**Assessment 10 – VPC**

**Trainee Name : Gargi Sharma**

**Mentor Name : Mr. Ravi Kumar**

**College Name : UPES**

1. When to use Elastic IP over Public IP

Public IP addresses are dynamic - i.e. if you stop/start your instance you get reassigned a new public IP.Elastic IPs get allocated to your account, and stay the same - it's up to you to attach them to an instance or not. You could say they are static public IP addresses.Elastic IP address is a public static IPv4 address which is reachable from the Internet.Basically Elastic IP addresses are used by AWS to manage its dynamic cloud computing services. Within the AWS infrastructure, customers have virtual private clouds (VPC), within the VPCs, users have instances. So when you launch an EC2 instance, you receive a Public IP address by which that instance is reachable from the internet. Once you stop that instance and restart the instance you get a new Public IP for the same instance. So it's basically a problem to connect your instance from the internet for not having a static IP. To overcome this problem, we attach an Elastic IP to an Instance which doesn't change after you stop / start the instance.

1. Valid IP Ranges for LAN, Implication of using Public IP ranges for Private Network.

Class A : 10.0.0.0 – 10.0.255.255

Class B : 172.16.0.0 – 172.13.255.255

Class C : 192.168.0.0 – 192.168.255.255

If you ever expect to connect these systems to an Internet-facing router, though, then you could experience the following issues if you don't stick with private IP ranges:

● Traffic destined for another host may leak out on to the Internet.

● You might want to get to the IANA-assigned host on that IP and may not be

able to do it if it's an internal host.

● If you aren't the only one mantaining this network, you could horribly confuse someone who is doing troubleshooting.

1. List down the things to keep in mind while VPC peering.

To create a VPC peering connection with another VPC, be aware of the following

limitations and rules:

● You cannot create a VPC peering connection between VPCs that have matching or overlapping IPv4 or IPv6 CIDR blocks. Amazon always assigns your VPC an unique IPv6 CIDR block. If your IPv6 CIDR blocks are unique but your IPv4 blocks are not, you cannot create the peering connection.

● You have a quota on the number of active and pending VPC peering connections that you can have per VPC. For more information, see ​Amazon VPC Quotas​ in the Amazon VPC User Guide

● VPC peering does not support transitive peering relationships. In a VPC peering connection, your VPC does not have access to any other VPCs with which the peer VPC may be peered. This includes VPC peering connections that are established entirely within your own AWS account. For more information about unsupported peering relationships, see ​Unsupported VPC Peering Configurations​. For examples of supported peering relationships, see ​VPC Peering Scenarios​.

● You cannot have more than one VPC peering connection between the same two VPCs at the same time.

● Unicast reverse path forwarding in VPC peering connections is not supported. For more information, see ​Routing for Response Traffic​.

● Any tags that you create for your VPC peering connection are only applied in the account or region in which you create them.

● If the IPv4 CIDR block of a VPC in a VPC peering connection falls outside of the private IPv4 address ranges specified by ​RFC 1918​, private DNS hostnames for that VPC cannot be resolved to private IP addresses. To resolve private DNS hostnames to private IP addresses, you can enable DNS resolution support for the VPC peering connection. For more information, see ​Enabling DNS Resolution Support for a VPC Peering Connection​.

● You cannot connect to or query the Amazon DNS server in a peer VPC.

1. CIDR of a VPC is [10.0.0.0/16](http://10.0.0.0/16), if the subnet mask is /20 calculate the number of subnets that could be created from the VPC. Also find the number of IP in subnet.

\*10.0.0.0/16: 00001010.00000000.00000000.00000000 (In /16, first 2 octets are fixed).

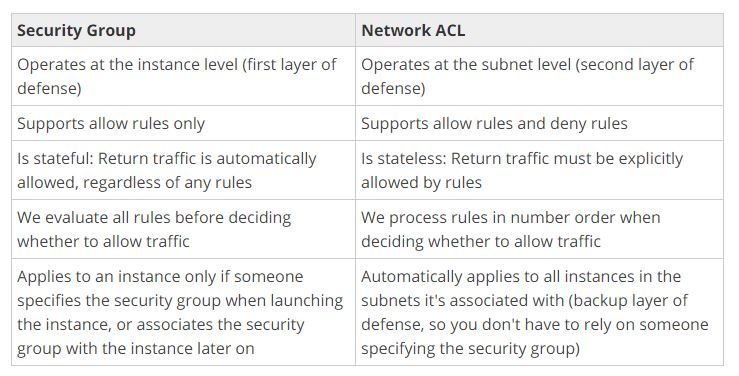
00001010.00000000.00000000.00000000 (In /20, extra 4 bits are borrowed from hosts)

\* These extra 4 bits are subnetting bits.

So, total number of subnets = 2^(20-16)=(16)

And, total IP’S in each subnet = 2^(32-20)= (4096)

1. Differentiate between NACL and Security Groups.



1. Implement a 2-tier vpc with following requirements:

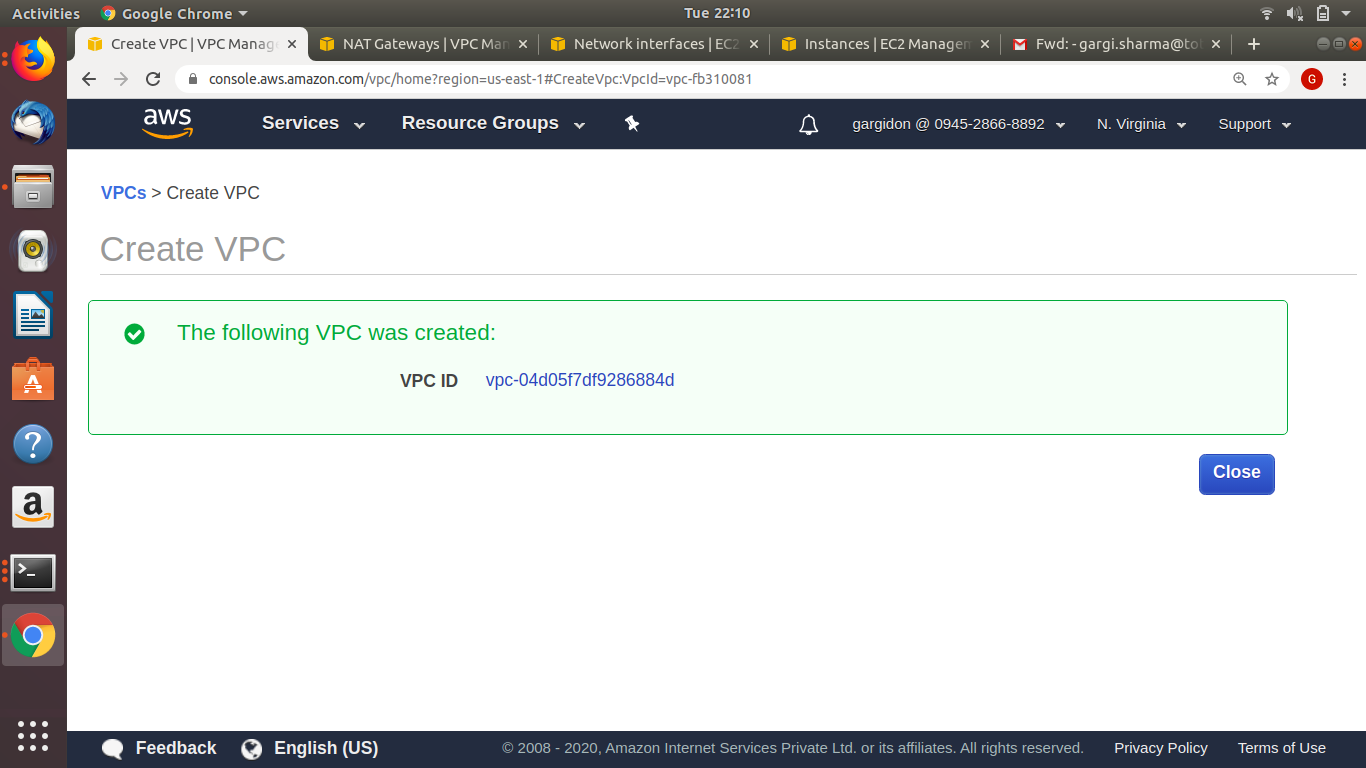
1. Create a private subnet, attach NAT, and host an application server(Tomcat)

