Cloud Computing-Open Stack

Task No. 1:

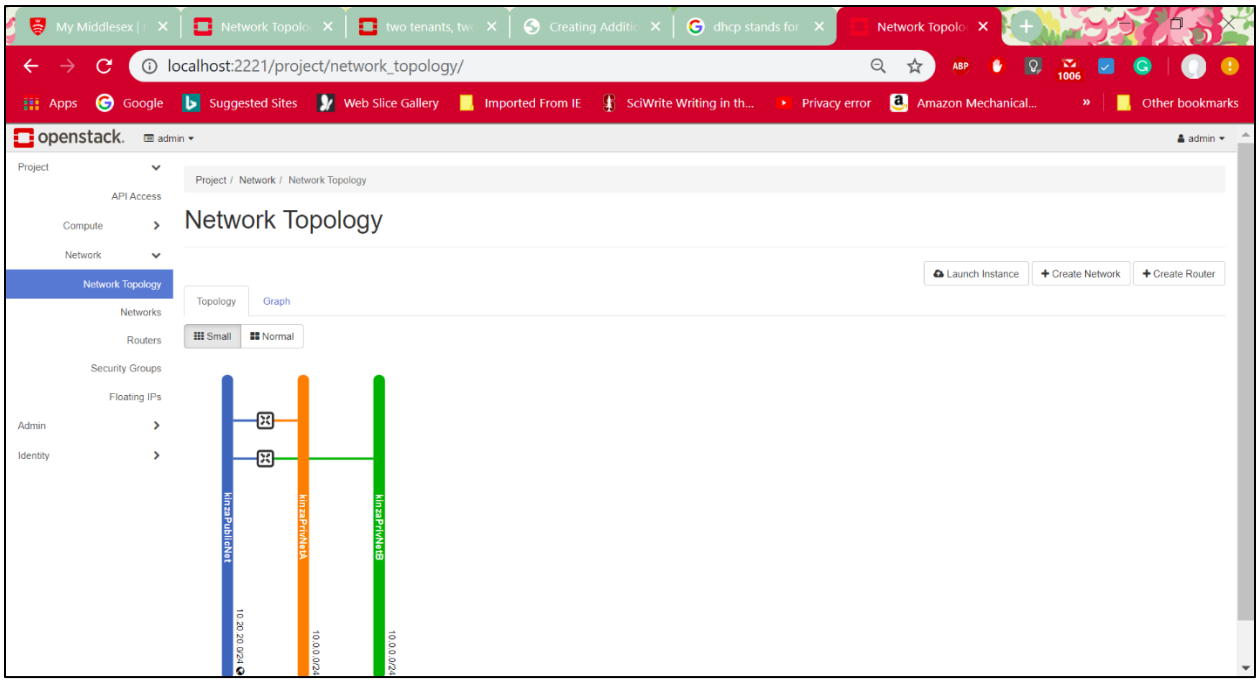
In this task, Creating two private networks and connecting them through routers to external/public network.

Steps:

Private Nework Creation:

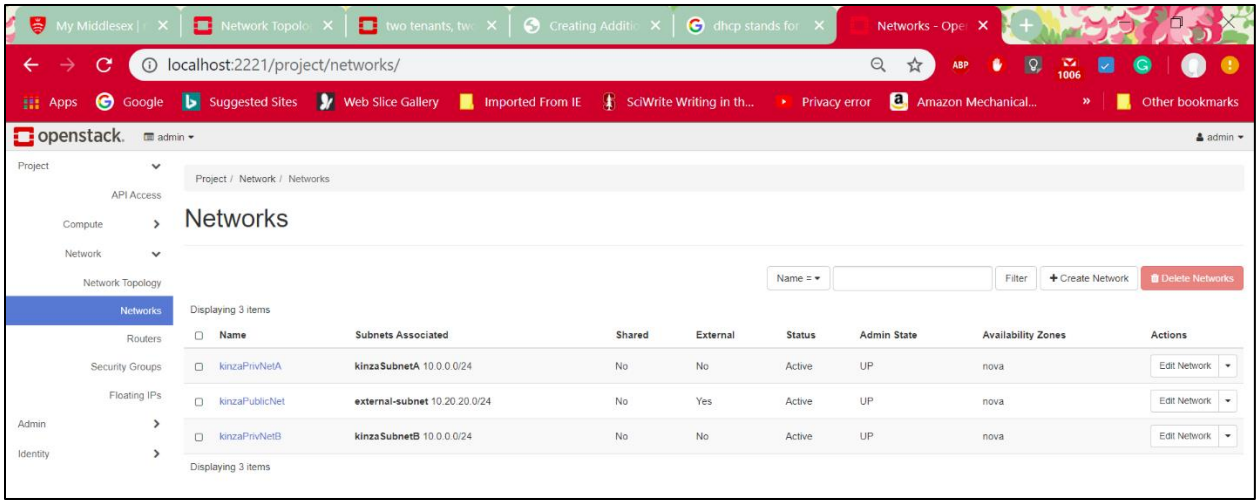
The following two options can be used to go to **Create Network** button.

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

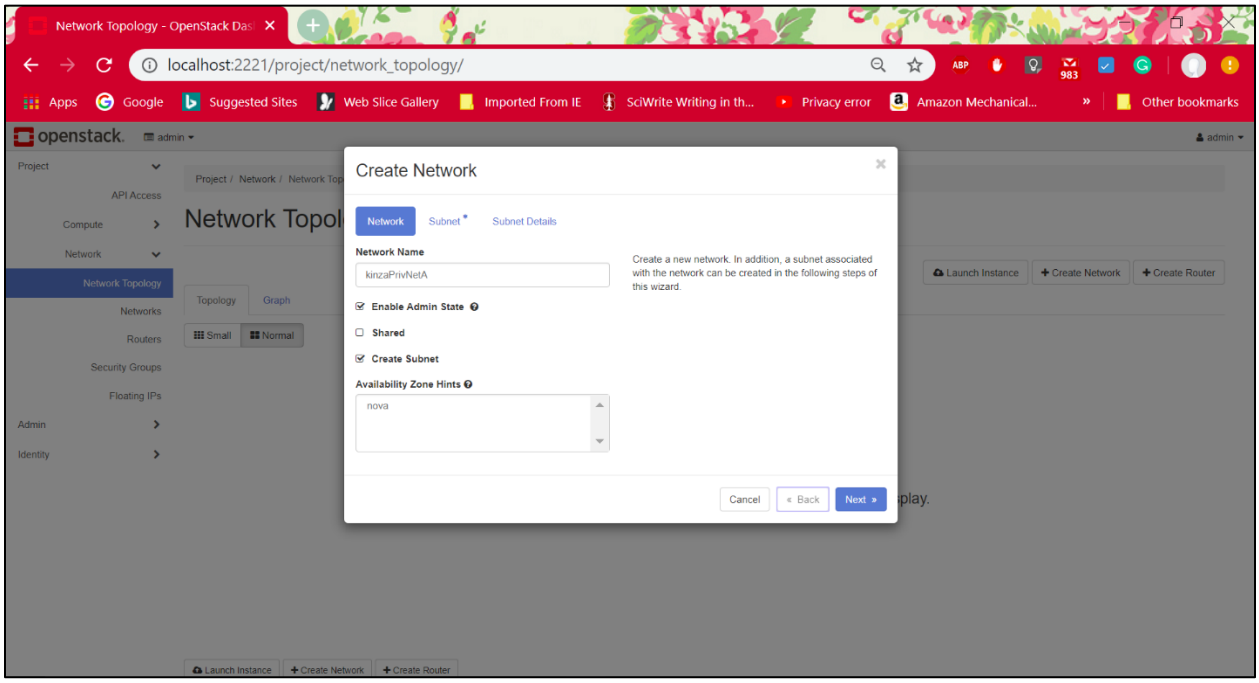


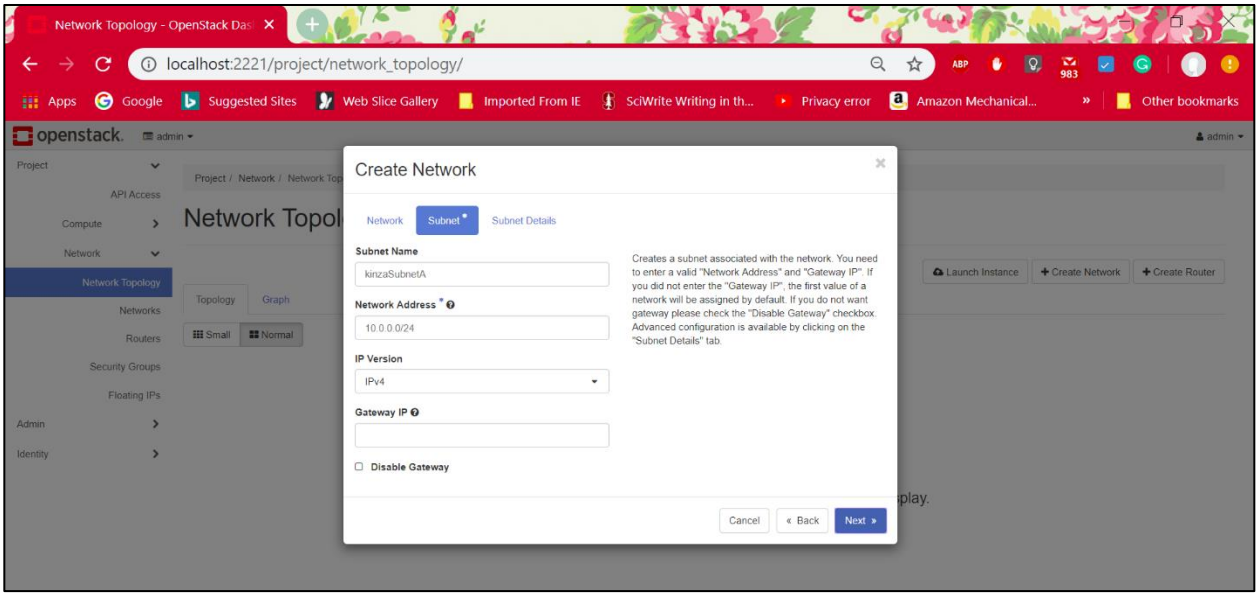
Option 2: Clicked on Project>>Network>> Networks

1. 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.

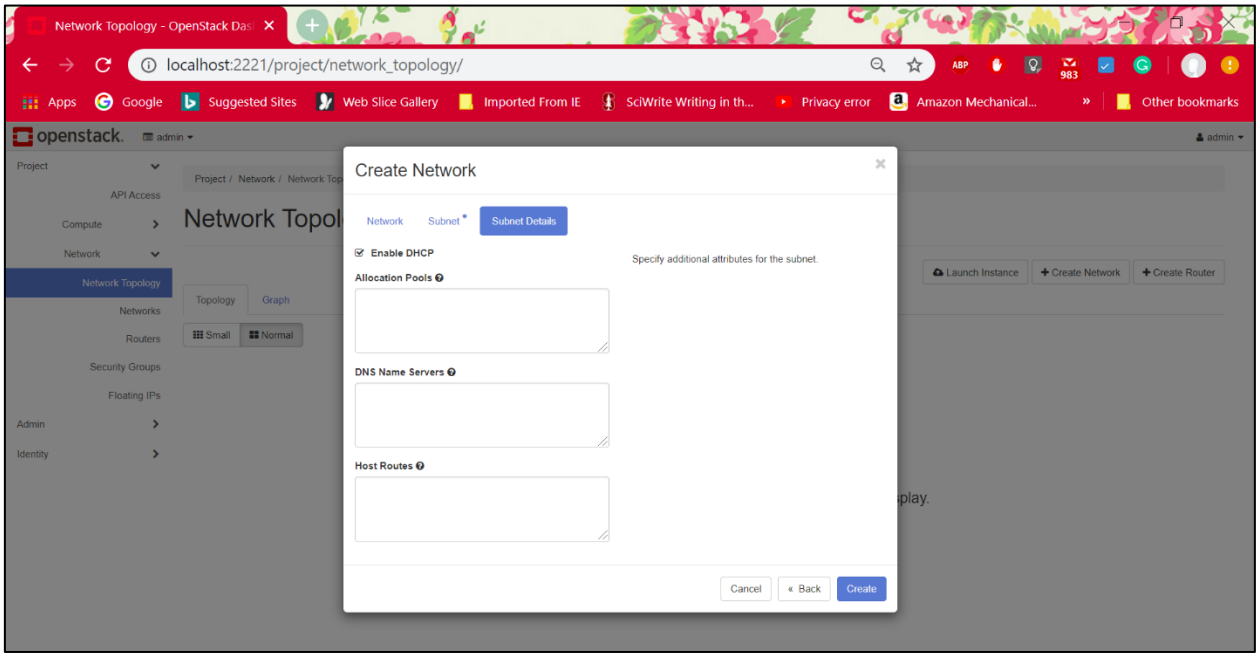


2. Clicked on **Next** button and added subnet details. **Network address: 10.0.0.0/24** which will assign IP addresses randomly from range 10.0.0.1 to 10.0.0.254. Creating the subnet is important because without subnet it is not possible to attach instances and routers with the network.



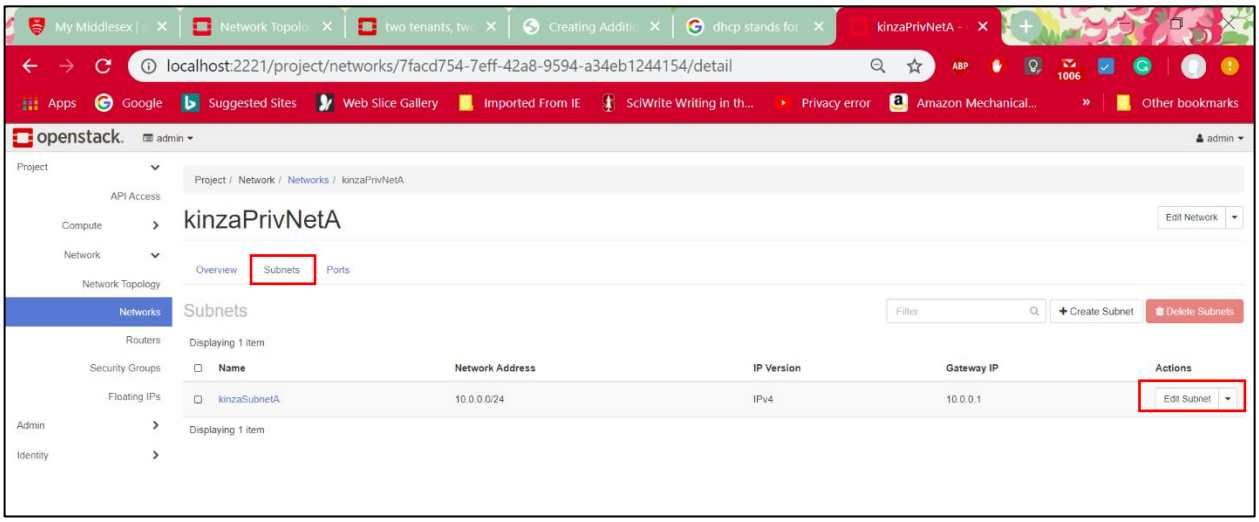


3. On clicking Next in the previous, the following screen can be seen . By default DHCP is enabled which will allocate IP addresses to virtual machine dynamically.
   Allocation pools is optional: The IP addresses specified in this pool are used to assign to the subnets. It makes sure that there is no overlapping assignement. 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 projects[1].



[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**.



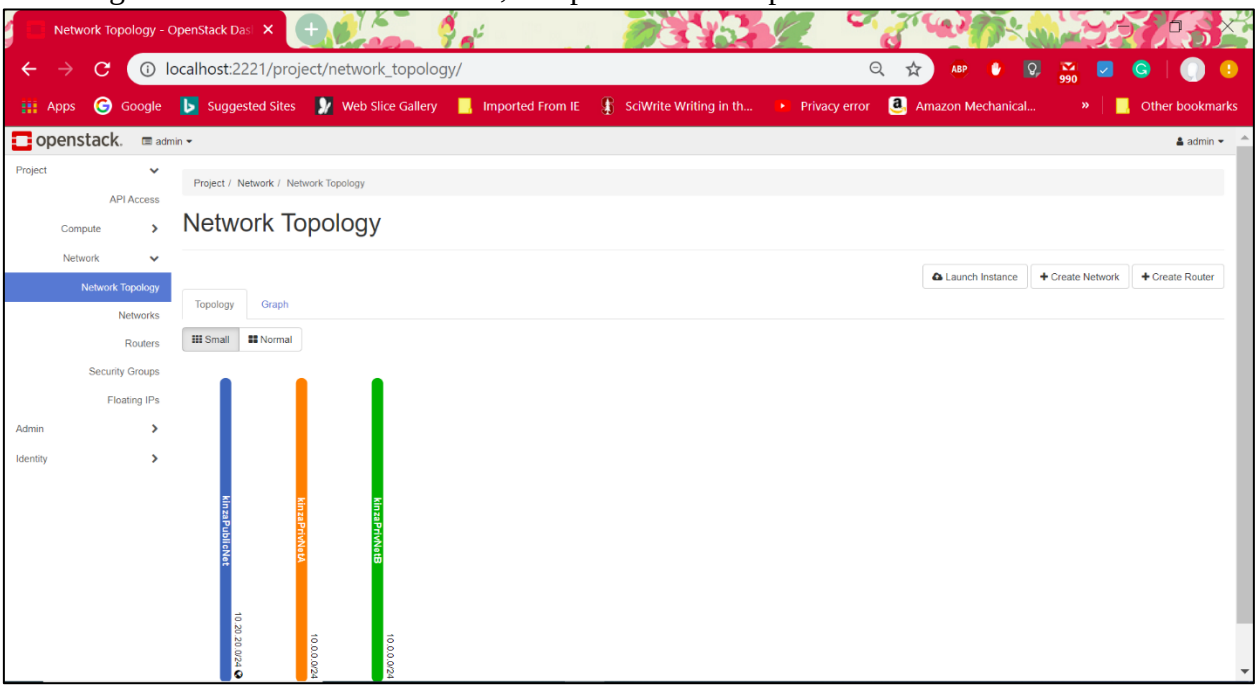
5. Repeated the same steps for creating second private network with the following details

   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.

