-a00c-62d00cf7eb15
image	Ubuntu_2204 (8b25ae9f-ee7b-407a-8ecc-57d094142433)
key_name	ece577
name	web1
progress	0
project_id	b10605b9b7ab441dbca06aa35be986fc
properties	
security_groups	name='default'
status	BUILD
updated	2023-06-30T01:07:26Z
user_id	5ad60eb04ca14be0aa920c7eb6376e94
volumes_attached	
stack@controller:~/devstack$
Right Ctrl




```
ubuntu@web1:~$ ping 172.24.4.76
PING 172.24.4.76 (172.24.4.76) 56(84) bytes of data.
64 bytes from 172.24.4.76: icmp_seq=1 ttl=62 time=5.18 ms
64 bytes from 172.24.4.76: icmp_seq=2 ttl=62 time=7.51 ms
64 bytes from 172.24.4.76: icmp_seq=3 ttl=62 time=7.86 ms
64 bytes from 172.24.4.76: icmp_seq=4 ttl=62 time=5.42 ms
64 bytes from 172.24.4.76: icmp_seq=5 ttl=62 time=5.47 ms
64 bytes from 172.24.4.76: icmp_seq=6 ttl=62 time=5.48 ms
64 bytes from 172.24.4.76: icmp_seq=7 ttl=62 time=5.76 ms
64 bytes from 172.24.4.76: icmp_seq=8 ttl=62 time=5.11 ms
64 bytes from 172.24.4.76: icmp_seq=9 ttl=62 time=6.38 ms
64 bytes from 172.24.4.76: icmp_seq=10 ttl=62 time=5.53 ms
64 bytes from 172.24.4.76: icmp_seq=11 ttl=62 time=5.48 ms
64 bytes from 172.24.4.76: icmp_seq=12 ttl=62 time=5.32 ms
^C
--- 172.24.4.76 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11059ms
rtt min/avg/max/mdev = 5.112/5.875/7.855/0.867 ms
ubuntu@web1:~$
```



ASSIGNMENT PART 2

Keep the pings running and attempt to explain the actual path of traffic from web0 to web1. Using tcpdump would be pretty useful here, as well as commands like brctl and ovs-vsctl too, since the software defined networks are implemented using Open vSwitch.

Command : sudo tcpdump -i ens3 -n

sudo tcpdump -r web1_to_web0.pcap
sudo tcpdump -r web1_to_web0.pcap

```
ubuntu@web0:~$ sudo tcpdump -i ens3 -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
01:34:01.096264 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 3275167130:3275167238, ack 234636559
, win 502, options [nop,nop,TS val 1674540733 ecr 2359627645], length 108
01:34:01.097870 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 108, win 501, options [nop,nop,TS val
 2359628208 ecr 1674540733], length 0
01:34:01.098711 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 108:144, ack 1, win 502, options [no
p,nop,TS val 1674540735 ecr 2359628208], length 36
01:34:01.099554 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 144, win 501, options [nop,nop,TS val
 2359628210 ecr 1674540735], length 0
01:34:01.101713 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 144:196, ack 1, win 502, options [no
p,nop,TS val 1674540738 ecr 2359628210], length 52
01:34:01.102715 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 196, win 501, options [nop,nop,TS val
 2359628213 ecr 1674540738], length 0
01:34:01.103249 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 196:264, ack 1, win 502, options [no
p,nop,TS val 1674540740 ecr 2359628213], length 68
01:34:01.104378 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 264, win 501, options [nop,nop,TS val
 2359628215 ecr 1674540740], length 0
01:34:01.106099 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 264:332, ack 1, win 502, options [no
p,nop,TS val 1674540743 ecr 2359628215], length 68
01:34:01.106866 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 332, win 501, options [nop,nop,TS val
 2359628218 ecr 1674540743], length 0
01:34:01.107868 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 332:368, ack 1, win 502, options [no
p,nop,TS val 1674540744 ecr 2359628218], length 36
01:34:01.109239 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 368, win 501, options [nop,nop,TS val
 2359628219 ecr 1674540744], length 0
01:34:01.130389 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 368:580, ack 1, win 502, options [no
p,nop,TS val 1674540767 ecr 2359628219], length 212
01:34:01.131497 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 580, win 501, options [nop,nop,TS val
 2359628242 ecr 1674540767], length 0
01:34:01.132239 IP 10.77.80.50.22 > 10.0.1.36.50614: Flags [P.], seq 580:616, ack 1, win 502, options [no
p,nop,TS val 1674540769 ecr 2359628242], length 36
01:34:01.133550 IP 10.0.1.36.50614 > 10.77.80.50.22: Flags [.], ack 616, win 501, options [nop,nop,TS val
 2359628244 ecr 1674540769], length 0
```

```
ubuntu@web1:~$ sudo tcpdump -i ens3 -n
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
02:11:19.218000 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 1865912202:1865912310, ack 193062012, win 502, opt
ions [nop,nop,TS val 3592460733 ecr 1218511217], length 108
02:11:19.220072 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 108, win 501, options [nop,nop,TS val 1218511768 ec
r 3592460733], length 0
02:11:19.221750 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 108:144, ack 1, win 502, options [nop,nop,TS val 3
592460737 ecr 1218511768], length 36
02:11:19.223068 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 144, win 501, options [nop,nop,TS val 1218511771 ec
r 3592460737], length 0
02:11:19.223672 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 144:196, ack 1, win 502, options [nop,nop,TS val 3
592460739 ecr 1218511771], length 52
02:11:19.224715 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 196, win 501, options [nop,nop,TS val 1218511773 ec
r 3592460739], length 0
02:11:19.225731 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 196:264, ack 1, win 502, options [nop,nop,TS val 3
592460741 ecr 1218511773], length 68
02:11:19.226817 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 264, win 501, options [nop,nop,TS val 1218511775 ec
r 3592460741], length 0
02:11:19.228462 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 264:332, ack 1, win 502, options [nop,nop,TS val 3
592460744 ecr 1218511775], length 68
02:11:19.229654 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 332, win 501, options [nop,nop,TS val 1218511778 ec
r 3592460744], length 0
02:11:19.241877 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 332:552, ack 1, win 502, options [nop,nop,TS val 3
592460757 ecr 1218511778], length 220
02:11:19.243039 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 552, win 501, options [nop,nop,TS val 1218511792 ec
r 3592460757], length 0
02:11:19.243489 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 552:588, ack 1, win 502, options [nop,nop,TS val 3
592460759 ecr 1218511792], length 36
02:11:19.244430 IP 10.0.1.36.46758 > 10.77.80.121.22: Flags [.], ack 588, win 501, options [nop,nop,TS val 1218511793 ec
r 3592460759], length 0
02:11:19.245804 IP 10.77.80.121.22 > 10.0.1.36.46758: Flags [P.], seq 588:768, ack 1, win 502, options [nop,nop,TS val 3
592460761 ecr 1218511793], length 180
```

```
ubuntu@web0:~$ sudo tcpdump -r web0_to_web1.pcap
reading from file web0_to_web1.pcap, link-type EN10MB (Ethernet), snapshot length 262144
01:33:12.098162 IP web0.ssh > 10.0.1.36.50614: Flags [P.], seq 3275165534:3275165578, ack 234635659, win 502, options [nop,nop,TS val 1674491735 ecr 2359578475], length 44
01:33:12.099984 IP 10.0.1.36.50614 > web0.ssh: Flags [.], ack 44, win 501, options [nop,nop,TS val 2359579232 ecr 1674491735], length 0
01:33:12.100824 IP web0.ssh > 10.0.1.36.50614: Flags [P.], seq 44:96, ack 1, win 502, options [nop,nop,TS val 1674491737 ecr 2359579232], length 52
01:33:12.102108 IP 10.0.1.36.50614 > web0.ssh: Flags [.], ack 96, win 501, options [nop,nop,TS val 2359579234 ecr 1674491737], length 0
01:33:12.104555 IP web0.ssh > 10.0.1.36.50614: Flags [P.], seq 96:164, ack 1, win 502, options [nop,nop,TS val 1674491741 ecr 2359579234], length 68
01:33:12.105819 IP 10.0.1.36.50614 > web0.ssh: Flags [.], ack 164, win 501, options [nop,nop,TS val 2359579238 ecr 1674491741], length 0
01:33:12.106757 IP web0.ssh > 10.0.1.36.50614: Flags [P.], seq 164:232, ack 1, win 502, options [nop,nop,TS val 1674491743 ecr 2359579238], length 68
01:33:12.107780 IP 10.0.1.36.50614 > web0.ssh: Flags [.], ack 232, win 501, options [nop,nop,TS val 2359579240 ecr 1674491743], length 0
01:33:12.108322 IP web0.ssh > 10.0.1.36.50614: Flags [P.], seq 232:268, ack 1, win 502, options [nop,nop,TS val 1674491745 ecr 2359579240], length 36
01:33:12.109533 IP 10.0.1.36.50614 > web0.ssh: Flags [.], ack 268, win 501, options [nop,nop,TS val 2359579241 ecr 1674491745], length 0
01:33:30.686205 IP web0.40638 > pugot.canonical.com.ntp: NTPv4, Client, length 48
01:33:30.794396 IP pugot.canonical.com.ntp > web0.40638: NTPv4, Server, length 48
ubuntu@web0:~$
```

