

Cloud Computing-Open Stack

Task No. 1:

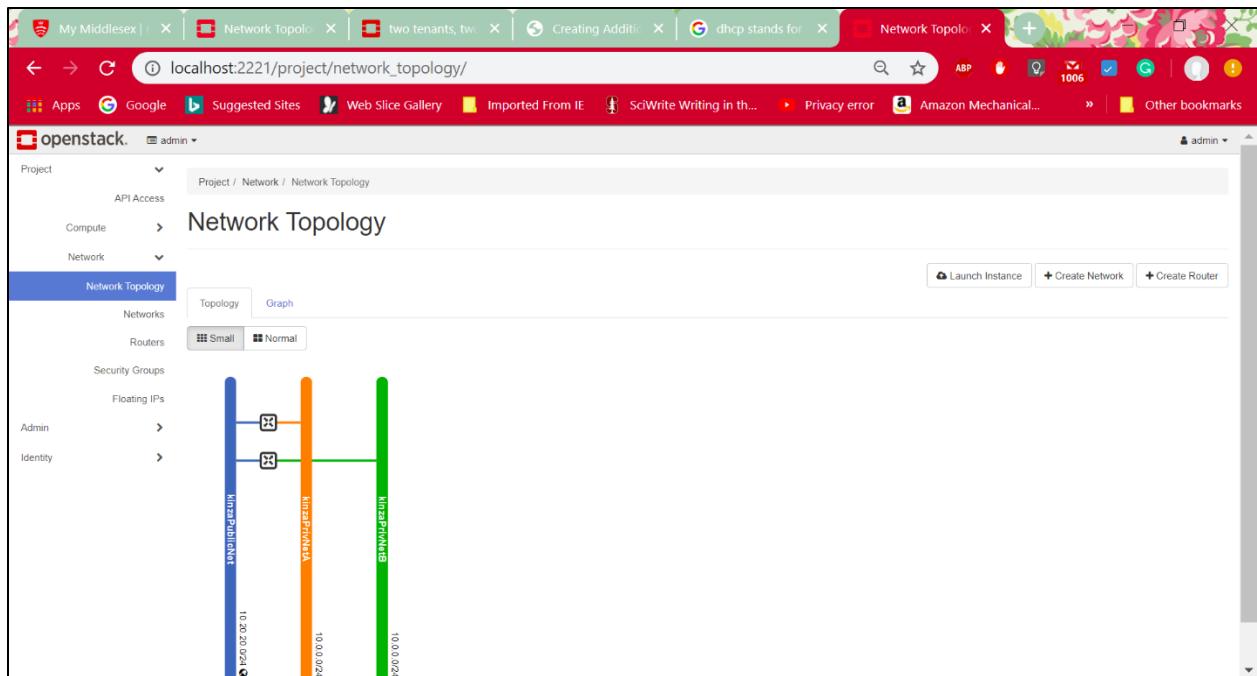
In this task, Creating two private networks and connecting them through routers to external/public network.

Steps:

Private Nework Creation:

The following two options can be used to go to **Create Network** button.

Option 1: Clicked on Project>>Network>>Network Topology



Option 2: Clicked on Project>>Network>> Networks

- Clicked on button **Create Network** and it showed the following screen. Given kinzaPrivNetA name to network. Default setting is **Enabled Admin state** and **Create Subnet**. If **Create Subnet** is enabled it asks for the subnet details while creating the network as can be seen in the following screen.

Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
kinzaPrivNetA	kinzaSubnetA 10.0.0.0/24	No	No	Active	UP	nova	<button>Edit Network</button>
kinzaPublicNet	external-subnet 10.20.20.0/24	No	Yes	Active	UP	nova	<button>Edit Network</button>
kinzaPrivNetB	kinzaSubnetB 10.0.0.0/24	No	No	Active	UP	nova	<button>Edit Network</button>

- Clicked on **Next** button and added subnet details. **Network address: 10.0.0.0/24** which will assign IP addresses randomly from range 10.0.0.1 to 10.0.0.254. Creating the subnet is important because without subnet it is not possible to attach instances and routers with the network.

Network Name: kinzaPrivNetA

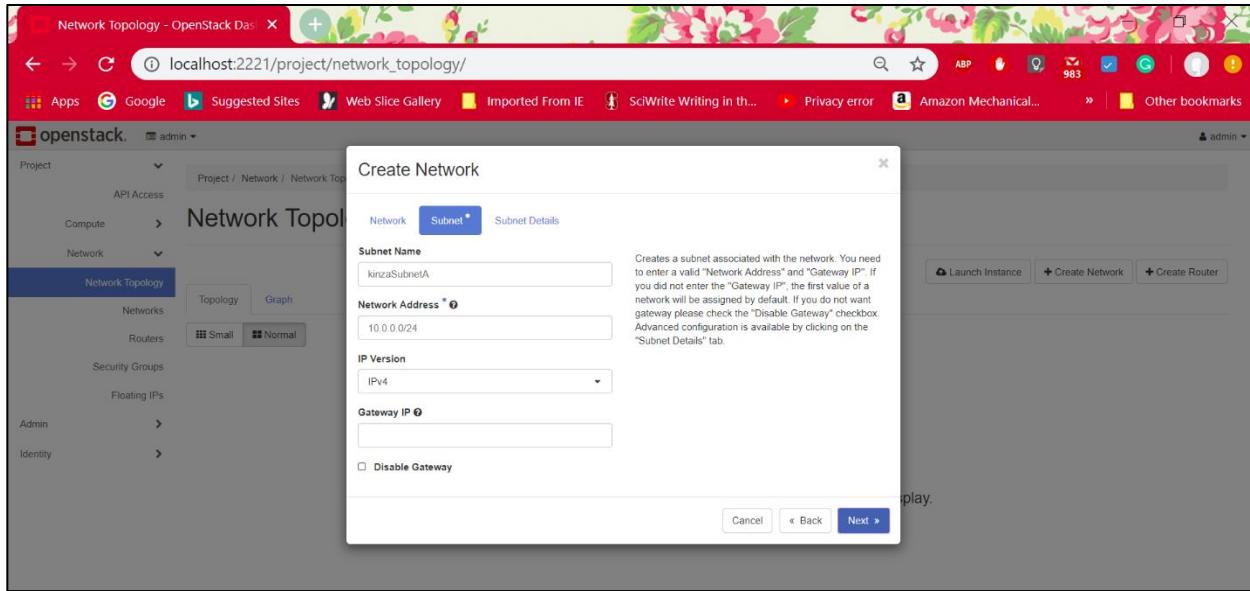
Enable Admin State:

Shared:

Create Subnet:

Availability Zone Hints: nova

Next >

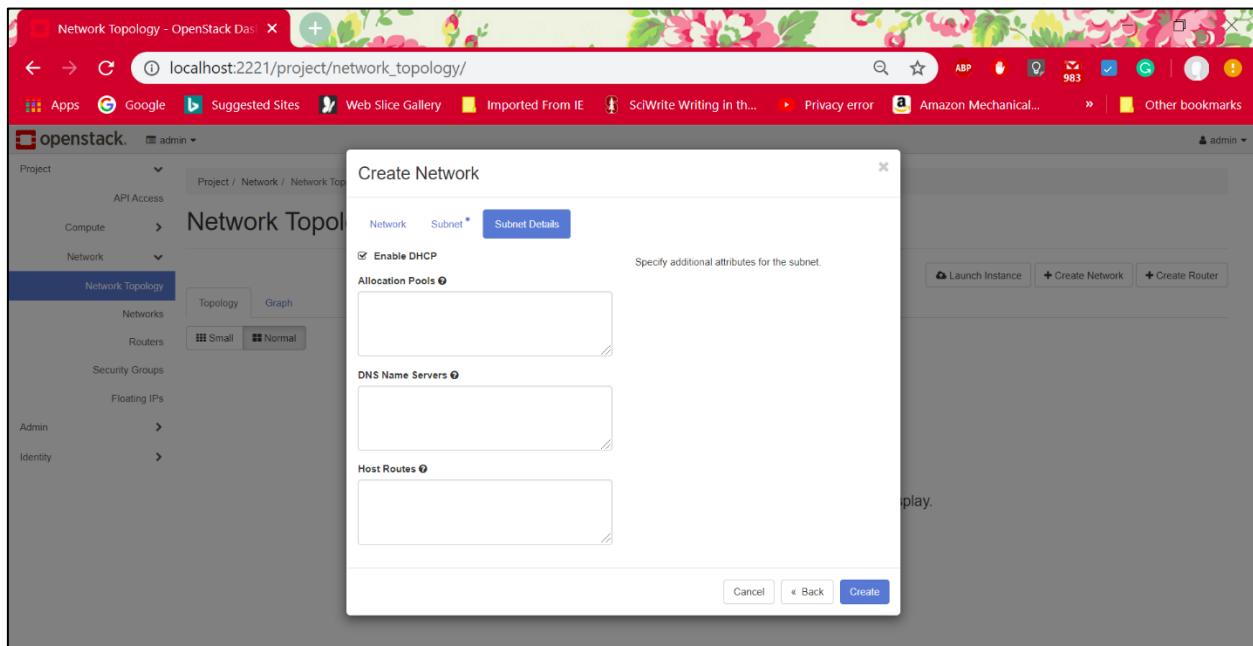


3. On clicking Next in the previous, the following screen can be seen . By default DHCP is enabled which will allocate IP addresses to virtual machine dynamically.

Allocation pools is optional:The IP addresses specified in this pool are used to assign to the subnets. It makes sure that there is no overlapping assignment. The allocation pool can be created in the following two ways:

Project: If the allocation pool is created with in the regular project then it doesn't show to any other network.

Admin: If the allocation pool is created by Admin then it can be shared by other projects¹.



¹ <https://docs.openstack.org/newton/networking-guide/config-subnet-pools.html>

4. **Alternative way of adding subnet:** If **Create subnet** is not enabled while creating network, the subnet details can be entered later by going to **Networks** and then selecting particular network to update the subnet details by clicking on **Edit Subnet**.

Name	Network Address	IP Version	Gateway IP	Actions
kinzaSubnetA	10.0.0.0/24	IPv4	10.0.0.1	Edit Subnet

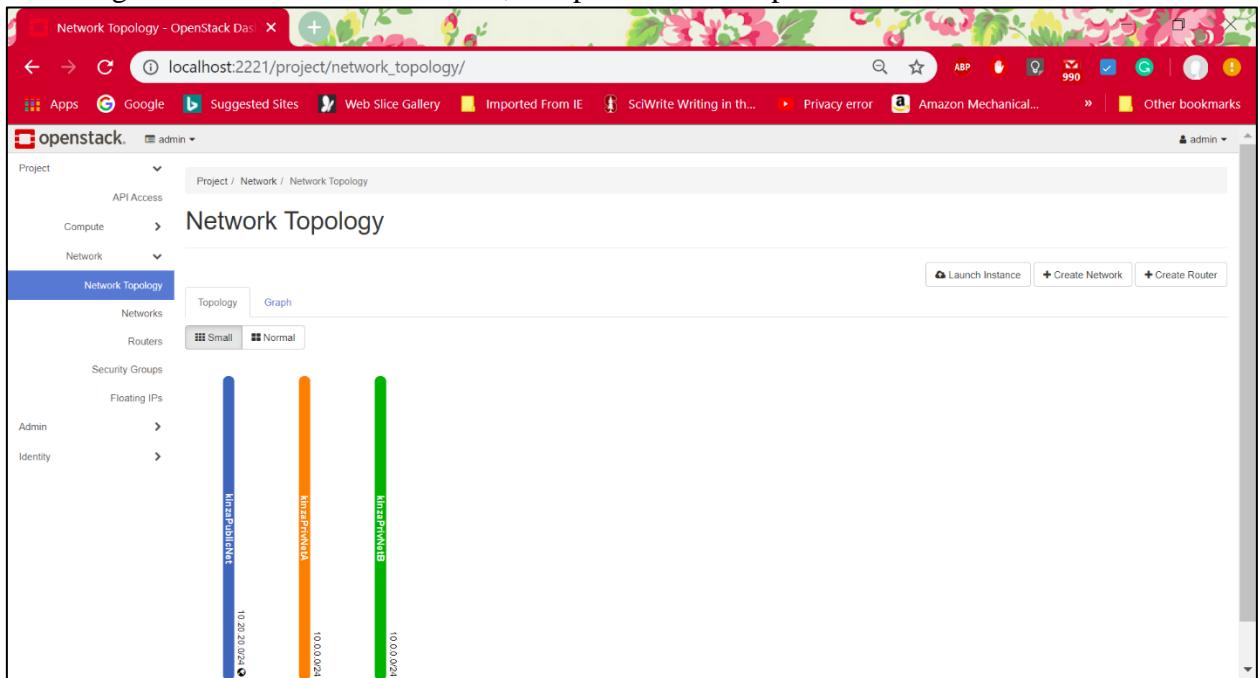
5. Repeated the same steps for creating second private network with the following details

Private Network Name: **kinzaPrivNetB**

Subnet Name: **kinzaSubnetB**

Network address: **10.0.0.0/24**

The following screen shows three networks, one public and two private.

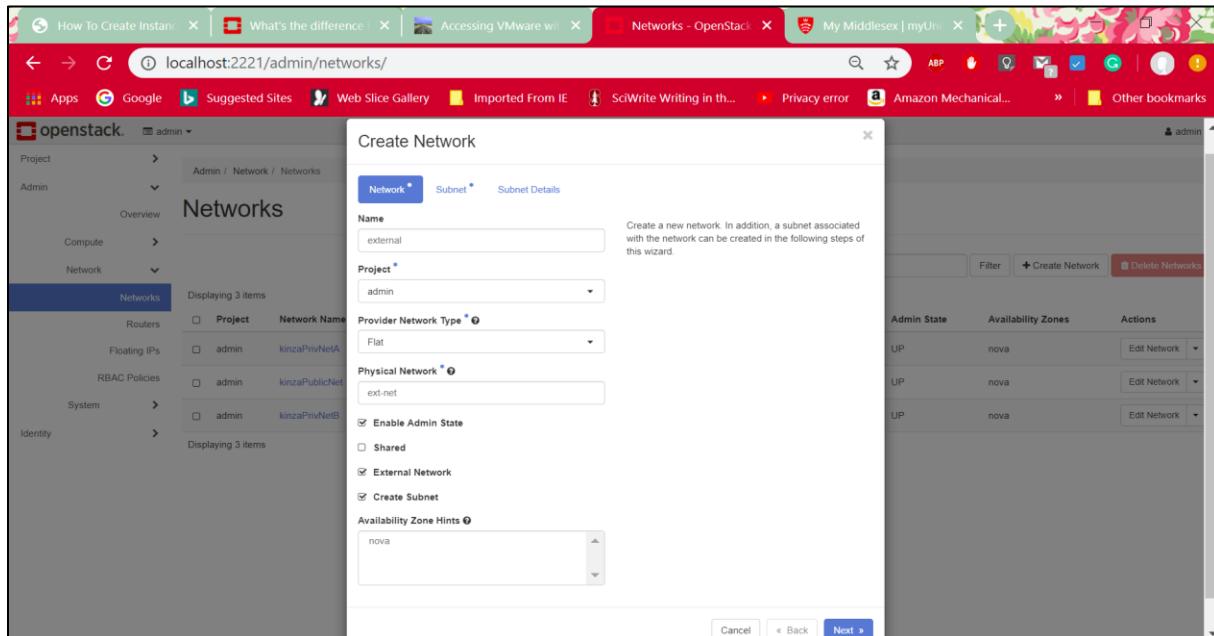


Public Network :

