

Openstack Guide to get started

General Openstack terminologies

- a. **Nova** – Nova is a tool for managing Virtual Machines. It uses the underlying system's(hypervisor such as KVM, Vmware etc.). It can be accessed via Horizon by CLI or APIs
- b. **Swift** – Swift is used for object storage in Openstack. It is horizontally scalable and is ideal for storing unstructured data such as backups, videos, pictures, user data etc.
- c. **Cinder** – Cinder is also a block storage application in openstack but it is used to store structured data or persistent storage volumes for the VMs. Its like a hard drive for the VMs and the storage for VMs can be added by adding cinder volumes.
- d. **Neutron** – Neutron is the networking section of openstack. All networking configurations like Virtual networks, subnets, routing etc. are provisioned via Neutron. Its like the SDN controller in Openstack
- e. **Glance** – Where Nova was a tool to provision VMs, Glance is a tool that actually stores and manages those VMs. Templating of VMs and creating snapshots of VMs is something that Glance does.
- f. **Keystone** – Keystone is Openstack's authentication component. It generates tokens to clients for identity validation. This is for overall infrastructure security
- g. **Horizon** – Horizon is the front end GUI application of Openstack. The User will land at horizon and horizon will abstract all settings for the user using APIs

1. Difference between Users and Roles?

Users: Users are the different end users that are operating on open stack. Each user may have different levels of permissions and allowed access.

Roles: Roles are like user profiles that can be created per project. Users can be then added to be a part of these roles.

2. What is a hypervisor and which hypervisors are supported in OpenStack?

A hypervisor is a software that runs on a bare metal server and manages resources such as compute, storage and memory. It also manages VMs and management of resources among them. OpenStack supports multiple hypervisors such as Hyper-V, Vmware Vsphere, Xen, etc. But it supposedly works best with KVM which is again open source.

3. Explain the meaning of ‘flavor’ in OpenStack.

Flavors are like templates for VMs in openstack. We can have different flavors with different levels of resources depending on the functionality of the VM. We can then create with VMs with a certain flavor to quickly spin up rather than configuring each VM manually.

Creating a new Network

1. Go to the Network topology Tab on openstack.
2. Click on Create new network. Give a Network Name
3. Then go to the subnet tab and give subnet name, network address and gateway IP

Create Network

Network Subnet Subnet Details

Subnet Name
Objective 1

Network Address Source
Enter Network Address manually

Network Address 10.0.0.0/26

IP Version IPv4

Gateway IP 10.0.0.1

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. Go to the subnet details tab and then Enable DHCP, and allocate the pool:

Create Network

Network Subnet Subnet Details

Enable DHCP

Specify additional attributes for the subnet

Allocation Pools 10.0.0.2,10.0.0.33

DNS Name Servers

Host Routes

Cancel

5. Now click Create. In the Networks tab you will now see the new network created

Creating and configuring a logical Router

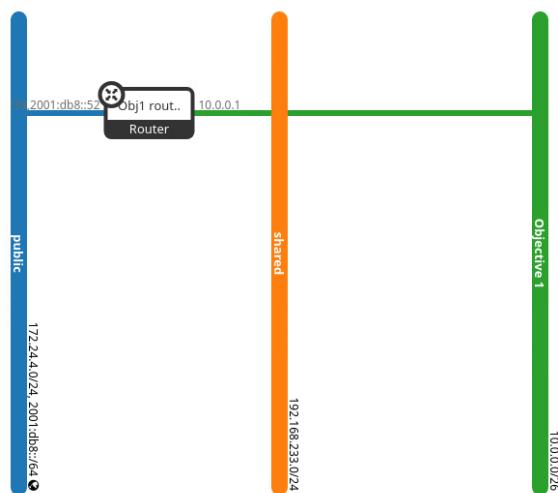
1. Under the Network tab click on Routers.

Click on Create Router

2. Give a router name, then select the external network that you want to connect it to and then enable SNAT and then create router. This will automatically connect your router to the external network

3. In order to connect your router to your new internal network, go to the routers tab>>select your router>>go to the interfaces tab>>Add interface>>then assign an IP address to the interface and then select the internal network from the dropdown.