ECE577 - Lab 06 - Data Center Networks - Part 2 - Live Migration

In this part of the lab, we will show how VMs can be live migrated from any compute host to any other compute host while retaining their network parameters.

Step A - Configuring Live-Migration

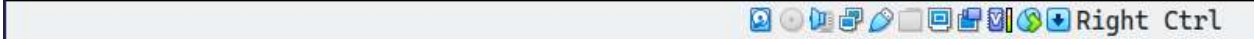
Live migration uses SSH to transport the VM state/disk from one machine to the other. To do so requires that the stack user on each host has root SSH access to the other host. Live migration requires host names to work correctly

1. Login as ubuntu controller type:
   $sudo passwd stack
   and assign the stack user the password stack and logout

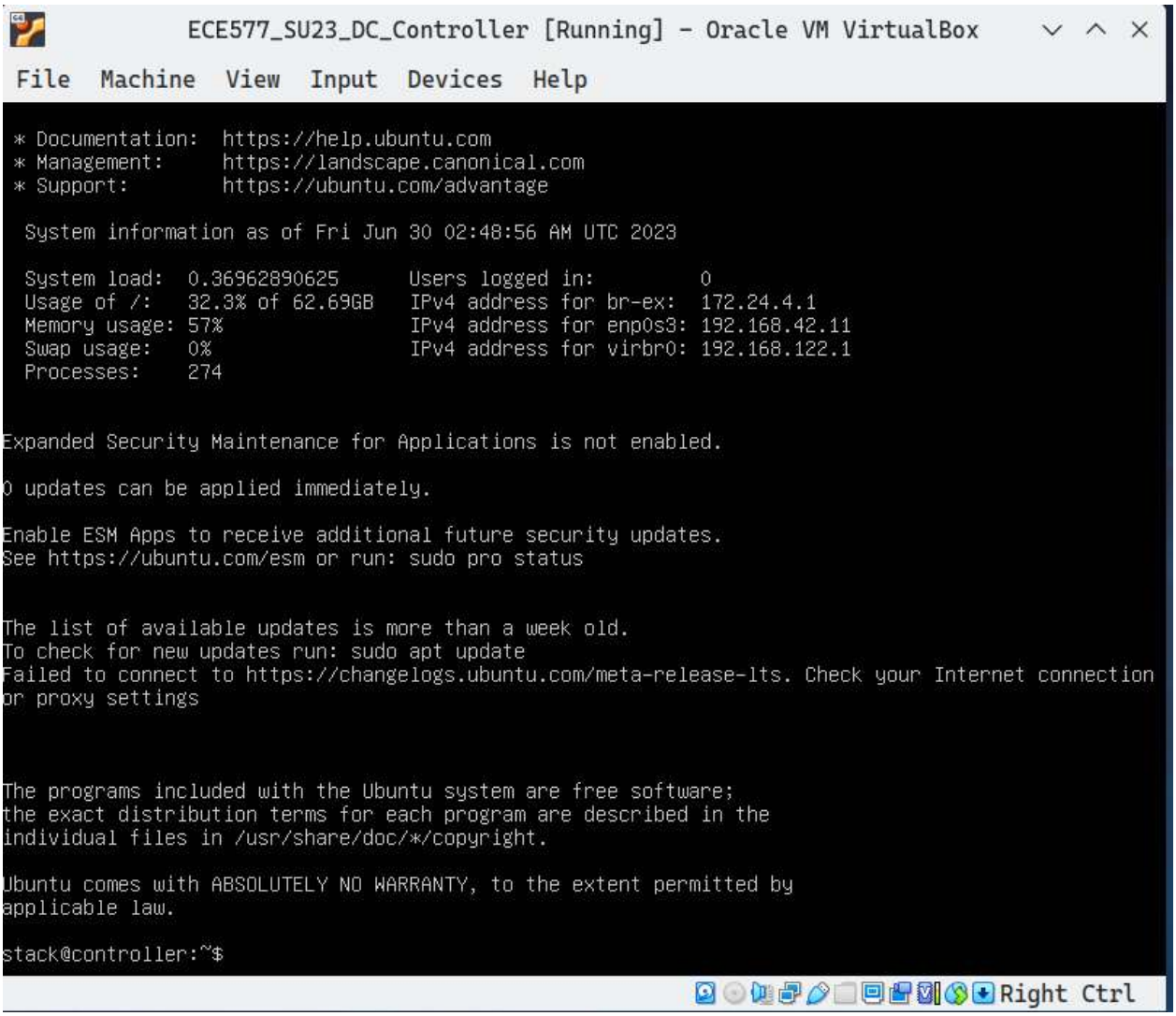
```
ubuntu@controller:~$ sudo passwd stack
New password:
Retype new password:
passwd: password updated successfully
ubuntu@controller:~$ _
```

2. Repeat the above step for compute0

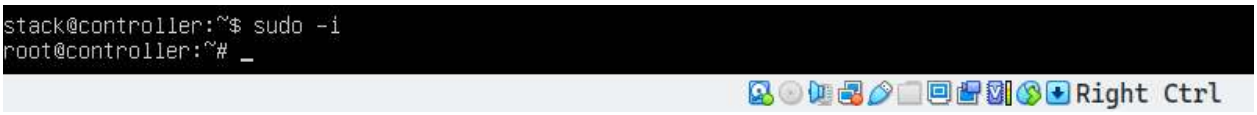
```
ubuntu@compute0:~$ sudo passwd stack
[sudo] password for ubuntu:
New password:
Retype new password:
passwd: password updated successfully
ubuntu@compute0:~$
```



3.Log in as the stack user on controller.