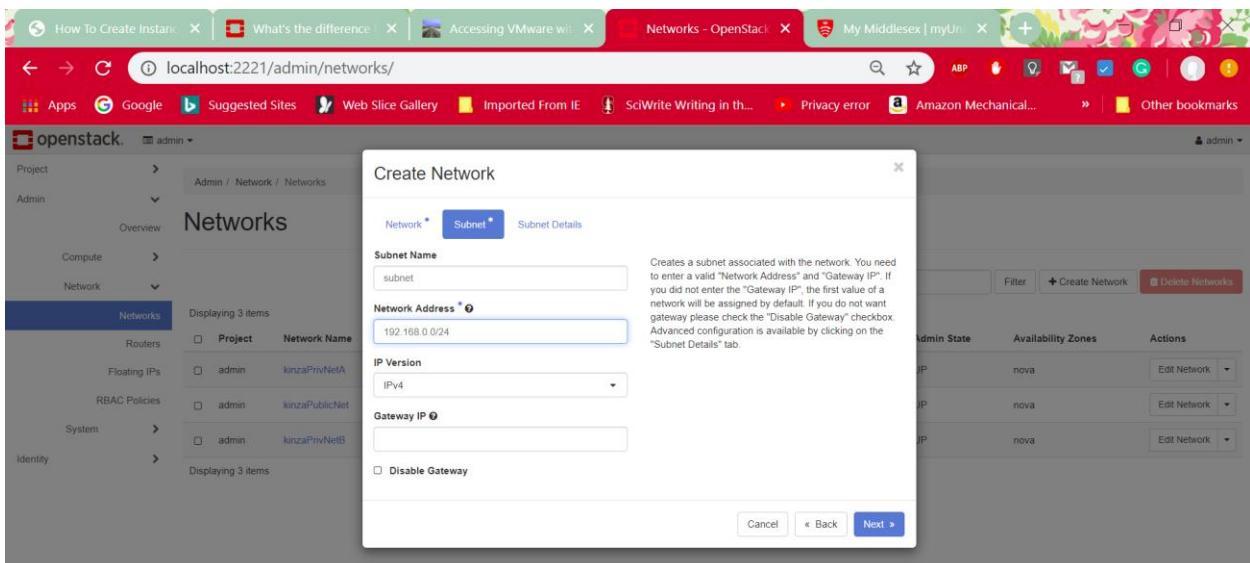
Option 1(Creating Public Network):

Clicked on Admin>>Network>>Networks and can go to create network

- **Name:** external
- **Project:** admin because admin can only create the public network.
- **Provider Network type:** Flat which allows connection to outside world.
- Checked the box next to External network



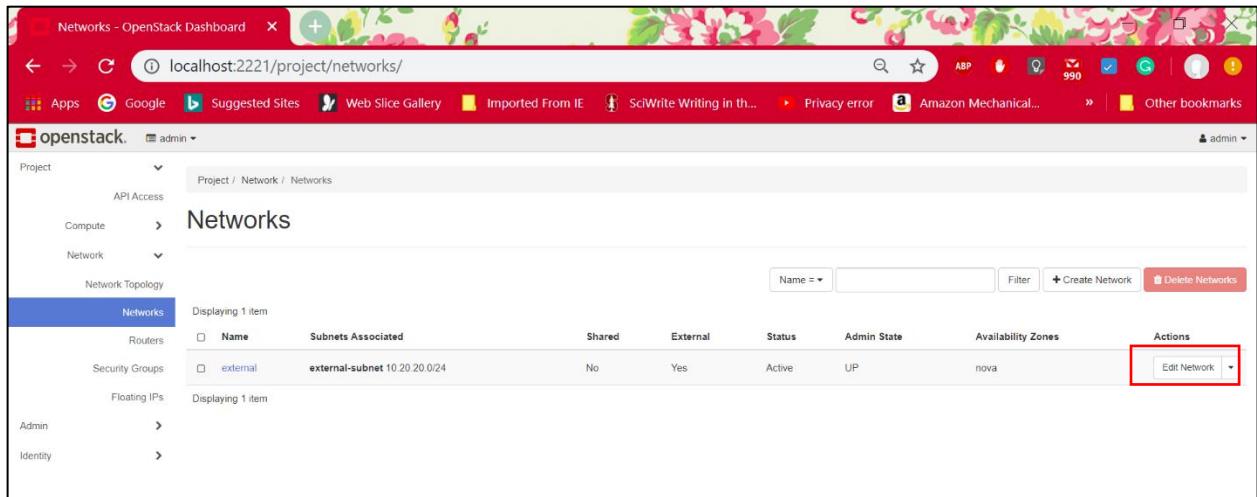
2. The following screen shows adding subnet details which will be used to assign floating IP's to instances while associating.



Option 2(Default External Network):

Clicked on Project>>Network>>Network Topology

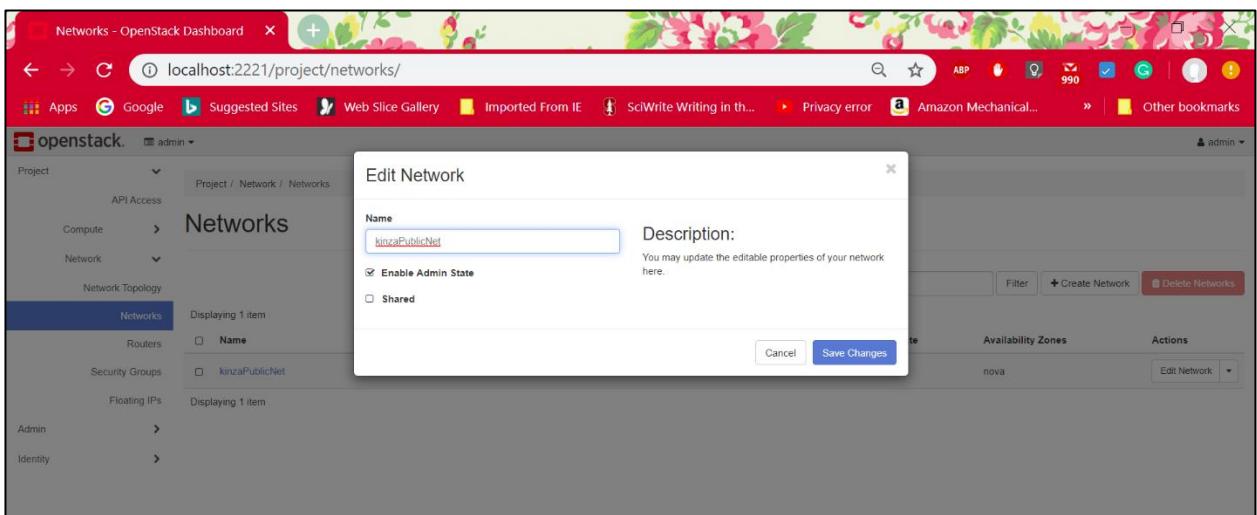
1. On Network Topology page, already have a public/external network. Renaming the public network



The screenshot shows the OpenStack Dashboard with the URL `localhost:2221/project/networks/`. The left sidebar is collapsed. The main content area is titled "Networks". A table displays one item:

Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
external	external-subnet 10.20.20.0/24	No	Yes	Active	UP	nova	Edit Network

2. Renamed to *kinzaPublicNet*.



The screenshot shows the OpenStack Dashboard with the URL `localhost:2221/project/networks/`. The left sidebar is collapsed. A modal dialog box is open, titled "Edit Network". Inside the dialog:

- Name:** kinzaPublicNet
- Description:** You may update the editable properties of your network here.
- Enable Admin State:** checked
- Shared:** unchecked

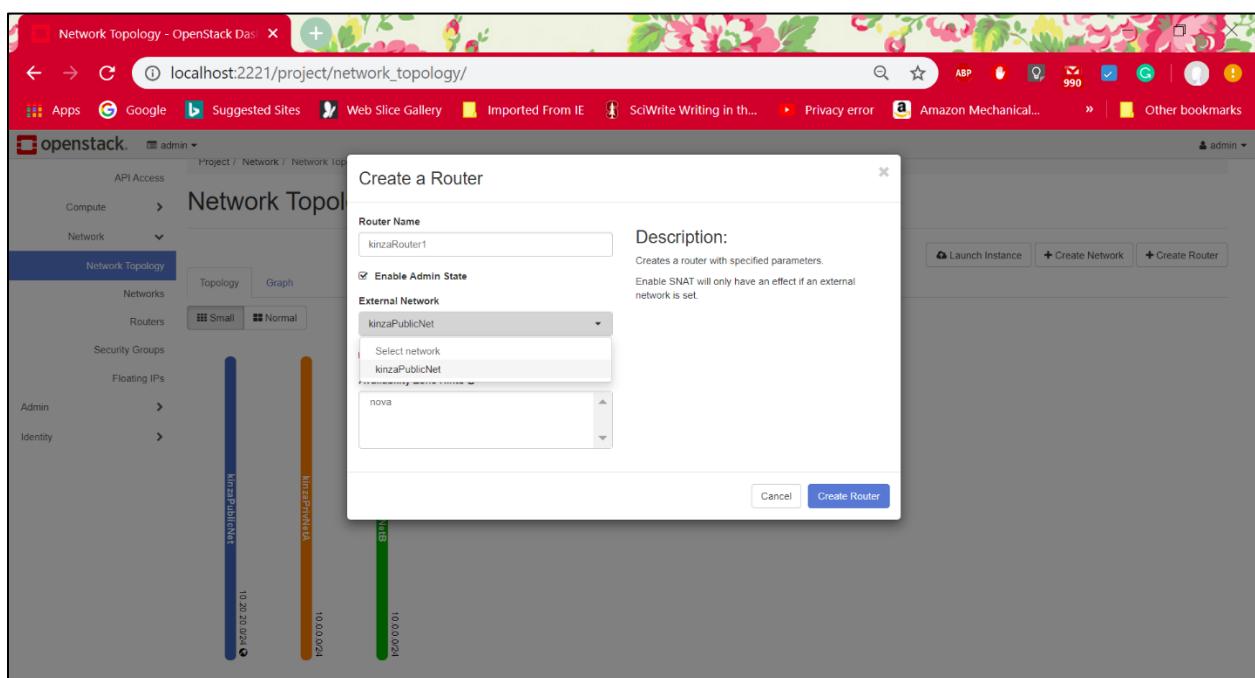
At the bottom right of the dialog are "Cancel" and "Save Changes" buttons. The background shows the same Networks table as the previous screenshot, with the "external" network now renamed.

Routers Creation:

Now, connecting the both private network to public network so that private networks can make connection to the outside world. This is possible using Routers. Routers perform packets forwarding between network and Network Address Translation(NAT) which maps private IP to Floating IP. The reason behind this is the servers on the public network can not process private IP address and respond. So, Routers maps private IP address of the network device if it wants to send request to a server on the public network².

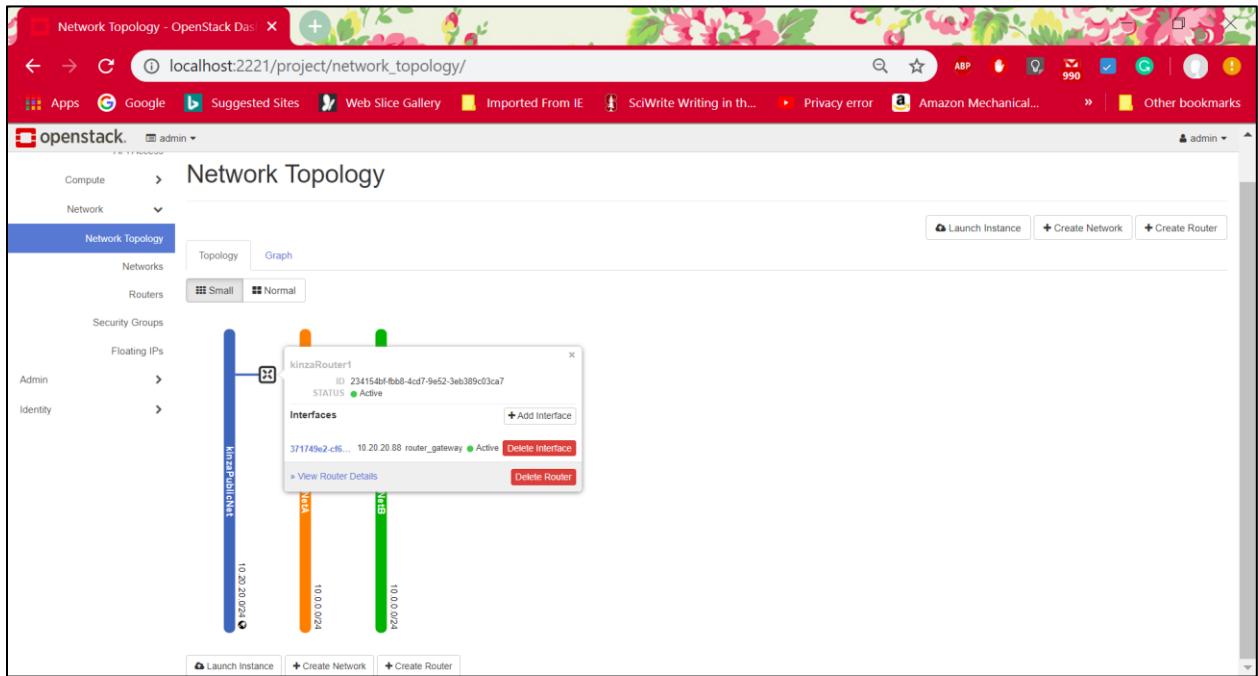
For that purpose creating the Routers in the following steps.

- i) Click on the following **Project>Network> Network Topology** and clicked on **Create Router** button. Gave Router Name: kinzaRouter1 and enabled admin state and assigned public network to which there is need to connect the private networks.

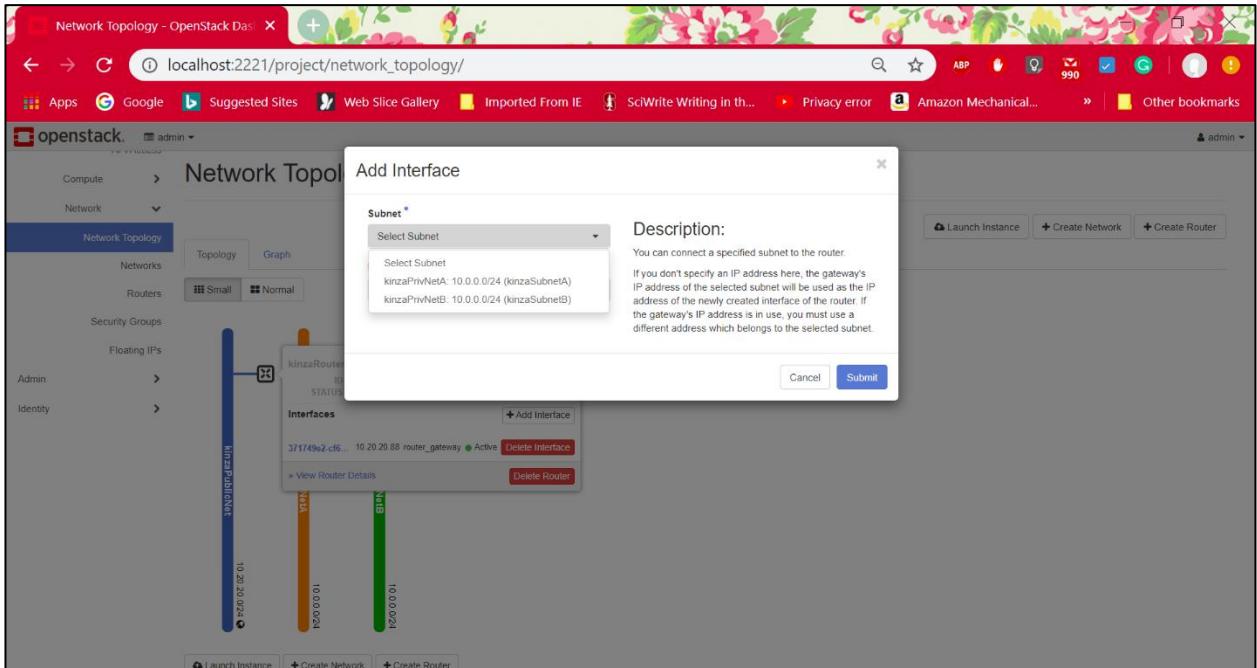


² <https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/router.html>

- ii) The following screen shows router attached to the public/external network. On Router, to attach private networks can use the **Add Interface** option.