2. Create a public subnet, and host a web server(Nginx), also proxypass to Tomcat from Nginx

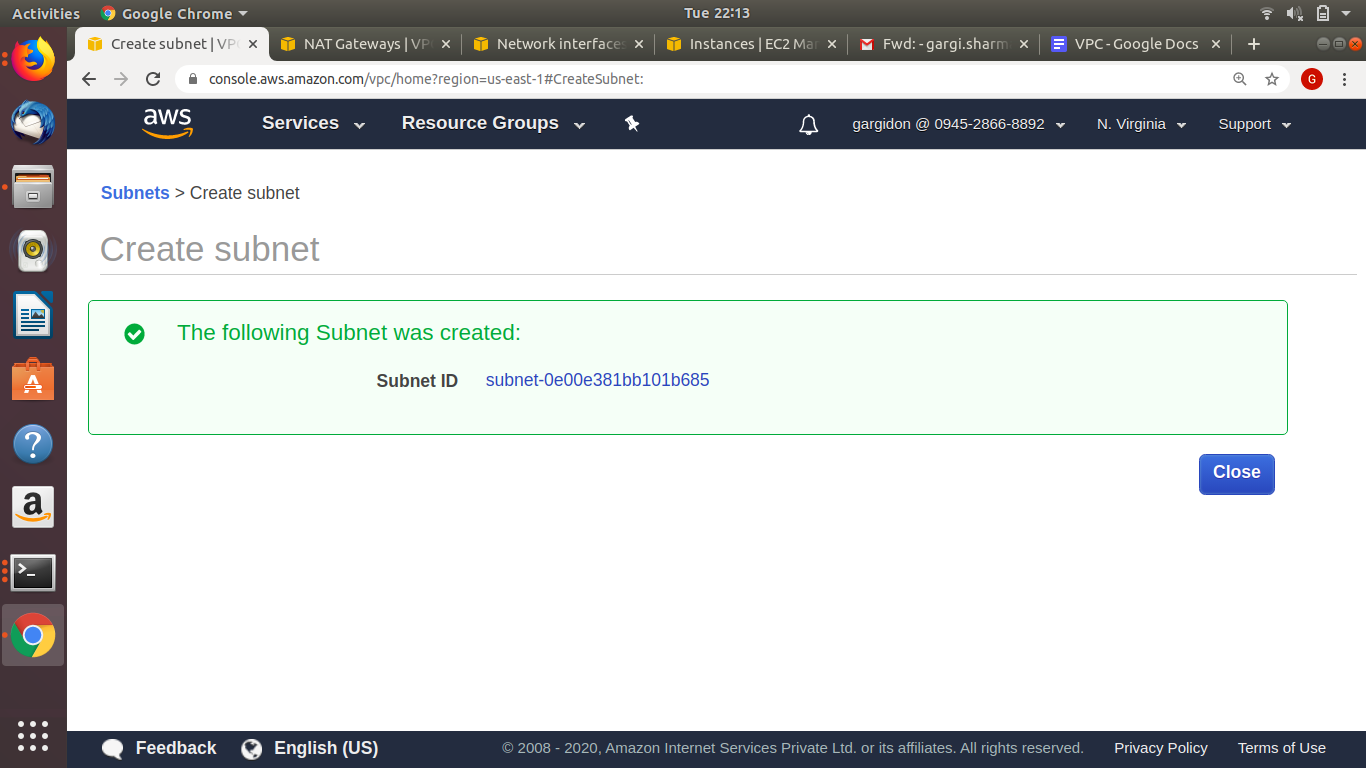
After Implementing this on AWS, create an architecture diagram for this use case.

Note: For hosting Nginx in public subnet, use Elastic IP.

