

# ASSIGNMENT –1

Name: T. Vishnuvardhan

Mail : [thappetlavishnuvardhan9@gmail.com](mailto:thappetlavishnuvardhan9@gmail.com)

Batch No : 129

Phono: 8520081327

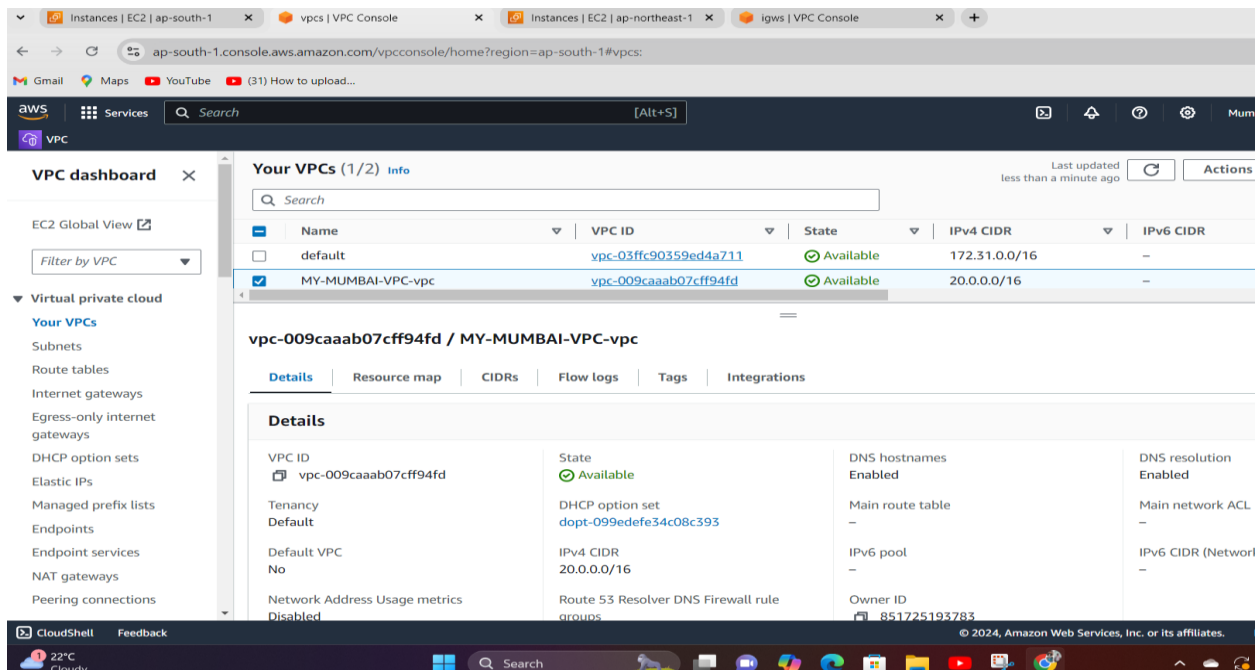
## **TASK:**

Create two VPC`s in different regions and connect the two VPC`s using peering.

Step 1: Create a VPC in 1 region [Mumbai]

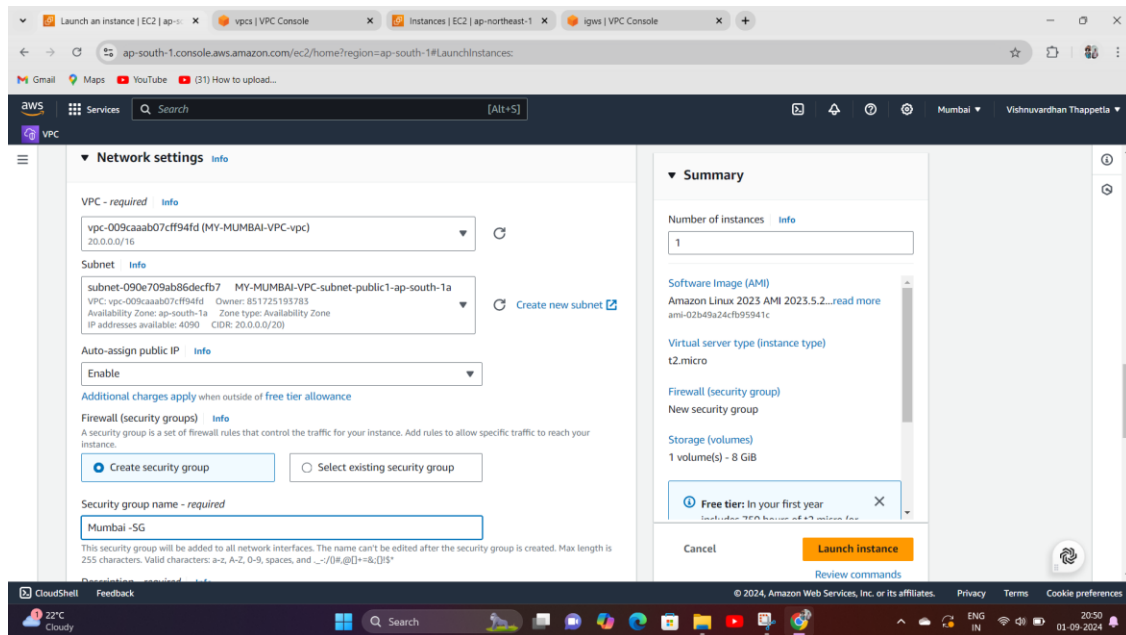
Select/fill options as

- \* VPC and More option
- \* Name of the VPC
- \* IPv4 CIDR block
- \* No of availability zons (2)
- \* 1 NAT Gateway (in 1 AZ)
- \* VPC endpoints NONE
- \* Click on Create VPC