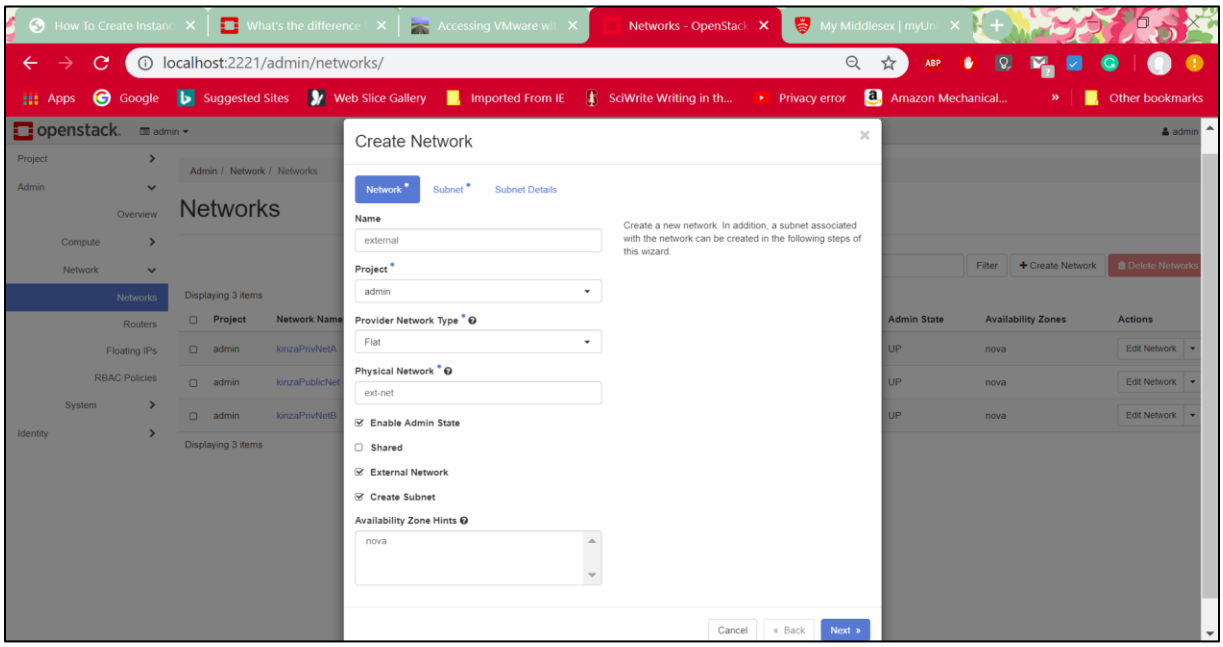


Public Network :

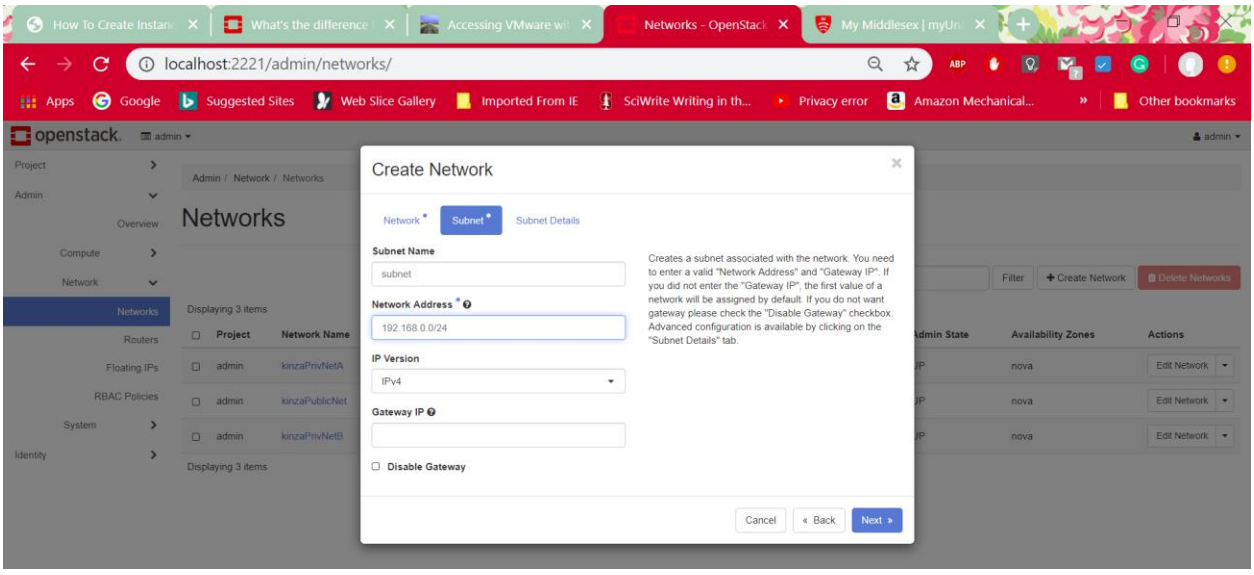
Option 1(Creating Public Network):

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



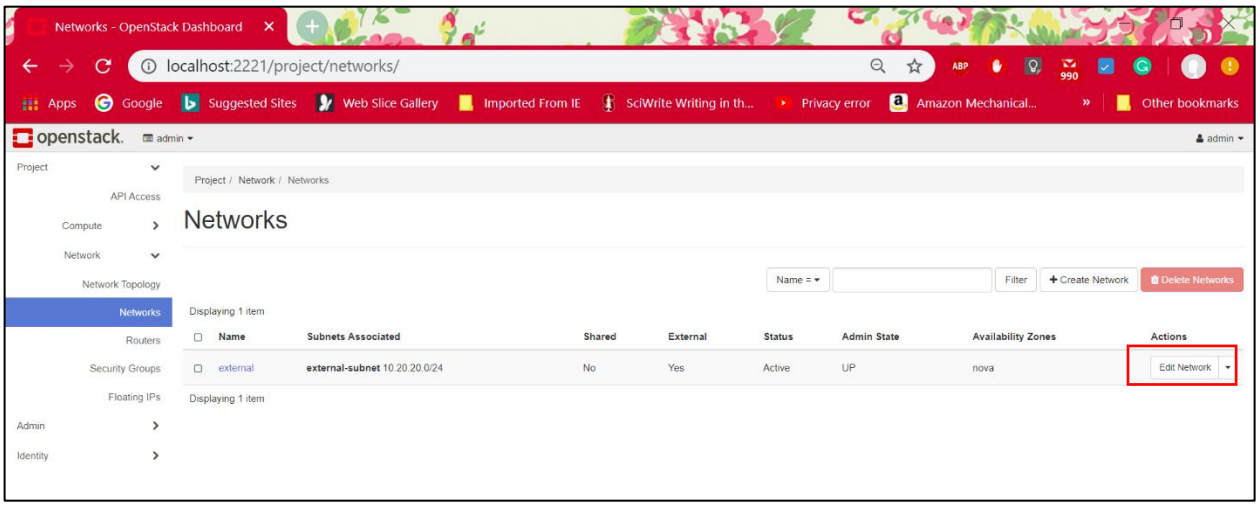
2. The following screen shows adding subnet details which will be used to assign floating IP's to instances while associating.



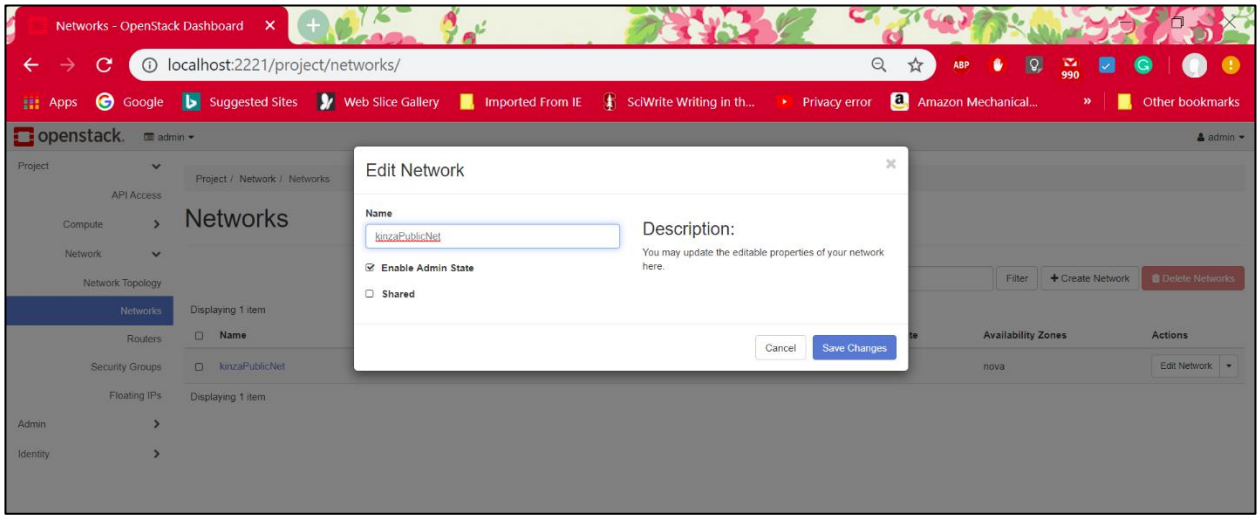
Option 2(Default External Network):

Clicked on **Project>>Network>>Network Topology**

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



2. Renamed to ***kinzaPublicNet.***

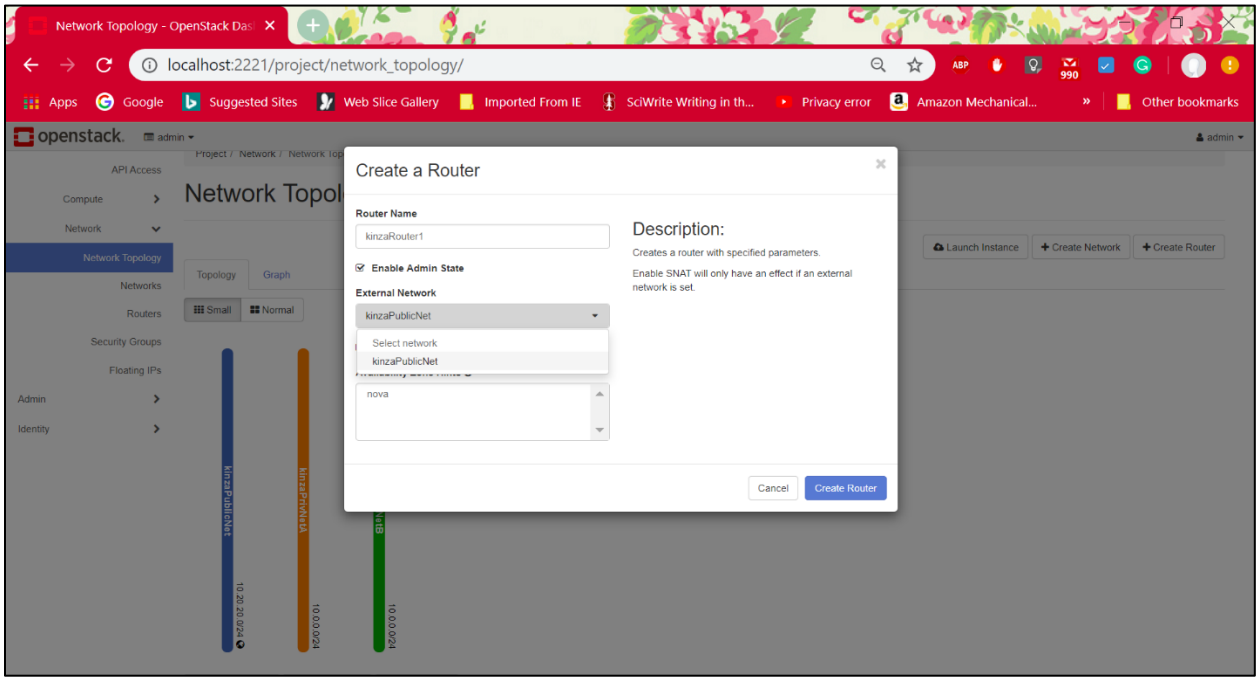


Routers Creation:

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 network[2].

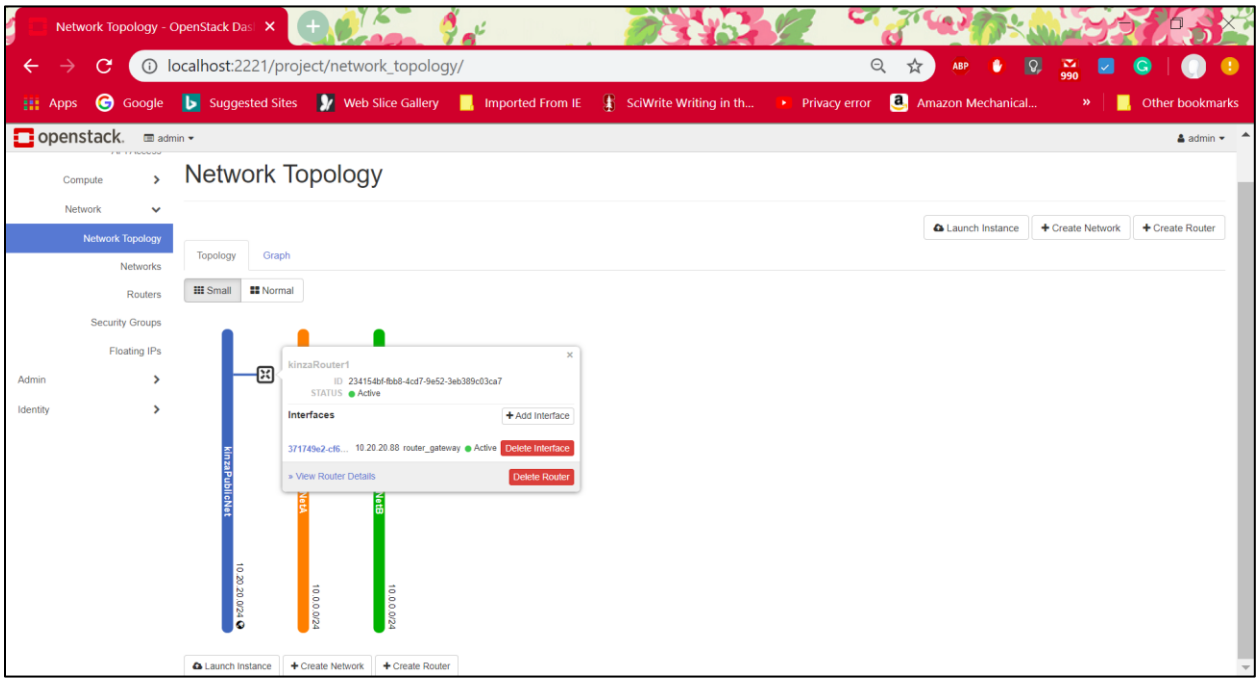
For that purpose creating the Routers in the following steps.

i) Clicke on the following **Project>Network> Network Topology** and clicked on **Create Router** button. Gave Router Name: kinzaRouter1and enabled admin state and assigned public network to which there is need to connect the private networks.