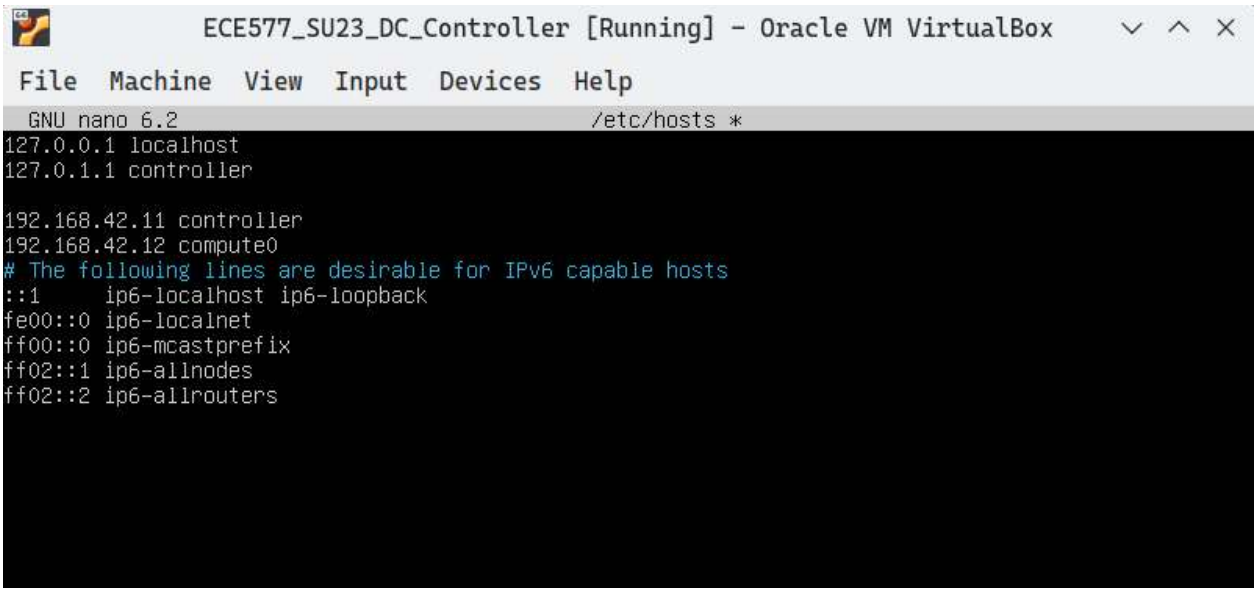


4. Issue an $sudo -i to drop into controller

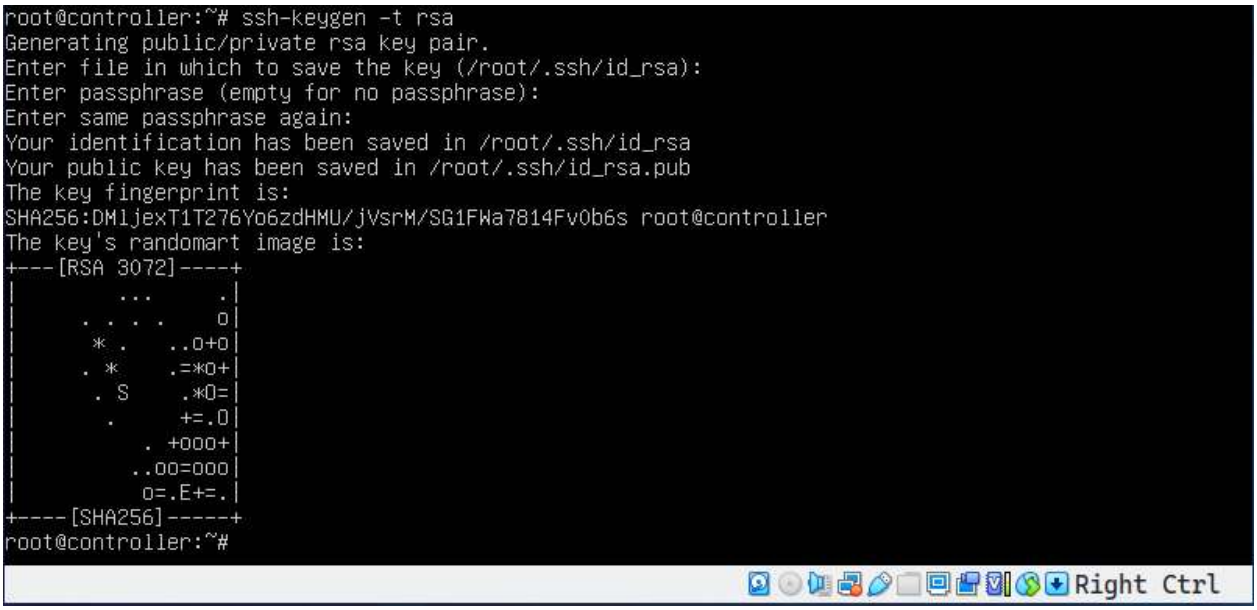


5. As root, modify to /etc/hosts file to add entries for both controller and compute0 with the following lines:

192.168.42.11 controller
192.168.42.12 compute0



6. Next,as root, type:
ssh-keygen -t rsa
And hit enter a couple times to create the key file without a password