4. In the Network Topology tab, you will now see the public and the objective 1 networks connected to each other with the router:



Spinning up new instances and configuring them to network

1. Under the compute tab, click on images and check if the Cirros image is present or not.
2. Now click on the instances tab and then click on Launch Instance

Fill out the fields:

Launch Instance

Details	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.
Source	Project Name: admin
Flavor *	Instance Name *: instance1
Networks *	Description:
Network Ports	Availability Zone: nova
Security Groups	Count *: 2
Key Pair	
Configuration	
Server Groups	
Scheduler Hints	
Metadata	

Launch Instance

Source *	Select Boot Source: Image	Create New Volume: Yes
Flavor *	Volume Size (GB) *: 1	Delete Volume on Instance Delete: Yes
Networks *	Volume Type: lvmdriver-1	
Network Ports	Allocated: Displaying 0 items	
Security Groups		
Key Pair		
Configuration		
Server Groups	Name Updated Size Format Visibility	Select an item from Available items below
Scheduler Hints		
Metadata		

Select one

Displaying 1 item					
Name: cirros-0.6.3-x86_64-disk	Updated: 1/18/26 10:08 PM	Size: 20.69 MB	Format: QCOW2	Visibility: Public	<input type="button" value="Up"/>
Displaying 1 item					

Then choose any one flavor

Launch Instance

Details

Allocated

Displaying 1 item

Name	vCPUs	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.micro	1	256 MB	1 GB	1 GB	0 GB	Yes

Network Ports

Displaying 1 item

Security Groups

▼ Available (1)

Select one

Name	vCPUs	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.nano	1	192 MB	1 GB	1 GB	0 GB	Yes
cirros256	1	256 MB	1 GB	1 GB	0 GB	Yes
m1.tiny	1	512 MB	1 GB	1 GB	0 GB	Yes
ds512M	1	512 MB	5 GB	5 GB	0 GB	Yes
ds1G	1	1 GB	10 GB	10 GB	0 GB	Yes
m1.small	1	2 GB	20 GB	20 GB	0 GB	Yes
ds2G	2	2 GB	10 GB	10 GB	0 GB	Yes
m1.medium	2	4 GB	40 GB	40 GB	0 GB	Yes
ds4G	4	4 GB	20 GB	20 GB	0 GB	Yes

Key Pair

Configuration

Displaying 11 items

Server Groups

Scheduler Hints

Metadata

Select a network

Launch Instance

Details

Allocated

Displaying 1 item

Network	Subnets Associated	Shared	Admin State	Status
Objective 1	Objective 1	No	Up	Active

Network Ports

Displaying 1 item

Security Groups

▼ Available (2)

Select one or more

Network	Subnets Associated	Shared	Admin State	Status
shared	shared-subnet	No	Up	Active
public	public-subnet ipv6-public-subnet	No	Up	Active

Key Pair

Configuration

Displaying 2 items

Server Groups

Scheduler Hints

Metadata

Displaying 2 items

Cancel

Back

Next

Launch Instance

Now click on Launch Instance

You will then see the instances being created

Instances										
Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
Instance1-2	-	10.0.0.20	m1.micro	-	Active	us-west-1a	None	Running	0 minutes	Create Snapshot
Instance1-1	-	10.0.0.24	m1.micro	-	Active	us-west-1a	None	Running	0 minutes	Create Snapshot

We can also see that the IP addresses have been automatically assigned for each instance because we had enabled DHCP

Making your host VM talk to openstack instances.

Now that we have Private IP addresses associated with the instances and the router. We need to do the following.

1. Create a new security group and add the following rules for the instances and the router to be able to ping and allow ssh communication.

Manage Security Group Rules: allowall (80ae54cb-4699-490d-bc58-eb84442ae803)

Displaying 4 items								+ Add Rule	Delete Rules
Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions		
Egress	IPv4	Any	Any	0.0.0.0/0	-	-			Delete Rule
Egress	IPv6	Any	Any	::/0	-	-			Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-			Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-			Delete Rule

Assign this security group for all instances and the router interfaces.