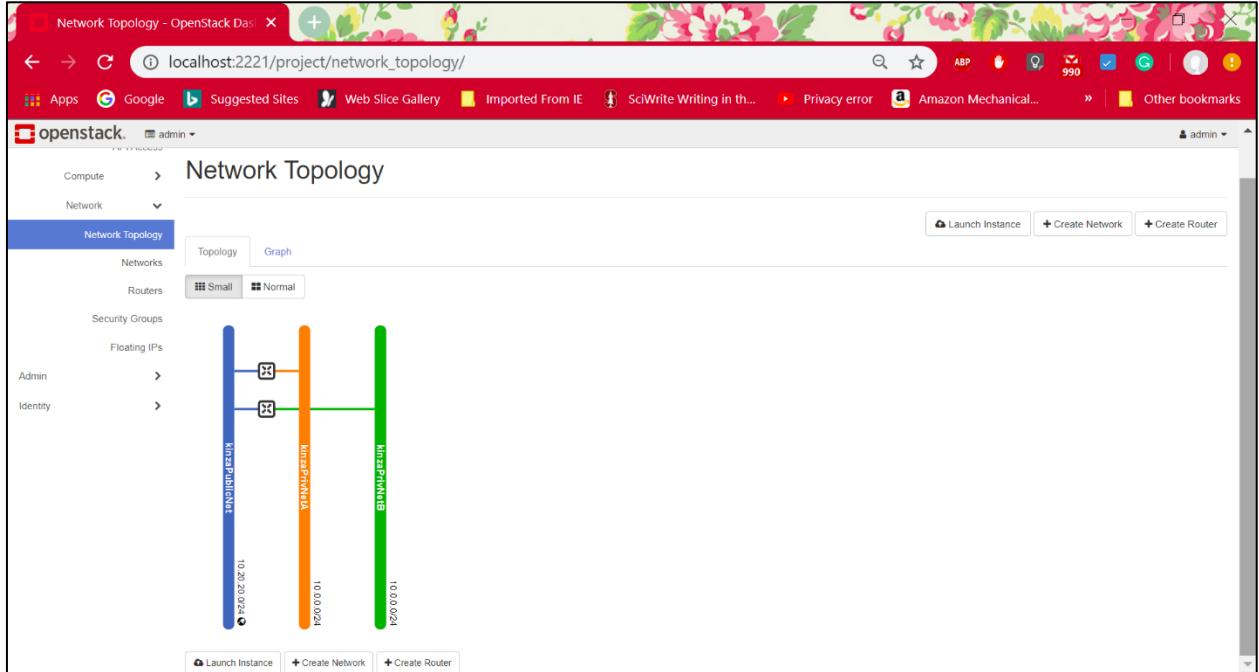


- iii) After clicking on Add Interface, It shows list of private networks exists to add to interface. Selecting the **kinzaPrivNetA** for interface will add this network to the router and connect to the public network. Clicked on **Submit** button



- iv) Repeated the same procedure to create second router with name **kinzaRouter2** and used it to connect **kinzaPrivNetB** to the external/public network **kinzaPublicNet**. The final network topology is shown in the following screen. It can be seen links between

blue (public network) to other two private networks connecting through routers. Now the private networks are able to communicate the outside world.



- Creating private network/ Internal network allows instances to communicate with each other within same private network and transfer or copy the files.
- External Network/public network will be used by instances of private networks to communicate outside world and can send and receive traffic from outside.
- Only Admin user can create the external network/public network
- Provider Network type : Flat/vlan allows to connect to router

Task 4:

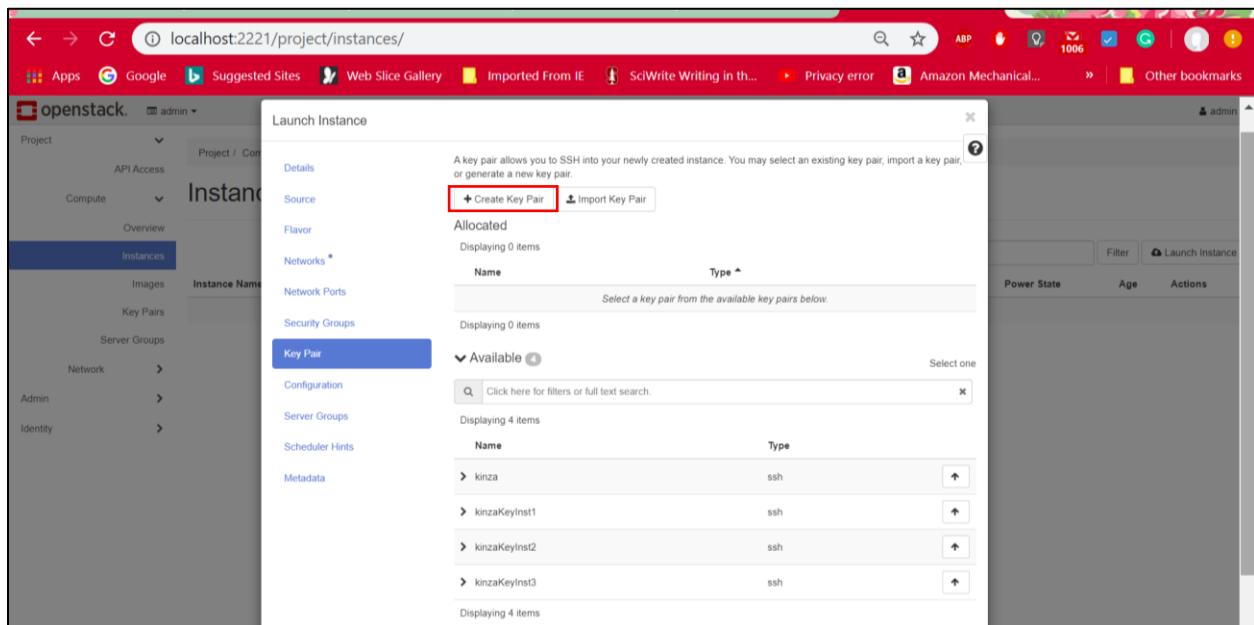
Creating three different keypairs which will be assigned to instance while launching them. This task has been performed before Task 2 and Task 3 because creating key pairs is best to do and then can assign key pair while creating instances.

Public key is injected into instance while launching and private key can be used to make ssh connection instead of giving password authentication. Open stack automatically downloads the private key file which must be kept secure to make a connection. Private key is more strong way to have security over network. One keypair can be used for many instance within project.

The ways to create keypairs:

Option1:

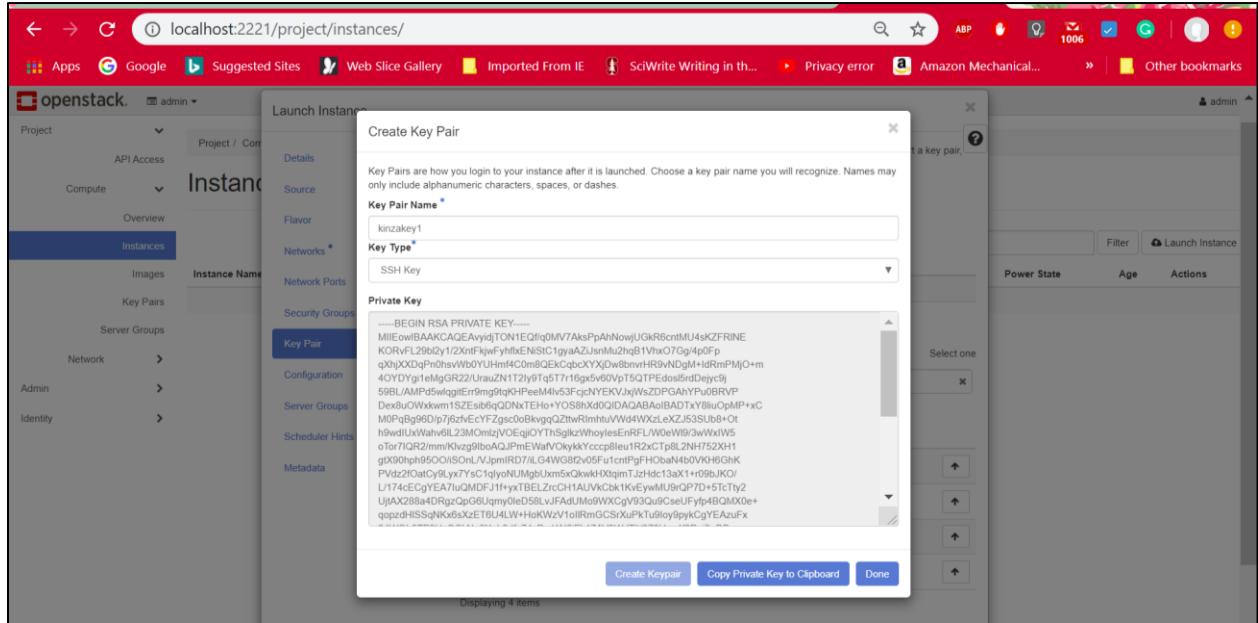
One way to create keypair is creating while launching an instance. Selected
Project>>Compute>>Instances>>Launch Instance



The screenshot shows the 'Launch Instance' dialog box from the OpenStack Compute interface. The 'Key Pair' tab is selected. A red box highlights the '+ Create Key Pair' button. Below it, the 'Available' section lists four key pairs: 'kinza', 'kinzaKeyInst1', 'kinzaKeyInst2', and 'kinzaKeyInst3', all of type 'ssh'. The 'Allocated' section shows 0 items.

Name	Type
kinza	ssh
kinzaKeyInst1	ssh
kinzaKeyInst2	ssh
kinzaKeyInst3	ssh

When the **Create keypair** is selected, it shows the following screen . Then keypair can be seen.



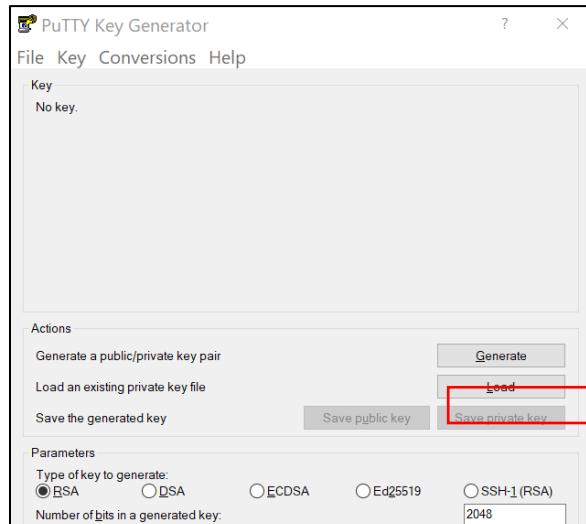
Limitations: It doesn't automatically download the keypair file. The user has to copy the key in some new file and save it manually.

Option 2:

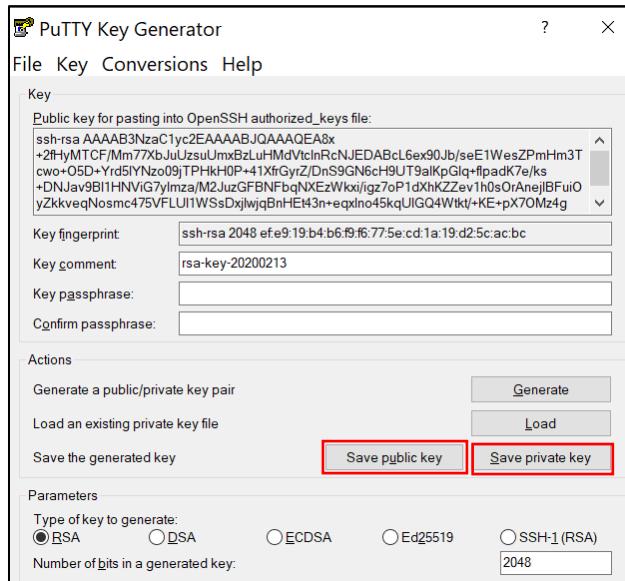
Importing the key:

Importing key is another option to use while launching the instance. The advantage of importing key is that the keypairs which are used by other sources can be used for openstack as well making it more convenient for the user. For example, PuttYgen which is putty key generator can be used to generate public and private key and then can be imported public key in openstack.

1. Clicked on **Generate** button



2. The following screen shows the generated public key. Can copy this key in the open stack or either save it clicking on **Save Public key** and **Save Private Key** button. Then, public key can be imported into openstack whereas private key can be kept safe.



3. Clicked on **Import Public Key** button on the following screen

The screenshot shows the OpenStack Dashboard under the 'Key Pairs' tab. The left sidebar has categories like Project, API Access, Compute, Network, Admin, and Identity. Under Compute, 'Key Pairs' is selected. The main area displays a table of key pairs. The first row shows 'kinzakKeyInst1' as an ssh type. The 'Import Public Key' button at the top right of the table is highlighted with a red box. The table includes columns for Name, Type, and actions (Delete Key Pair).