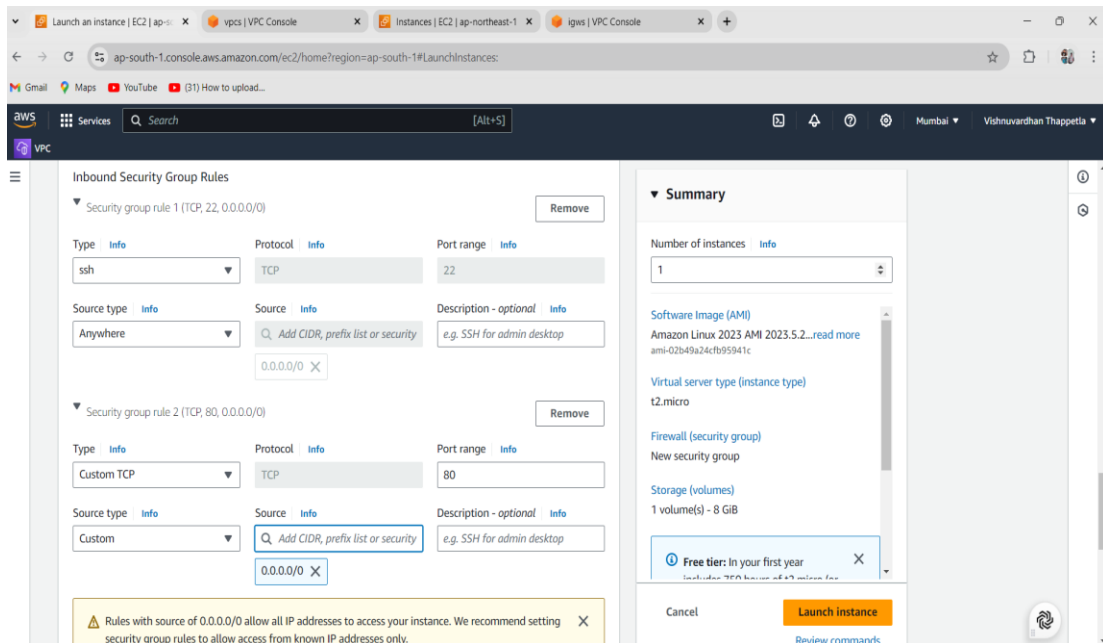
## Step 2: Create EC2 instance

- Name of instance
- Choose Amazon Linux
- Create a Key Pair and Downloaded file Paste it on Desktop
- Click on Network settings

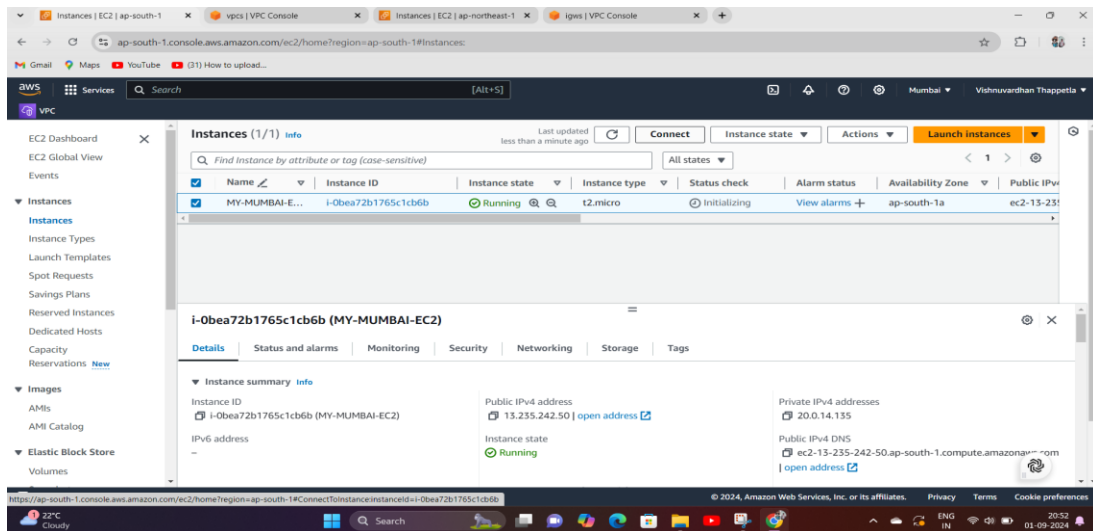


Step 3: From the above figure

- Select Created VPC
- Select Public Subnet
- Public IP Enable
- Create a Security Group
- Add Inbound Security Group Rules with “80 ” Port Range