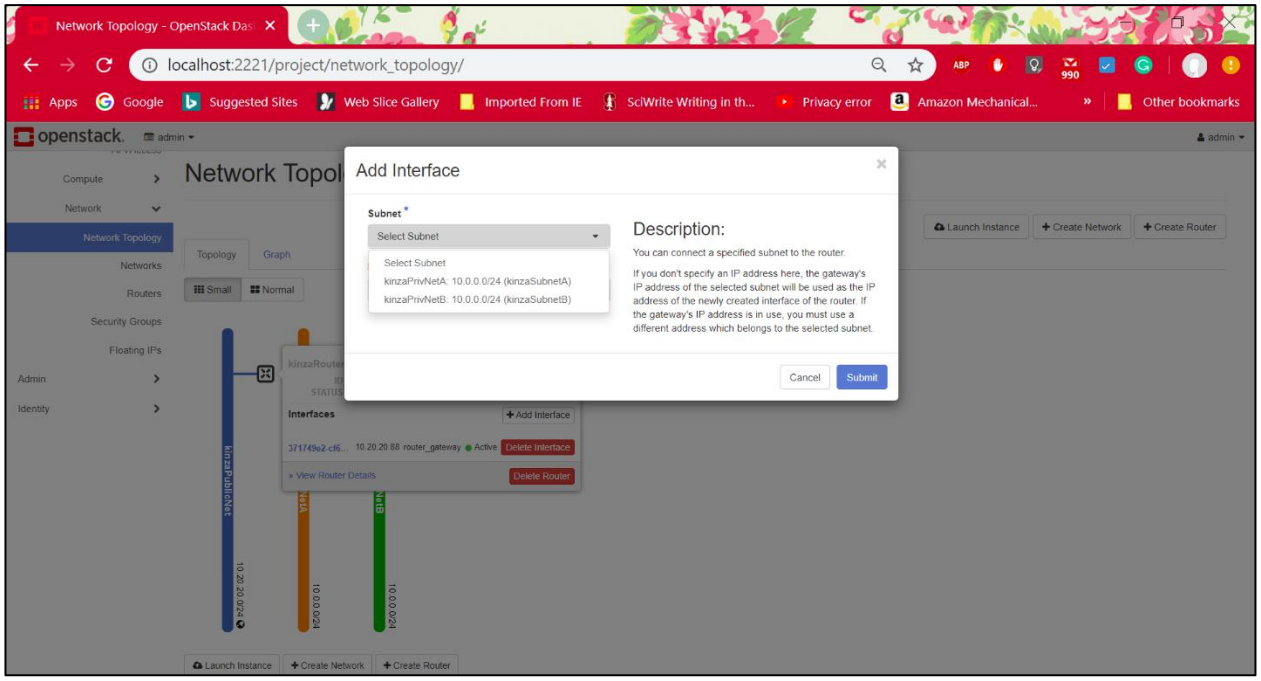


[2] https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/router.html

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

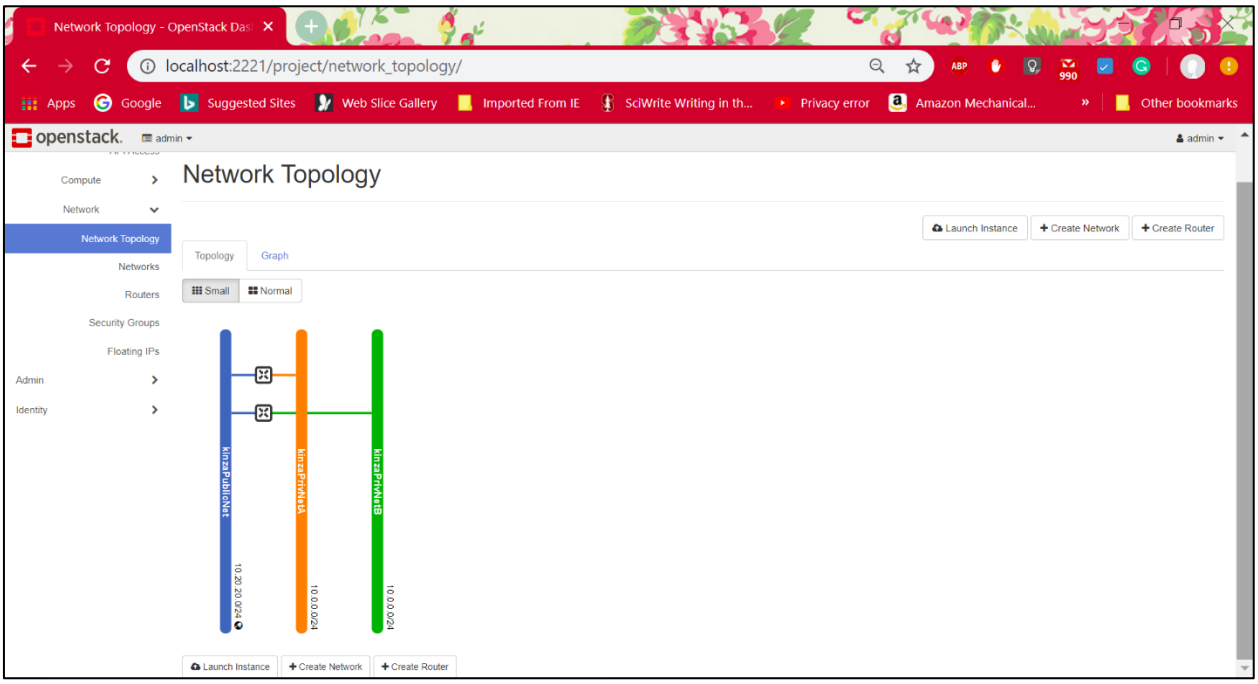


iii) After clicking on Add Interface, It shows list of private networks exits to add to interface. Selecting the **kinzaPrivNetA** for interface will add this network to the router and connect to the public network. Clicked on **Submit** button



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 privte 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:

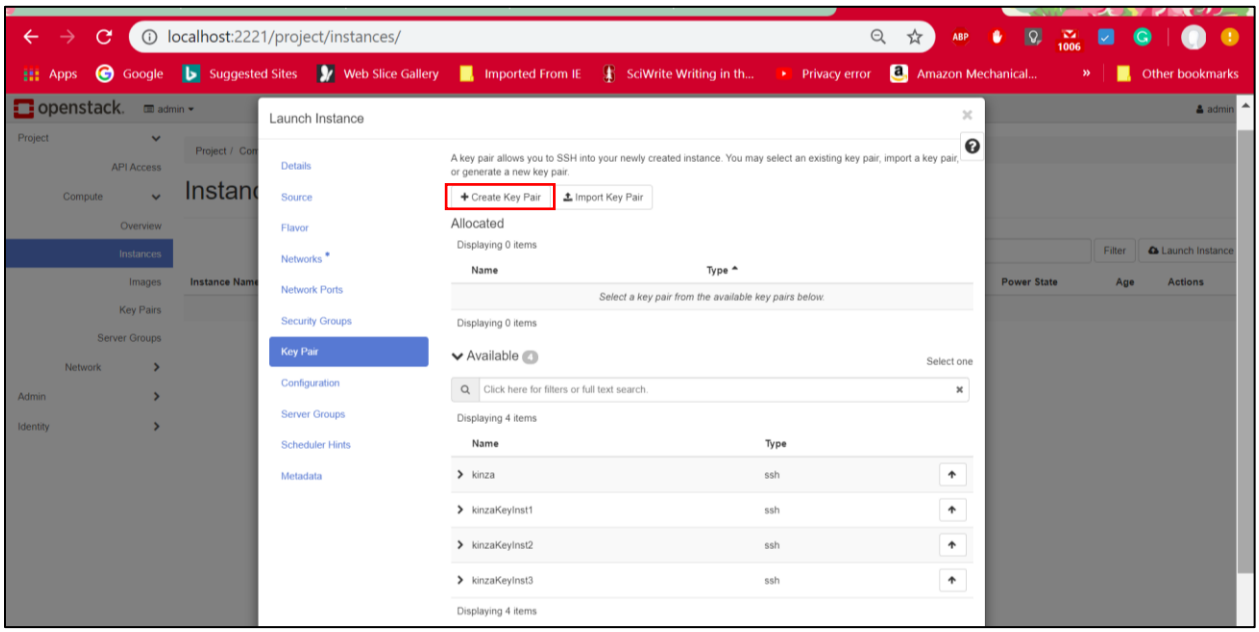
Creating three different keypairs which will be assigned to instance while launching them. This task has been performed before Task 2 and Task 3 because creating key pairs is best to do and then can assign key pair while creating instances.

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 canbe used for many instance within project.

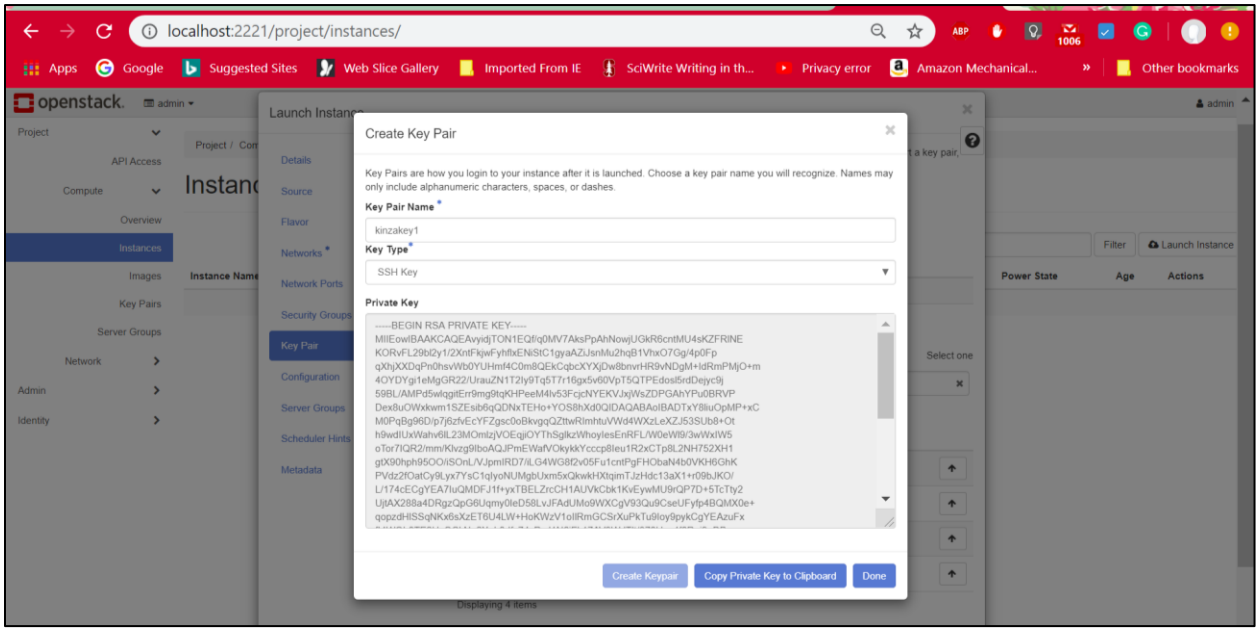
The ways to create keypairs:

Option1:

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



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



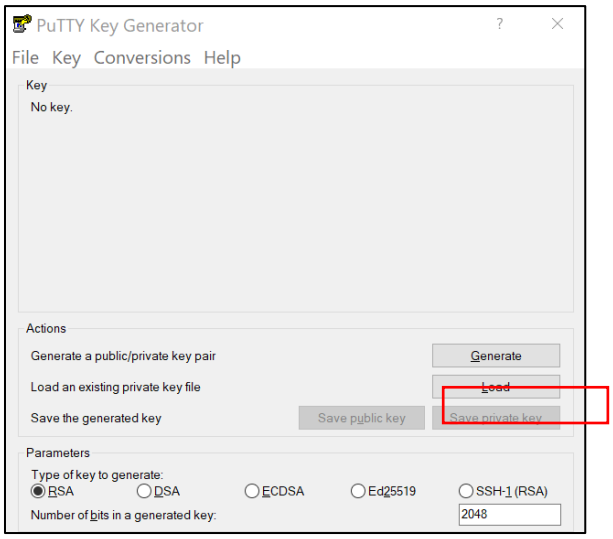
Limitations: It doesn't automatically download the keypair file. The user has to copy the key in some new file and save it manually.

Option 2:

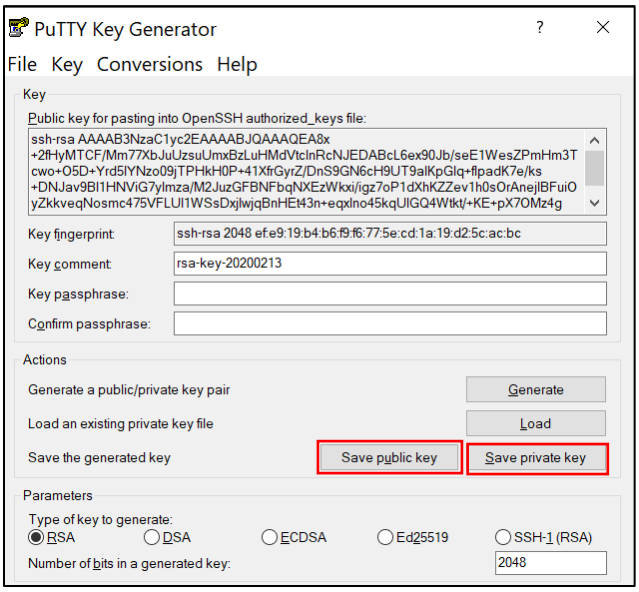
Importing the key:

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.

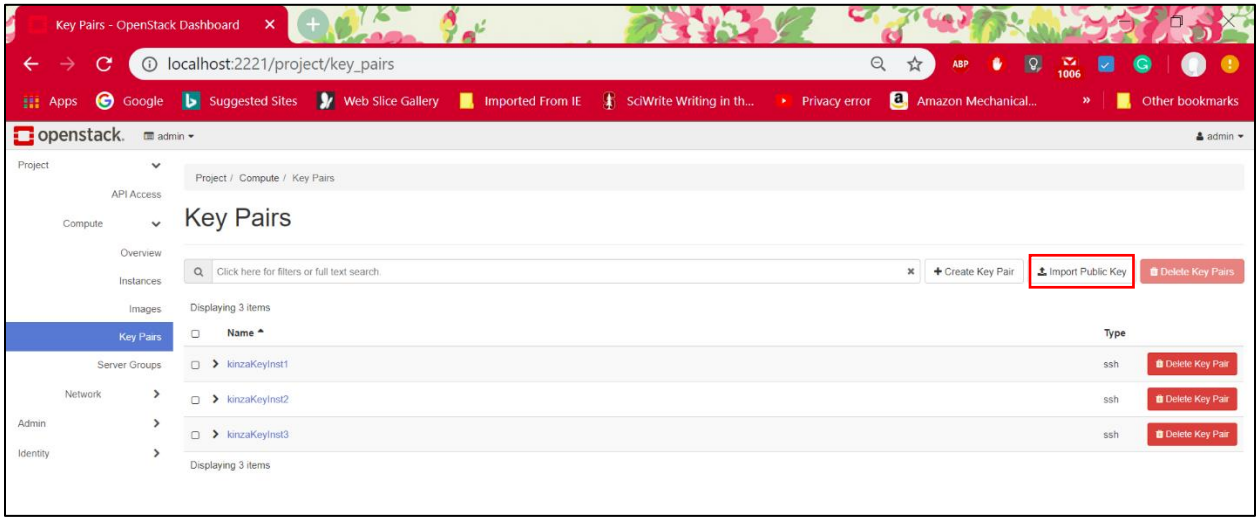
1. Clicked on **Generate** button



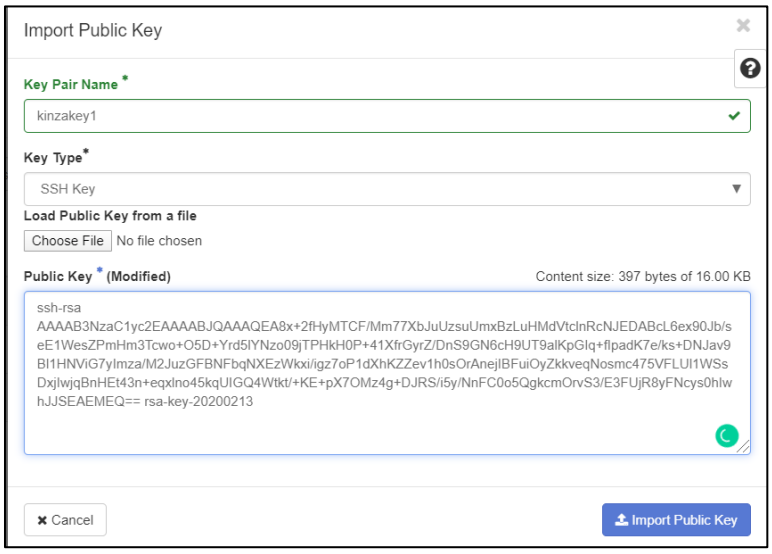
2. The following screen shows the generated public key. Can copy this key in the open stack or either save it clicking on **Save Public key** and **Save Private Key** button. Then, public key can be imported into openstack whereas private key can be kept safe.



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



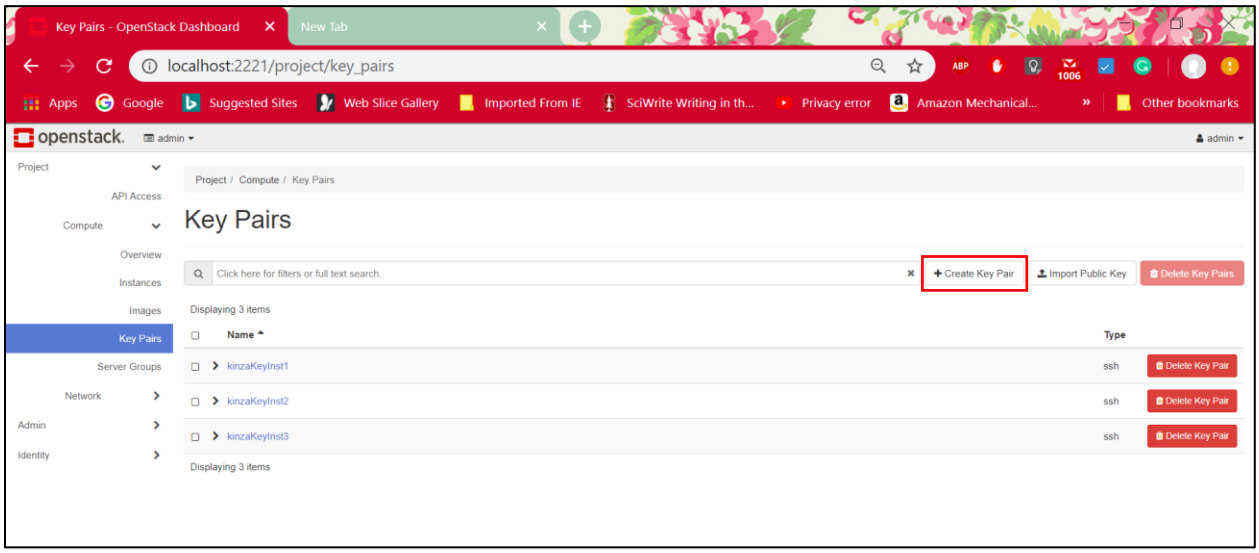
4. After clicking import public key, the following screen shows copied public key generated from PuttYgen software.