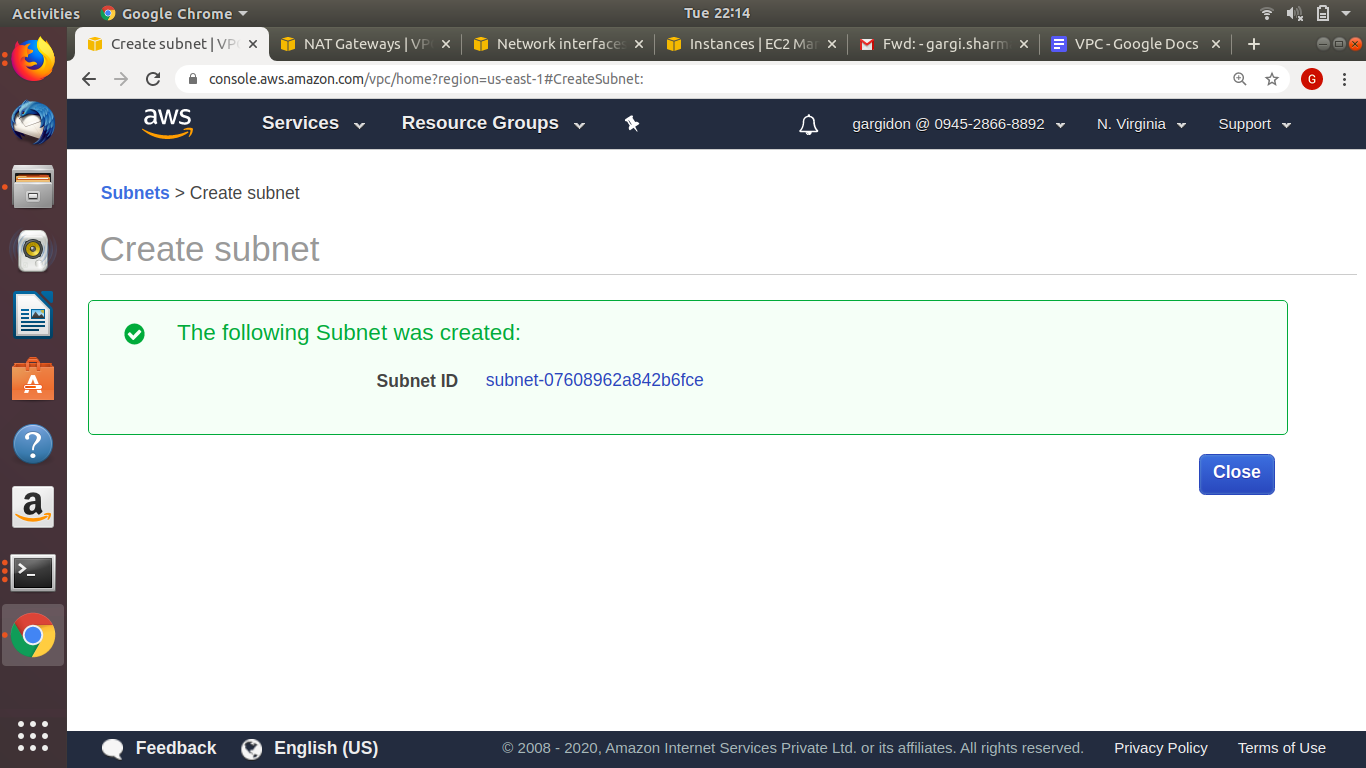
Step1: create a VPC



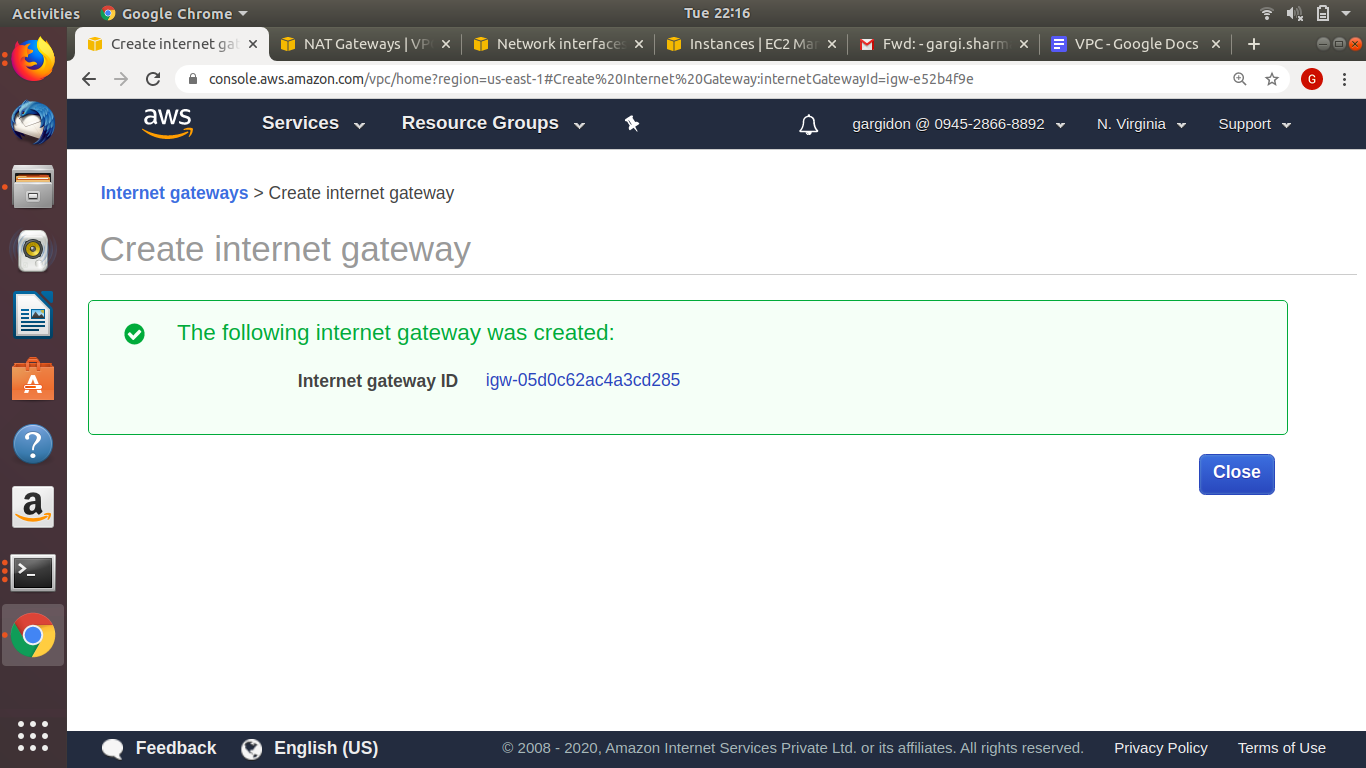
Step2: Creating private subnet: CIDR(10.0.0.0/28)

￼￼

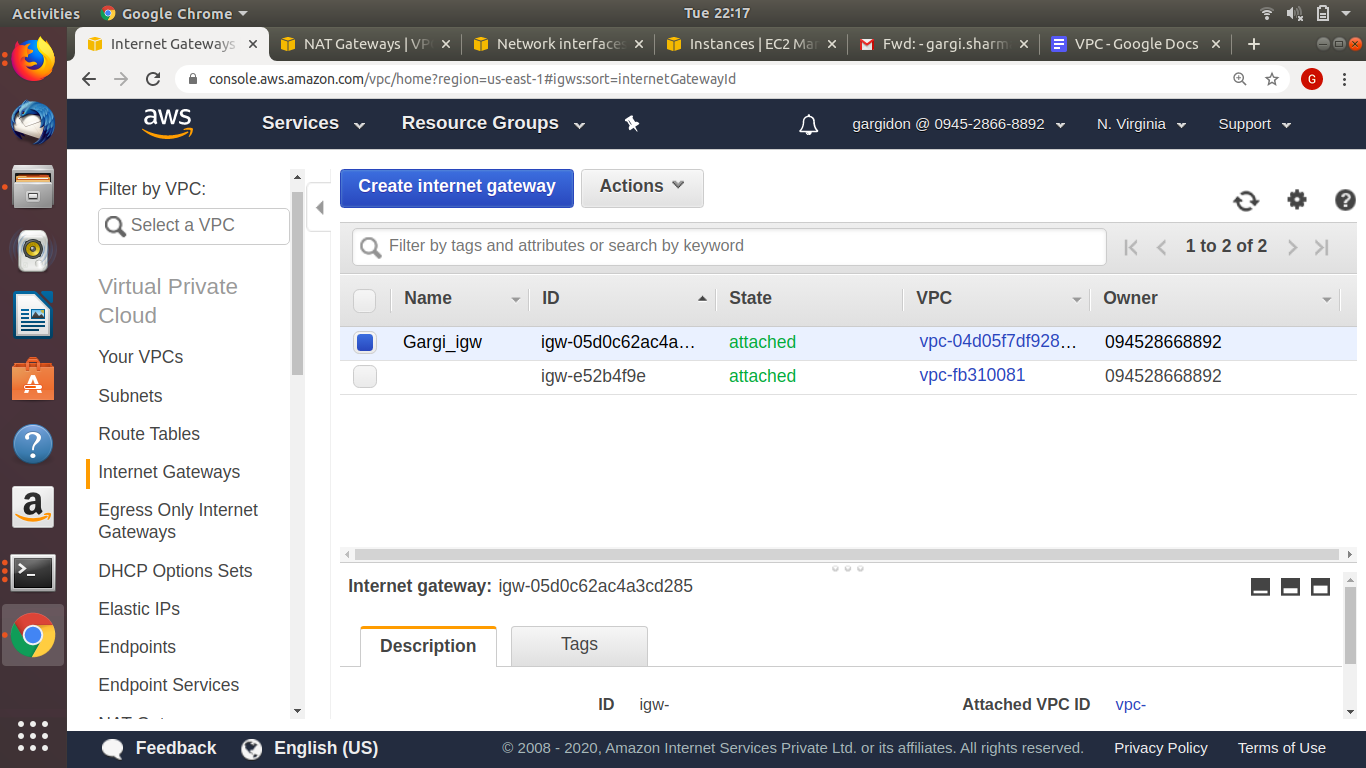
Step3: Create public subnet: CIDR(10.0.2.0/28)