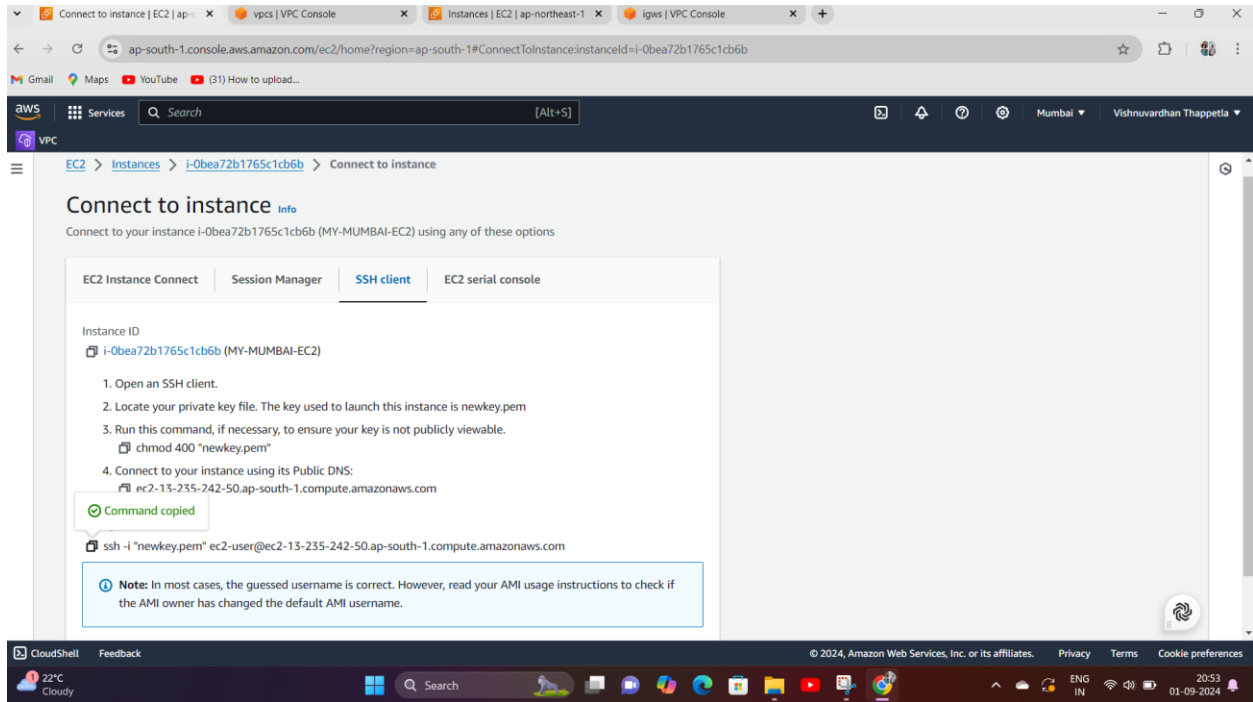


- Click on Launch Instance



#### Step 4:

- Connect to EC2 Instance Through SSH Client Link
- Copy the SSH Client Link



#### Step 5:

- Back to Desktop
- Open GitBash and
- Paste the SSH Client Link
- Press ENTER Button
- Type “Yes “
- Now you are connected to EC2 Instance



```
root@ip-20-0-14-135:/usr/share/nginx/html
Verifying      : nginx-1:1.24.0-1.amzn2023.0.2.x86_64      4/7
Verifying      : nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64 5/7
Verifying      : nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch 6/7
Verifying      : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 7/7

Installed:
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
libunwind-1.4.0-5.amzn2023.0.2.x86_64
nginx-1:1.24.0-1.amzn2023.0.2.x86_64
nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

complete!
[root@ip-20-0-14-135 ~]# cd /usr/share/nginx/html
[root@ip-20-0-14-135 html]# ls
404.html  50x.html  icons  index.html  nginx-logo.png  poweredby.png
[root@ip-20-0-14-135 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-20-0-14-135 html]# vi index.html
[root@ip-20-0-14-135 html]# systemctl restart nginx
[root@ip-20-0-14-135 html]# curl 20.0.14.135
This message from Mumbai
[root@ip-20-0-14-135 html]# |
```