Name	Type	Action
kinzakKeyInst1	ssh	Delete Key Pair
kinzakKeyInst2	ssh	Delete Key Pair
kinzakKeyInst3	ssh	Delete Key Pair

4. After clicking import public key, the following screen shows copied public key generated from PuttYgen software.

Import Public Key

Key Pair Name *

Key Type *

SSH Key

Load Public Key from a file

Choose File No file chosen

Public Key * (Modified)

Content size: 397 bytes of 16.00 KB

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAQEA8x+2fHyMTCF/Mm77XbJuUzsUUmxBzLuHMdVtclnRcNJEDABC6ex90Jb's
eE1WesZPmHm3Tcw+O5D+Yrd5YNz009jTPHkH0P+41XfrGyZ/DnSGN6cH9UT9alKpGlq+flpadK7e/ks+DNJava
B1HNvIG7yImza/M2JuzGFBNFbqNXEzWxki/gz7oP1dXhKZZev1h0sOrAnejlBfuiOyZkkveqNoscma475VFLU1WSs
DxjlwjqBnHEt43n+eqxlno45kqUIGQ4Vltkt/+KE+pX7OMz4g+DRS/5y/NnFC0c5QgkcmOrvS3/E3FUjR8yFNcys0hlw
hJSEAEEMEQ== rsa-key-20200213
```

Cancel **Import Public Key**

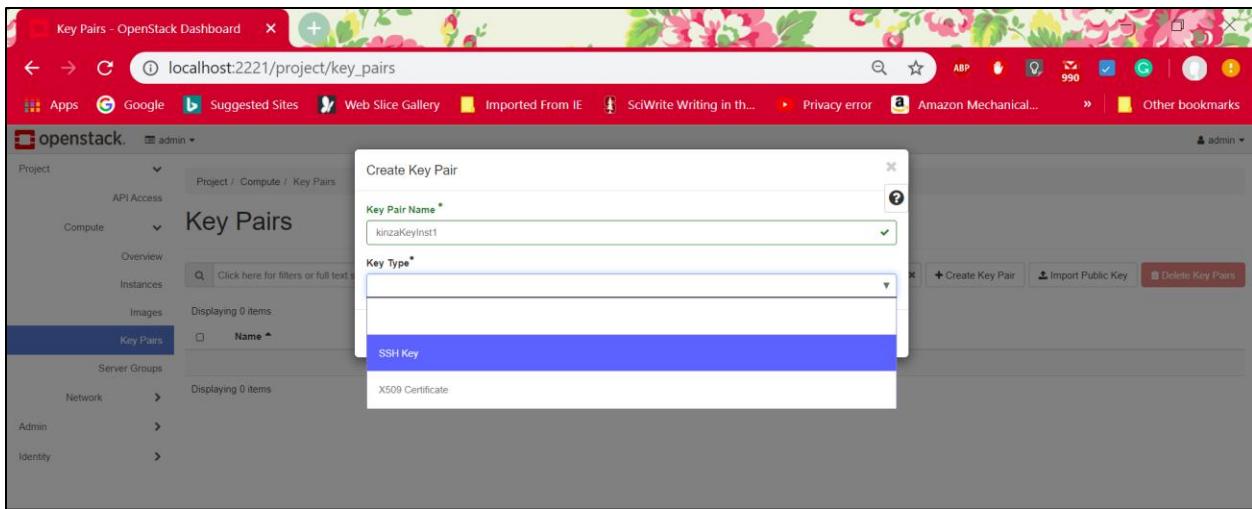
Option 3:

Used the following method to create three different key pairs for three instances.

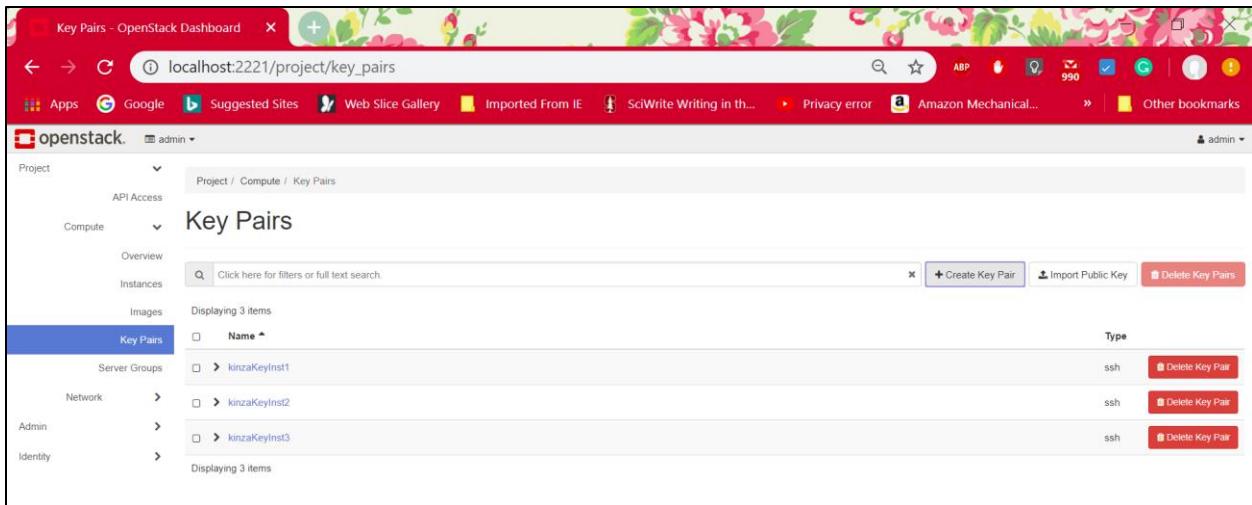
1. Clicked on Project>>Compute>>Key Pairs and clicked on Create Key Pair

Name	Type	Action
kinzaKeyInst1	ssh	Delete Key Pair
kinzaKeyInst2	ssh	Delete Key Pair
kinzaKeyInst3	ssh	Delete Key Pair

2. After clicking, the following screen displays where entered **kinzaKeyInst1** and type of the key is SSH key because this key will be used while creating SSH connection with the instances.



3. Repeated the same procedure for creating two more keys. And the following screen shows all three different key pairs.



Task 2 and Task 3:

Creating instances and adding them to networks.

Flavor Creation:

Flavor defines the computational power, storage and main memory of virtual server. Flavors is giving virtual resources to the virtual server. Creating flavour because default flavors donot have the same specification required by the task.

Steps:

1. Clicked Admin>Compute>Flavor>Create Flavor and the following screen shows the details to provide. Name of the flavour, ID which is automatically generated, VCPU's means number of Virtual CPUs. RAM, Root disk Ephemeraldisk which is lost with the end

of lifecycle of an instance. By default Ephemeral disk, swap disk are 0 of not entered. TX/RX factor is the bandwidth of the network and by default is 1³.

The screenshot shows the OpenStack Dashboard with the URL localhost:2221/admin/flavors/. On the left, the navigation menu is visible with 'Compute' selected under 'Instances'. A modal window titled 'Create Flavor' is open, showing fields for 'Name' (KinzaFlavor), 'ID' (auto), 'VCPU's (1), 'RAM (MB)' (64), 'Root Disk (GB)' (1), 'Ephemeral Disk (GB)' (0), 'Swap Disk (MB)' (0), and 'RX/TX Factor' (1). To the right, a table lists existing flavors: m1.large, m1.medium, m1.small, m1.tiny, and m1.xlarge.

ID	Public	Metadata	Actions
4	Yes	No	<button>Update Metadata</button>
3	Yes	No	<button>Update Metadata</button>
2	Yes	No	<button>Update Metadata</button>
1	Yes	No	<button>Update Metadata</button>
5	Yes	No	<button>Update Metadata</button>

2. After Creating Flavor, the added flavour can be seen in the following screen shot.

The screenshot shows the 'Flavors' list after creating 'KinzaFlavor'. The table includes columns: ID, Flavor Name, VCPUs, RAM, Root Disk, Ephemeral Disk, Swap Disk, RX/TX factor, ID, Public, Metadata, and Actions. The 'KinzaFlavor' row is highlighted with a red box. The table now displays 6 items.

ID	Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	RX/TX factor	ID	Public	Metadata	Actions
4	m1.large	4	8GB	20GB	0GB	0MB	1.0	4	Yes	No	<button>Update Metadata</button>
3	m1.medium	2	4GB	20GB	0GB	0MB	1.0	3	Yes	No	<button>Update Metadata</button>
2	m1.small	1	2GB	20GB	0GB	0MB	1.0	2	Yes	No	<button>Update Metadata</button>
1	m1.tiny	1	512MB	1GB	0GB	0MB	1.0	1	Yes	No	<button>Update Metadata</button>
5	m1.xlarge	8	16GB	20GB	0GB	0MB	1.0	5	Yes	No	<button>Update Metadata</button>
	KinzaFlavor	1	64MB	10GB	0GB	0MB	1.0	d921b478-1a26-4b15-b0fd-c4f9a0f6950a	Yes	No	<button>Update Metadata</button>

Instances Creation:

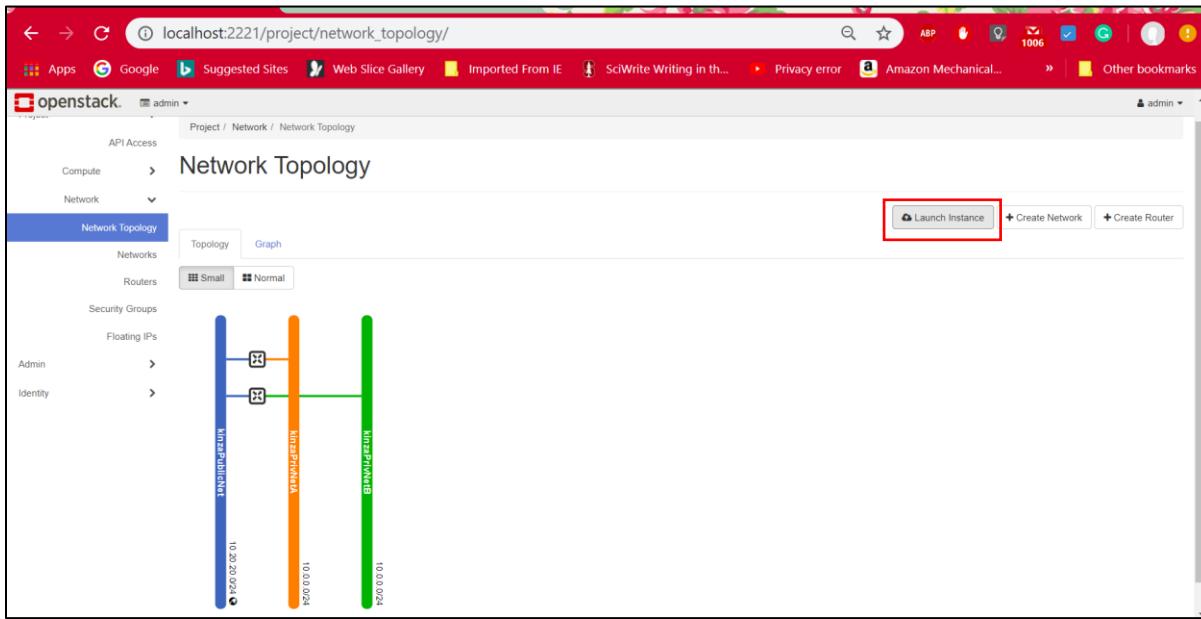
Instances are virtual machine created inside a cloud. The instance **source** could be image, snapshot which contains only description/template for creating a virtual machine. Then, selecting **flavor** which decides the computational, storage and memory specification for the instance. **Networks** is

³ <https://docs.openstack.org/horizon/latest/admin/manage-flavors.html>

attaching a network with the instance. **Key pairs** defines to make ssh connection to the instances. A **security group** is the rules defining the type of incoming and outgoing traffic from network.

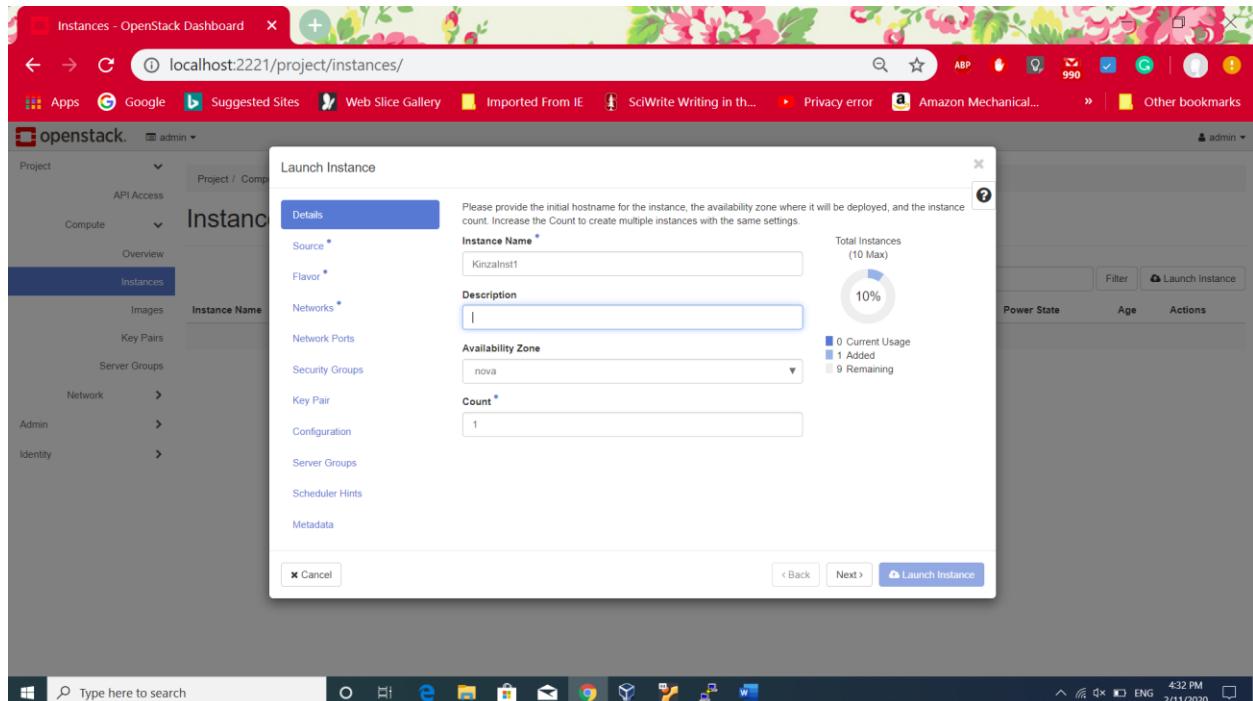
Option 1:

Instance can be created going to **Project>> Network>>Network topology** and clicking **Launch Instance** button.

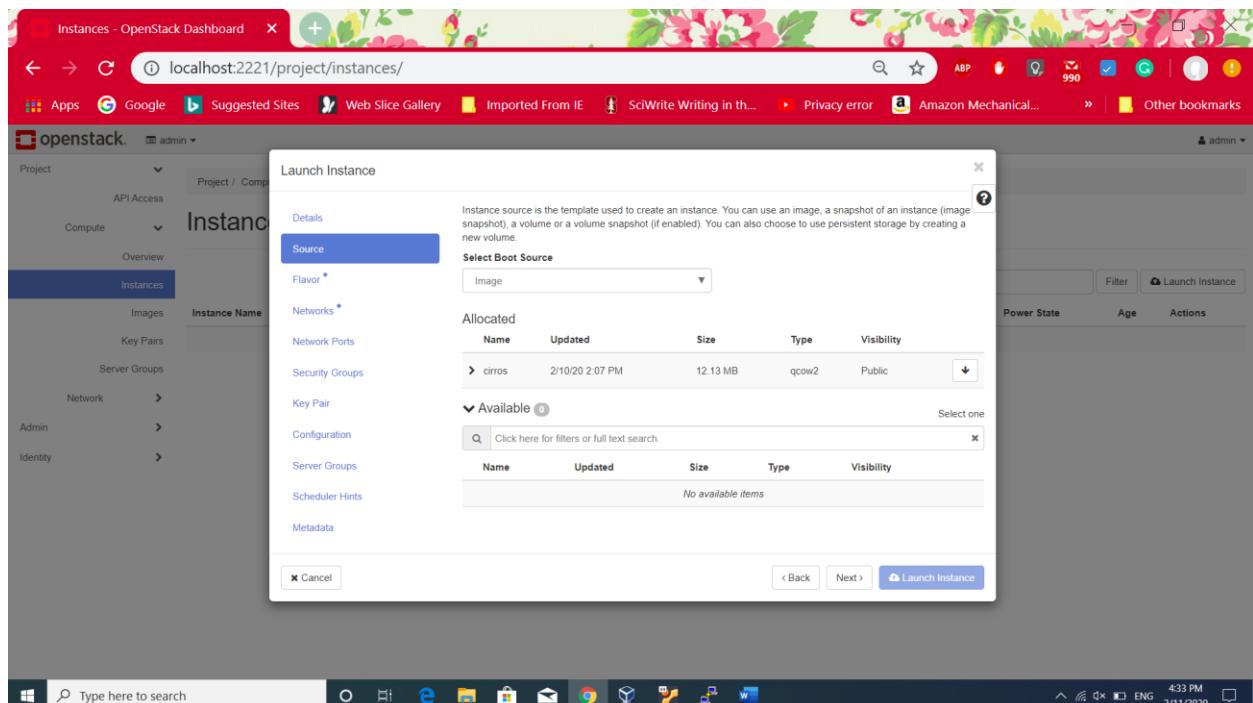


Option 2:

- Clicked on Project>Compute>Instances>Launch Instance. The following screen appears.



- Clicked **Next** after entering **KinzaInst1**. And selected the **cirros** image which is the only available option.



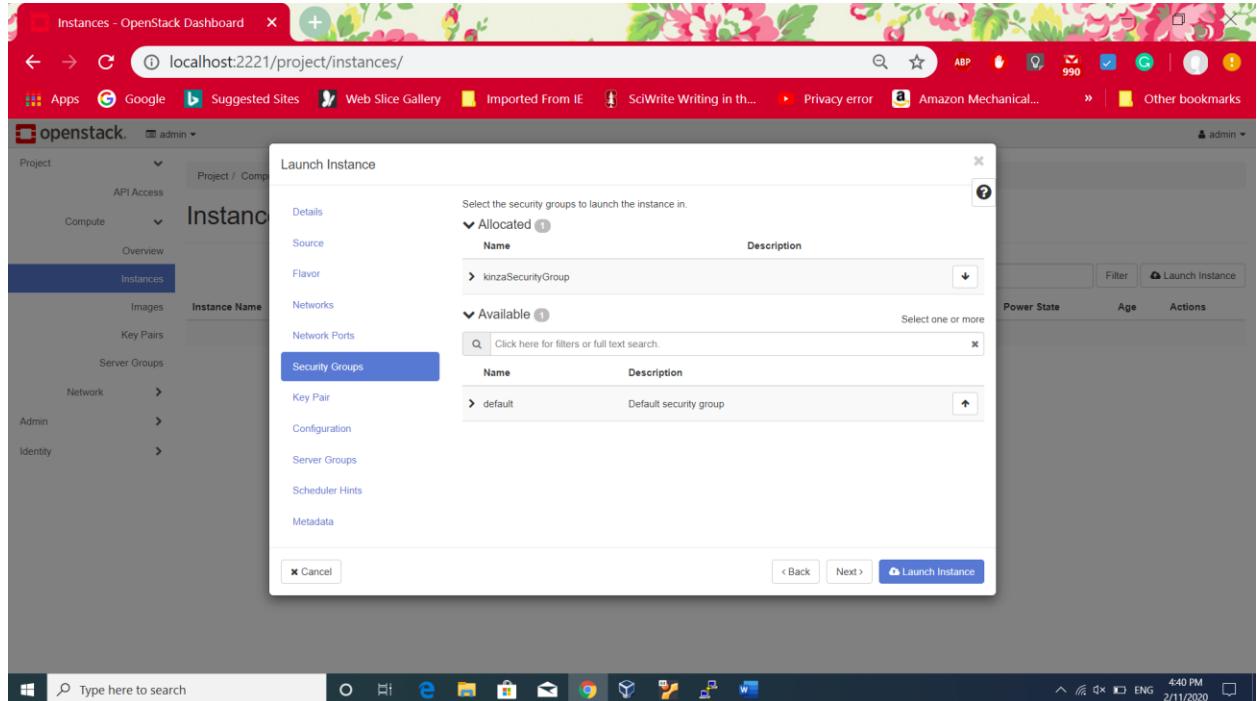
3. On Next screen, Selected the previously created **KinzaFlavor** .

The screenshot shows the 'Launch Instance' dialog box over a web browser window. The URL is `localhost:2221/project/instances/`. The left sidebar shows 'Instances' selected under 'Compute'. The main dialog has 'Flavor' selected in the 'Source' dropdown. The 'Allocated' section shows one entry: 'KinzaFlavor' with 1 vCPU, 64 MB RAM, 1 GB Total Disk, 1 GB Root Disk, 0 GB Ephemeral Disk, and Yes Public. The 'Available' section lists several other flavors: m1.tiny, m1.small, m1.medium, m1.large, and m1.xlarge, each with their respective specifications. At the bottom right of the dialog is a blue 'Launch Instance' button.

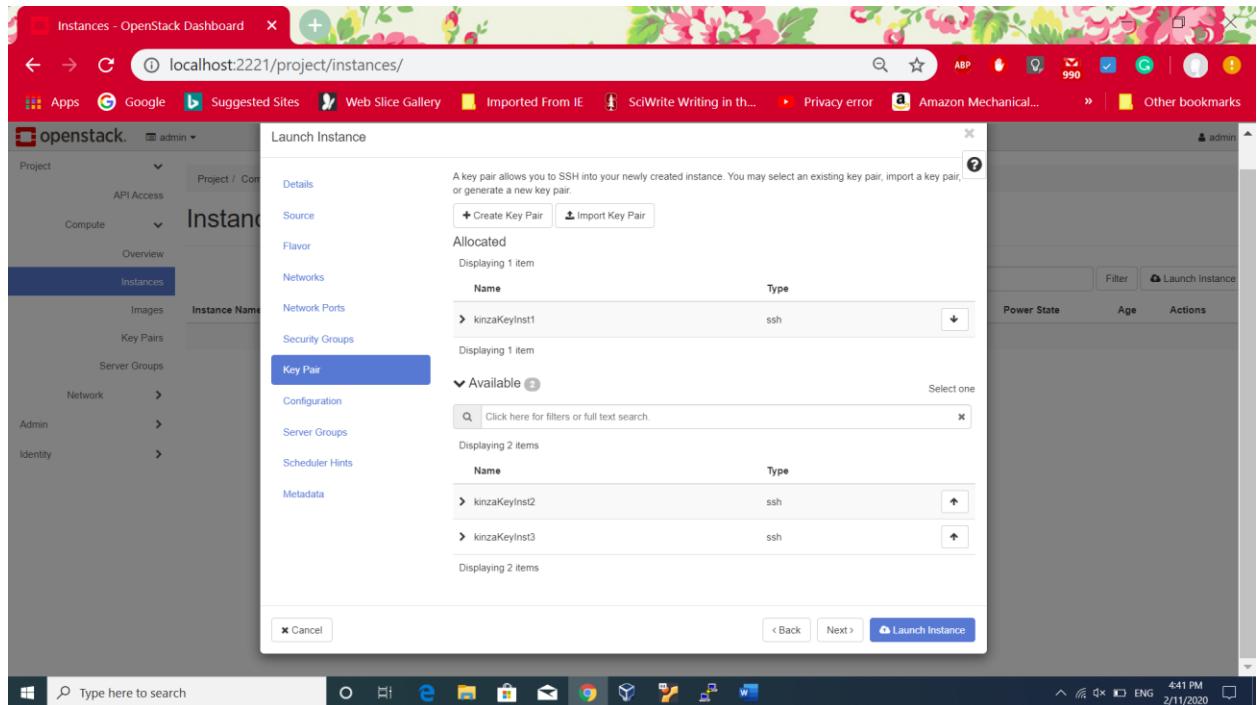
3. Next step, selected **kinzaPrivNetA** for the KinzaInst1 as required by the task.

The screenshot shows the 'Launch Instance' dialog box over a web browser window. The URL is `localhost:2221/project/instances/`. The left sidebar shows 'Instances' selected under 'Compute'. The main dialog has 'Networks' selected in the 'Source' dropdown. The 'Allocated' section shows one entry: 'kinzaPrivNetA' associated with 'kinzaSubnetA', marked as Shared No, Admin State Up, and Status Active. The 'Available' section lists two networks: 'kinzaPublicNet' associated with 'external-subnet', and 'kinzaPrivNetB' associated with 'kinzaSubnetB', both marked as Shared No, Admin State Up, and Status Active. At the bottom right of the dialog is a blue 'Launch Instance' button.

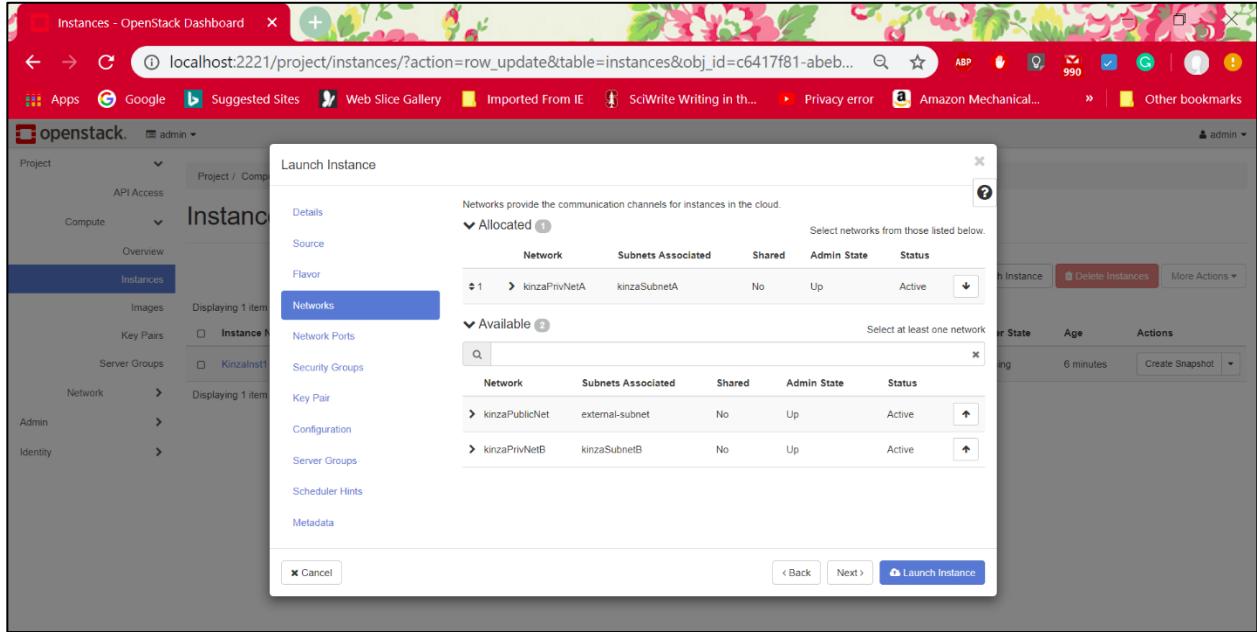
4. Now adding **Security Group**, the default security group allow only ssh connection to the instance. So, created new security group with **KinzaSecurityGroup** having ingress which is incoming and egress which is outgoing traffic.



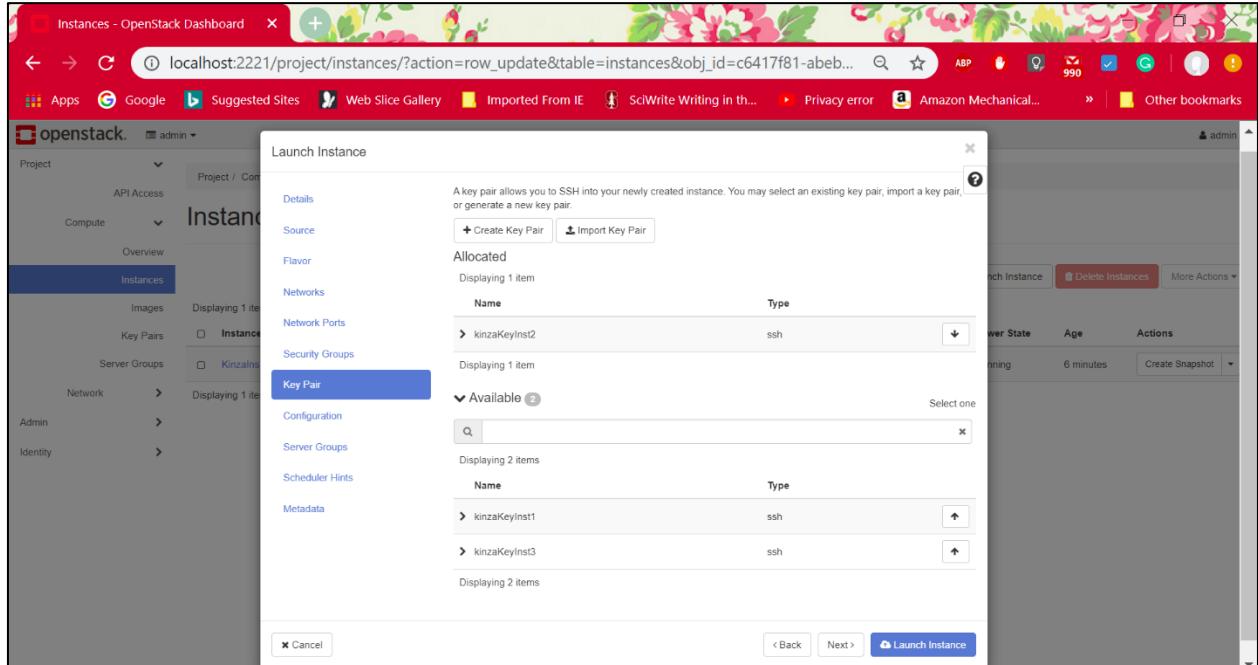
5. Next , selected the first keypair **kinzaKeyInst1** for the first instance.



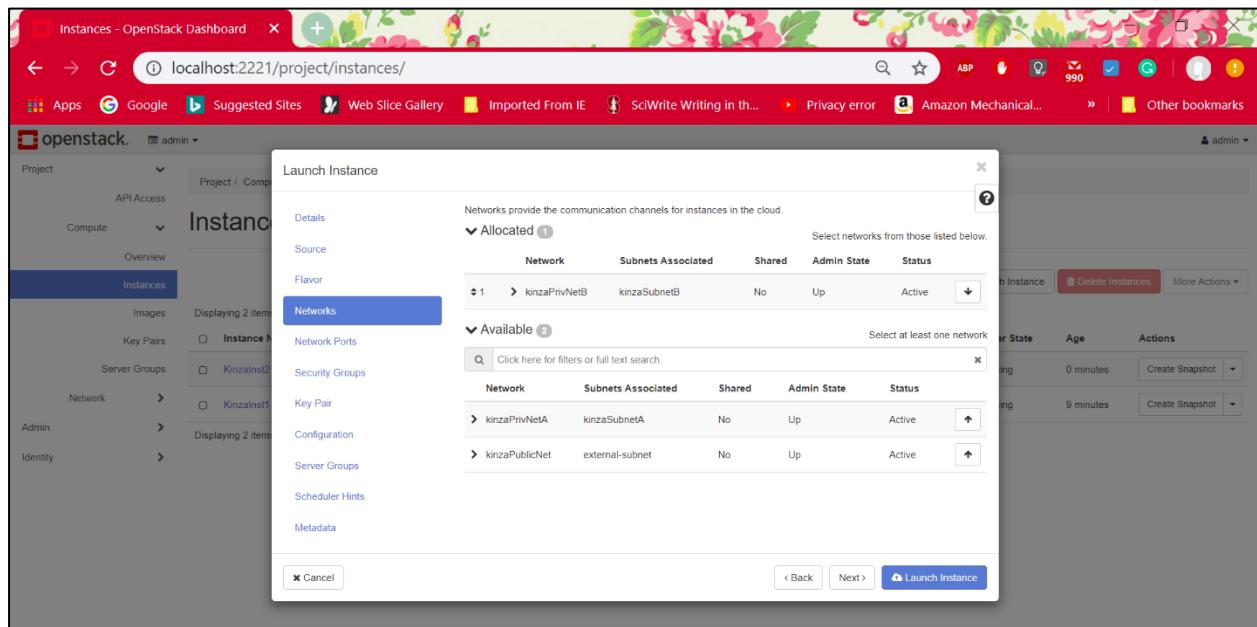
6. Clicked on Launch Instance. Repeated the same procedure for creating second instance except assigning the same private network **kinzaPrivNetA** to **KinzaInst2**.



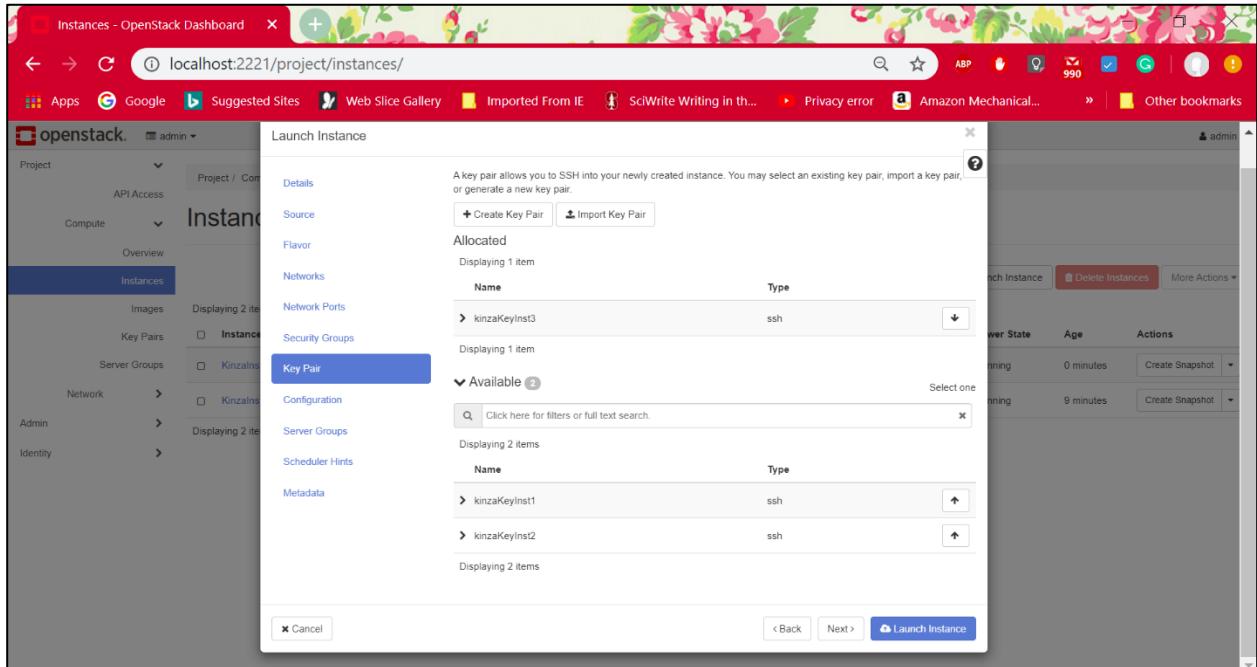
7. The **kinzakeyInst2** has been assigned to **kinzaInst2**.



8. For Instance3 , assigned private network with name **kinzaPrivNetB**