2. Floating IPs

Go to the instances section and for each instance, create a floating IP which would be reachable from our local machine

Lets Consider a Scenario:

You are working in a cloud firm that has a single instance of an application running on OpenStack cloud platform. The firm is planning to add a functionality to the

single running instance of the application that can autoscale/replicate itself to multiple instances whenever the compute capacity (eg. CPU cycles or memory) reaches a pre-defined threshold. Since you are familiar with the Python programming and REST API, you are being assigned a following task:

- a. Write a simple Python application that can ssh into the available “cirros” instance that was created in the above objective and extract the CPU utilization information. [As an alternative, you may use ceilometer service for retrieving this telemetry data]
- b. If the CPU utilization or memory usage exceeds a threshold value, for example 20%, spin up additional instances of cirros. The creation of cirros instances should be triggered whenever the usage of CPU or memory exceeds a predefined threshold. Select CPU or memory usage to your interest to define your condition to trigger the creation of additional instances. In order to collect the utilization data, you’ll have to monitor its usage using appropriate commands.
- c. The Python application can use Nova REST API to create additional “Cirros” instances whenever the above condition occurs.
- d. The auto scaling of the instances should be handled considering following requirements:

Max scaling size: 4 (this value denotes the maximum number of instances that should be spun)

Increment size: 1 (this value denotes the number of instances that should be spun whenever CPU utilization exceeds threshold)

Evaluation period: 40 (this value denotes the time period in seconds for monitoring CPU usage)

2. You can use the [Linux stress tool](#) to raise the CPU utilization of an instance above the threshold.

Implementation:

I have created two different python scripts to each this objective: 1.

Cpu_utlization.py and 2. Instance_autoscaling.py

Cpu_utilization is a simple python script that uses paramiko to ssh into the remote instance and return a cpu utilization value.

The instance_autoscaling.py is another python script that imports the get_cpu_usage function from cpu_utilization.py and then uses that to check realtime cpu usage and spin up instances if needed.

```
#def get_token():
    url = "http://10.224.76.78/identity/v3/auth/tokens"
    payload = {
        "auth": {
            "identity": {
                "methods": ["password"],
                "password": {
                    "user": {
                        "name": "admin",
                        "domain": {"id": "default"},
                        "password": "pranav"
                    }
                },
                "scope": {
                    "project": {
                        "name": "admin",
                        "domain": {"id": "default"}
                    }
                }
            }
        }
    }
```

The instance_autoscaling.py has 3 functions and a loop logic

Functions:

1. Keystone authentication: It uses the identity API to create authentication tokens for the user and for the project

2. Creating instances using Nova API: We need to populate a few server data such as name, imageref, flavorref and network ID that we want to spin up the new server

```
def create_instance(token):
    nova_url = "http://10.224.76.78/compute/v2.1/servers"

    server_data = {
        "server": {
            "name": "autoscale-vm",
            "imageRef": "2caa938d-243a-42e8-a454-6e8b894af38a",
            "flavorRef": "84",
            "networks": [{"uuid": "38cf01ad-82d7-477c-b115-b5a8231c7354"}]
        }
    }

    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/json"
    }

    r = requests.post(nova_url, json=server_data, headers=headers)
    print("Nova API response:", r.json())
```

3. Get_instance_count() function: This basically polls how many instances are currently spun up for this project. It again uses the nova API to achieve the same

```
def get_instance_count(token):
    url = "http://10.224.76.78/compute/v2.1/servers/detail"
    headers = {"X-Auth-Token": token}

    r = requests.get(url, headers=headers)
    data = r.json()

    # Count ACTIVE instances only
    active = [s for s in data["servers"] if s["status"] == "ACTIVE"]
    return len(active)
```

Working of autoscaling

Instances before

Instances

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
instance1-2	cirros-0.6.3-x86_64-disk	10.0.0.20, 172.24.4.216	m1.micro	-	Active	nova	None	Running	4 hours, 47 minutes	Create Snapshot
instance1-1	cirros-0.6.3-x86_64-disk	10.0.0.24, 172.24.4.205	m1.micro	-	Active	nova	None	Running	4 hours, 47 minutes	Create Snapshot

```
student@csci5380-vm1-prpr4002:~/nvo_lab$ ./bin/python3 /home/student/nvo_lab/instance_autoscaling.py
CPU: 0.00%
CPU normal
CPU: 0.00%
CPU normal
```