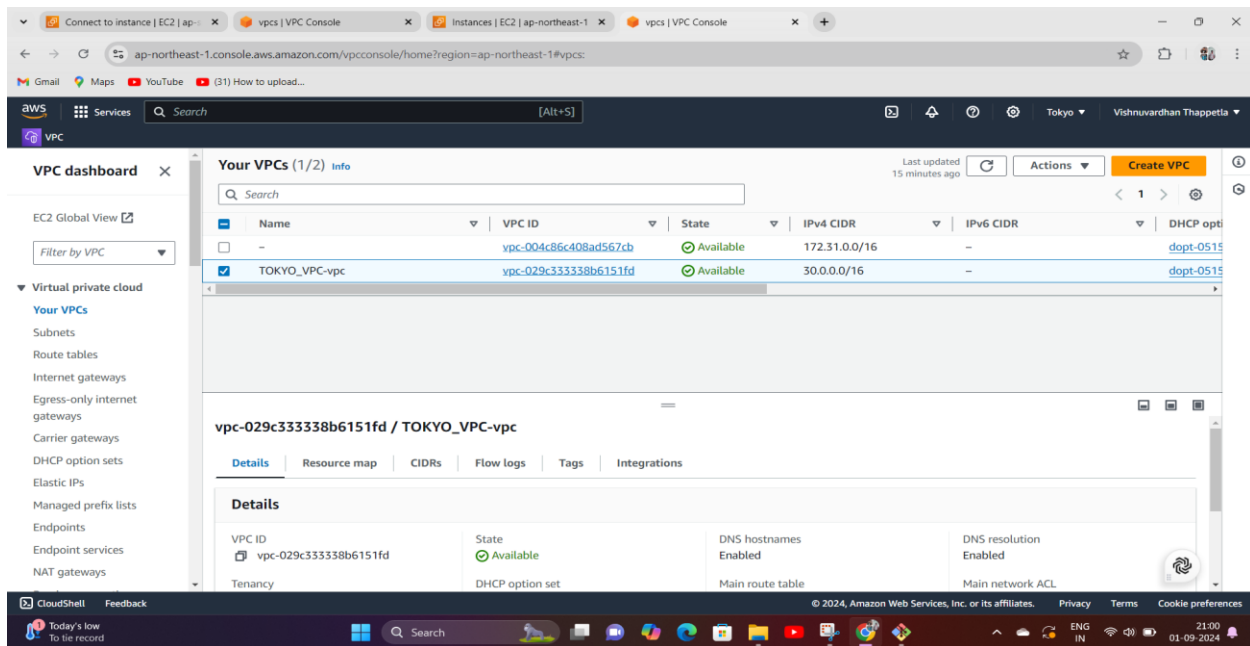
Step 7: Now use the “top” command to not hang the connection and Display all running process

```
root@ip-20-0-14-135:/usr/share/nginx/html
top - 15:29:36 up 7 min, 2 users, load average: 0.01, 0.11, 0.08
Tasks: 107 total, 1 running, 106 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.3 us, 2.3 sy, 0.0 ni, 95.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 949.5 total, 523.9 free, 130.8 used, 294.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 676.5 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 25707 root        20   0   223908    3348    2692 R   0.3   0.3   0:00.02 top
    1 root        20   0   105536   16936   10156 S   0.0   1.7   0:01.08 systemd
    2 root        20   0         0         0         0 S   0.0   0.0   0:00.00 kthreadd
    3 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_gp
    4 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_par_gp
    5 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 slub_flushwq
    6 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 netns
    7 root        20   0         0         0         0 I   0.0   0.0   0:00.02 kworker/0:0-c+
    8 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 kworker/0:0H-+
    9 root        20   0         0         0         0 I   0.0   0.0   0:00.11 kworker/u30:0+
   10 root         0 -20         0         0         0 I   0.0   0.0   0:00.00 mm_percpu_wq
   11 root        20   0         0         0         0 I   0.0   0.0   0:00.00 rcu_tasks_kth+
   12 root        20   0         0         0         0 I   0.0   0.0   0:00.00 rcu_tasks_rud+
   13 root        20   0         0         0         0 I   0.0   0.0   0:00.00 rcu_tasks_tra+
   14 root        20   0         0         0         0 S   0.0   0.0   0:00.09 ksoftirqd/0
   15 root        20   0         0         0         0 I   0.0   0.0   0:00.06 rcu_preempt
   16 root        rt    0         0         0         0 S   0.0   0.0   0:00.00 migration/0
   17 root        20   0         0         0         0 I   0.0   0.0   0:00.00 kworker/0:1-c+
```

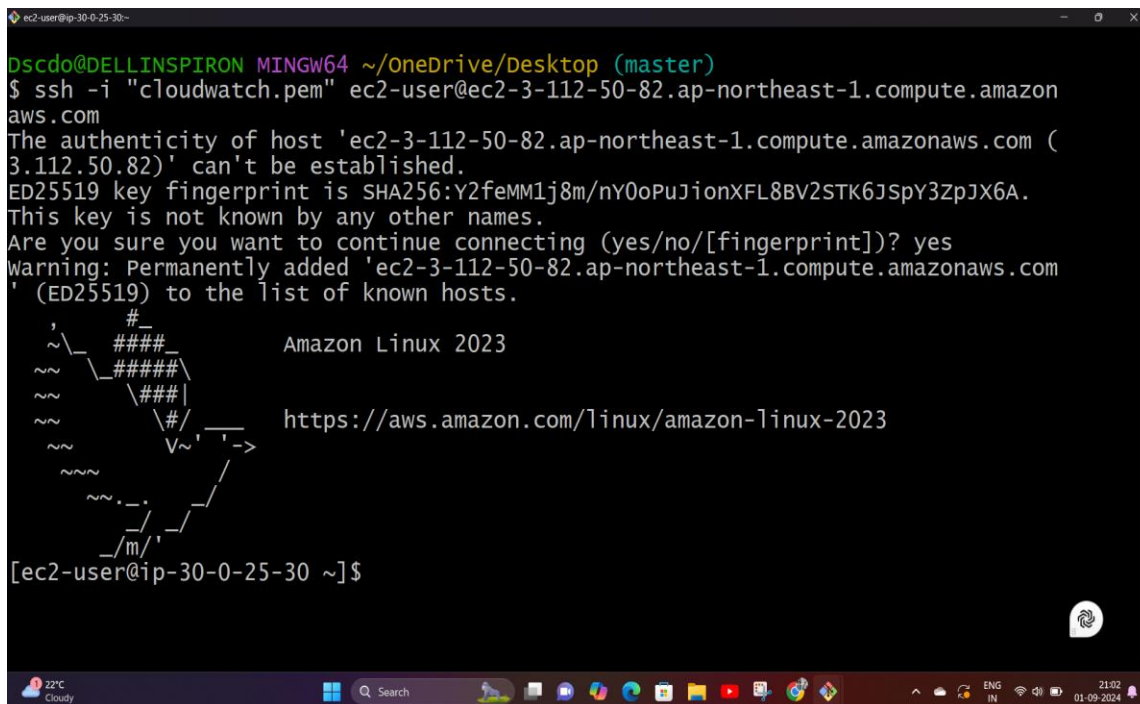
Step 8: Now goto another region and create a VPC. So, I Created in TOKYO region

- Follow the Step 1



## Step 9: Now Create a EC2 instance in Tokyo region

- Follow the Step 2, 3, 4, 5.