Step:4 Create Internet Gateway

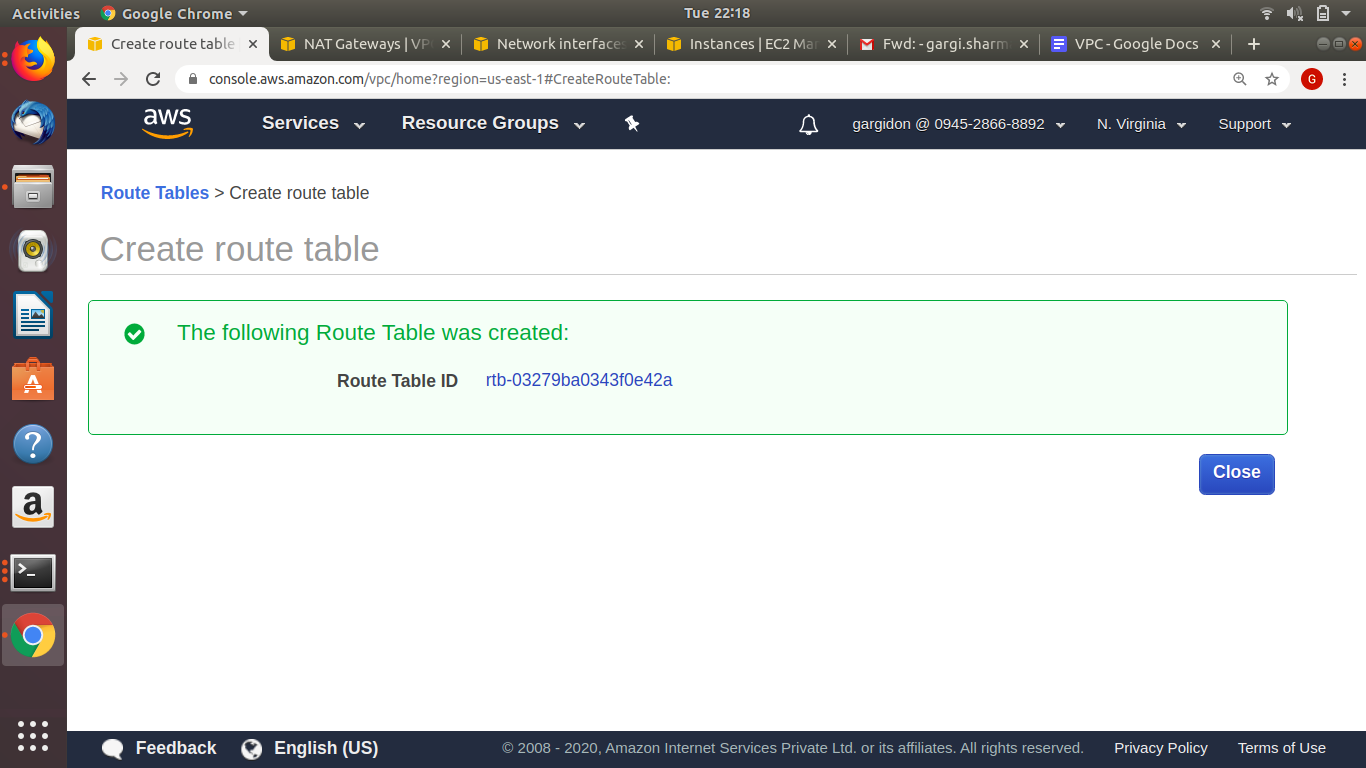


Step:5 Attach Internet Gateway to VPC

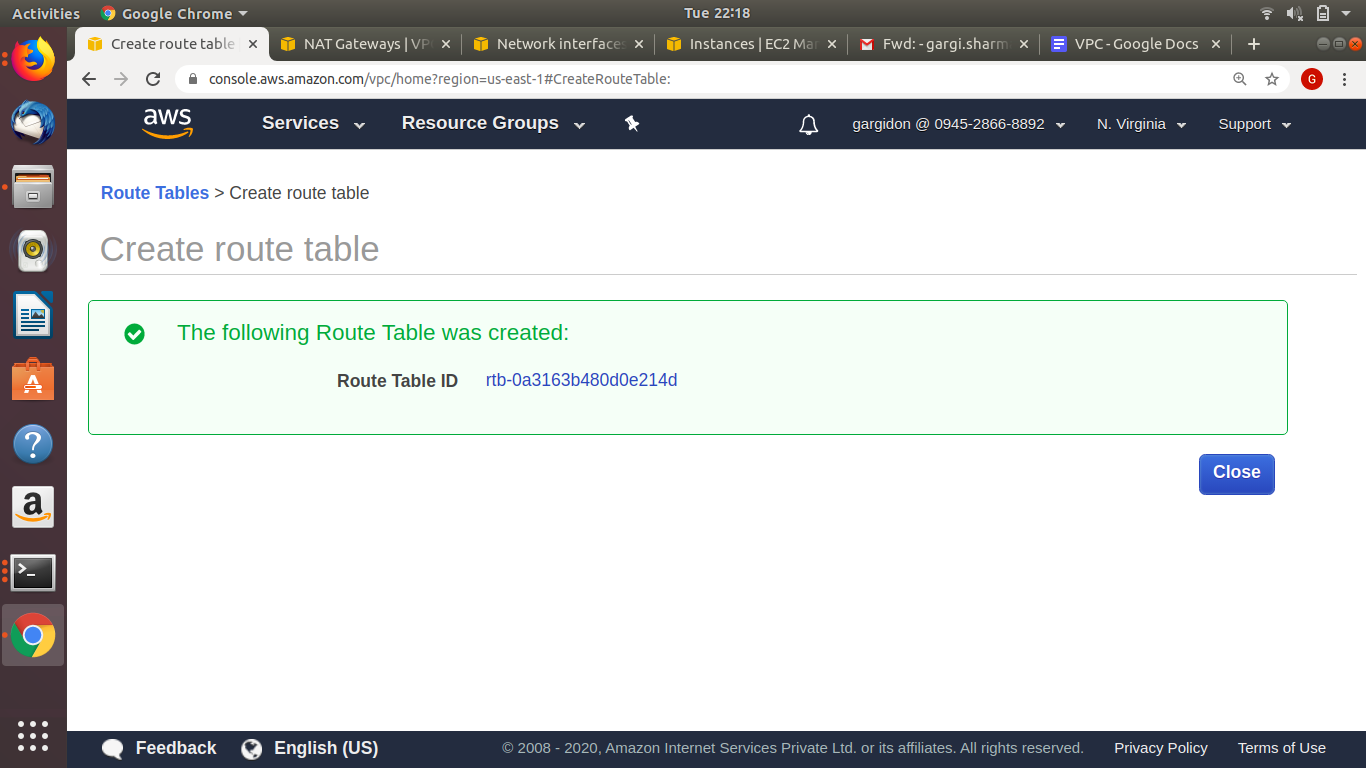


Step:6 Create route tables for public subnet

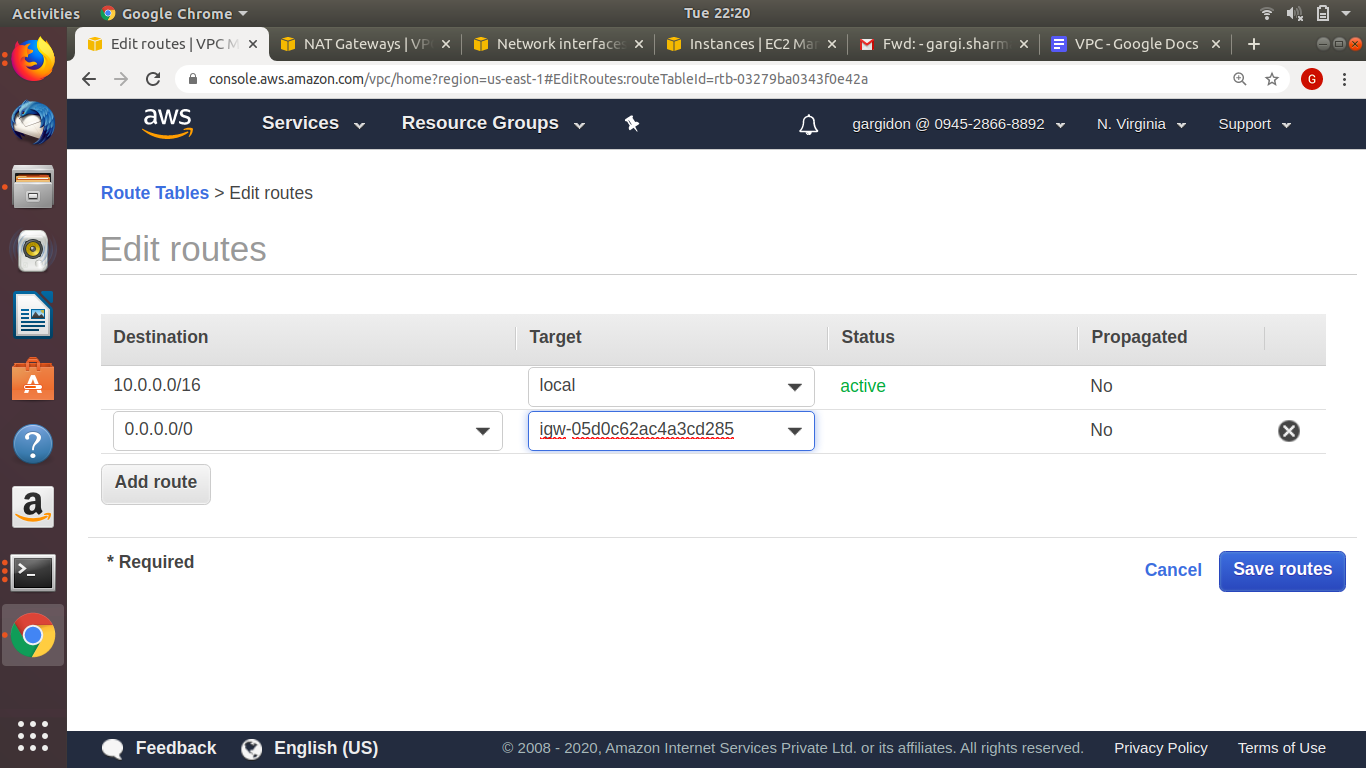
Route table for Public subnet