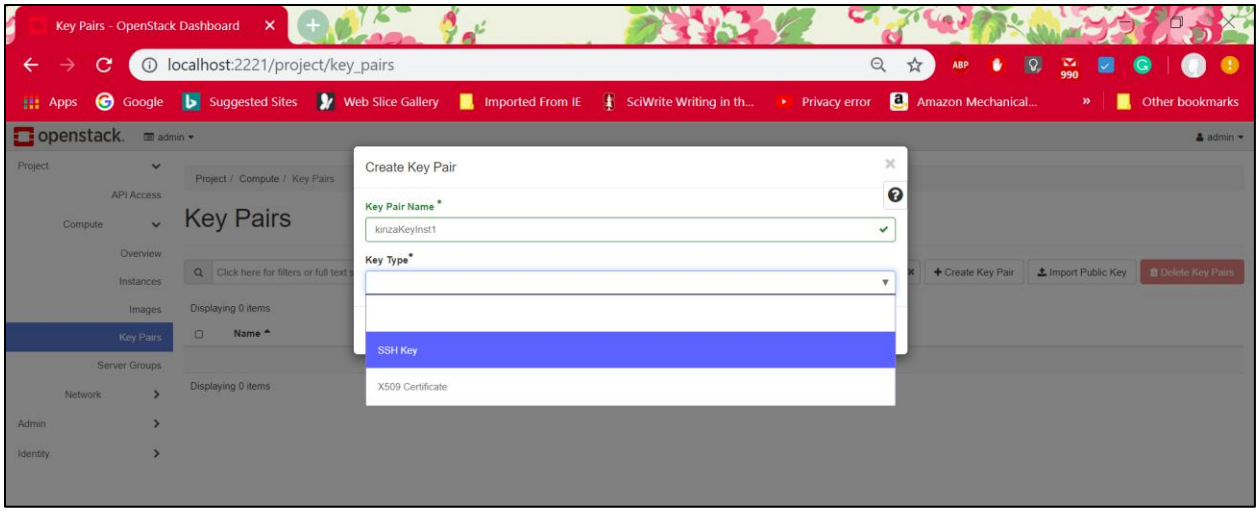
Option 3:

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

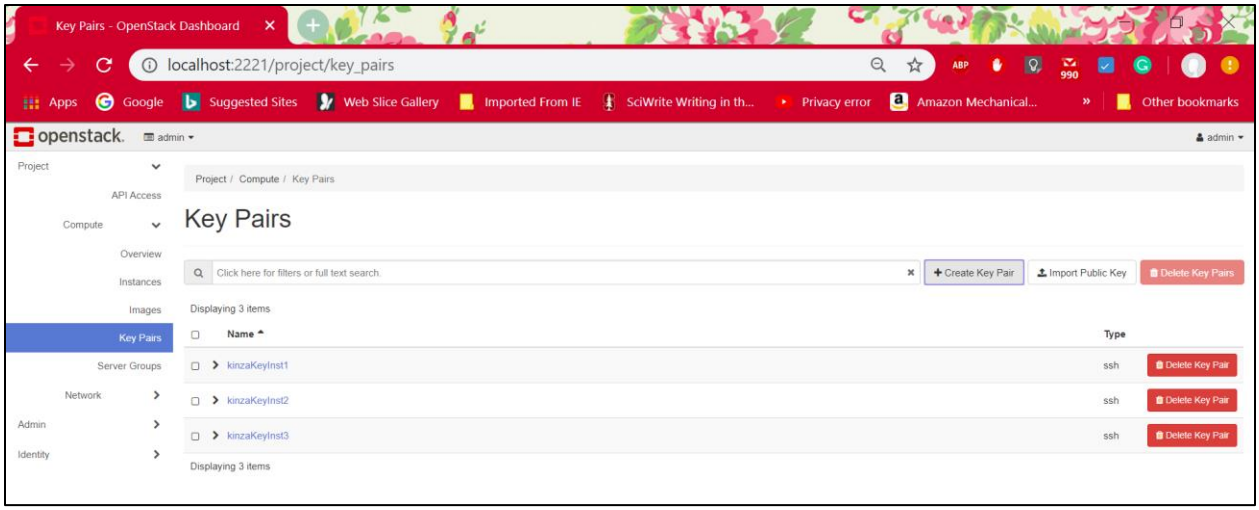
1. Clicked on **Project>>Compute>>Key Pairs** and clicked on **Create Key Pair**



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 intsances.



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.

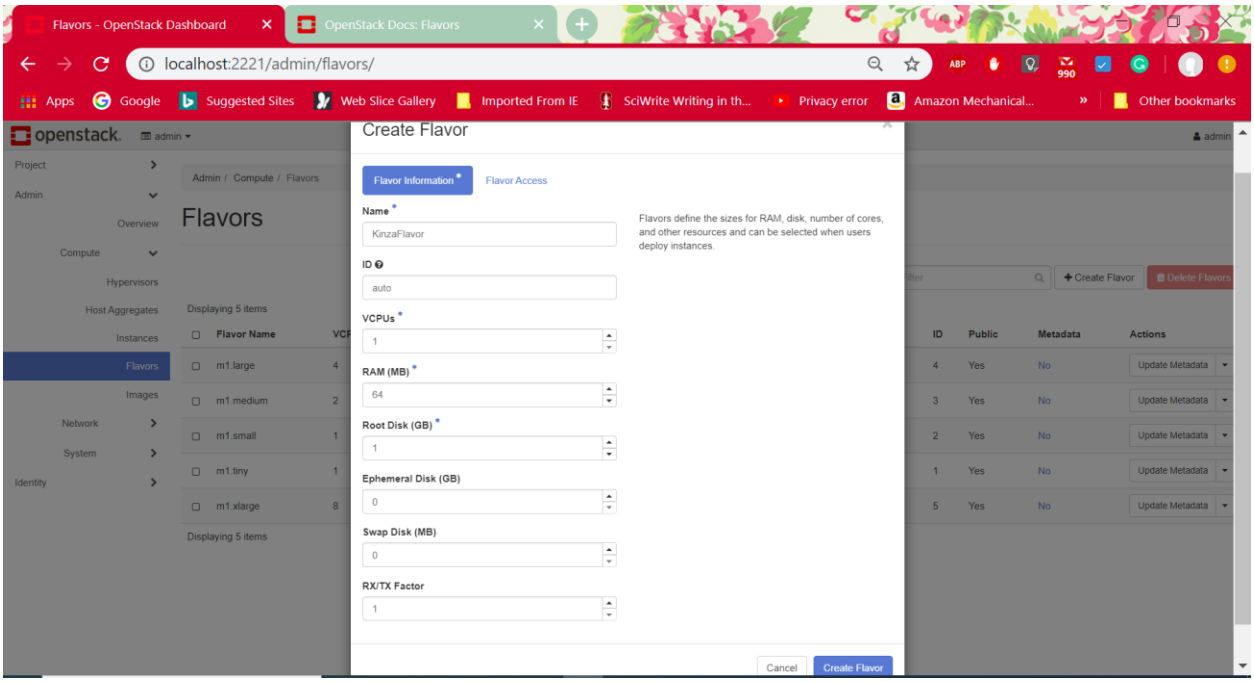
Flavor Creation:

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.

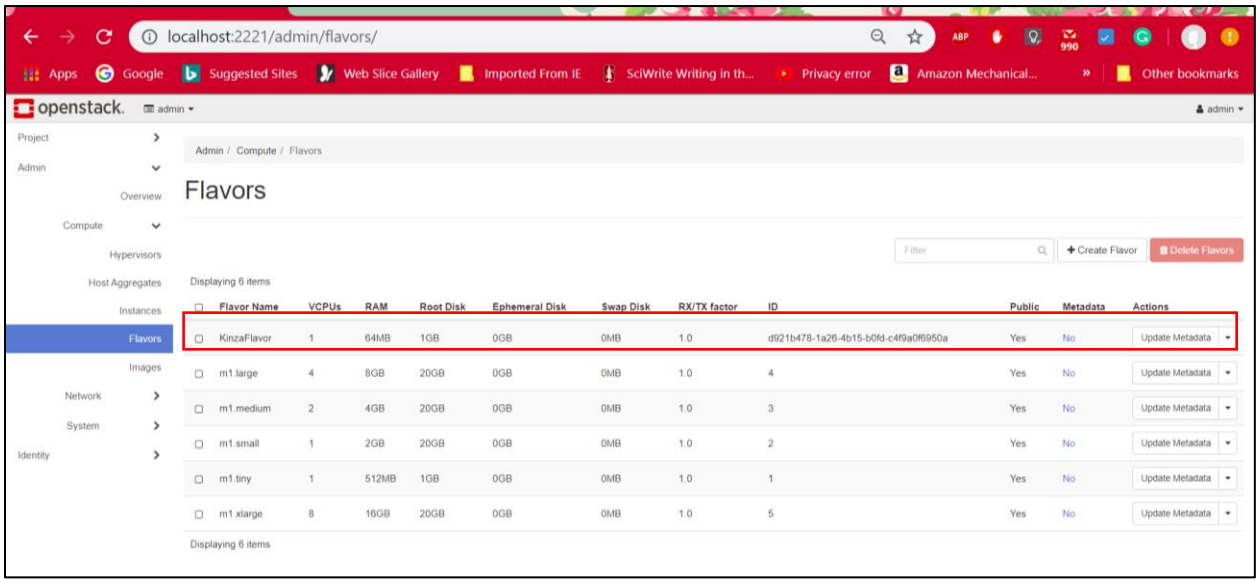
Steps:

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 1[3].



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



Instances Creation:

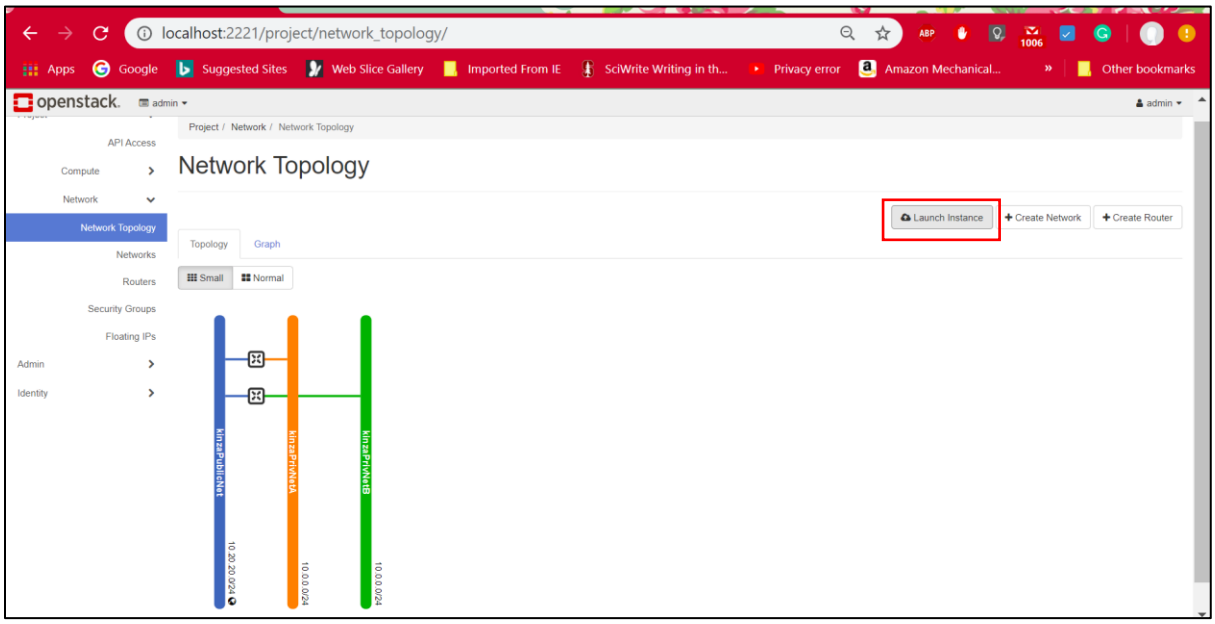
Instances are virtual machine created inside a cloud. The instance **source** could be image, snapshot which contains only description/template for creating a virtual machine. Then, selecting **flavor** which decides the computational, storage and memory specification for the instance. **Networks** is

[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.

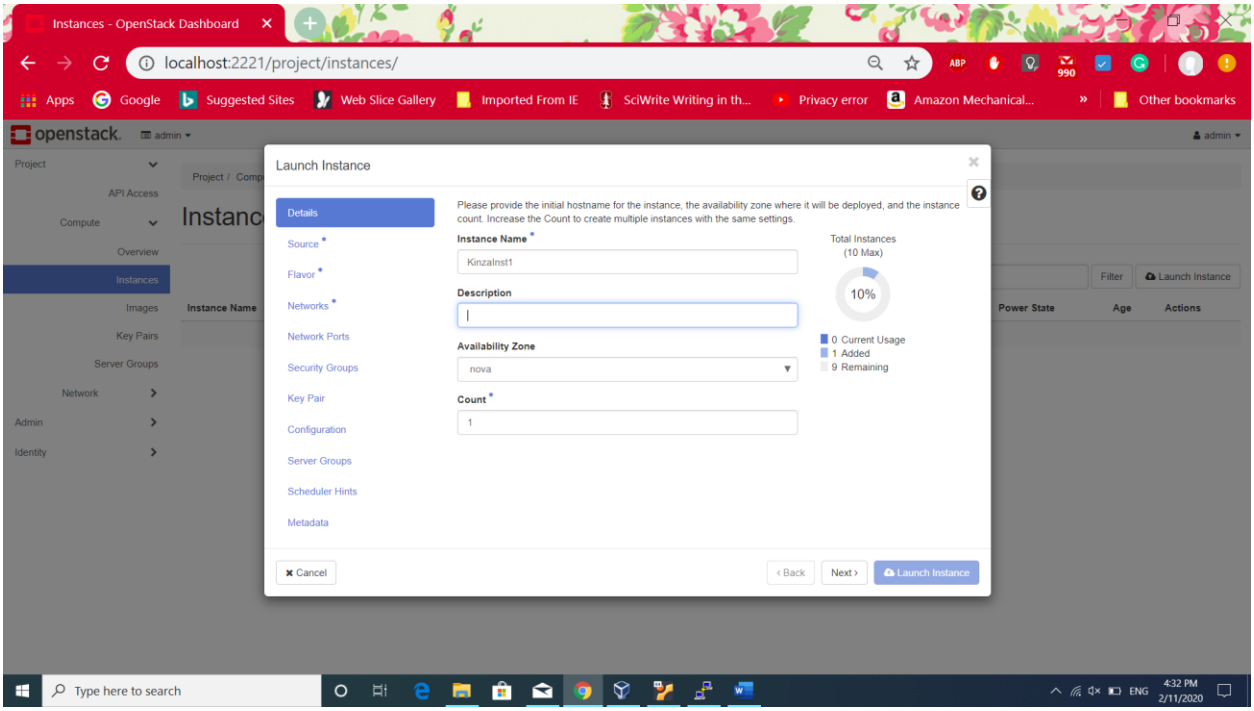
Option 1:

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

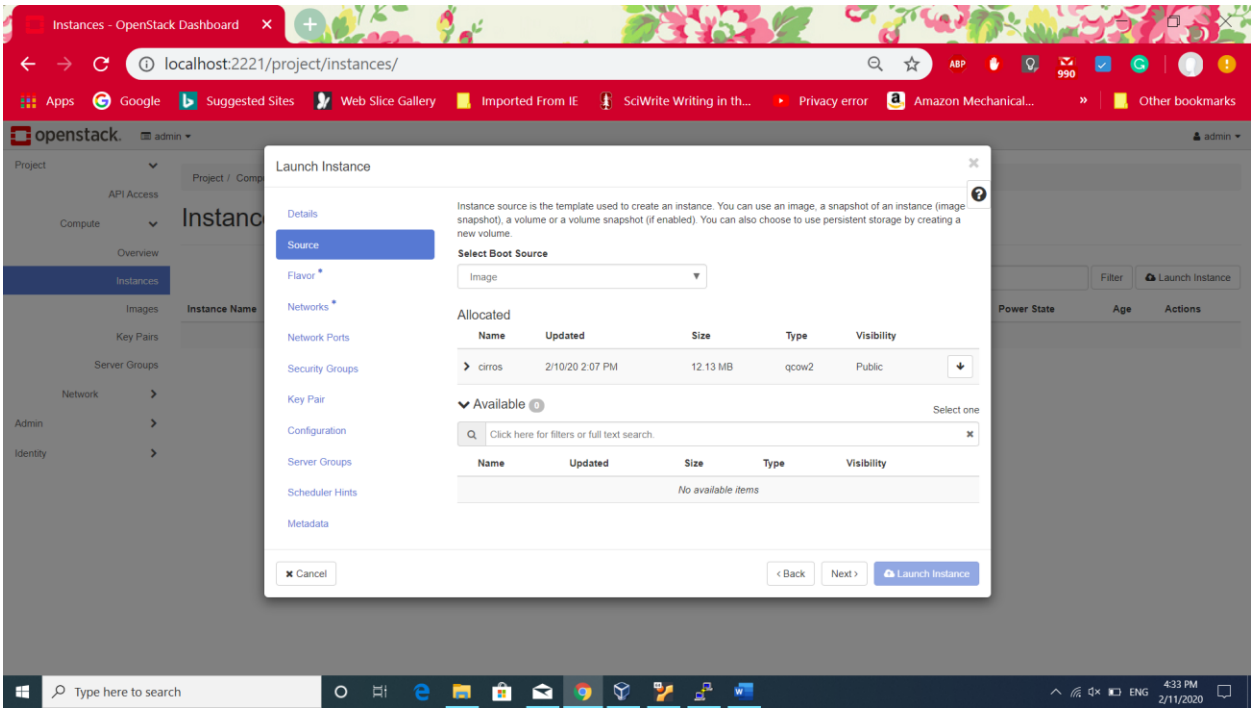


Option 2:

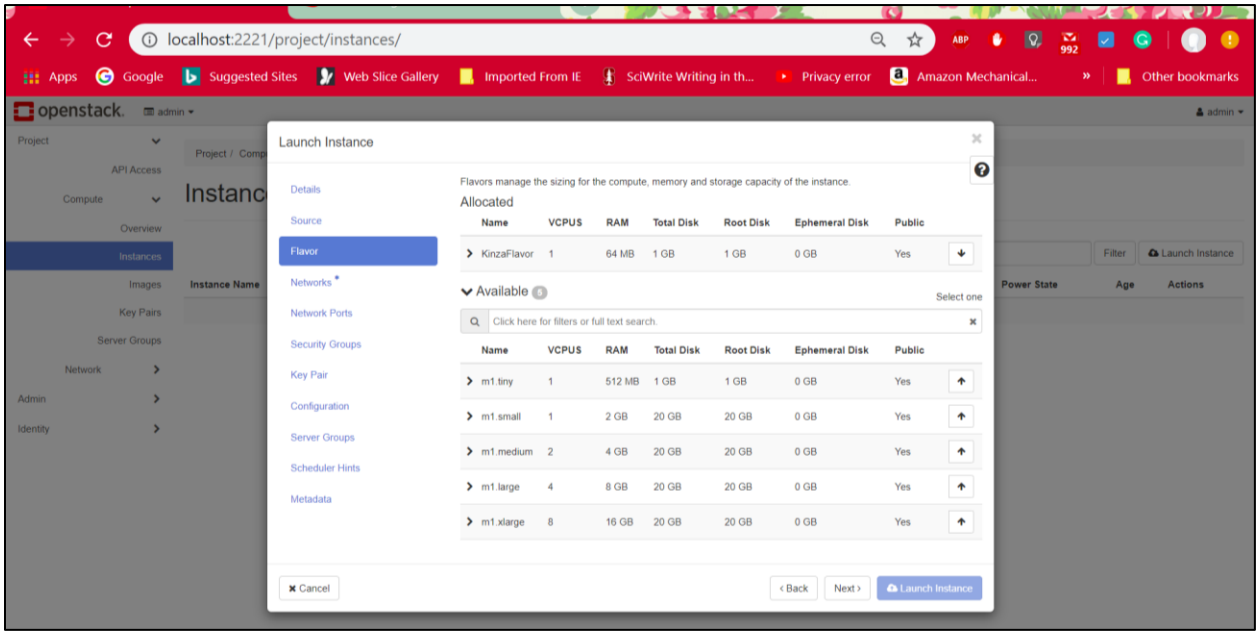
1. Clicked on Project>Compute>Instances>Launch Instance. The following screen appears.