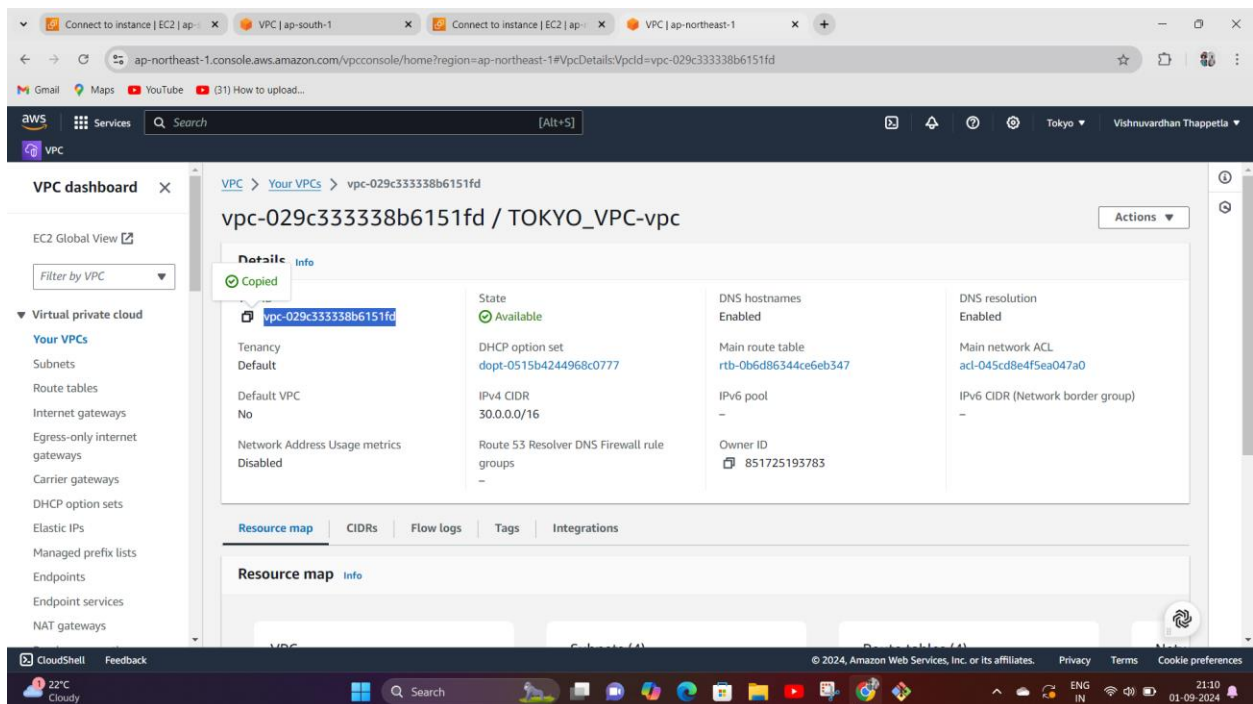


Step 10: Now follow the Step 6 but change the file content.

(This message from Tokyo) and follow Step 7

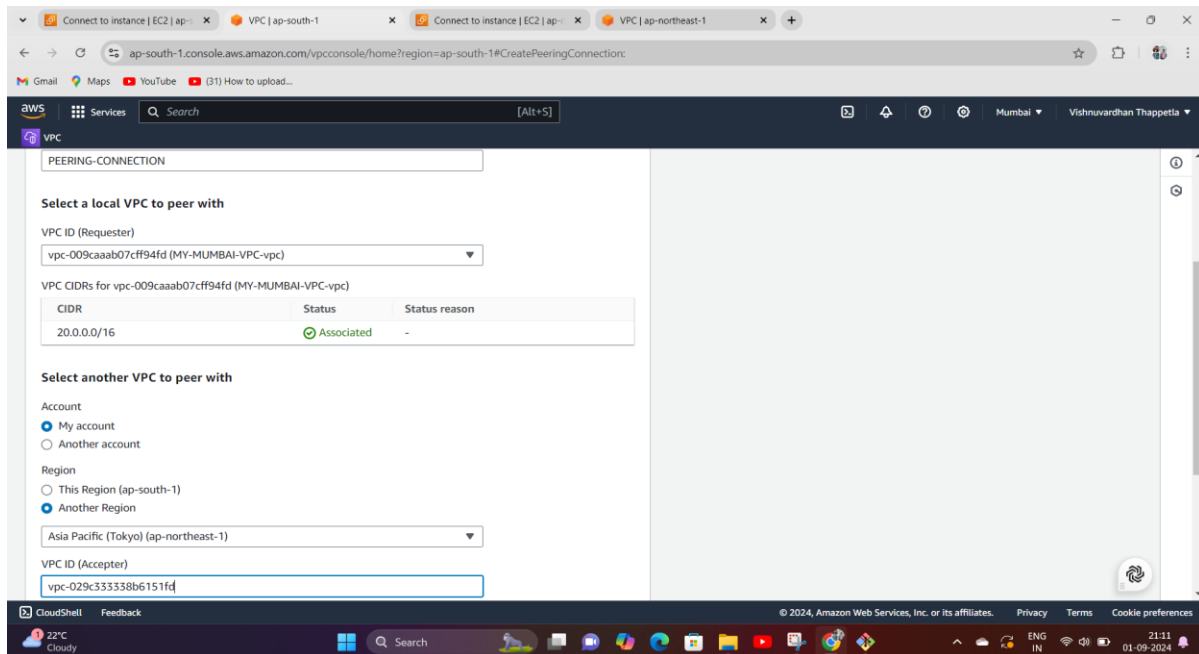
```
root@ip-30-0-25-30 html]# rm index.html
rm: remove regular file 'index.html'? yes
root@ip-30-0-25-30 html]# vi index.html
root@ip-30-0-25-30 html]# systemctl restart nginx
root@ip-30-0-25-30 html]# curl 30.0.25.30
This message from Tokyo
root@ip-30-0-25-30 html]#
```