After increasing CPU usage on the current running instance

```
student@csci5380-vm1-prpr4002:~/nvo_lab$ ./bin/python3 /home/student/nvo_lab/instance_autoscaling.py
CPU normal
CPU: 0.00%
CPU normal
CPU: 100.00%
High utilization detected
Current instance count: 2
Scaling out by 1 instance...
Nova API response: {'server': {'id': '9dcdb025-2f61-42a2-ab3c-6ac7fce6b481', 'links': [{'rel': 'self', 'href': 'http://10.224.76.78/compute/v2.1/servers/9dcdb025-2f61-42a2-ab3c-6ac7fce6b481'}, {'rel': 'bookmark', 'href': 'http://10.224.76.78/compute/servers/9dcdb025-2f61-42a2-ab3c-6ac7fce6b481'}], 'OS-DCF:diskConfig': 'MANUAL', 'security_groups': [{'name': 'default'}], 'adminPass': 'QXRaLa8403x3'}}
```

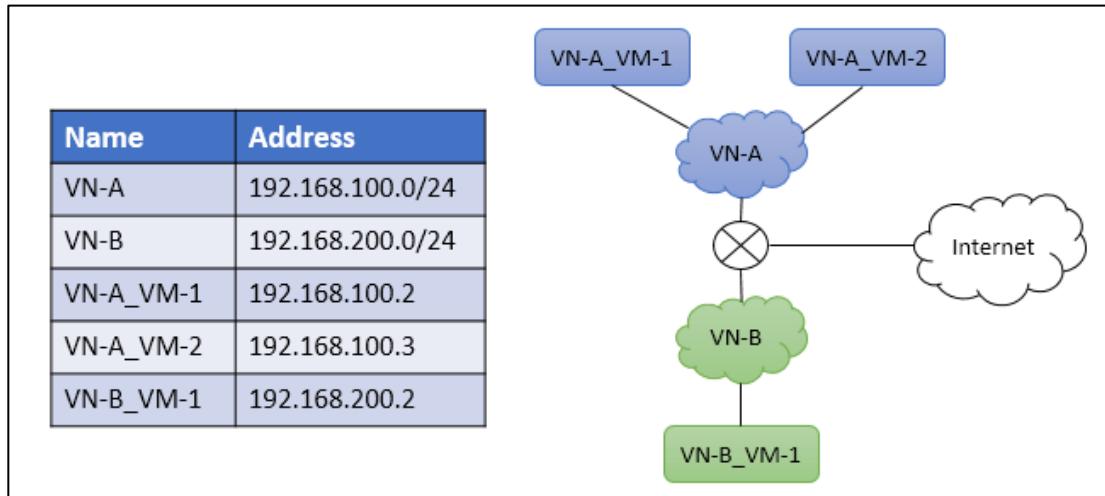
Instances

Displaying 3 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
autoscale-vm	cirros-0.6.3-x86_64-disk	10.0.0.25	m1.micro	-	Active	nova	None	Running	0 minutes	Create Snapshot
instance1-2	cirros-0.6.3-x86_64-disk	10.0.0.20, 172.24.4.216	m1.micro	-	Active	nova	None	Running	5 hours, 27 minutes	Create Snapshot
instance1-1	cirros-0.6.3-x86_64-disk	10.0.0.24, 172.24.4.205	m1.micro	-	Active	nova	None	Running	5 hours, 27 minutes	Create Snapshot

Displaying 3 items

Multi-tenants



Creating project, user, flavor and image

1. Within OpenStack UI Identity tag, create a project called lab2. Then create a user called lab2_admin and attach it to the project lab2.

Projects

Projects						
		Description	Project ID	Domain Name	Enabled	Actions
<input type="checkbox"/>	Name					Manage Members
<input type="checkbox"/>	demo		42a84c50fe484650a38a886743a73f53	Default	Yes	Manage Members
<input type="checkbox"/>	Lab2		58ce638151824f228d3d732a19aaefcb	Default	Yes	Manage Members
<input type="checkbox"/>	admin	Bootstrap project for initializing the cloud.	96ee5673c4454e21abfe97eff1fbfbf7	Default	Yes	Manage Members
<input type="checkbox"/>	service		abcaa1676e564d0ebde8946563791149	Default	Yes	Manage Members
<input type="checkbox"/>	alt_demo		ba9a97d2309a4986b20eb52d3c741912	Default	Yes	Manage Members
<input type="checkbox"/>	invisible_to_admin		fa30ef2cd93d4825b1db3615bf4f7bd5	Default	Yes	Manage Members