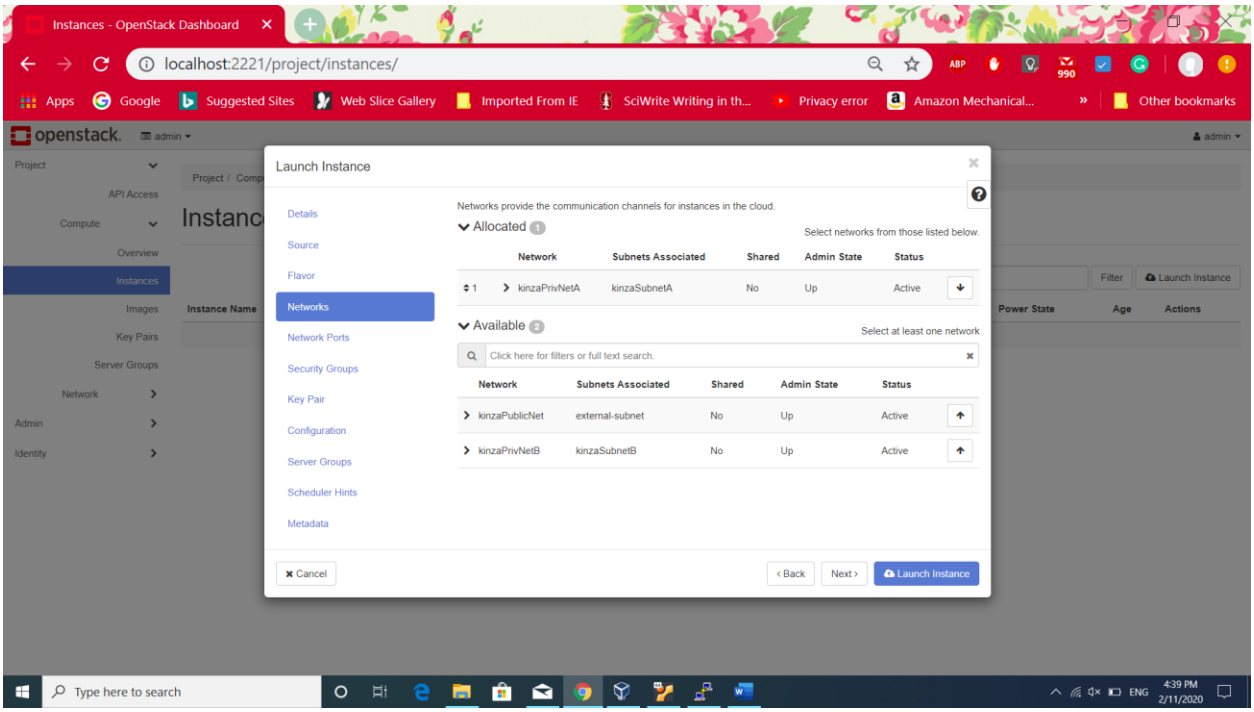
2. Clicked **Next** after entering **KinzaInst1**. And selected the **cirros** image which is the only available option.



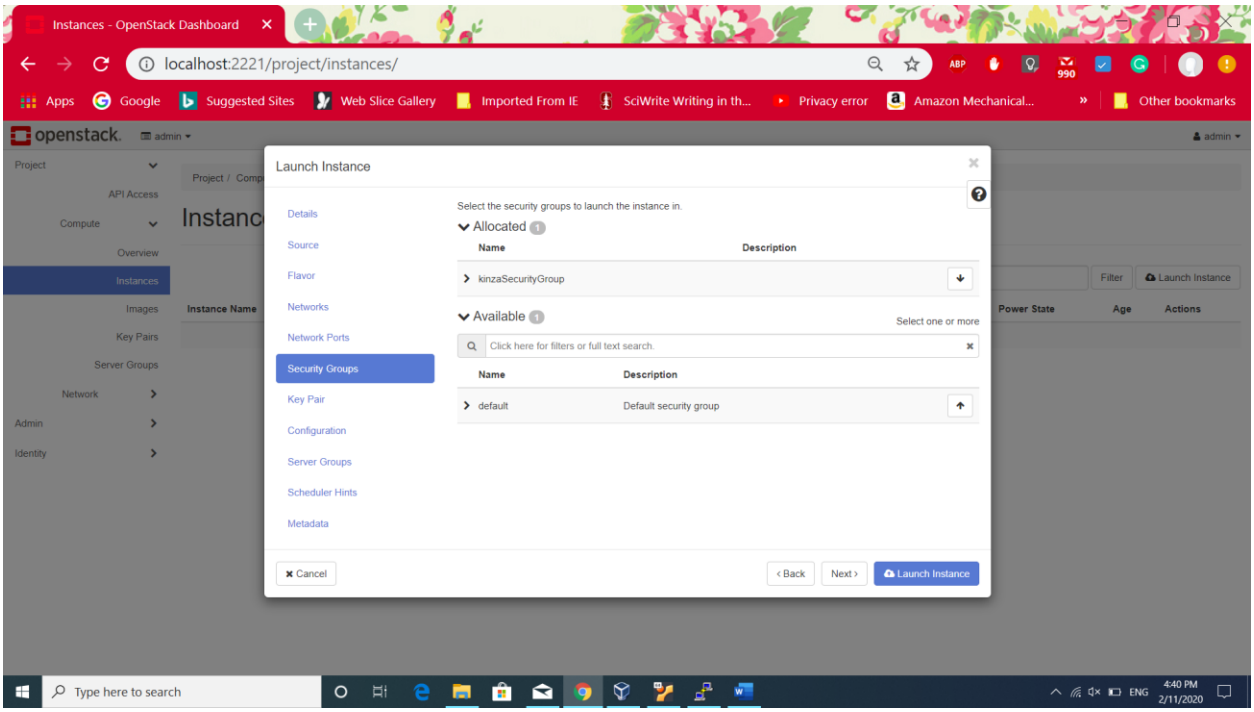
3. On **Next** screen, Selected the previously created **KinzaFlavor** .



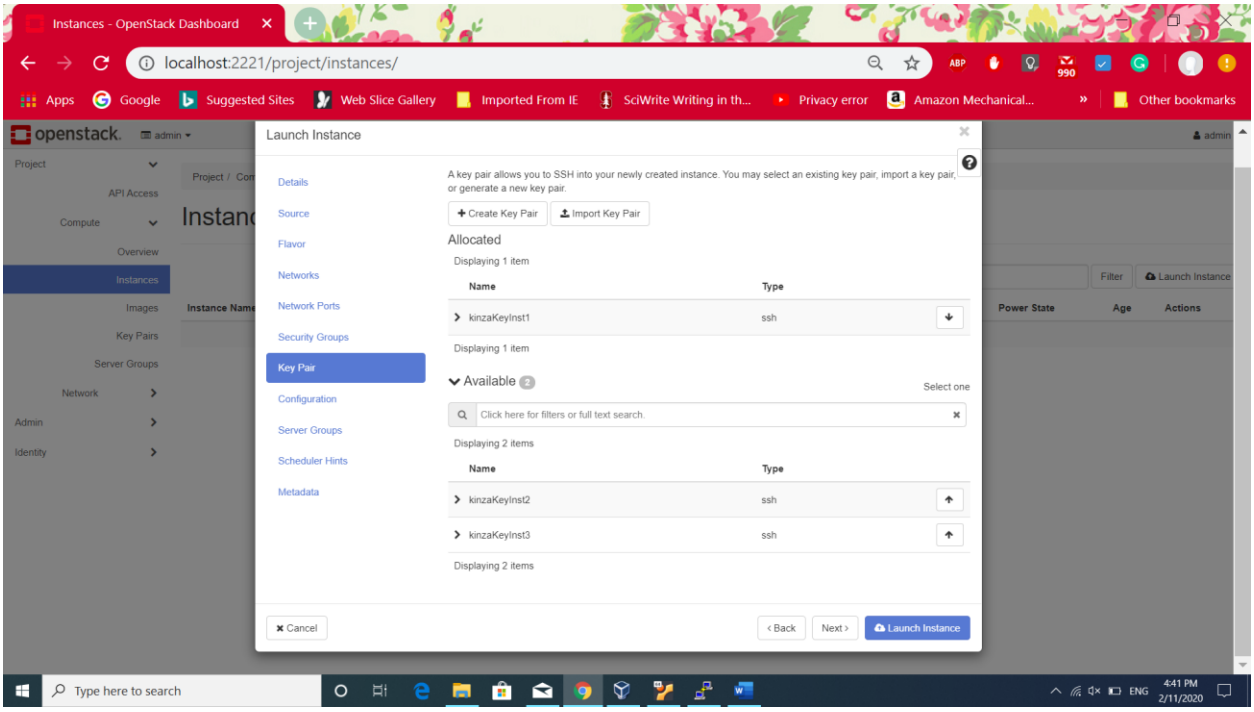
3. Next step, selected **kinzaPrivNetA** for the KinzaInst1 as required by the task.



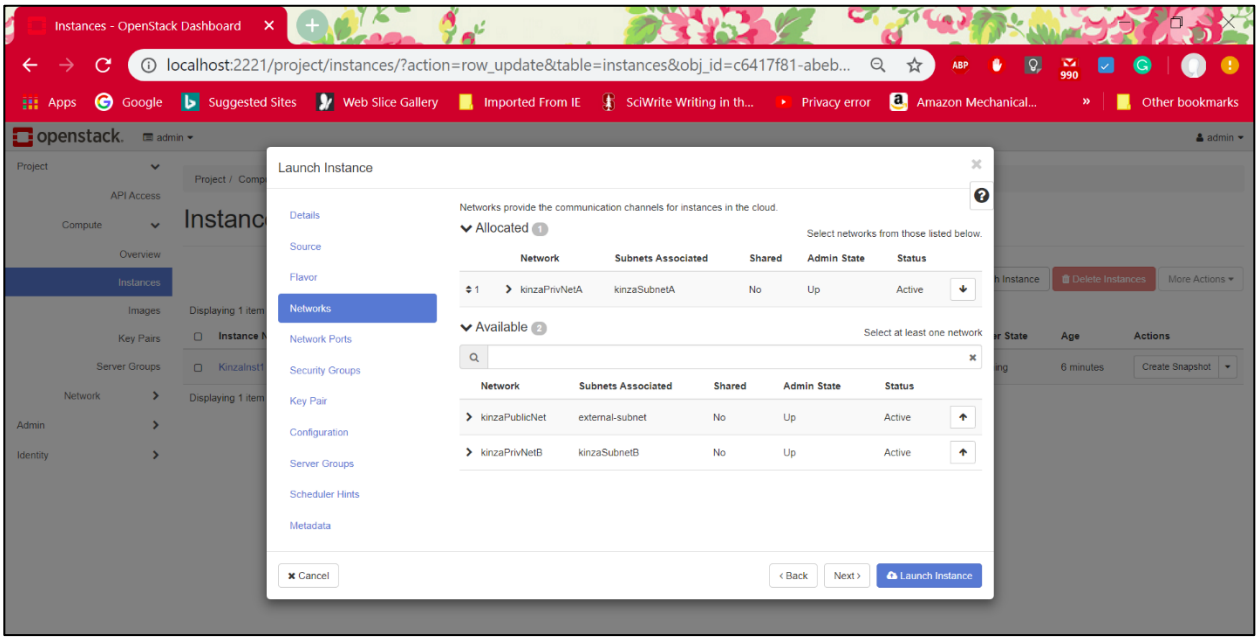
4. Now adding **Security Group,** the default security group allow only ssh connection to the instance. So, created new security group with **KinzaSecurityGroup** having ingress which is incoming and engress which is outgoing traffic.



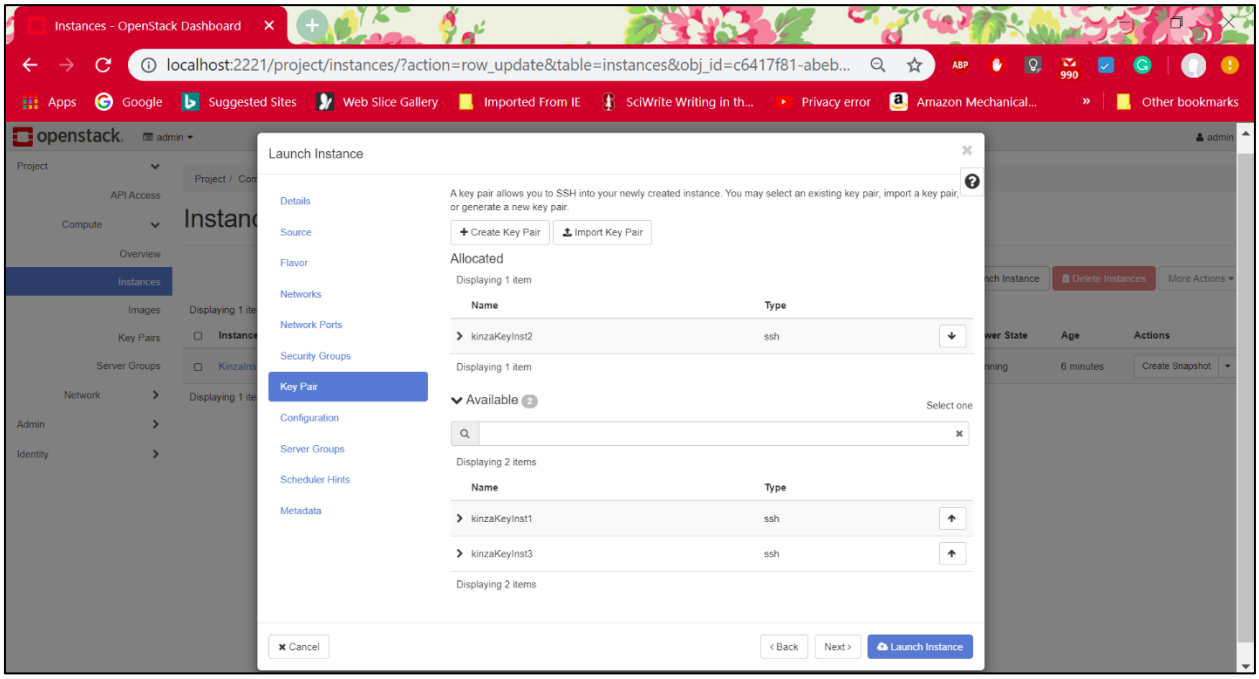
5. Next , selected the first keypair **kinzaKeyInst1** for the first instance.



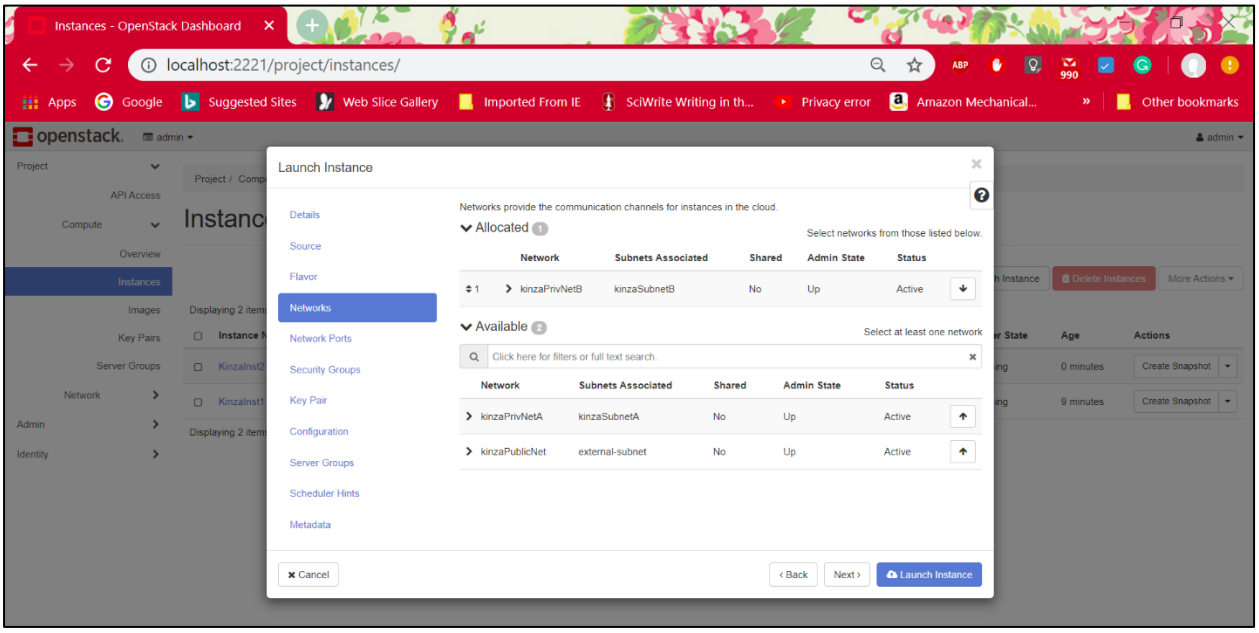
6. Clicked on Launch Instance. Repeated the same procedure for creating second instance except assigning the same private network **kinzaPrivNetA** to **KinzaInst2**.