9. For Instance3, assigned the key named **kinzaKeyInst3**



10. After creating three instance, the following screen shows created instances.

The screenshot shows the OpenStack Dashboard's Instances page. The left sidebar has 'Compute' selected under 'Instances'. The main area displays a table of instances:

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
kinzalinst3	cirros	10.0.0.203	KinzaFlavor	kinzaKeyInst3	Active	nova	None	Running	0 minutes	Create Snapshot
kinzalinst2	cirros	10.0.0.79	KinzaFlavor	kinzaKeyInst2	Active	nova	None	Running	1 minute	Create Snapshot
kinzalinst1	cirros	10.0.0.218	KinzaFlavor	kinzaKeyInst1	Active	nova	None	Running	2 minutes	Create Snapshot

Task No. 5:

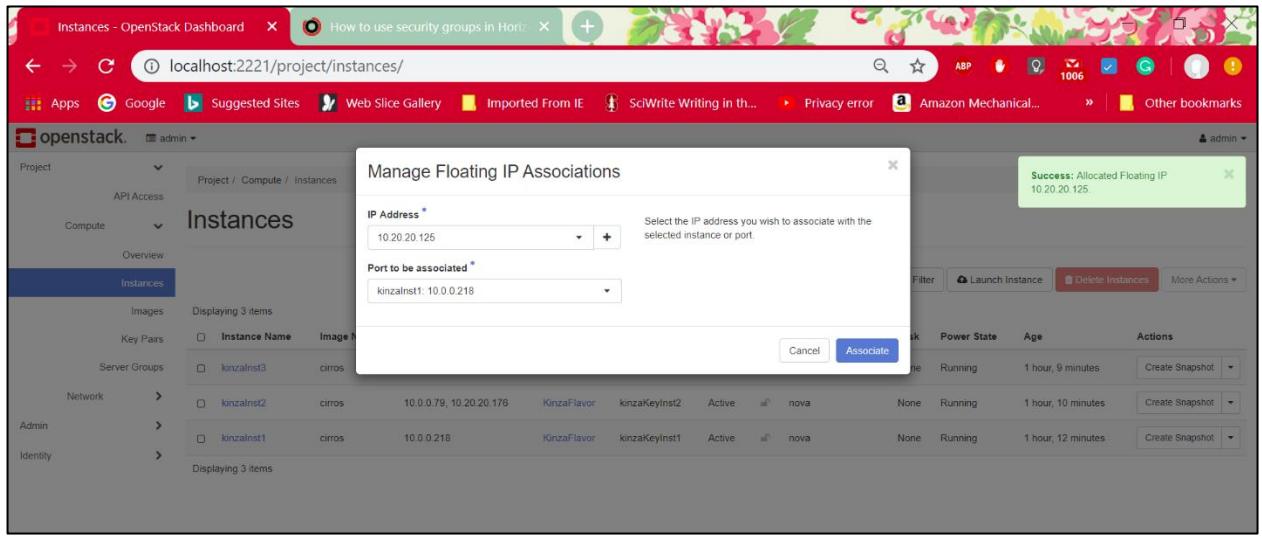
Associating the floating IP address and managing the security group

Associating the floating IP address:

1. A floating IP is assigned to instance(VM) which is used to connect to external network. It can be changed after the creation of instance. The following screen show the option available in front of each instance for associating IP address.

The screenshot shows the same OpenStack Dashboard Instances page as before. The context menu for the first instance, 'kinzalinst1', is open. The 'Associate Floating IP' option is highlighted in red.

2. After clicking **Associate IP address**, the following screen shows up. IP address is the floating IP address retrieved from selecting external network. And **port to be associated** option is the selection of instance. Click on **Associate**.

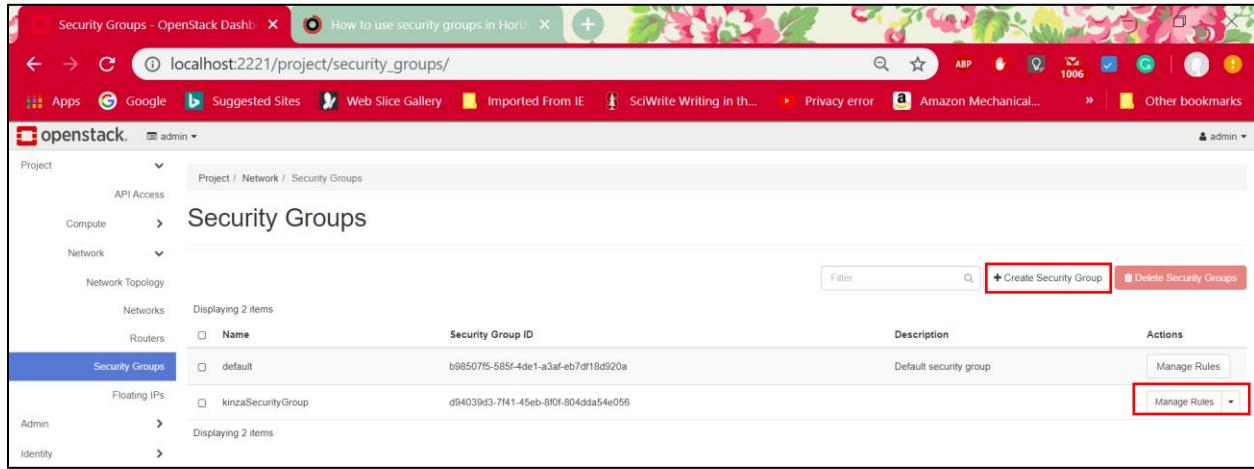


3. The above procedure repeated for all three instances and the following screen shows. Now, floating IP's can be seen associated with the private ip address of instances.

Instance ID		Filter	Launch Instance	Delete Instances	More Actions			
IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
10.0.0.203, 10.20.20.14	KinzaFlavor	kinzaKeyInst3	Active	nova	None	Running	1 hour, 10 minutes	Create Snapshot
10.0.0.79, 10.20.20.176	KinzaFlavor	kinzaKeyInst2	Active	nova	None	Running	1 hour, 11 minutes	Create Snapshot
10.0.0.218, 10.20.20.125	KinzaFlavor	kinzaKeyInst1	Active	nova	None	Running	1 hour, 12 minutes	Create Snapshot

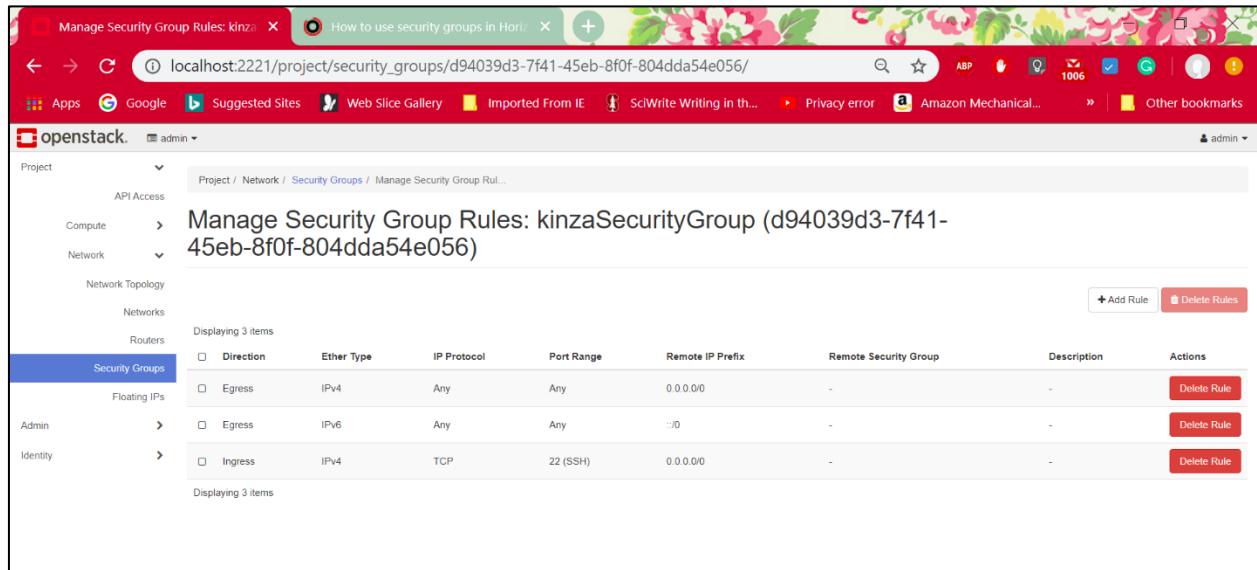
Managing the security group:

1. Clicked on Project>>Network>>Security Group and then clicked on Manage Rule button on the right of KinzaSecurityGroup.



The screenshot shows the OpenStack Dashboard under the 'Project' tab. In the 'Network' section, 'Security Groups' is selected. A table lists two security groups: 'default' and 'kinzaSecurityGroup'. The 'kinzaSecurityGroup' row has a 'Manage Rules' button highlighted with a red box. The URL in the browser is localhost:2221/project/security_groups.

2. After clicking on Manage Rule, the available rules are shown, **Egress** rules control the outgoing traffic which means any IPV4 and IPv6 address can send information to outside world. The current **Ingress** rule is only IPV4 traffic is allowed through making ssh connection on port 22 and remote IP prefix can be any. The task requires mangning security group by allowing ping test from all instance. For this purpose, new rule for ping will be added. Clicked on **Add Rule**.



The screenshot shows the 'Manage Security Group Rules' page for 'kinzaSecurityGroup'. It displays three existing rules: one Egress rule (IPV4 Any Any 0.0.0.0/0) and two Ingress rules (IPV4 Any Any ::/0 and TCP 22 (SSH) 0.0.0.0/0). The 'Ingress' rule for TCP port 22 is highlighted with a red box. The URL in the browser is localhost:2221/project/security_groups/d94039d3-7f41-45eb-8f0f-804dda54e056/.

3. After clicking the Add rule, the following screen shows. Rule is **All ICMP** which **Internet Control Message Protocol** which is an error reporting protocol sends an error to IP address when network does not send packets. So it means when we send request to an IP address and network could not send the packets to the IP it gives an error. Direction is **Ingress** which means rule is applied to incoming traffic only. **CIDR** is IP address block which allows to define only IP address which are allowed to ping the instances. Anyone can ping the instance. Then clicked on **Add**

4. After clicking the add, the following screen shows the added ICMP rule.

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	IPv6	Any	Any	::/0	-	-	<button>Delete Rule</button>
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	<button>Delete Rule</button>

Option2:

There is one more option to set the the ICMP rule to specific security group then only instances which are related to that security group are allowed to ping the instances. The following screen shows that

Rule is Custom ICMP Rule

Remote: Security group

Security Group: KinzaSecurityGroup

In this case, other instance of KinzaSecurityGroup are only allowed to ping.

The screenshot shows the OpenStack Network interface. On the left, a sidebar lists 'Project', 'API Access', 'Compute', 'Network' (selected), 'Network Topology', 'Networks', 'Routers', 'Security Groups' (selected), 'Floating IPs', 'Admin', and 'Identity'. In the center, the 'Manage Security Groups' page is displayed for a group named '45eb-8f0f-804d'. A modal window titled 'Add Rule' is open, showing the configuration for a 'Custom ICMP Rule'. The 'Description' field contains 'Custom ICMP Rule'. The 'Direction' dropdown is set to 'Ingress'. The 'Type' dropdown is set to '-1'. The 'Code' dropdown is set to '-1'. The 'Remote' dropdown is set to 'Security Group', which is expanded to show 'kinzaSecurityGroup (current)'. The 'Ether Type' dropdown is set to 'IPv4'. The right side of the screen shows a table of existing rules:

Group	Description	Actions
-	-	Delete Rule
-	-	Delete Rule
-	-	Delete Rule

The following screen shows Remote security group selected which is allowed to ping.

The screenshot shows the OpenStack Network interface. The sidebar and navigation bar are identical to the previous screenshot. The central area displays the 'Manage Security Group Rules' page for the 'kinzaSecurityGroup' (d94039d3-7f41-45eb-8f0f-804dda54e056). The table lists four rules:

Group	Description	Actions
kinzaSecurityGroup	-	Delete Rule
-	-	Delete Rule
-	-	Delete Rule

Limitation:

1. Default security group can not be deleted⁴ as can be seen in the following screen the error message on deleting the default security group.

The screenshot shows the OpenStack Dashboard at localhost:2221/project/security_groups. The left sidebar is collapsed. The main content area shows the 'Security Groups' page. A table lists two security groups: 'default' and 'kinzaSecurityGroup'. The 'default' row has a red border. An error message box is overlaid on the right side of the table, stating: 'Error: You are not allowed to delete security group: default'. The 'Actions' column for the 'default' row contains a 'Delete Security Group' button, which is also highlighted with a red box.

2. By default it only allows ssh connection to instances.
3. Other security group can not be deleted if they are attached to an instance.

Task No 6:

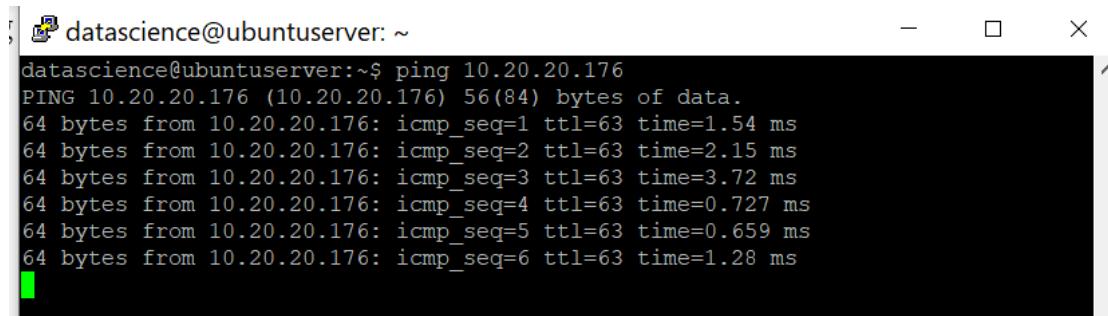
Ping Test for instances.

1. Before adding the ICMP rule, Pinging the kinzaInst1 which has floating IP: 10.20.20.14. It is not working because only ssh connection is allowed. Now after adding ICMP rule, pinging instance floating ip address is working.