2. Within OpenStack UI Admin tag, create a VM Flavor called **ngn.tiny** with the following setting (or the setting that works for your VM image):

vCPU	= 1
RAM	= 128MB
Root Disk	= 1GB

Ephemeral Disk = 1GB

Swap Disk = 1GB

Create Flavor

Flavor Information * Flavor Access

Name * ngn.tiny

ID * auto

vCPUs * 1

RAM (MB) * 128

Root Disk (GB) * 1

Ephemeral Disk (GB) * 1

Swap Disk (MB) 1024

Cancel Create Flavor

0 GB 20 GB 40 GB 60 GB 80 GB 100 GB

lab2_admin

Overview	Role assignments	Groups	Credentials																				
<table><tr><td>Name</td><td>lab2_admin</td></tr><tr><td>ID</td><td>f739a999d0ed496f9f88ed1ea640783d</td></tr><tr><td>Domain Name</td><td>Default</td></tr><tr><td>Domain ID</td><td>default</td></tr><tr><td>Description</td><td>-</td></tr><tr><td>Email</td><td>-</td></tr><tr><td>Enabled</td><td>Yes</td></tr><tr><td>Password Expires At</td><td>None</td></tr><tr><td>Lock password</td><td>No</td></tr><tr><td>Primary Project</td><td>Lab2</td></tr></table>	Name	lab2_admin	ID	f739a999d0ed496f9f88ed1ea640783d	Domain Name	Default	Domain ID	default	Description	-	Email	-	Enabled	Yes	Password Expires At	None	Lock password	No	Primary Project	Lab2			
Name	lab2_admin																						
ID	f739a999d0ed496f9f88ed1ea640783d																						
Domain Name	Default																						
Domain ID	default																						
Description	-																						
Email	-																						
Enabled	Yes																						
Password Expires At	None																						
Lock password	No																						
Primary Project	Lab2																						

3. Within OpenStack UI Admin tag, upload a VM image into OpenStack. You can use this URL: <http://tinycorelinux.net/7.x/x86/release/Core-current.iso> or <https://docs.openstack.org/image-guide/obtain-images.html>. Remember to make it public.

Create Image

Image Details
Metadata

Image Details

Image Name

Image Description

Image Source

File*

Format*

Image Requirements

Kernel <input type="text" value="Choose an image"/>	Ramdisk <input type="text" value="Choose an image"/>
Architecture <input type="text"/>	Minimum Disk (GB) <input type="text" value="0"/> <input type="button" value="▼"/>
	Minimum RAM (MB) <input type="text" value="0"/> <input type="button" value="▼"/>

Image Sharing

Image Visibility Private Shared Community Public

Protected Image Yes No

Images

Click here for filters or full text search.

Displaying 2 items							
<input type="checkbox"/>	Owner	Name	Type	Status	Visibility	Protected	Disk Format
<input type="checkbox"/>	admin	cirros-0.6.3-x86_64-disk	Image	Active	Public	No	QCOW2 20.69 MB
<input type="checkbox"/>	admin	Test image	Image	Active	Shared	No	ISO 10.60 MB

Displaying 2 items

- Before proceeding, logout and login with your newly created user lab2_admin.

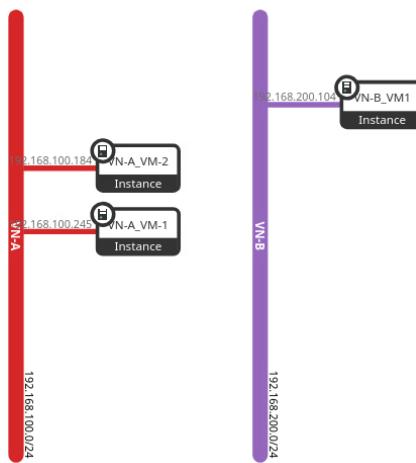
Setup Virtual Networks

- Login back into OpenStack UI, within the Project tag, create a new Network called VN-A with network address 192.168.100.0/24.
- Repeat the above steps to create a second network VN-B with network address 192.168.200.0/24.

Networks

Displaying 5 items							
Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
Objective 1	Objective 1 10.0.0.0/26	Yes	No	Active	UP	-	<button>Edit Network</button>
shared	shared-subnet 192.168.233.0/24	Yes	No	Active	UP	-	<button>Edit Network</button>
VN-A	VN-A 192.168.100.0/24	Yes	No	Active	UP	-	<button>Edit Network</button>
VN-B	VN-B 192.168.200.0/24	Yes	No	Active	UP	-	<button>Edit Network</button>
public	public-subnet 172.24.4.0/24 IPv6-public-subnet 2001:db8::/64	No	Yes	Active	UP	-	<button>Edit Network</button>

Displaying 5 items



Section 3: Launch VM instances

Launch the following VMs using the flavor and image created in Section 1.