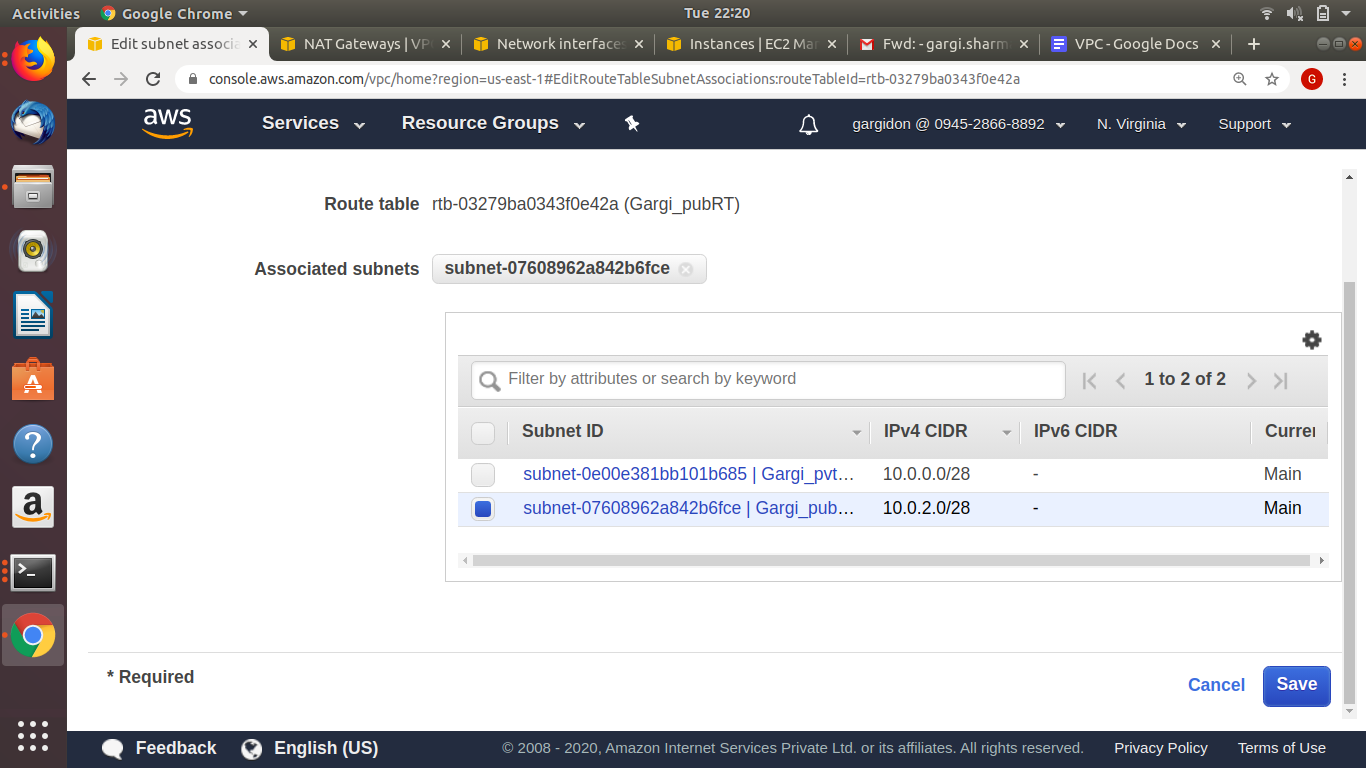
Route table for private subnet:



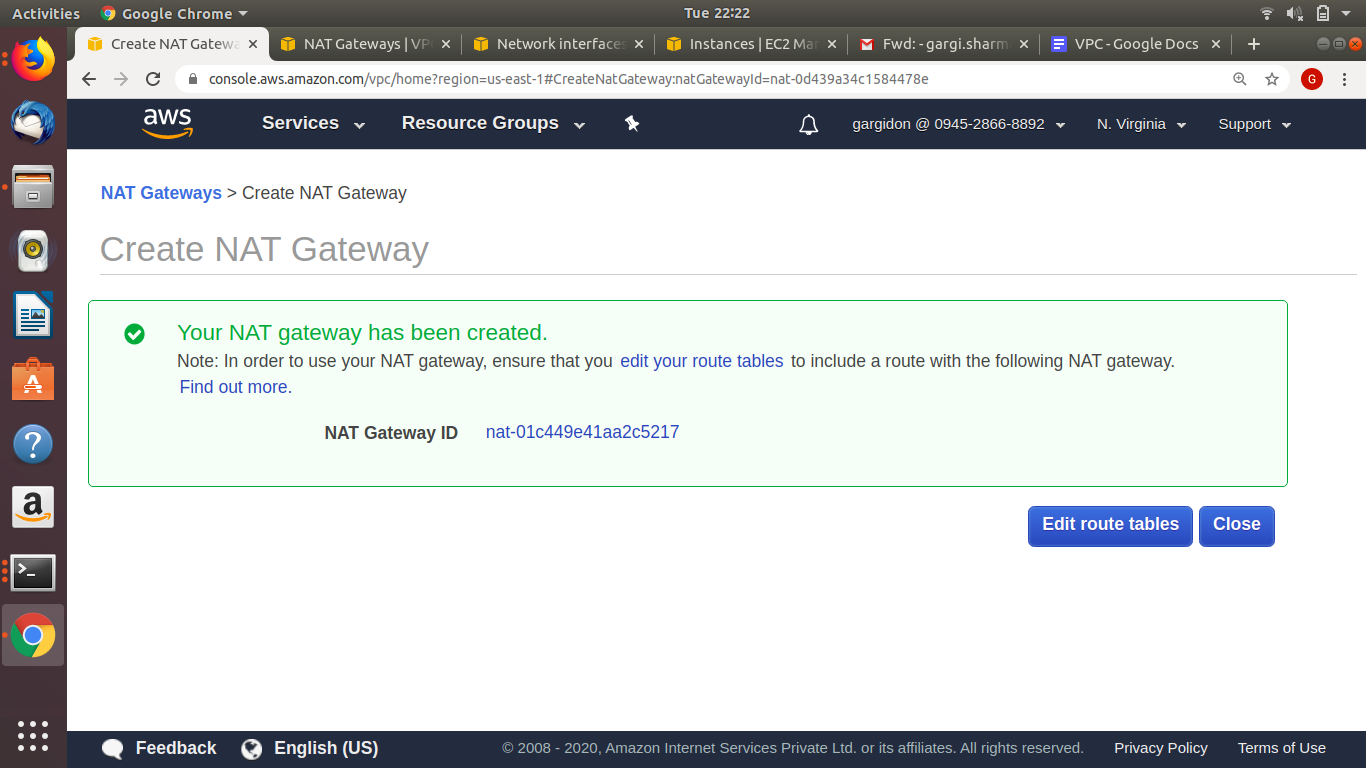
Step7: Edit the routes(for making publicly accessible,point it towards internet gateway)



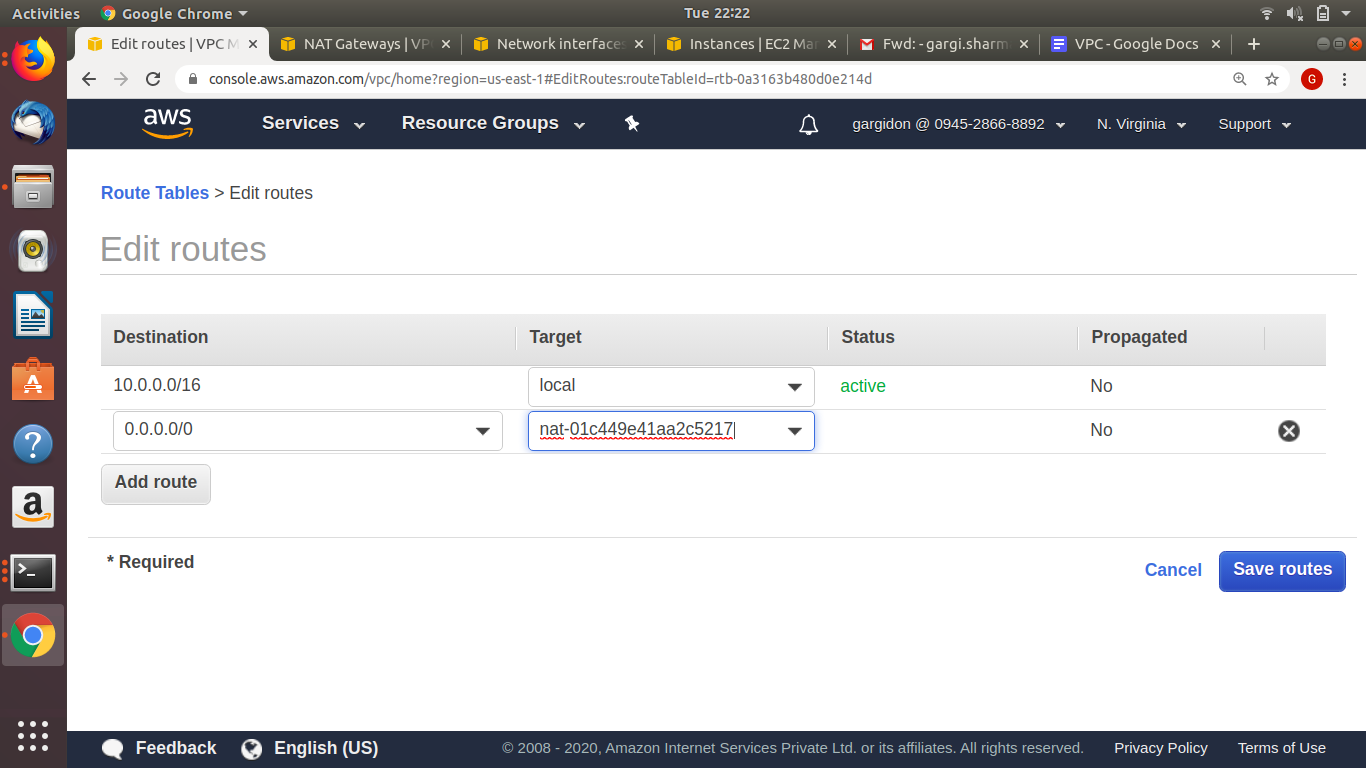
STEP 8: Associate public subnet with the public route table.