```
data@data:~$ ping 10.20.20.125
PING 10.20.20.125 (10.20.20.125) 56(84) bytes of data.
64 bytes from 10.20.20.125: icmp_seq=1 ttl=63 time=500 ms
64 bytes from 10.20.20.125: icmp_seq=2 ttl=63 time=60.5 ms
64 bytes from 10.20.20.125: icmp_seq=3 ttl=63 time=16.3 ms
64 bytes from 10.20.20.125: icmp_seq=4 ttl=63 time=2.47 ms
64 bytes from 10.20.20.125: icmp_seq=5 ttl=63 time=2.11 ms
```

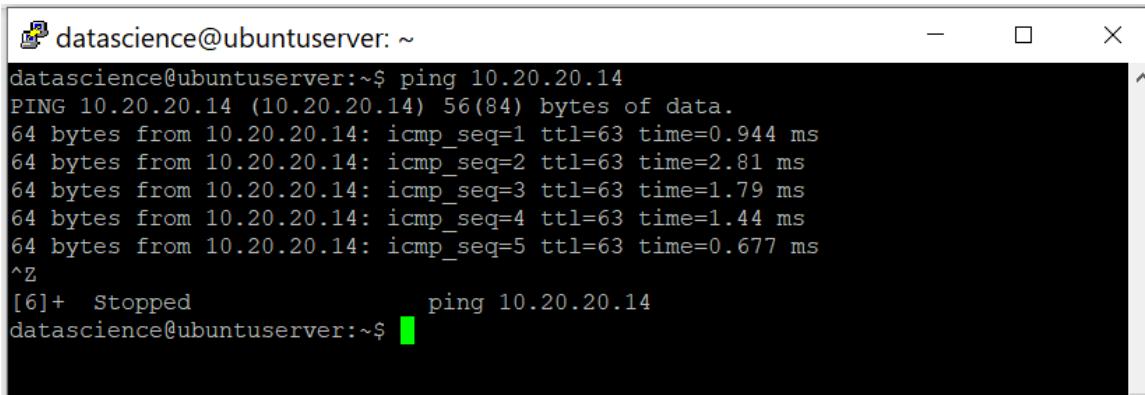
⁴ https://docs.openstack.org/mitaka/user-guide/cli_nova_configure_access_security_for_instances.html

2. Pinging second instance with Floating IP