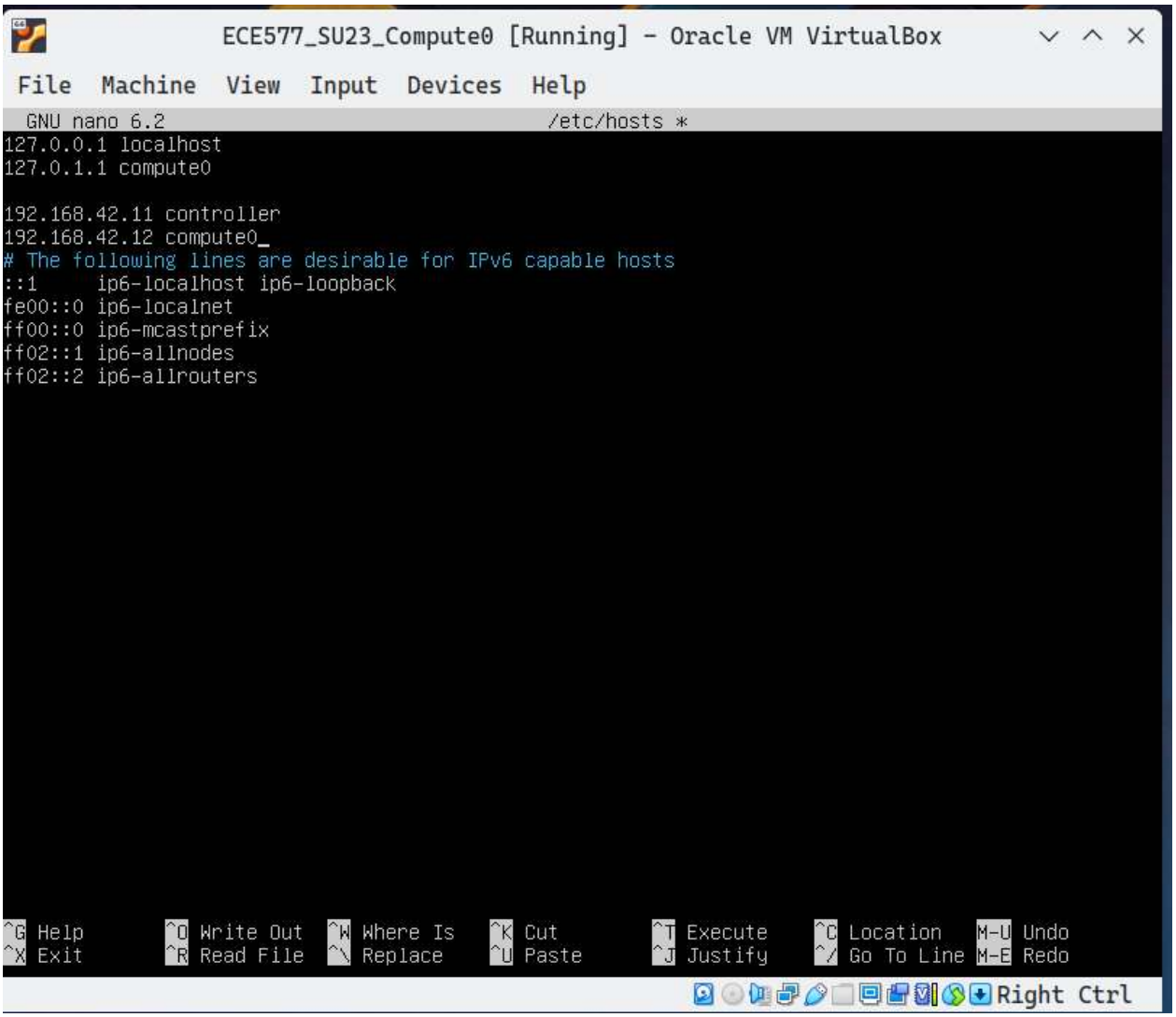


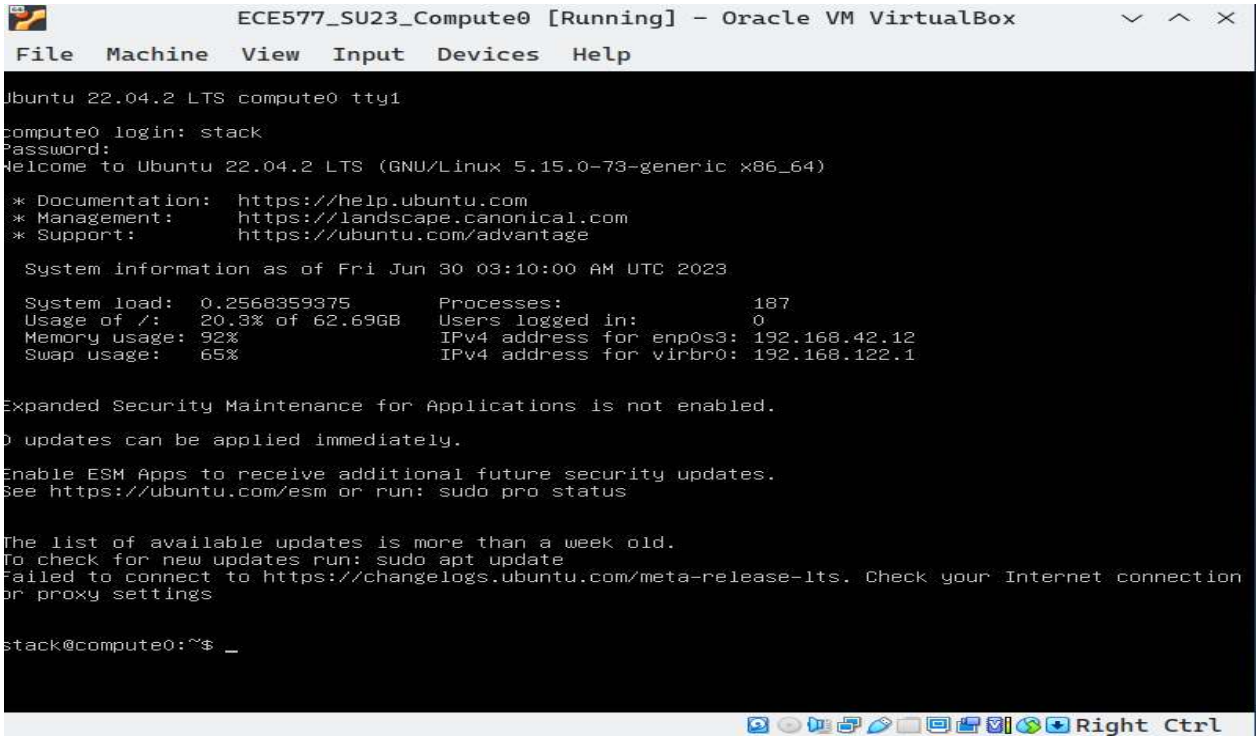
7. Still as root, copy root's public key to the stack user on compute0:
ssh-copy-id -i /root/.ssh/id_rsa.pub stack@compute0
The stack user should have the password "stack" unless you changed it.

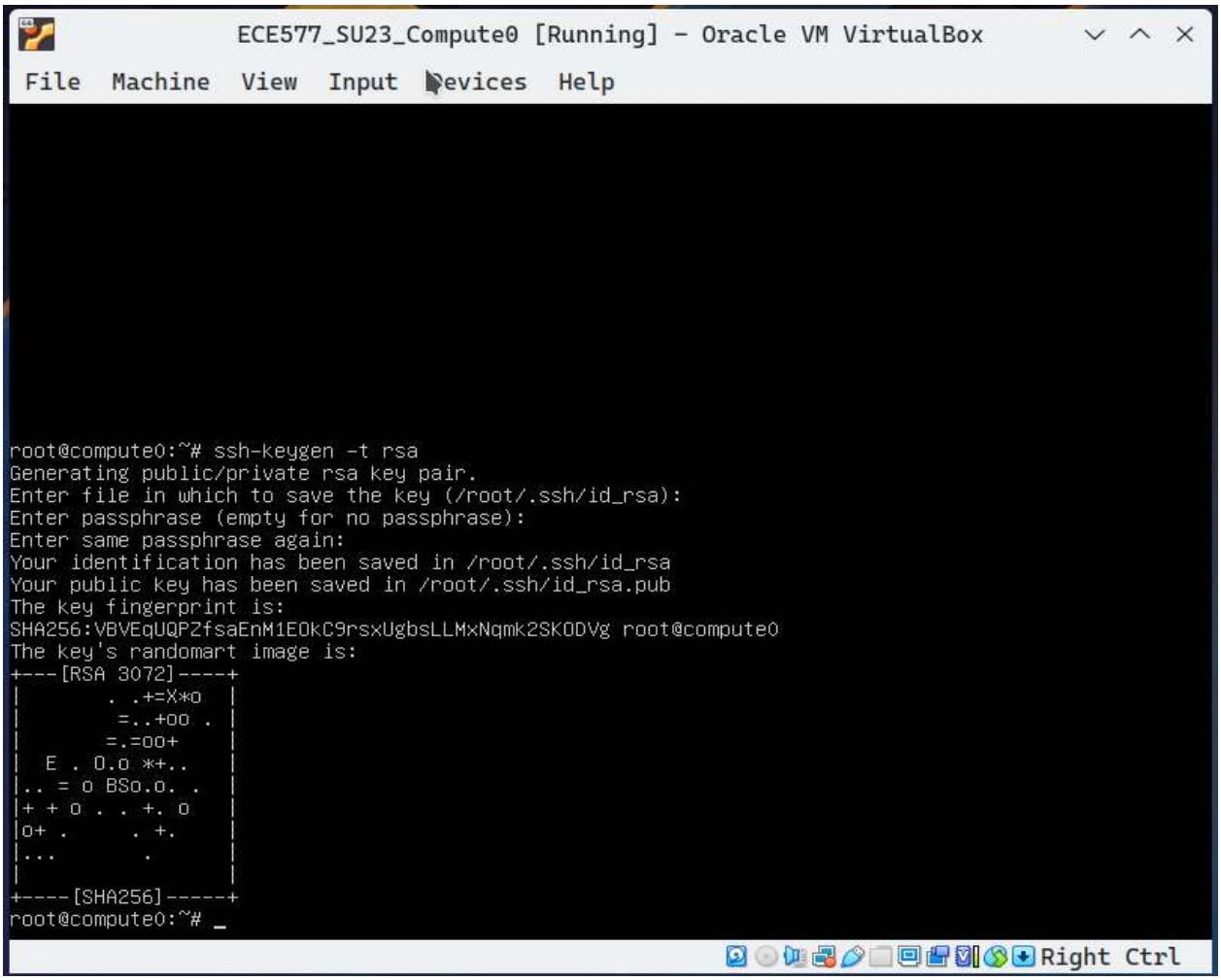
```
root@controller:~# ssh-copy-id -i /root/.ssh/id_rsa.pub stack@compute0
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host 'compute0 (192.168.42.12)' can't be established.
ED25519 key fingerprint is SHA256:suKUObvkdrgVW6J/FsNWokEc8dOw6CxsUVZEKdywXN8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
stack@compute0's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'stack@compute0'"
and check to make sure that only the key(s) you wanted were added.

root@controller:~# _
```