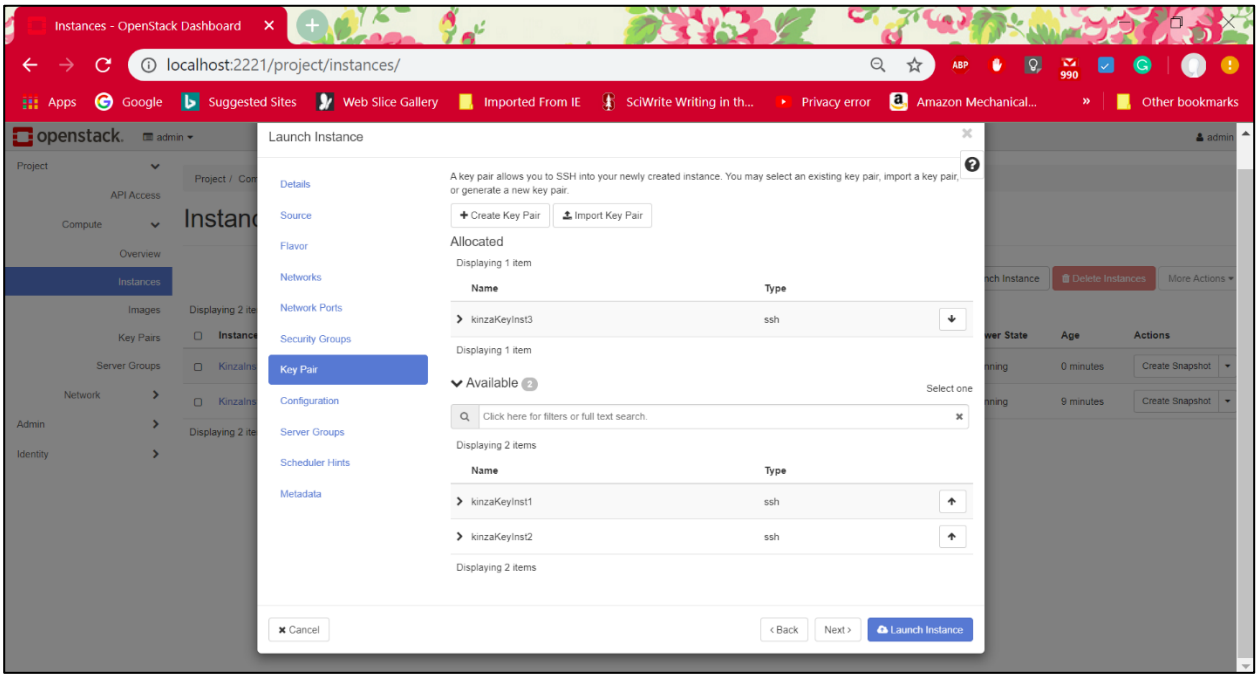
7. The **kinzakeyInst2** has been assigned to **kinzaInst2**.



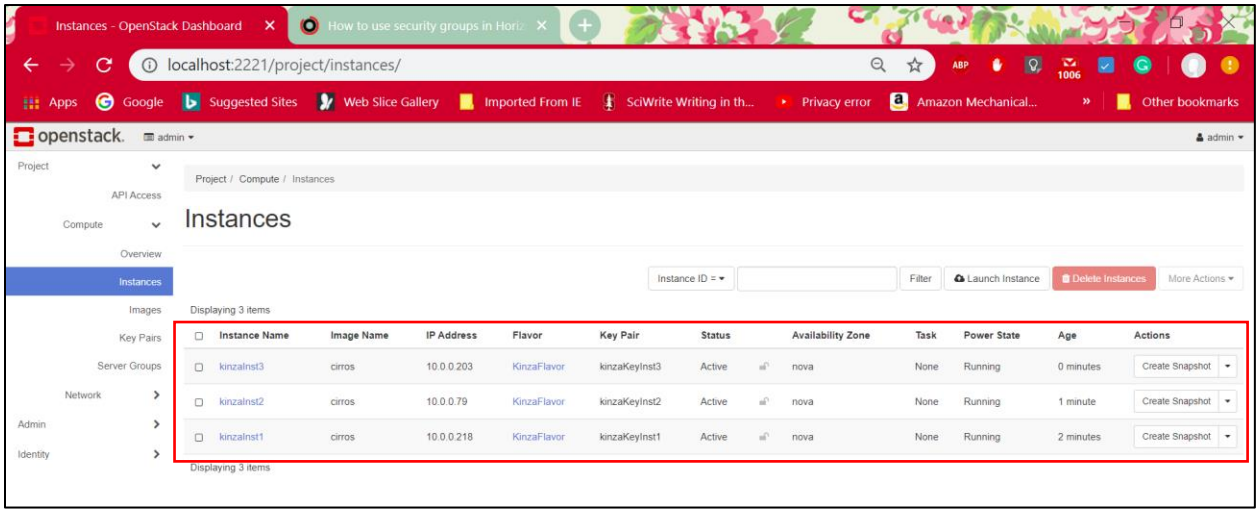
8. For Instance3 , assigned private network with name **kinzaPrivNetB**



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



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

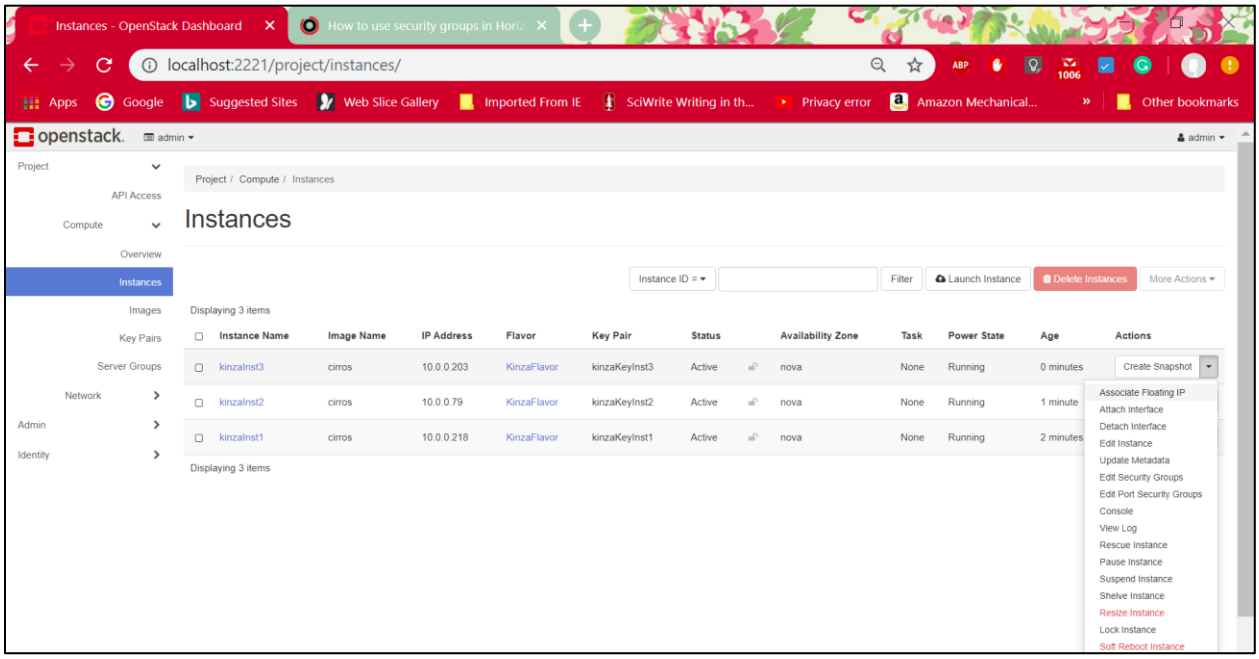


Task No. 5:

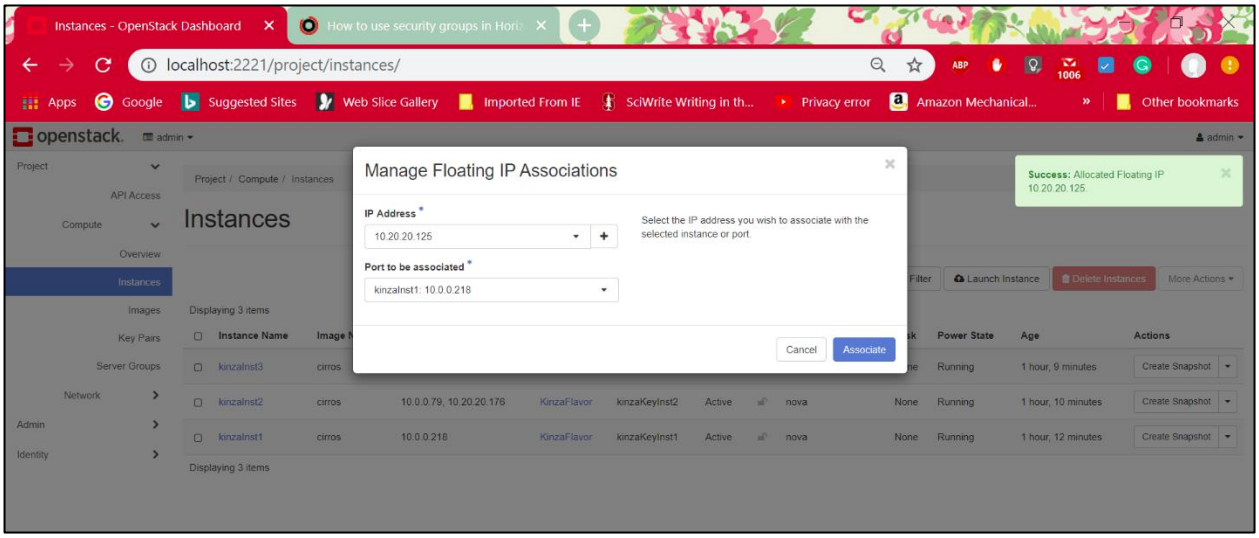
Associating the floating IP address and managing the security group

Associating the floating IP address:

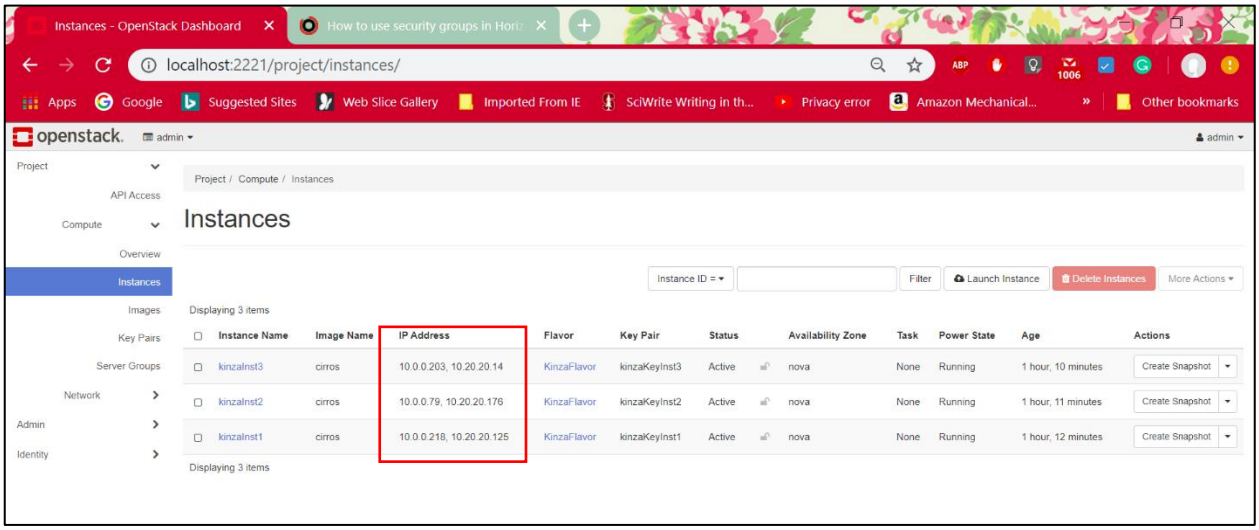
1. A floating IP is assigned to instance(VM) which is used to connect to external network. It can be changed after the creation of instance. The following screen show the option available infront of each instance for associating IP address.



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

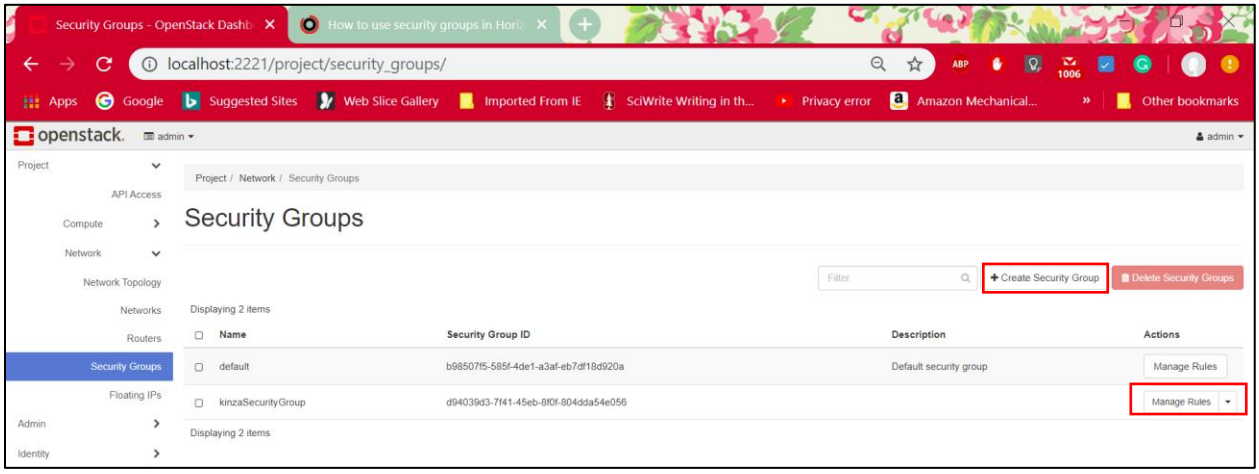


3. The above procedure repeated for all three instances and the following screen shows. Now, floating IP's can be seen associated with the private ip address of instances.

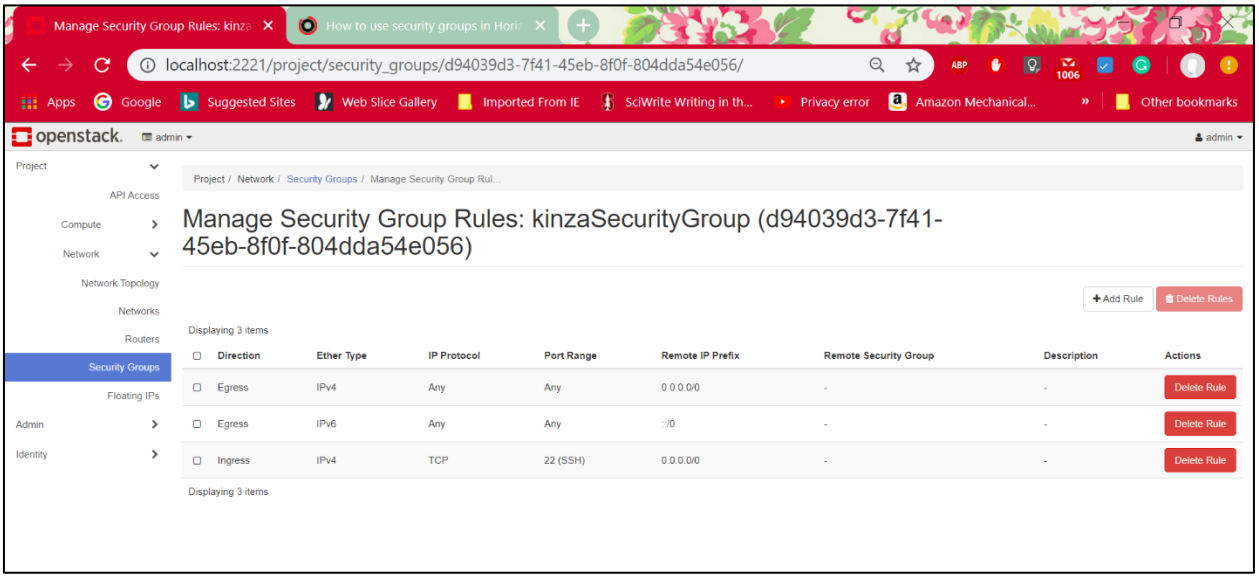


Managing the security group:

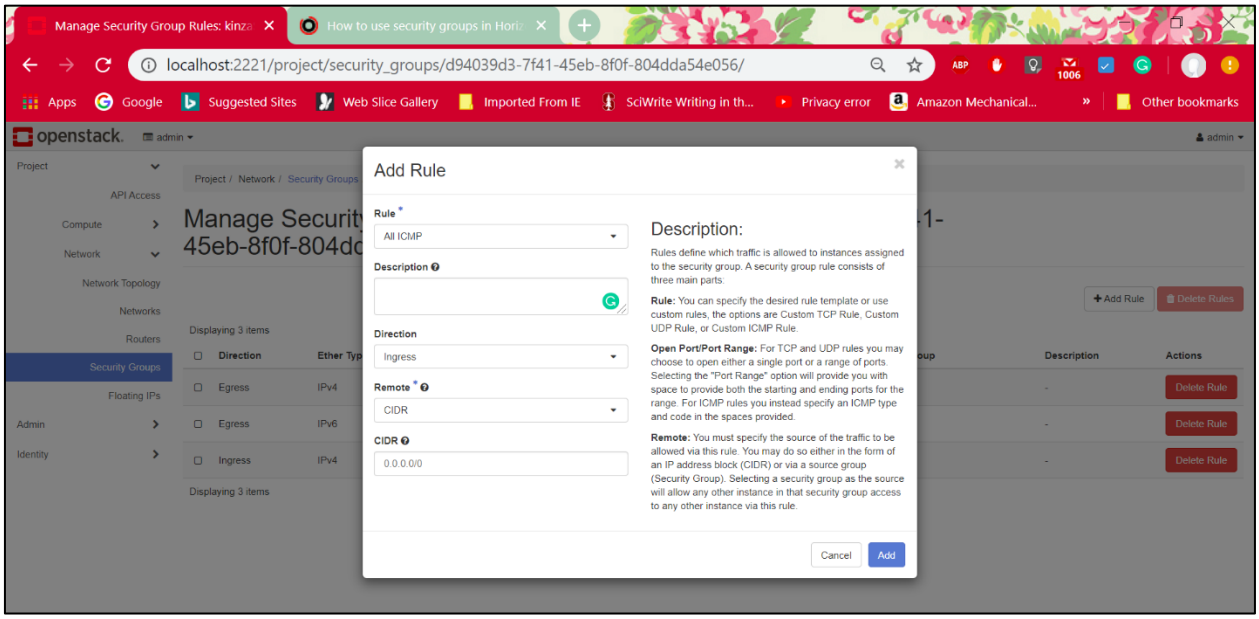
1. Clicked on **Project>>Network>>Security Group** and then clicked on **Manage Rule** button on the right of **KinzaSecurityGroup**.