```
data@ubuntuserver:~$ ping 10.20.20.176
PING 10.20.20.176 (10.20.20.176) 56(84) bytes of data.
64 bytes from 10.20.20.176: icmp_seq=1 ttl=63 time=1.54 ms
64 bytes from 10.20.20.176: icmp_seq=2 ttl=63 time=2.15 ms
64 bytes from 10.20.20.176: icmp_seq=3 ttl=63 time=3.72 ms
64 bytes from 10.20.20.176: icmp_seq=4 ttl=63 time=0.727 ms
64 bytes from 10.20.20.176: icmp_seq=5 ttl=63 time=0.659 ms
64 bytes from 10.20.20.176: icmp_seq=6 ttl=63 time=1.28 ms
```

3. Pinging third instance

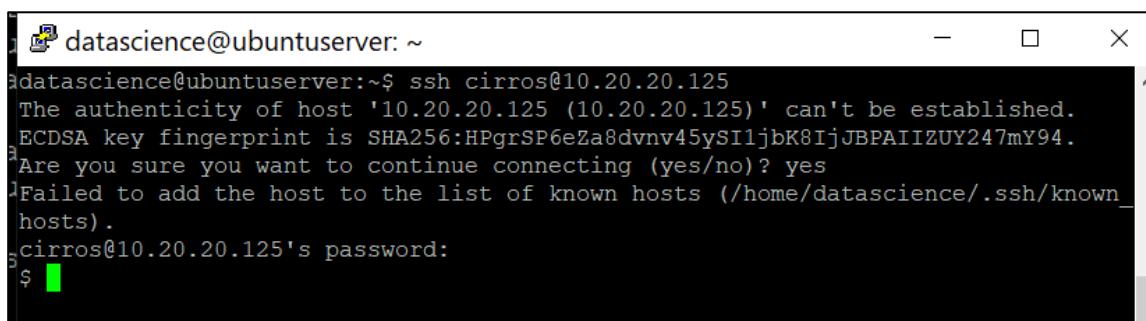


```
data@ubuntuserver:~$ ping 10.20.20.14
PING 10.20.20.14 (10.20.20.14) 56(84) bytes of data.
64 bytes from 10.20.20.14: icmp_seq=1 ttl=63 time=0.944 ms
64 bytes from 10.20.20.14: icmp_seq=2 ttl=63 time=2.81 ms
64 bytes from 10.20.20.14: icmp_seq=3 ttl=63 time=1.79 ms
64 bytes from 10.20.20.14: icmp_seq=4 ttl=63 time=1.44 ms
64 bytes from 10.20.20.14: icmp_seq=5 ttl=63 time=0.677 ms
^Z
[6]+  Stopped                  ping 10.20.20.14
data@ubuntuserver:~$
```

Task No 7:

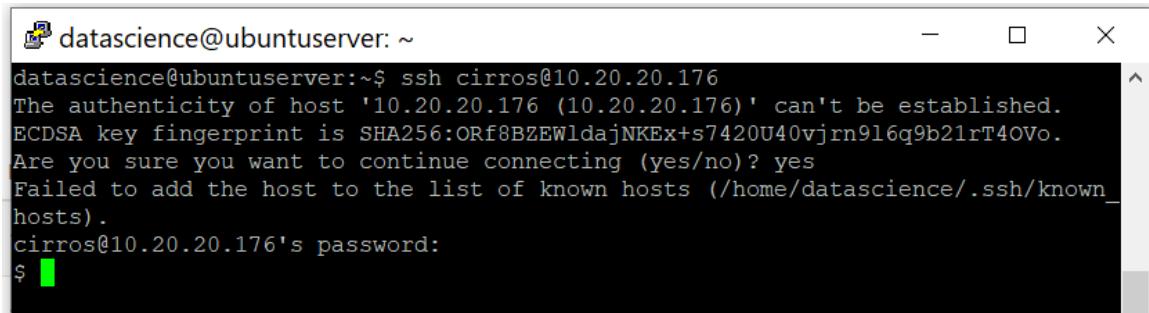
Connecting to instance via ssh

1. The following screen shows connection made via ssh to instance **kinzaInst1** as allowed in the security group.
ssh cirros@10.20.20.125 command: where cirros is the instance name and 10.20.20.125 is the floating IP.



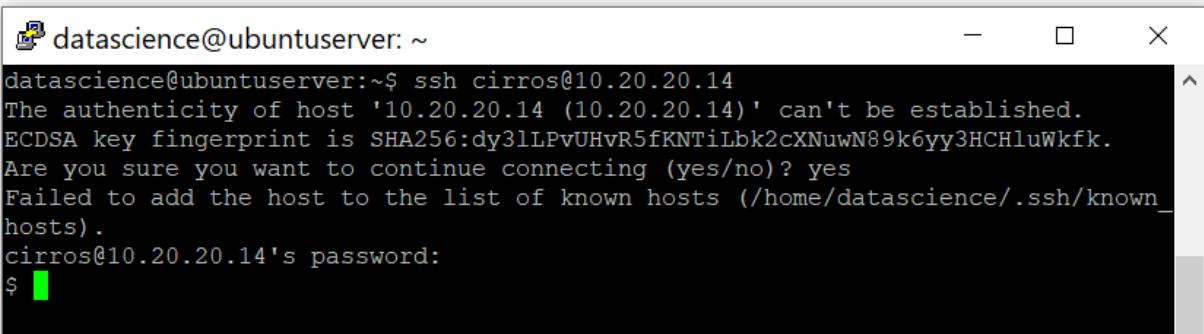
```
data@ubuntuserver:~$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.125's password:
$
```

2. The following screen shows ssh connection to kinzaInst2.



```
data@data:~$ ssh cirros@10.20.20.176
The authenticity of host '10.20.20.176 (10.20.20.176)' can't be established.
ECDSA key fingerprint is SHA256:ORf8BZEWldajNKEx+s7420U40vjr916q9b21rT40Vo.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.176's password:
$
```

3. The following screen shows ssh connection with third instance kinzaInst3.



```
data@data:~$ ssh cirros@10.20.20.14
The authenticity of host '10.20.20.14 (10.20.20.14)' can't be established.
ECDSA key fingerprint is SHA256:dy3lLPvUHvR5fKNTiLbk2cXNuwN89k6yy3HChluWkf.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.14's password:
$
```

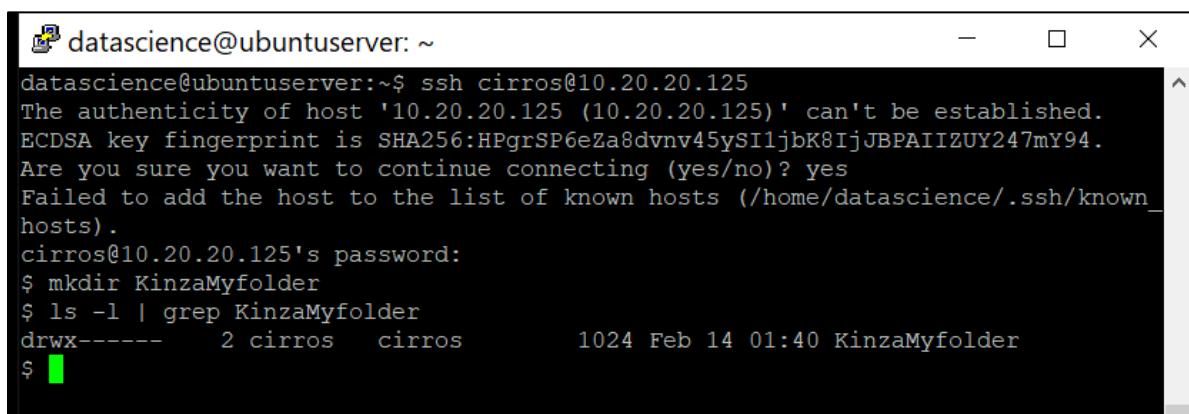
Task No. 7:

Creating a Folder and checking the permissions:

\$**mkdir KinzaMyfolder** creates the directory with the given name.

\$**ls -l | grep KinzaMyfolder** : ls -l shows the list long of directories (KinzaMyfolder) and file size(1024) and date and time modified (feb 14 01:40). | **grep KinzaMyfolder** makes sure only KinzaMyfolder details and permissions should be displayed.

drwx----- 2 cirros cirros : d is to represent the directory permissions. Only sudo/super user(**cirros**) has authority to read, write and execute the file where as **group(cirros)** and everybody else don't have any access to the file.



```
data@data:~$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.125's password:
$ mkdir KinzaMyfolder
$ ls -l | grep KinzaMyfolder
drwx----- 2 cirros cirros 1024 Feb 14 01:40 KinzaMyfolder
$
```

Task No. 9:

Creating python file in created folder

The first command is **touch**. The following explains the purpose of the touch command.

1. It is good check to know whether the user has permission to create the file or its read only permission allowed.
2. It also informs if one user is accessing the file and other user access the same file, it will give warning that first user is also reading the file and may update it.
3. It updates the timestamp of the file which means if the file already exists, then it will update timestamp to current date and time. If the file doesn't exist, it will create new file and give the current date and time as timestamp.

Vi command: Vi is text editor to write and update the file contents.

Steps:

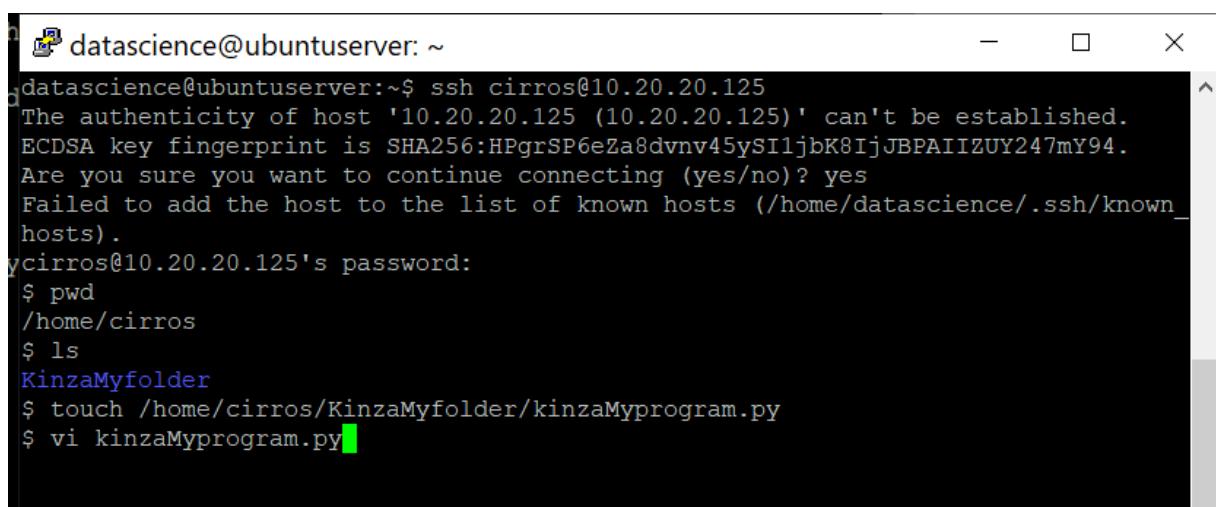
1. The following screen shows commands to create a python file

pwd: shows the current directory which is /home/cirros

ls : shows the list of folder/files

touch /home/cirros/KinzaMyfolder/kinzaMyprogram.py: is written to create a python file in the KinzaMyfolder. Path is given to create a file in specific folder.

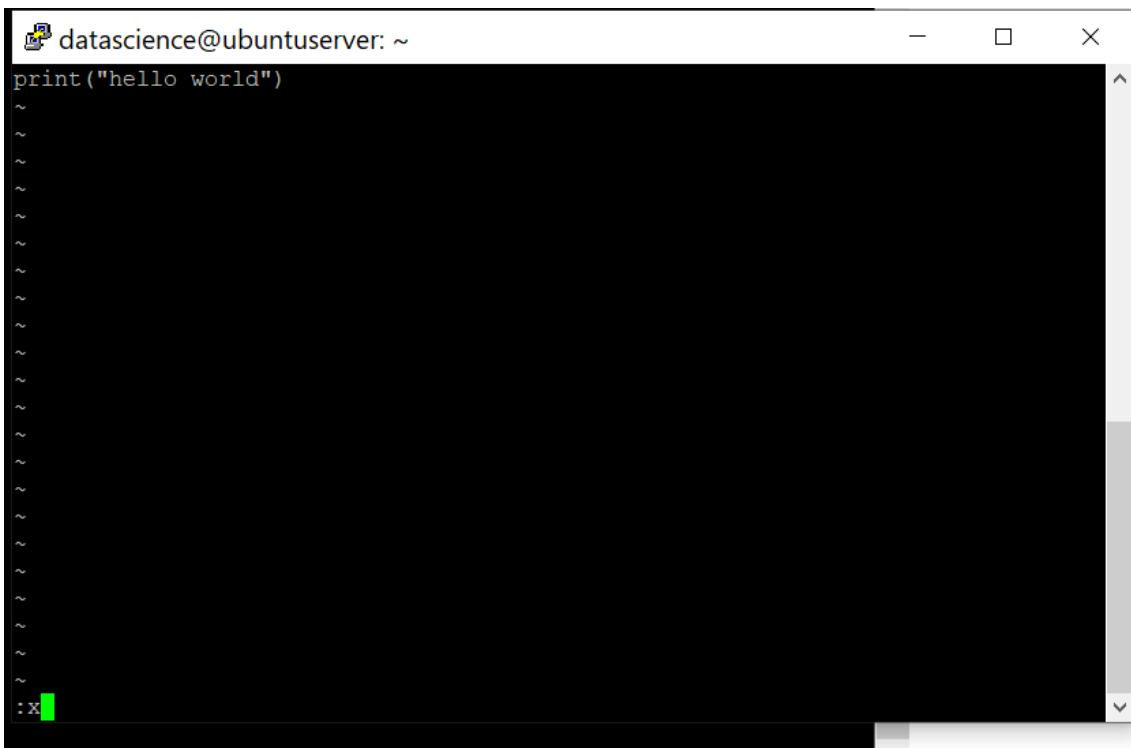
Vi kinzaMyprogram.py: in this command, vi is used to open editor which is powerful and complicated text editor than nano.



A screenshot of a terminal window titled "datascience@ubuntuserver: ~". The terminal shows the following session:

```
datascience@ubuntuserver:~$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data-science/.ssh/known_hosts).
cirros@10.20.20.125's password:
$ pwd
/home/cirros
$ ls
KinzaMyfolder
$ touch /home/cirros/KinzaMyfolder/kinzaMyprogram.py
$ vi kinzaMyprogram.py
```

2. After entering, vi text editor opens, Press **i** from the keyboard to get cursor and writing . After writing the desired text, to save the file Press **ESC key** from the keyboard. It will save the file and type **:x** and Enter. The editor will be closed.



A screenshot of a terminal window titled "dataScience@ubuntuserver: ~". The window contains a single line of Python code: "print("hello world")". The terminal is black with white text, and there is a vertical scroll bar on the right side.

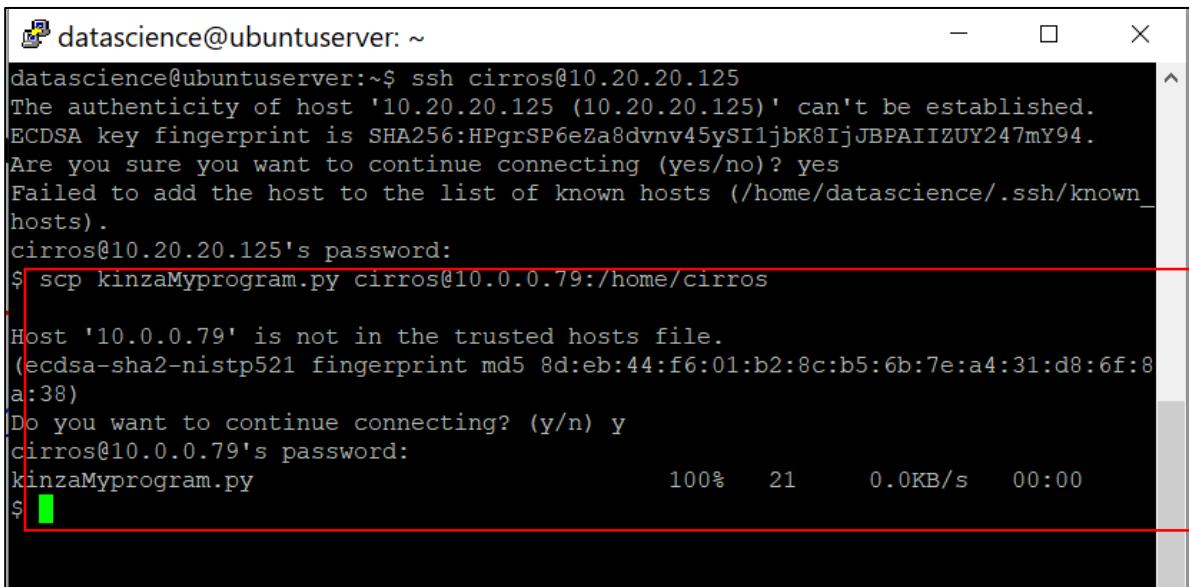
Task No. 10:

Copying file from one instance to other

SCP (Secure Copy) is a command which transfers file or a directory between two remote hosts. It works same as SSH(secure shell) because it runs ssh in the background while transferring the file. SCP works as both protocol and program to describe how to transfer the file and command to transfer. This command also requires authentication such as password and passphrase before sending file to host machine. It performs encryption on the file and password shared over the network to avoid loss of message.⁵

1. \$scp kinzaMyprogram.py cirros@10.0.0.79 :/home/cirros :
- **SCP kinzaMyprogram.py:** is secure copy transfer command with file name next to it.
- **cirros@10.0.0.79:** Username and private IP address of the second instance kinzaInst2 and kinzaInst1 and kinzaInst2 both are on the same kinzaPrivnetA so private IP address has been used because file is transferred on the same network.
- **:/home/cirros** specifies the directory in the host where file should be stored.

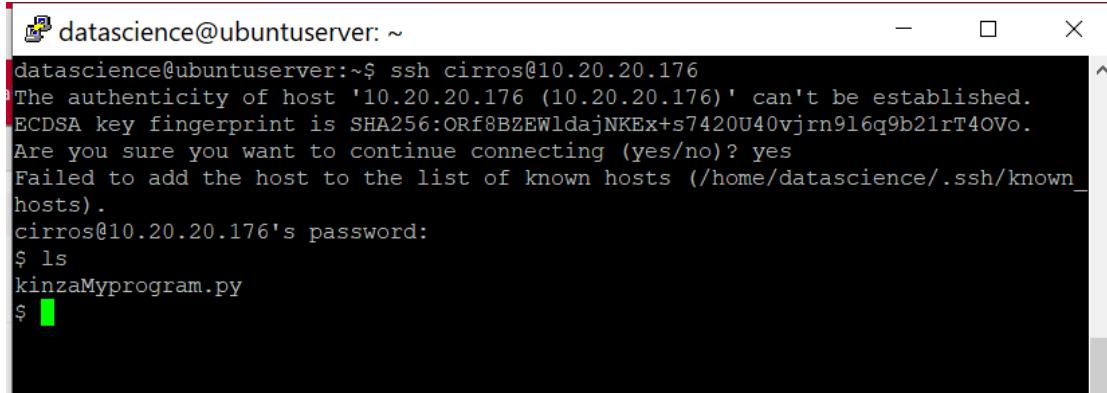
⁵ <https://imyuvii.com/posts/scp-command-transferring-files/>



```
data@ubuntuserver: ~
data$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.125's password:
$ scp kinzaMyprogram.py cirros@10.0.0.79:/home/cirros

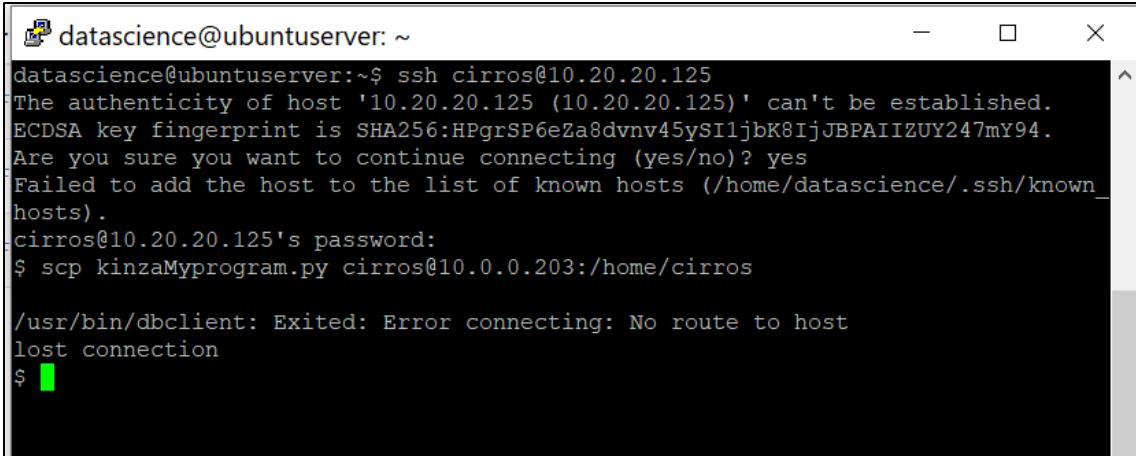
Host '10.0.0.79' is not in the trusted hosts file.
(ecdsa-sha2-nistp521 fingerprint md5 8d:eb:44:f6:01:b2:8c:b5:6b:7e:a4:31:d8:6f:8a:38)
Do you want to continue connecting? (y/n) y
cirros@10.0.0.79's password:
kinzaMyprogram.py                                100%   21      0.0KB/s   00:00
$
```

2. Checking the transferred file on the **kinzaInst2**. After making connection with the **kinzaInst2** using Floating ip, **ls** command has been used to check the files. It can be seen the **kinzaMyprogram.py** file has been transferred.



```
data@ubuntuserver: ~
data$ ssh cirros@10.20.20.176
The authenticity of host '10.20.20.176 (10.20.20.176)' can't be established.
ECDSA key fingerprint is SHA256:ORf8BZEWldajNKE+s7420U40vjrn916q9b21rT40Vo.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/data/.ssh/known_hosts).
cirros@10.20.20.176's password:
$ ls
kinzaMyprogram.py
$
```

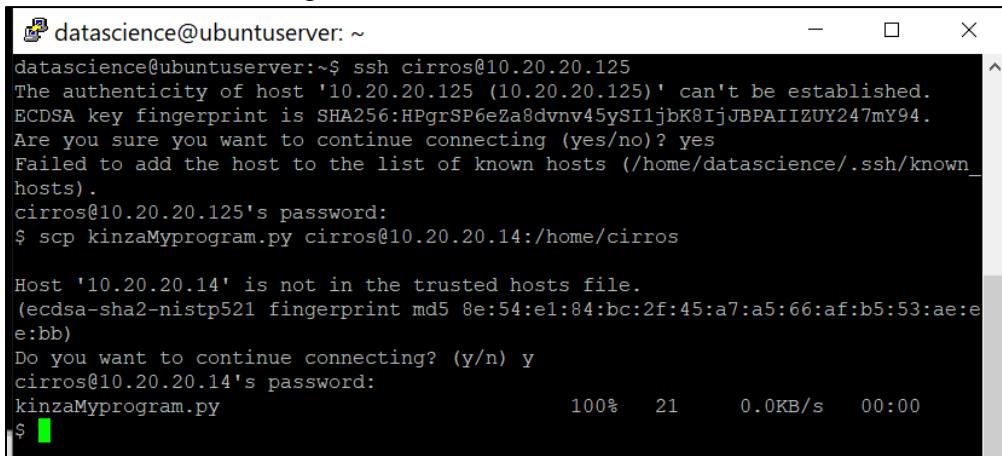
3. To transfer file to **kinzaInst3**, private ip address has been used to send file using SCP command in the following screen. It can be seen that transfer failed . Error in connecting which means no route has been found by the network because **kinzaInst3** does not exist on the network. So, using private IP address doesnot work.



```
datascience@ubuntuserver:~$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/datacience/.ssh/known_hosts).
cirros@10.20.20.125's password:
$ scp kinzaMyprogram.py cirros@10.0.0.203:/home/cirros

/usr/bin/dbclient: Exited: Error connecting: No route to host
lost connection
$
```

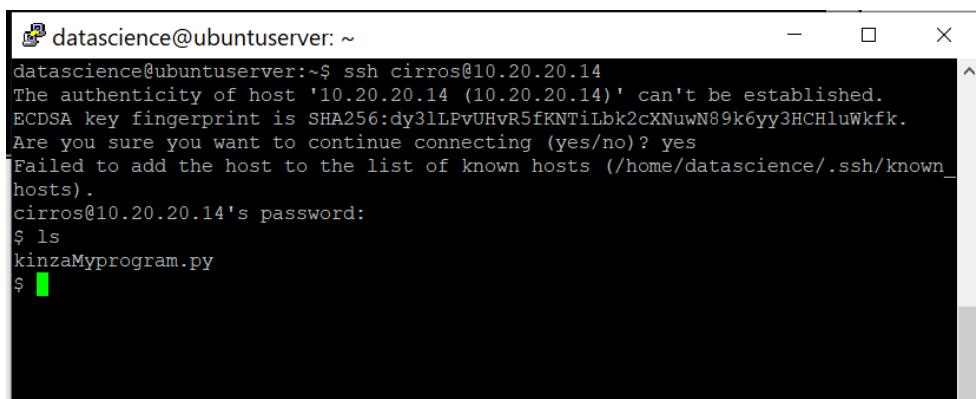
4. Now, transferring the file to kinzaInst3 using floating ip address because it exits on kinzaPrivNetB. The following screen shows transfer is successful.



```
datascience@ubuntuserver:~$ ssh cirros@10.20.20.125
The authenticity of host '10.20.20.125 (10.20.20.125)' can't be established.
ECDSA key fingerprint is SHA256:HPgrSP6eZa8dvnv45ySI1jbK8IjJBPAIIZUY247mY94.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/datacience/.ssh/known_hosts).
cirros@10.20.20.125's password:
$ scp kinzaMyprogram.py cirros@10.20.20.14:/home/cirros

Host '10.20.20.14' is not in the trusted hosts file.
(ecdsa-sha2-nistp521 fingerprint md5 8e:54:e1:84:bc:2f:45:a7:a5:66:af:b5:53:ae:e
e:bb)
Do you want to continue connecting? (y/n) y
cirros@10.20.20.14's password:
kinzaMyprogram.py                                100%   21      0.0KB/s  00:00
$
```

5. Now checking, whether the file is received at kinzaInst3, the following screen shows ls of file on the instance3.



```
datascience@ubuntuserver:~$ ssh cirros@10.20.20.14
The authenticity of host '10.20.20.14 (10.20.20.14)' can't be established.
ECDSA key fingerprint is SHA256:dy31LPvUHvR5fKNTiLbk2cXNuwN89k6yy3HChluWkfk.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/datacience/.ssh/known_hosts).
cirros@10.20.20.14's password:
$ ls
kinzaMyprogram.py
$
```

Limitation:

If two instances are within same private network, then file can be transferred using private IP address as for instance1 and instance2 in this case because both are on same private network.

However, if instances are on different network, then there is need to use floating IP address for transferring the file.