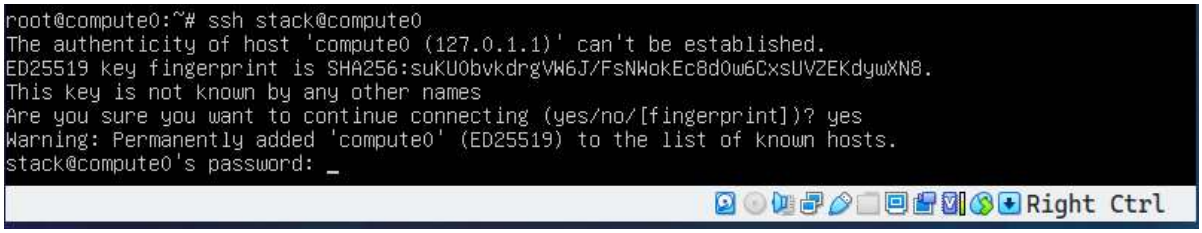
Right Ctrl

8. Repeat steps 1-5 on the compute0 node, changing step 5 to be: # ssh-copy-id -i /root/.ssh/id_rsa.pub stack@controller



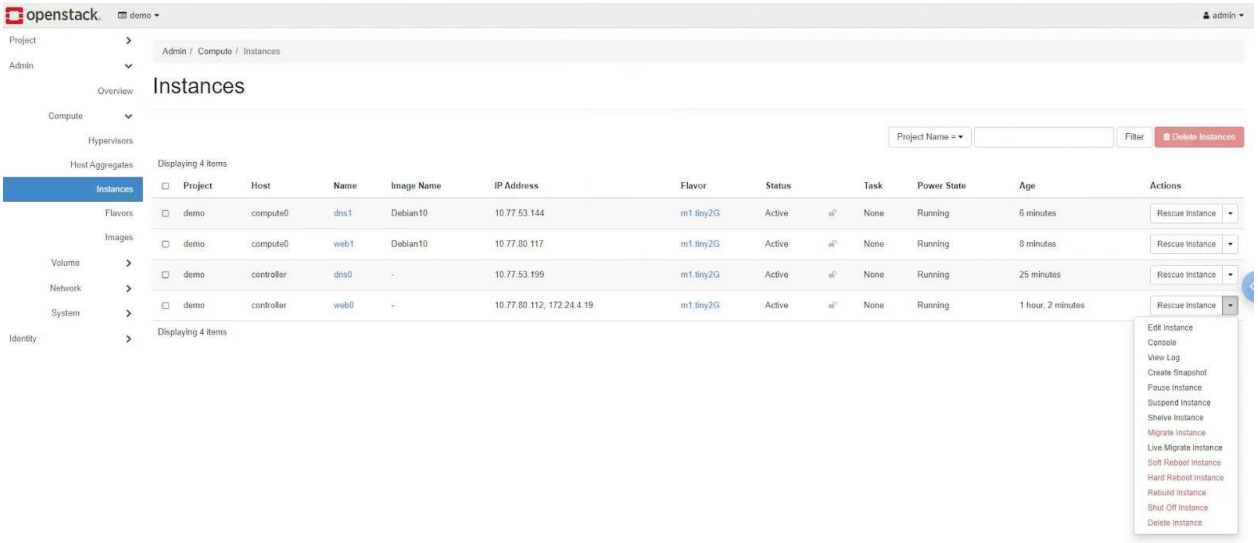

ECE577_SU23_Compute0 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu 22.04.2 LTS compute0 tty1
compute0 login: stack
Password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-73-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Jun 30 03:10:00 AM UTC 2023
System load: 0.2568359375 Processes: 187
Usage of /: 20.3% of 62.69GB Users logged in: 0
Memory usage: 92% IPv4 address for enp0s3: 192.168.42.12
Swap usage: 65% IPv4 address for virbr0: 192.168.122.1
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings
stack@compute0:~$ _
Right Ctrl