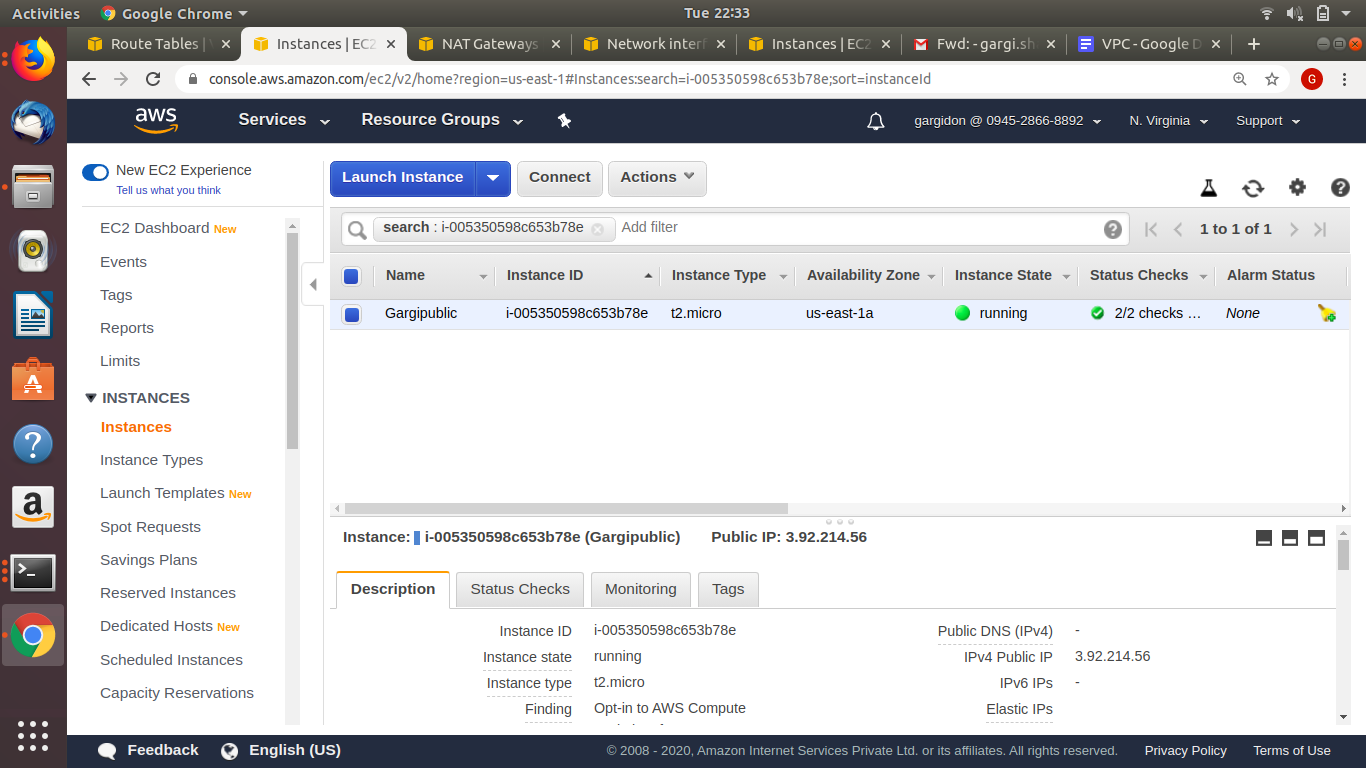
STEP9: create a NAT Gateway in public subnet



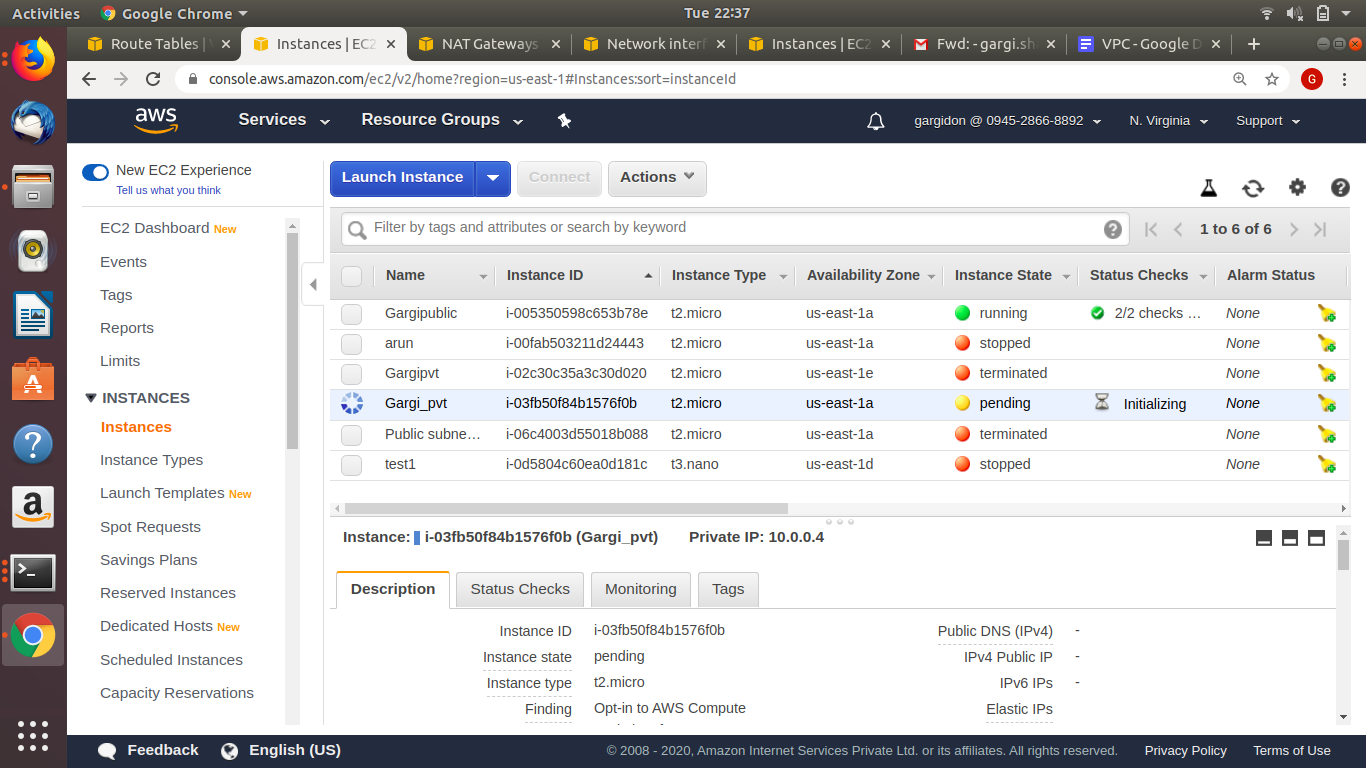
STEP10: Edit route table of pvt subnet and attach NAT gateway and associate private subnet to the route table



STEP11: Launch an instance in public subnet and specify user data for installation of nginx and enable auto assign IP