Step 11: Now Back to Tokyo region VPC and copy the VPC ID

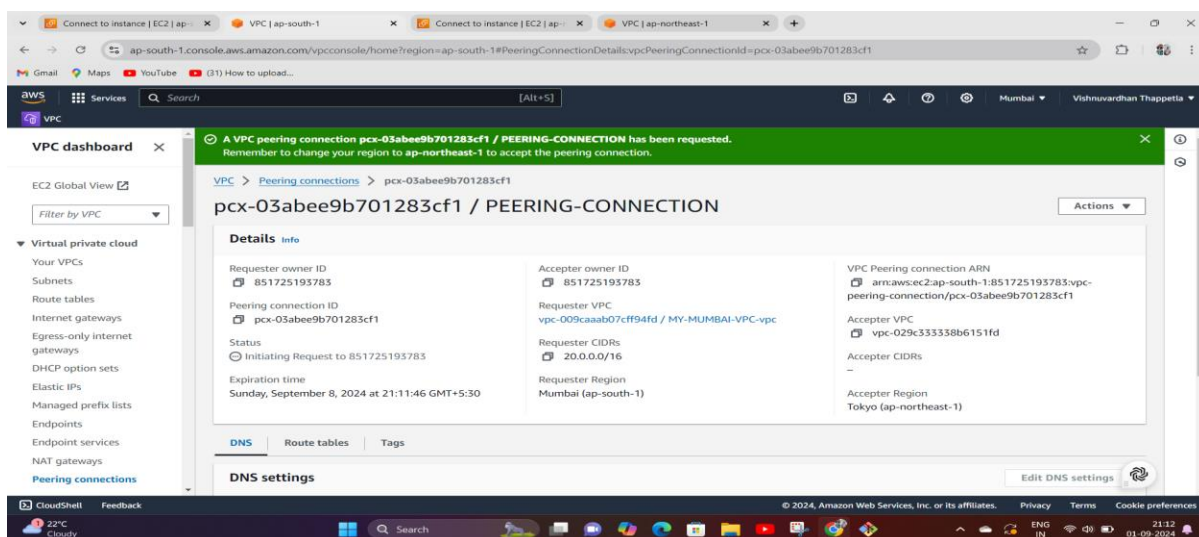




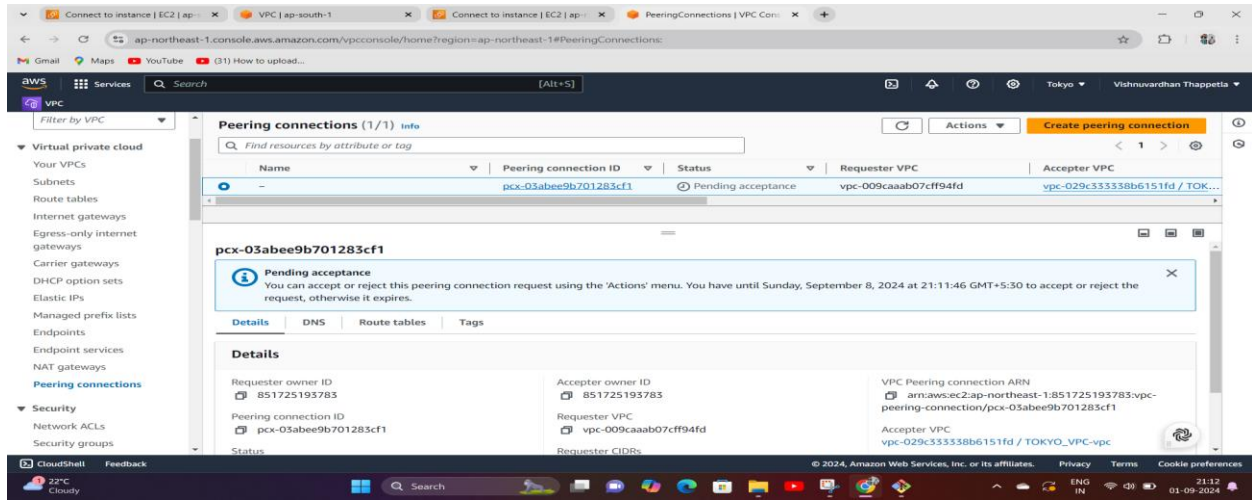
Step 12: Now create a peering connection in any one of the regions. So, i created in Mumbai region.