1. Launch VN-A_VM-1 and VN-A_VM-2 into virtual network VN-A.
2. Launch VN-B_VM-1 into virtual network VN-B.

Instances

Displaying 3 items										
Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
VN_B_VM1	-	192.168.200.104	rzn.tiny	-	Build	u1 nova	Networking	No State	0 minutes	<button>Associate Floating IP</button>
VN_A_VM-2	Test image	192.168.100.184	rzn.tiny	-	Active	u1 nova	None	Running	0 minutes	<button>Create Snapshot</button>
VN_A_VM-1	Test image	192.168.100.245	rzn.tiny	-	Active	u1 nova	None	Running	2 minutes	<button>Create Snapshot</button>

Displaying 3 items

Section 4: Ping testing

1. Use the console within OpenStack UI to test if VMs in VN-A can ping each other, while the VM in VN-B cannot reach VMs in VN-A.

VN-A_VM-1 to VN-A_VM-2 ping

```
tc@box:~$ ping 192.168.100.245
PING 192.168.100.245 (192.168.100.245): 56 data bytes
64 bytes from 192.168.100.245: seq=0 ttl=64 time=3.668 ms
64 bytes from 192.168.100.245: seq=1 ttl=64 time=0.810 ms
64 bytes from 192.168.100.245: seq=2 ttl=64 time=0.849 ms
64 bytes from 192.168.100.245: seq=3 ttl=64 time=0.427 ms
64 bytes from 192.168.100.245: seq=4 ttl=64 time=0.380 ms
64 bytes from 192.168.100.245: seq=5 ttl=64 time=0.494 ms
64 bytes from 192.168.100.245: seq=6 ttl=64 time=0.658 ms
^C
--- 192.168.100.245 ping statistics ---
13 packets transmitted, 13 packets received, 0% packet loss
```

VN-B_VM-1 to VN-A_VM1 ping fail

```
tc@box:~$ ping 192.168.100.184
PING 192.168.100.184 (192.168.100.184): 56 data bytes
^C
--- 192.168.100.184 ping statistics ---
13 packets transmitted, 0 packets received, 100% packet loss
tc@box:~$
```

2. Assign floating IP's to the VM's both in VN-A and VN-B, and test connectivity to the Internet.

To create floating IPs, we need to create a router which connects both these networks and then associate floating Ips. We will also need to create rules to permit ICMP and SSH

Router						
Overview	Interfaces	Static Routes				
Displaying 3 items						
Name	Fixed IPs	Status	Type	Admin State	Actions	
(0308c5f4-e276)	• 192.168.200.1	Active	Internal Interface	UP	Delete Interface	
(60183bf3-62c2)	• 192.168.100.1	Active	Internal Interface	UP	Delete Interface	
(f358a061-c77c)	• 172.24.4.47 • 2001:db8:3:3:7	Active	External Gateway	UP	Delete Interface	



Manage Security Group Rules: default (f32976aa-d5fe-47ba-9cce-b2b745209b74)

<input type="checkbox"/>	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
<input type="checkbox"/>	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Egress	IPv4	ICMP	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	Any	Any	-	default	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	<button>Delete Rule</button>
<input type="checkbox"/>	Ingress	IPv6	Any	Any	-	default	-	<button>Delete Rule</button>

Displaying 7 items

Floating IP addresses:

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
<input type="checkbox"/>	VN-B_VM1	Test image	192.168.200.104, 172.24.4.173	ngn.tiny	-	Active	nova	None	Running	34 minutes	<button>Create Snapshot</button>
<input type="checkbox"/>	VN-A_VM-2	Test image	192.168.100.184, 172.24.4.50	ngn.tiny	-	Active	nova	None	Running	35 minutes	<button>Create Snapshot</button>
<input type="checkbox"/>	VN-A_VM-1	Test image	192.168.100.245, 172.24.4.198	ngn.tiny	-	Active	nova	None	Running	36 minutes	<button>Create Snapshot</button>