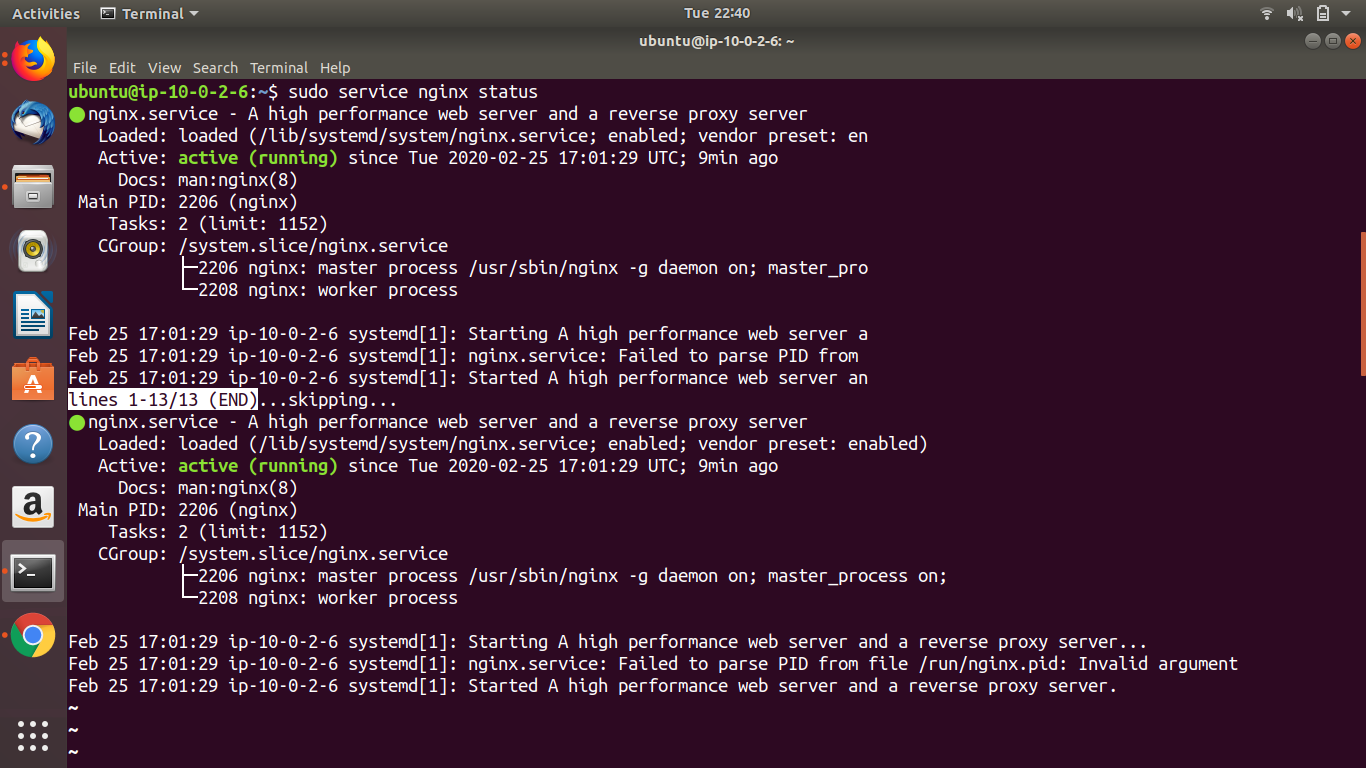


STEP12: Launch an instance in private subnet and disable public IP and install tomcat using user data



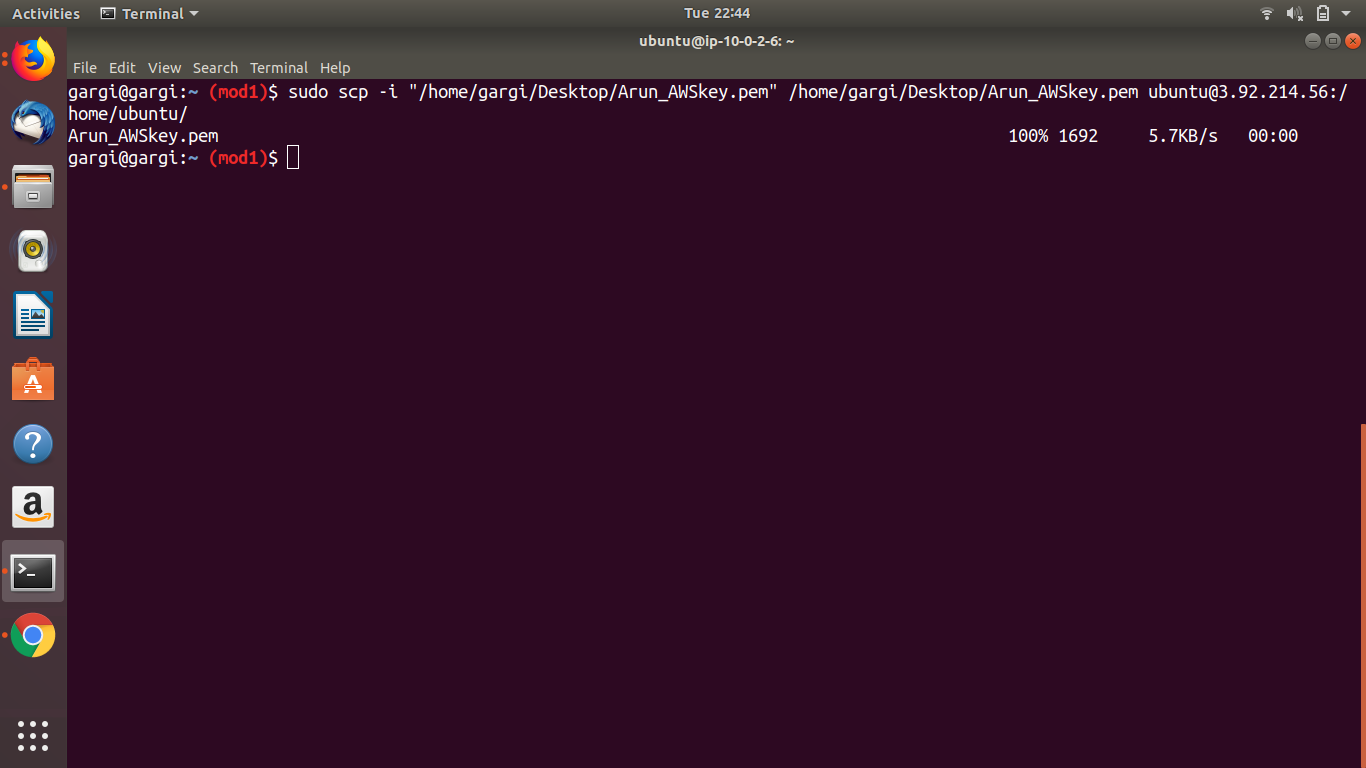
STEP13: Check if the private instance has tomcat9 installed and public instance has nginx installed.

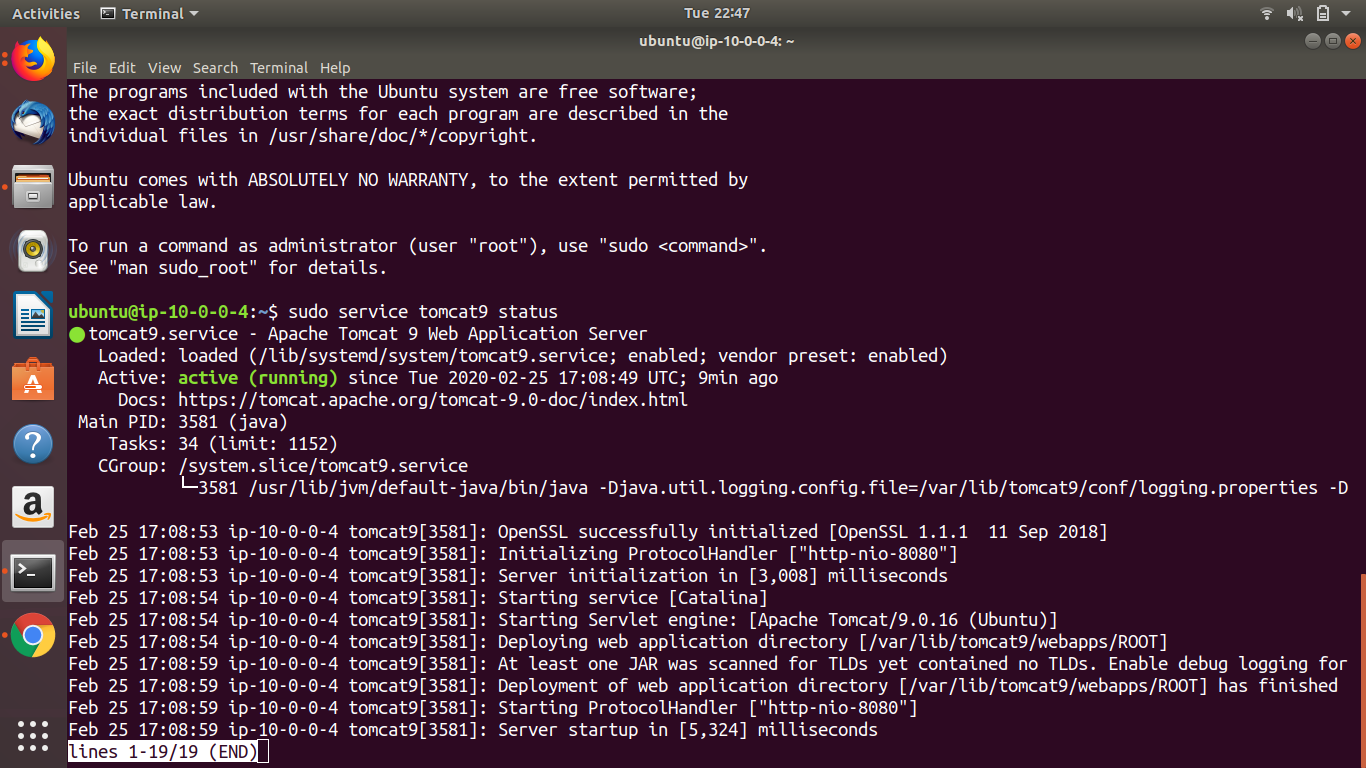
For public instance(nginx),ssh into it and check status:



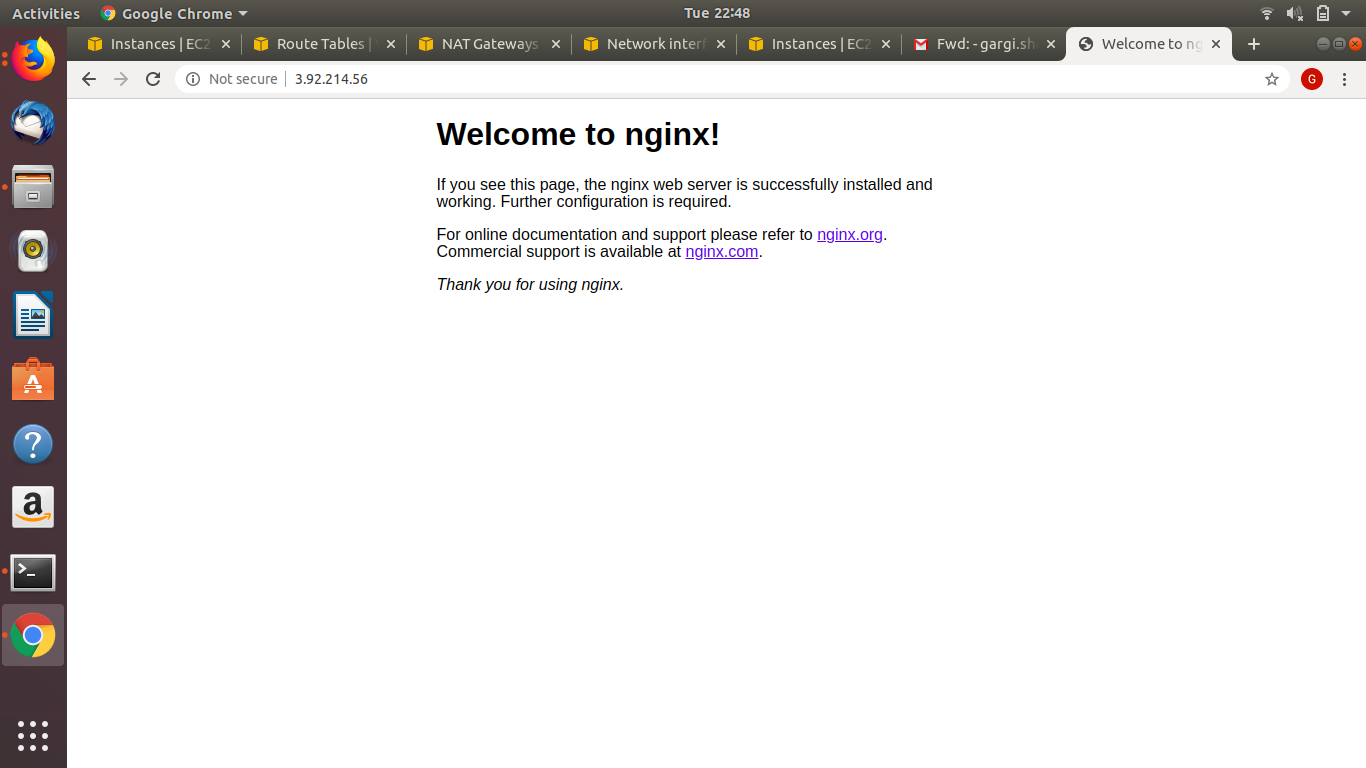
For private instance: scp the pem file to the public instance and then ssh from public instance to the private instance. Then check for tomcat9 status

Scp command: sudo scp -i “/home/gargi/Desktop/Arun\_AWSkey.pem” /home/gargi/Desktop/Arun\_AWSkey.pem ubuntu@3.92.214.56:/home/ubuntu/



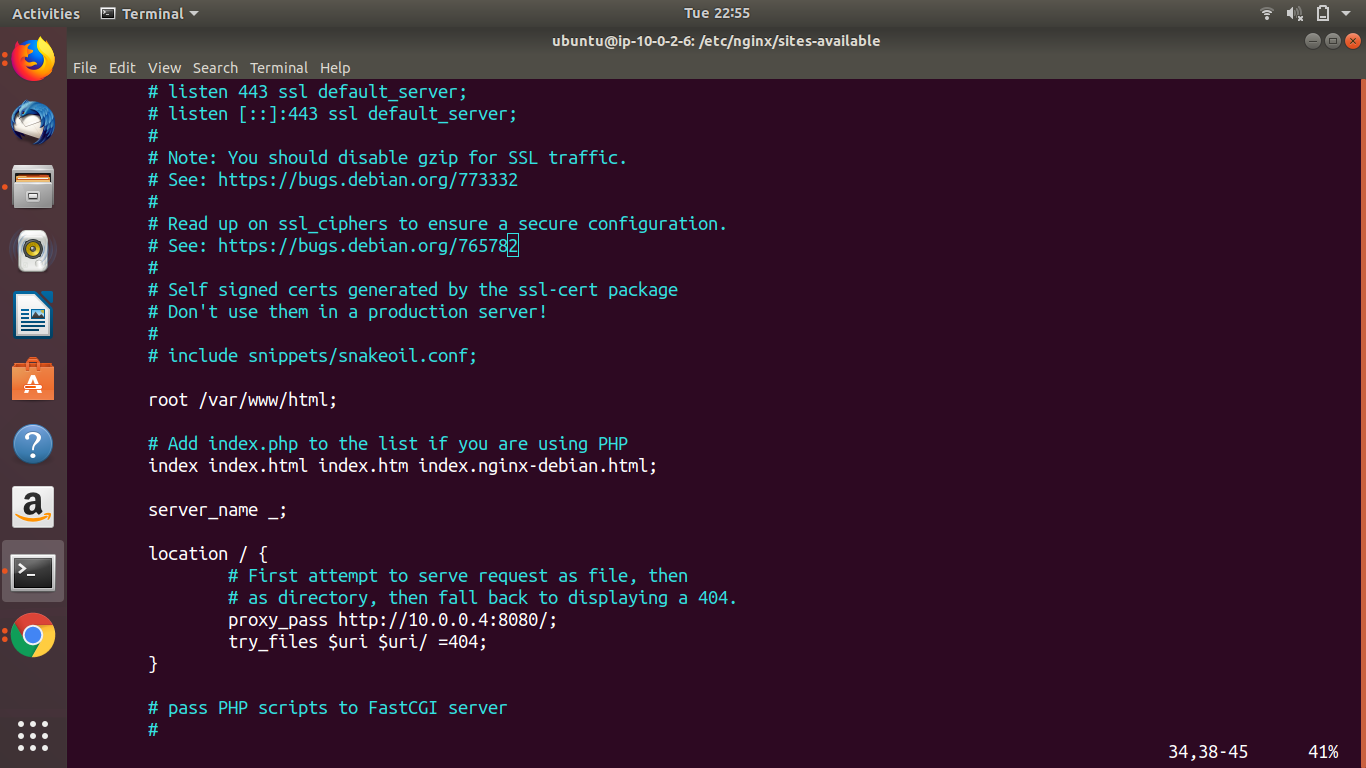


STEP14: Paste the public IP of your instance and check for the default nginx page



STEP15: Now login back to the public instance and add proxy\_pass in the location block of /etc/nginx/sites-available

**Proxy\_pass http://10.0.0.4:8080/;**



STEP16: Now reload nginx service and browse the public IP once again. This time it should display the tomcat9 page.