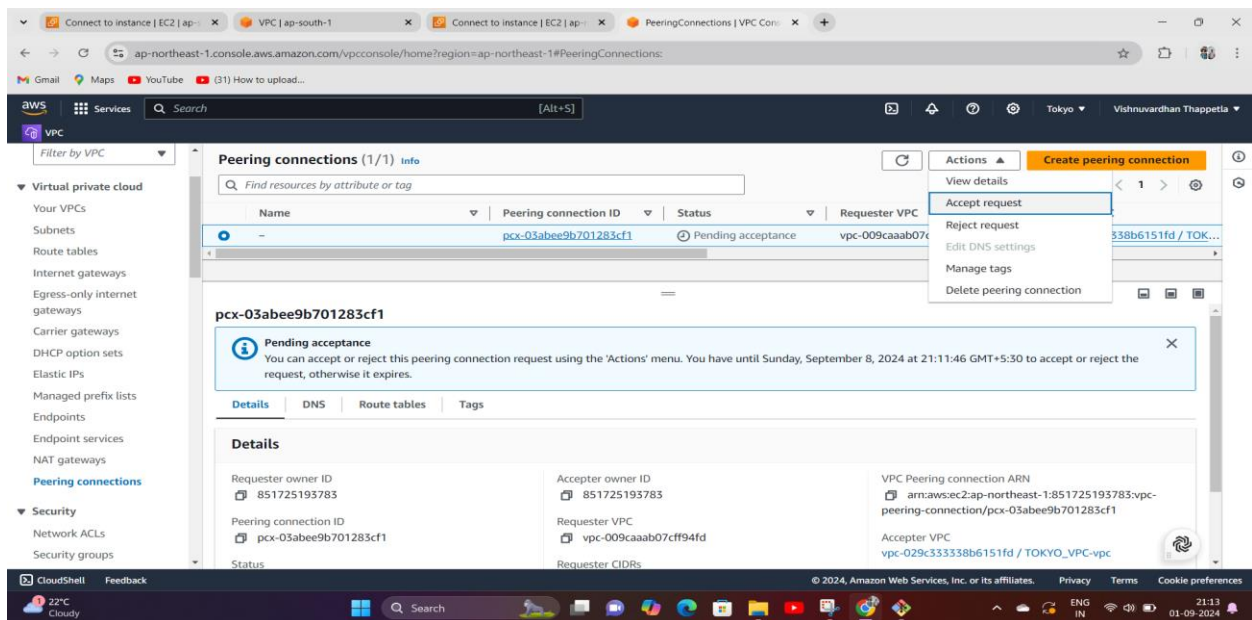
- While creating a peering connection VPC ID Requester give Mumbai VPC
- Select another VPC to peer with click on “another region”
- Add region
- Add VPC ID in Acceptor
- Now click on create Peering connection
- Now you will see Initiating Request



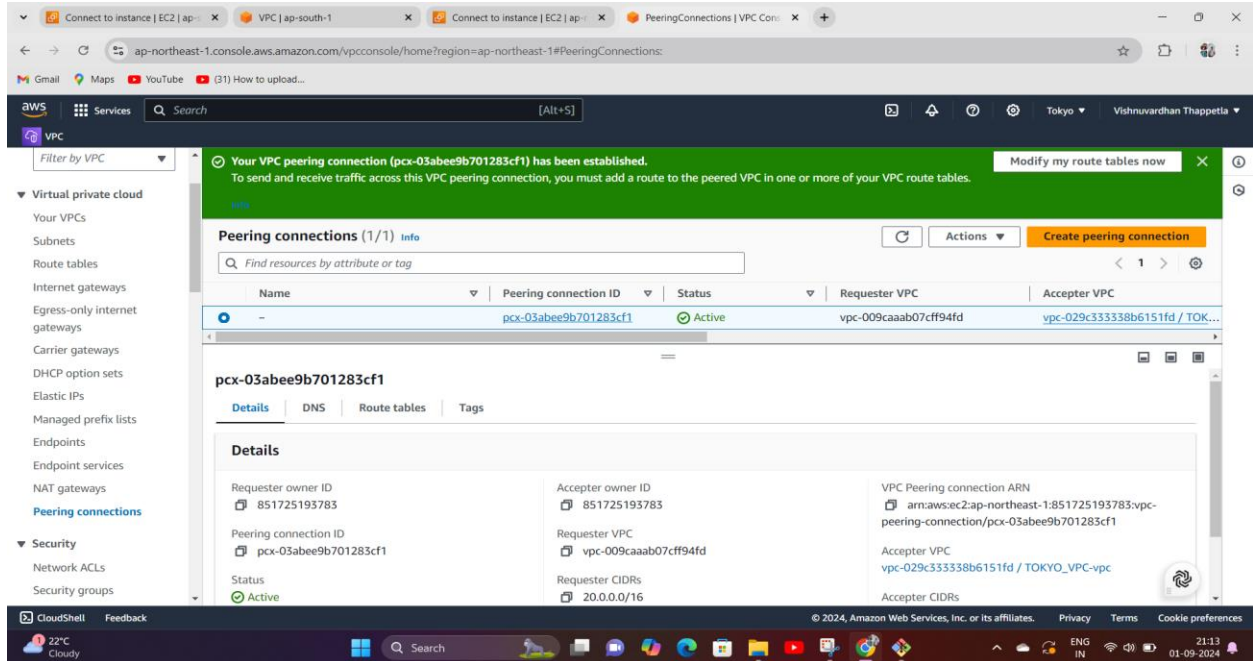
Step 13: Now Back to Another region “TOKYO” check whether the peeing connection request have been come or not.