Displaying 3 items

Ping test to google.com

```

ping: bad address google.com
tc@box:~$ ping 142.250.72.206
PING 142.250.72.206 (142.250.72.206): 56 data bytes
64 bytes from 142.250.72.206: seq=0 ttl=111 time=8.612 ms
64 bytes from 142.250.72.206: seq=1 ttl=111 time=4.670 ms
64 bytes from 142.250.72.206: seq=2 ttl=111 time=5.030 ms
64 bytes from 142.250.72.206: seq=3 ttl=111 time=6.739 ms
64 bytes from 142.250.72.206: seq=4 ttl=111 time=4.757 ms
64 bytes from 142.250.72.206: seq=5 ttl=111 time=2.381 ms
64 bytes from 142.250.72.206: seq=6 ttl=111 time=3.457 ms
64 bytes from 142.250.72.206: seq=7 ttl=111 time=4.810 ms
^C
--- 142.250.72.206 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 2.381/5.057/8.612 ms
tc@box:~$ S

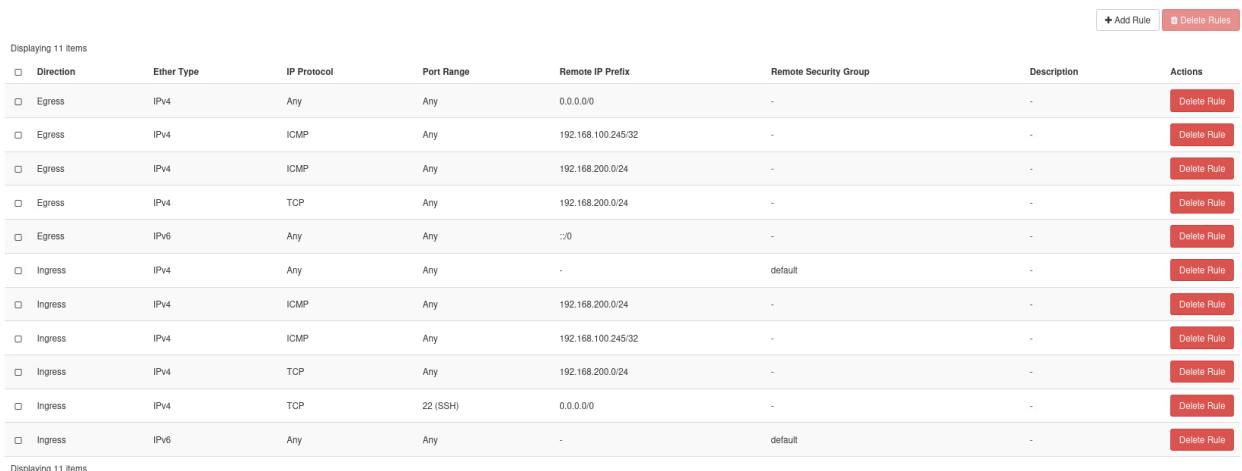
```

Network policies in OpenStack

Now manage the network policy inside OpenStack, such that:

1. VN-A_VM-1 can ping VN-B_VM-1, but VN-A_VM-2 cannot ping VN-B_VM-1.
2. VN-B_VM-1 can go out to the Internet, but VN-A_VM-1 and VN-A_VM-2 cannot.

Manage Security Group Rules: default (f32976aa-d5fe-47ba-9cce-b2b745209b74)



The screenshot shows a table of security group rules. At the top right are two buttons: '+ Add Rule' and 'Delete Rules'. The table has columns: Direction, Ether Type, IP Protocol, Port Range, Remote IP Prefix, Remote Security Group, Description, and Actions. The 'Actions' column contains red 'Delete Rule' buttons. The rules listed are:

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
Egress	IPv4	ICMP	Any	192.168.100.245/32	-	-	<button>Delete Rule</button>
Egress	IPv4	ICMP	Any	192.168.200.0/24	-	-	<button>Delete Rule</button>
Egress	IPv4	TCP	Any	192.168.200.0/24	-	-	<button>Delete Rule</button>
Egress	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
Ingress	IPv4	Any	Any	-	default	-	<button>Delete Rule</button>
Ingress	IPv4	ICMP	Any	192.168.200.0/24	-	-	<button>Delete Rule</button>
Ingress	IPv4	ICMP	Any	192.168.100.245/32	-	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	Any	192.168.200.0/24	-	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	<button>Delete Rule</button>
Ingress	IPv6	Any	Any	-	default	-	<button>Delete Rule</button>

I have created ICMP and TCP rules in order to satisfy the above two requirements