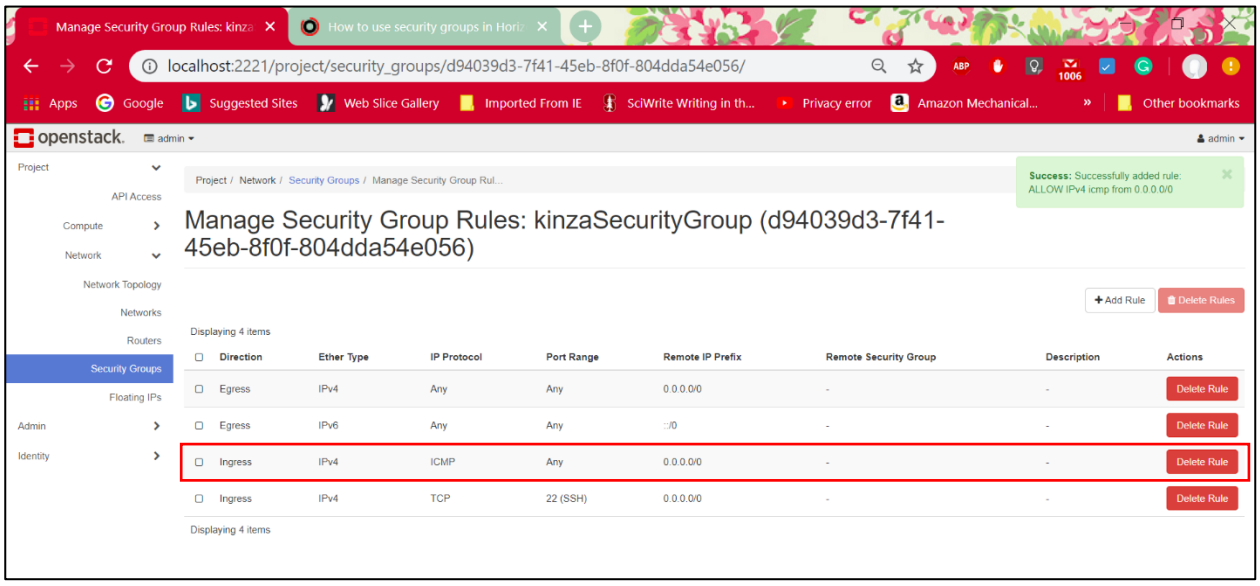
2. After clicking on Manage Rule, the available rules are shown, **Engress** rules control the outgoing traffic which means any IPV4 and IPv6 address can send information to outside world. The current **Ingress** rule is only IPv4 traffic is allowed through making ssh connection on port 22 and remote IP prefix can be any. The task requires manging security group by allowing ping test from all instance. For this purpose, new rule for ping will be added. Clicked on **Add Rule**.



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.



Option2:

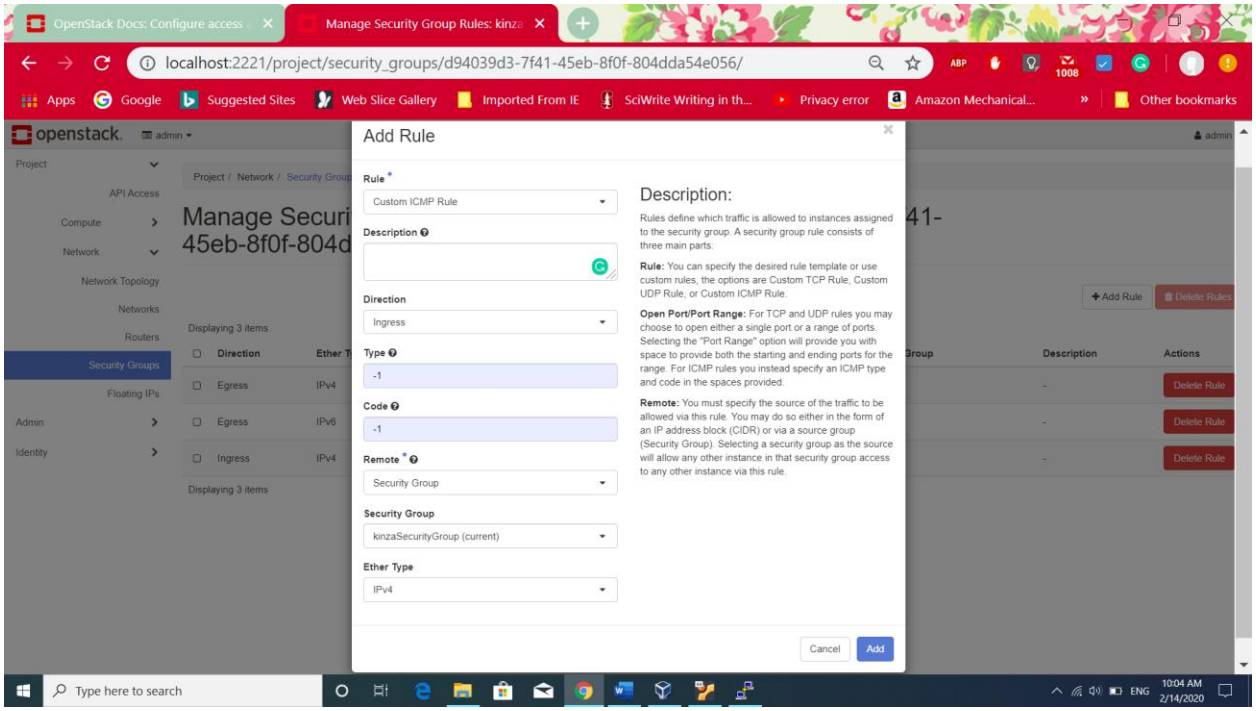
There is one more option to set the the ICMP rule to specific security group then only instances which are related to that security group are allowed to ping the instances. The following screen shows that

Rule is **Custom ICMP Rule**

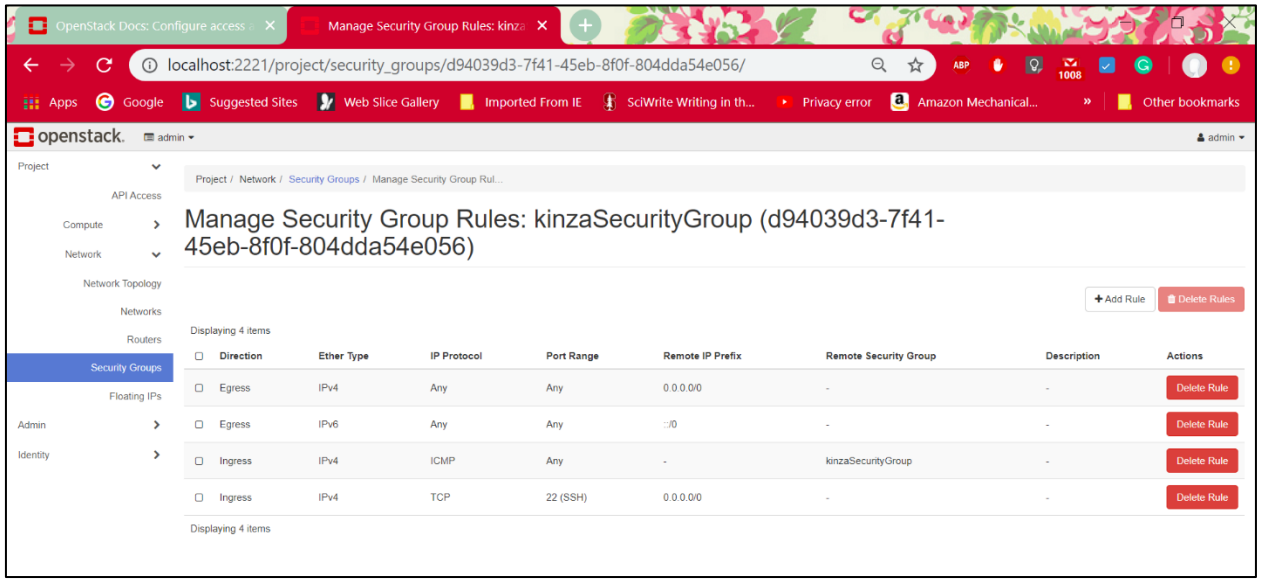
Remote: **Security group**

Security Group: **KinzaSecurityGroup**

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

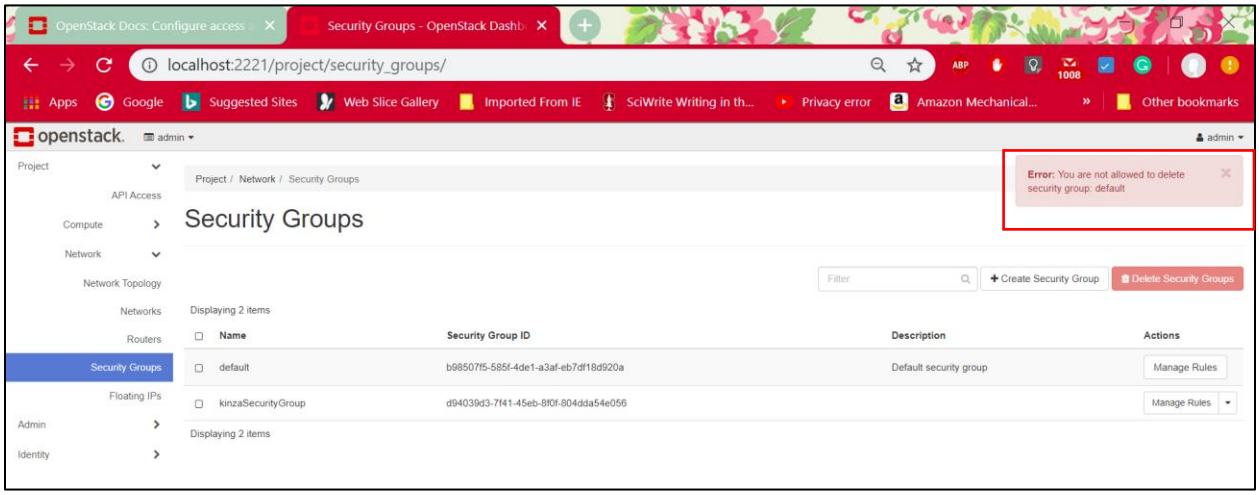


The following screen shows Remote security group selected which is allowed to ping.



Limitation:

1. Default security group can not be deleted[4] as can be seen in the following screen the error message on deleting the default security group.

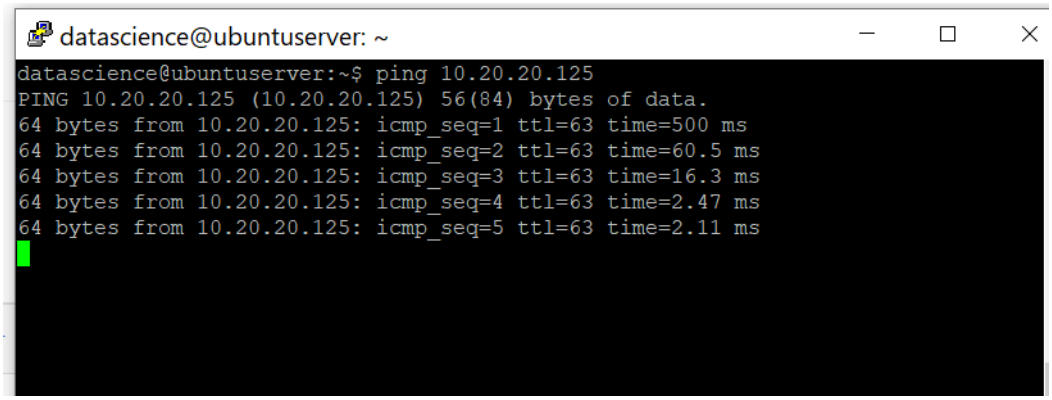


2. By default it only allows ssh connection to instances.
3. Other security group can not be deleted if they are attached to an instance.

Task No 6:

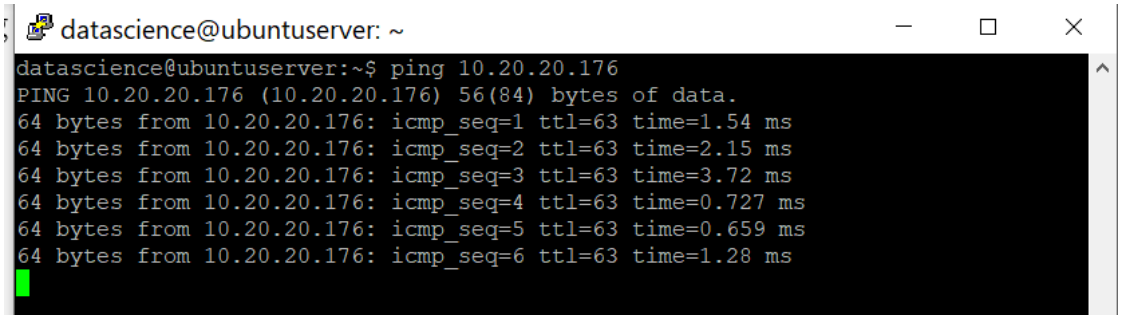
Ping Test for instances.

1. Before adding the ICMP rule, Pinging the kinzaInst1 which has floting 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.

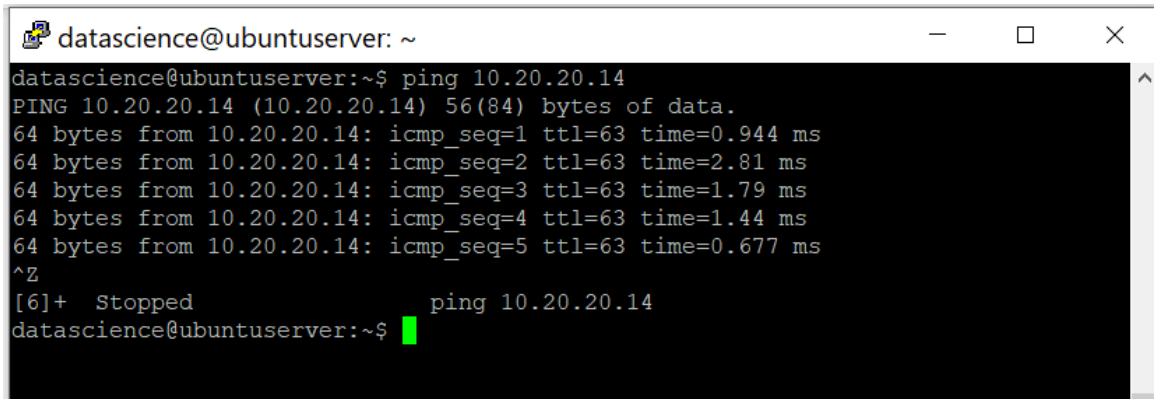


[4] https://docs.openstack.org/mitaka/user-guide/cli_nova_configure_access_security_for_instances.html

2. Pinging second instance with Floating IP



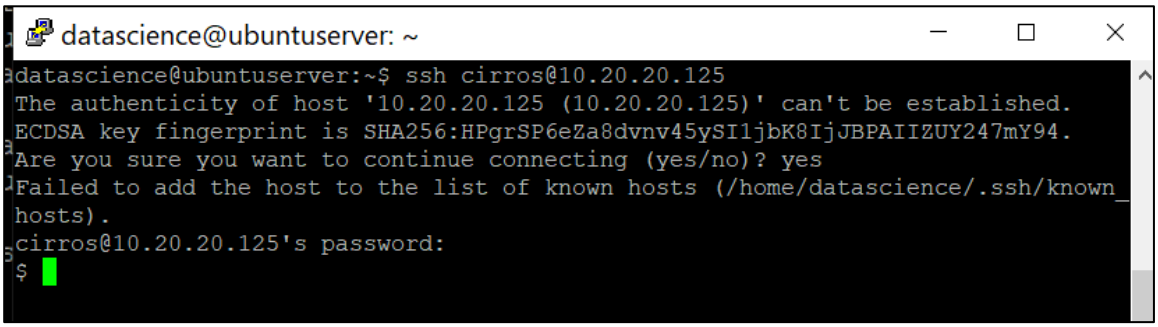
3. Pinging third instance



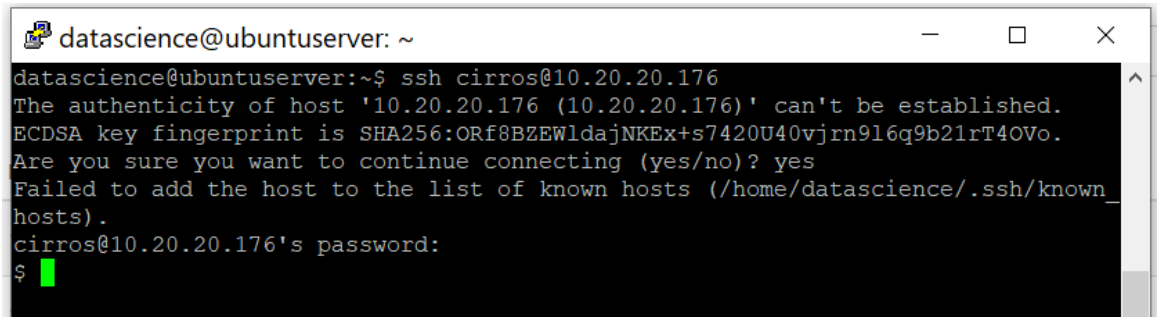
Task No 7:

Connecting to instance via ssh

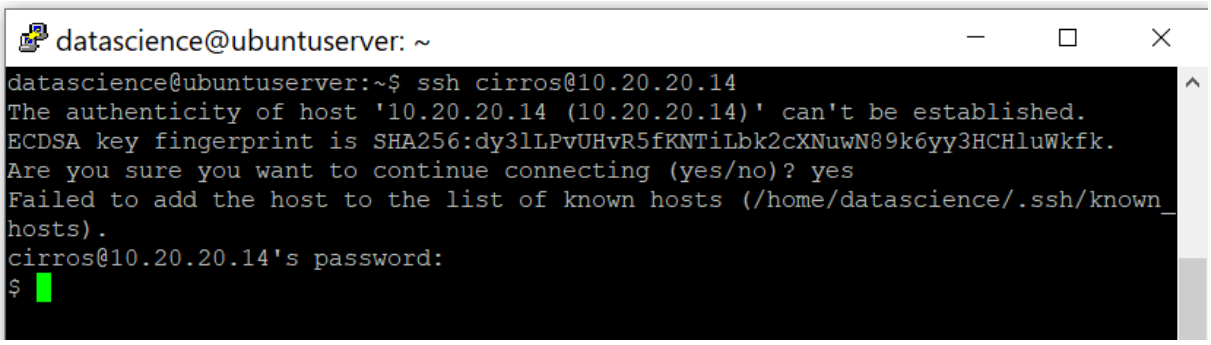
1. The following screen shows connection made via ssh to instance **kinzaInst1** as allowed in the security group.
   ssh cirros@10.20.20.125 command: where cirros is the instance name and 10.20.20.125 is the floating IP.