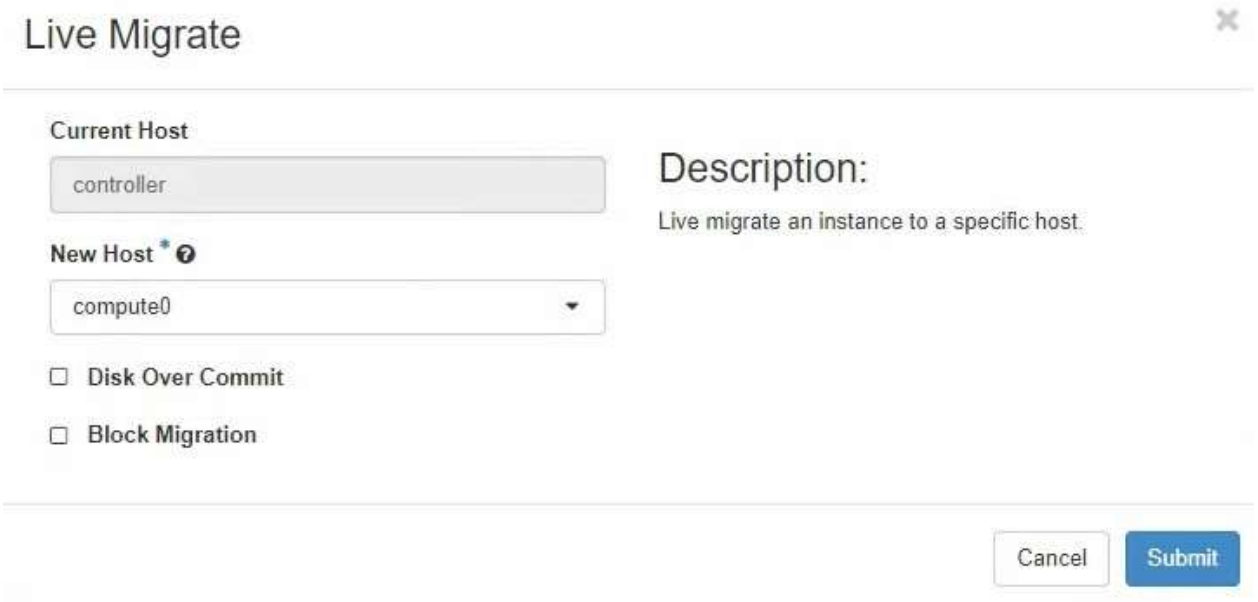


Step B - Live Migrating The Machine

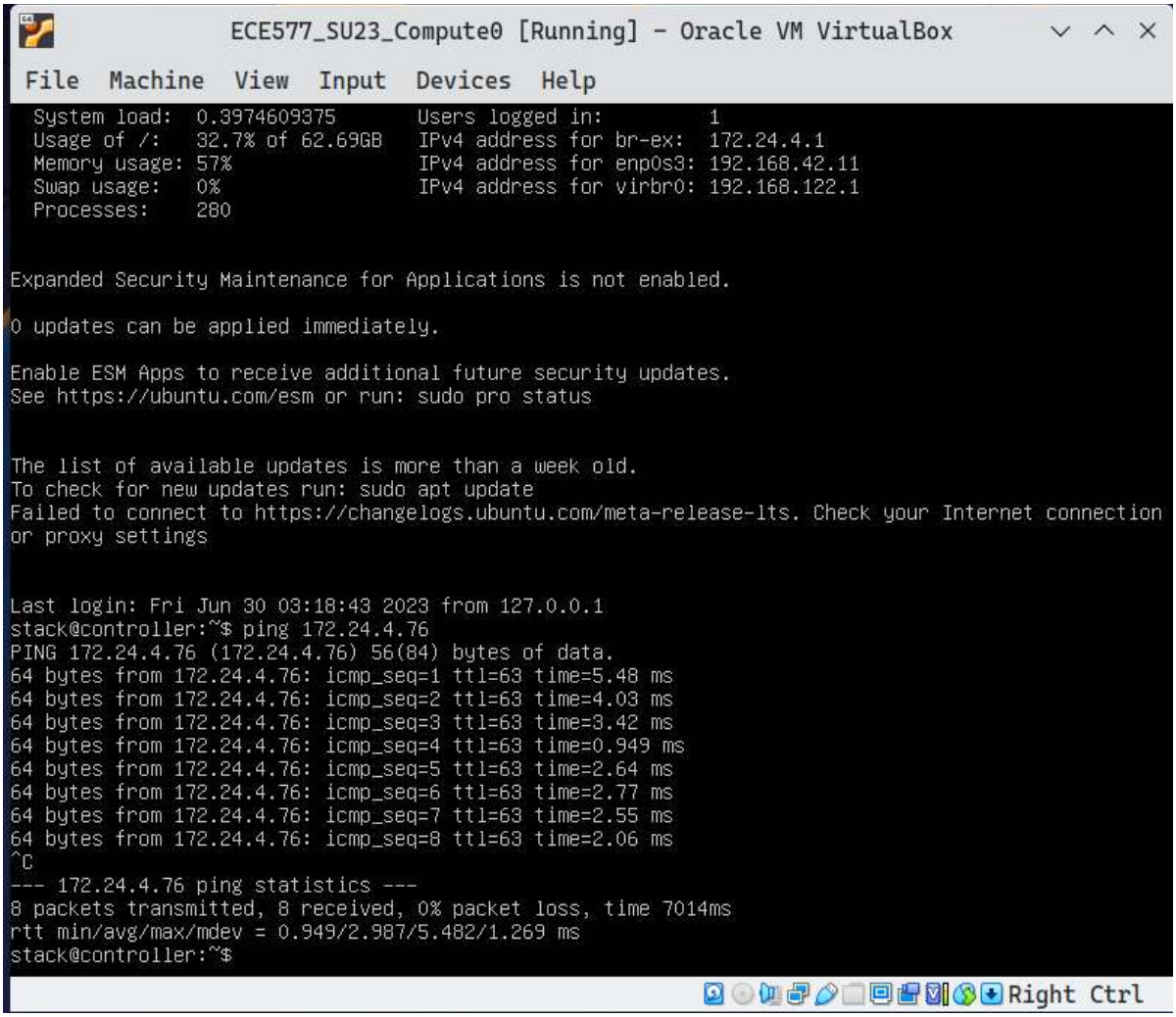
1. Make sure the web0 machine is assigned a floating IP and that you can ping that floating IP from the host.
2. Start a continuous ping on the floating IP ($ping -t in Windows). Keep an eye on this.
3. Go to Admin → Compute → Instances and select “Live Migration” from the drop-down next to web0:

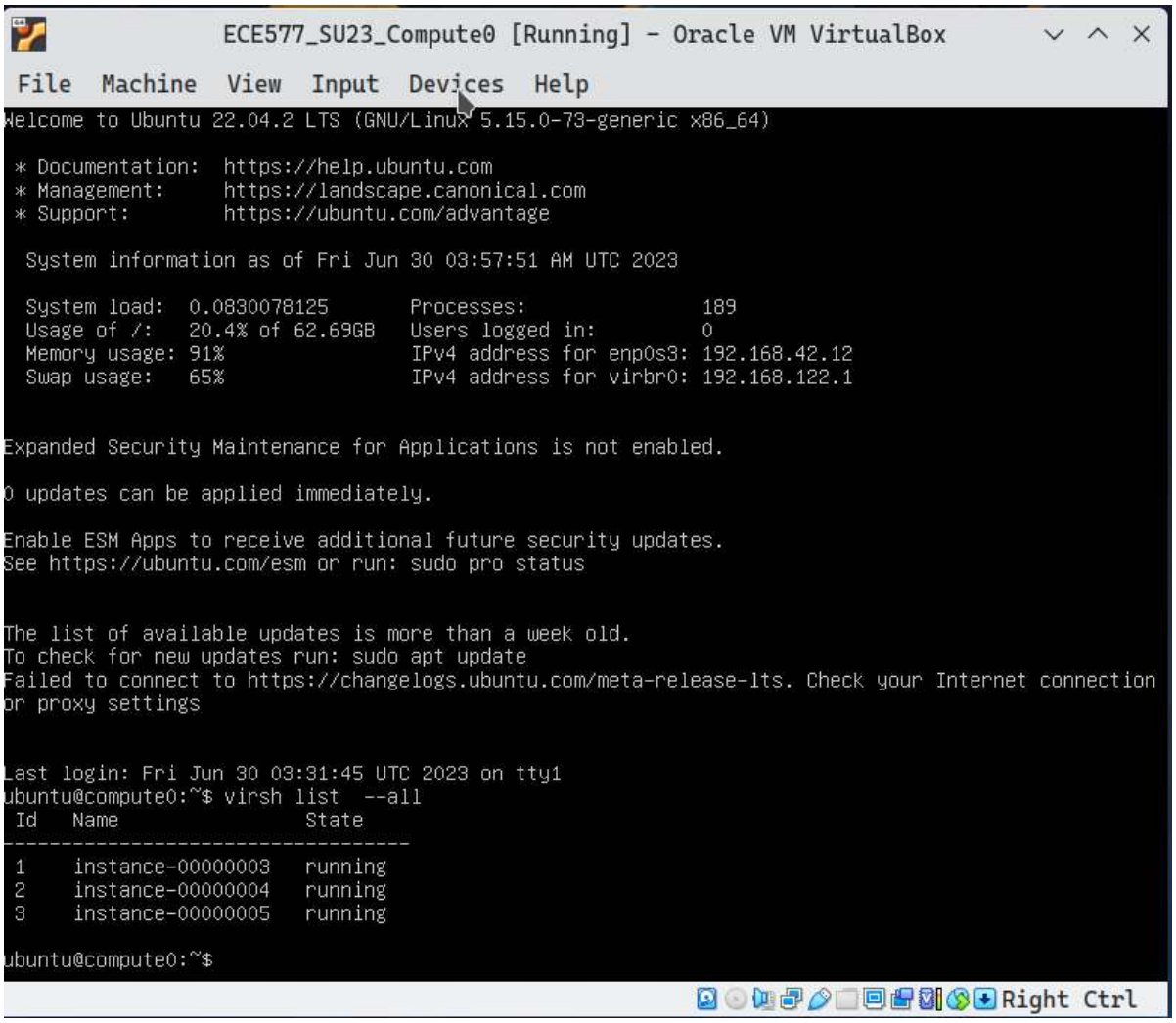


4. You will be prompted to select the new host node, so select compute0



5.Verify that web0 is now running on compute0


ECE577_SU23_Compute0 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
System load: 0.3974609375 Users logged in: 1
Usage of /: 32.7% of 62.69GB IPv4 address for br-ex: 172.24.4.1
Memory usage: 57% IPv4 address for enp0s3: 192.168.42.11
Swap usage: 0% IPv4 address for virbr0: 192.168.122.1
Processes: 280
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings
Last login: Fri Jun 30 03:18:43 2023 from 127.0.0.1
stack@controller:~$ ping 172.24.4.76
PING 172.24.4.76 (172.24.4.76) 56(84) bytes of data.
64 bytes from 172.24.4.76: icmp_seq=1 ttl=63 time=5.48 ms
64 bytes from 172.24.4.76: icmp_seq=2 ttl=63 time=4.03 ms
64 bytes from 172.24.4.76: icmp_seq=3 ttl=63 time=3.42 ms
64 bytes from 172.24.4.76: icmp_seq=4 ttl=63 time=0.949 ms
64 bytes from 172.24.4.76: icmp_seq=5 ttl=63 time=2.64 ms
64 bytes from 172.24.4.76: icmp_seq=6 ttl=63 time=2.77 ms
64 bytes from 172.24.4.76: icmp_seq=7 ttl=63 time=2.55 ms
64 bytes from 172.24.4.76: icmp_seq=8 ttl=63 time=2.06 ms
^C
--- 172.24.4.76 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7014ms
rtt min/avg/max/mdev = 0.949/2.987/5.482/1.269 ms
stack@controller:~$
Right Ctrl



Result: The project demonstrates how a data center can be virtualized using overlay networks and microsegmentation, ensuring secure communication between VMs. It also illustrates the effectiveness of live migration in maintaining service continuity.