2. The following screen shows ssh connection to kinzaInst2.



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



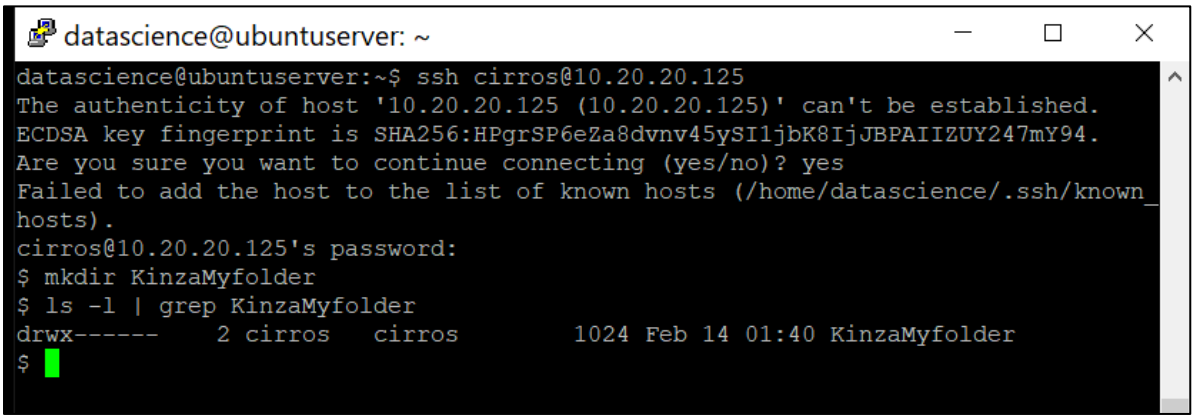
Task No. 7:

<u>Creating a Folder and checking the permissions</u>:

$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 KinzMyfolder 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.



Task No. 9:

Creating python file in created folder

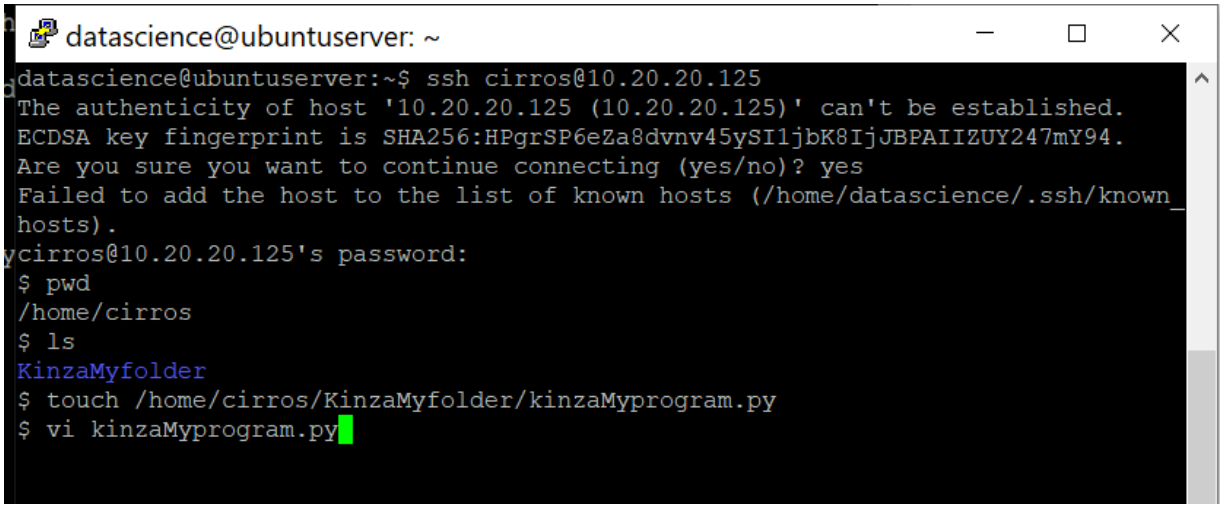
The first command is ***touch.*** The following explains the purpose of the touch command.

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.

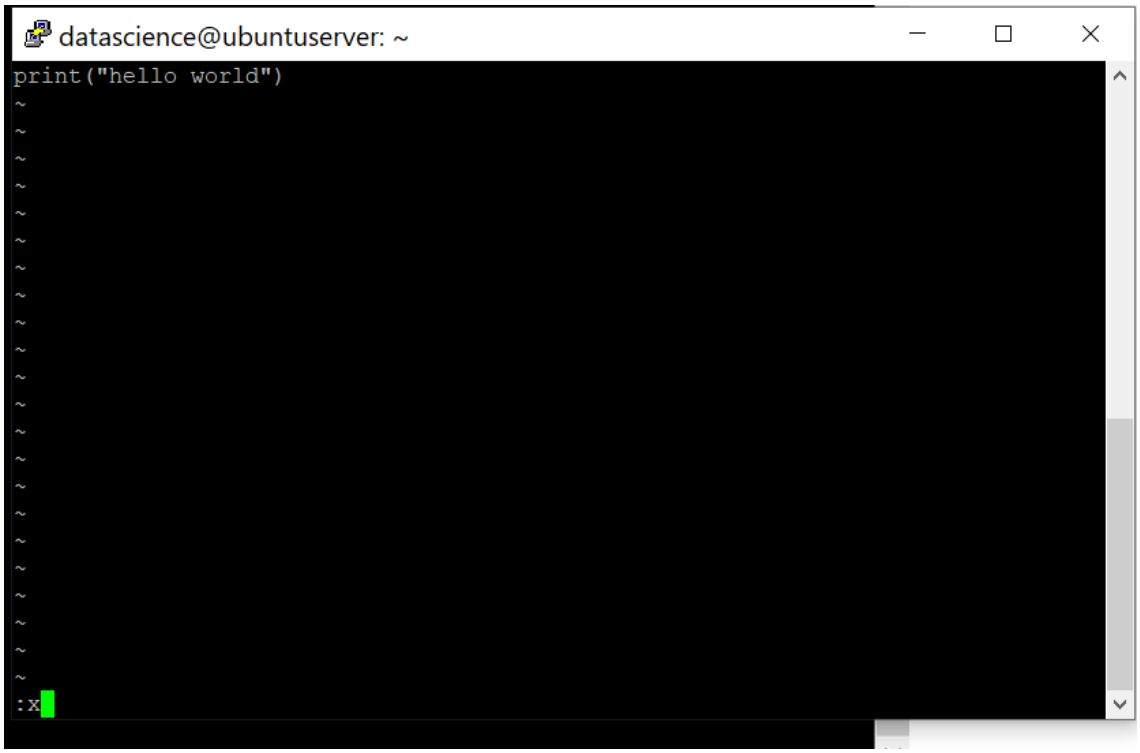
Vi command:Vi is text editor to write and update the file contents.

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.



2. After entering, vi text editor opens, Press ***i*** from the keybaord 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.



Task No. 10:

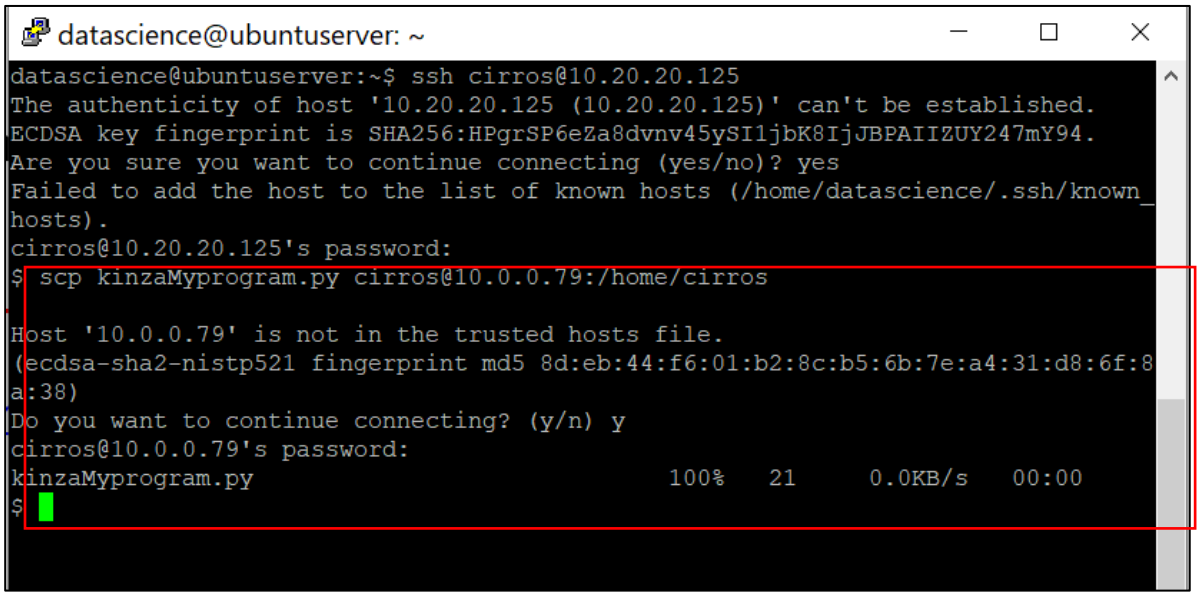
Copying file from one instance to other

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]

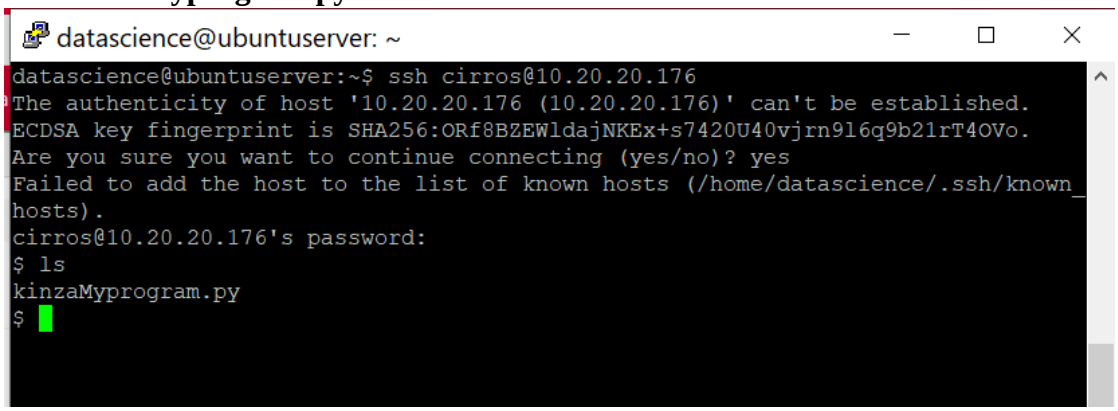
1. **$scp kinzaMyprogram.py cirros@10.0.0.79 :/home/cirros :**

- **SCP kinzaMyprogram.py:** is secure copy transfer command with file name next to it.
- **cirros@10.0.0.79**: Username and pirvate IP address of the second instance kinzaIns2 and kinzaInst1 and kinzaInst2 both are on the same kinzaPrivnetA so private IP address has been used because file is transferred on the samenetwork.
- **:/home/cirros** specifies the directory in the host where file should be stored.

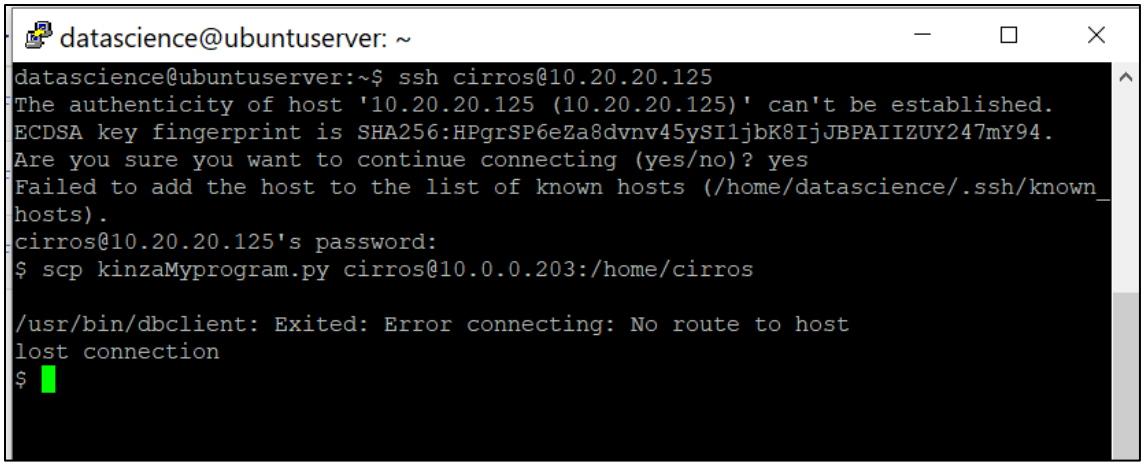
[5] https://imyuvii.com/posts/scp-command-transferring-files/



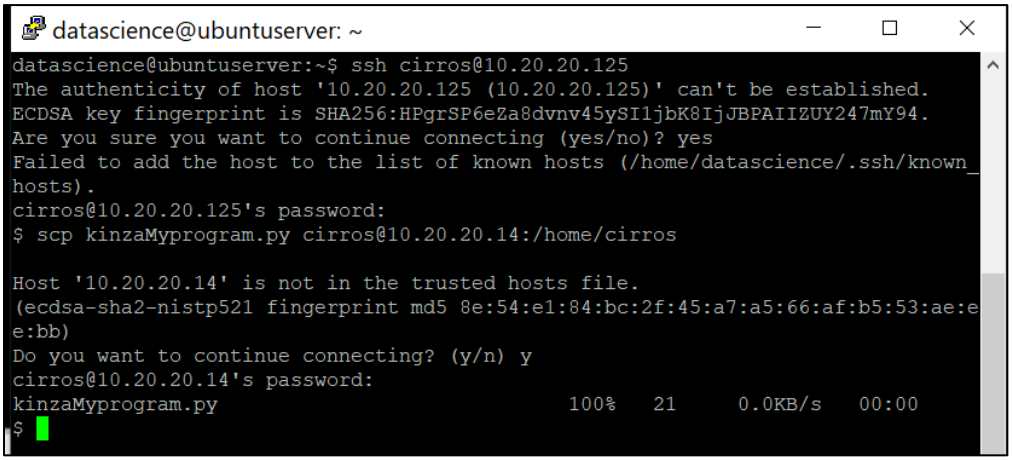
2. Checking the transferred file on the **kinzaInst2**. After making connection with the **kinzaInst2** using Floating ip, **ls** command has been used to check the files. It can be seen the **kinzaMyprogram.py** file has been transferred.



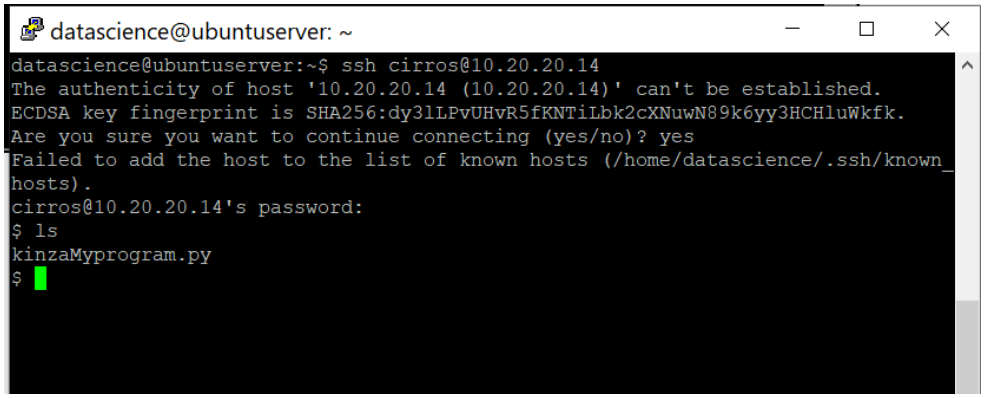
3. To transfer file to **kinzaInst3**, private ip address has been used to send file using SCP command in the following screen. It can be seen that transfer failed . Error in connecting which means no route has been found by the network because **kinzaInst3** does not exist on the network. So, using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



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



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



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.