- Now go to “Actions” and accept the request

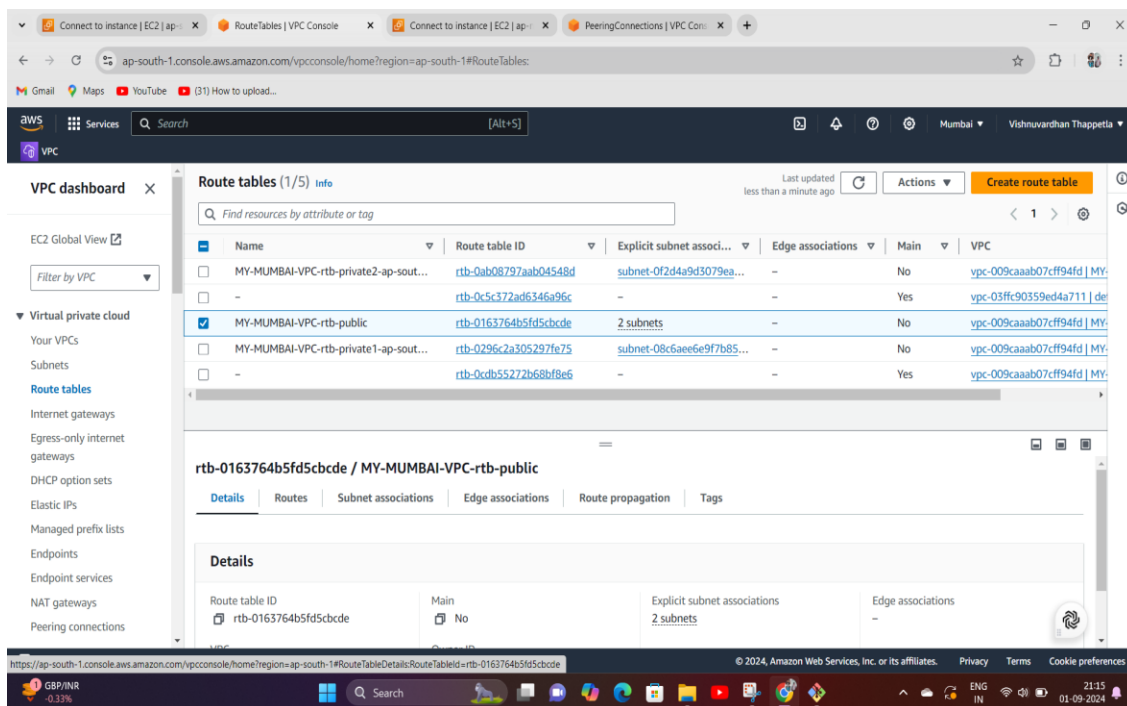


- After accepting the Peering connection will be active in both regions after refreshing

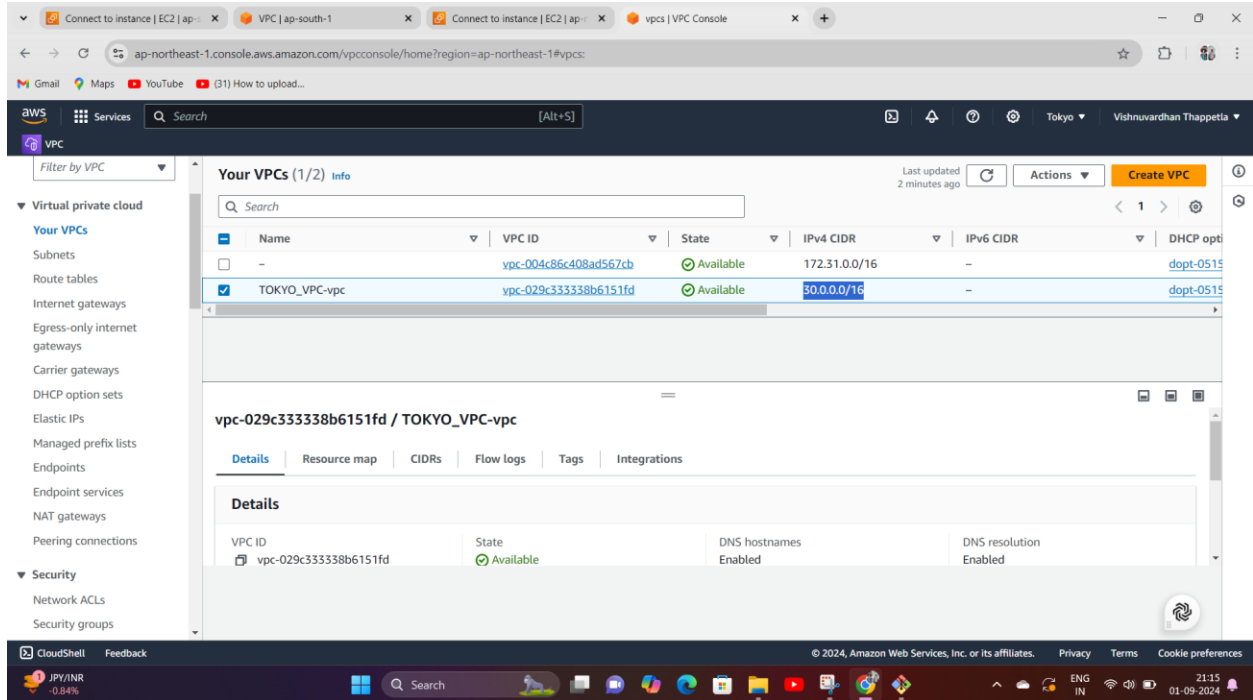


Step 14: Now back to Mumbai region "Open Public route table ID"

- Goto Actions and click edit route

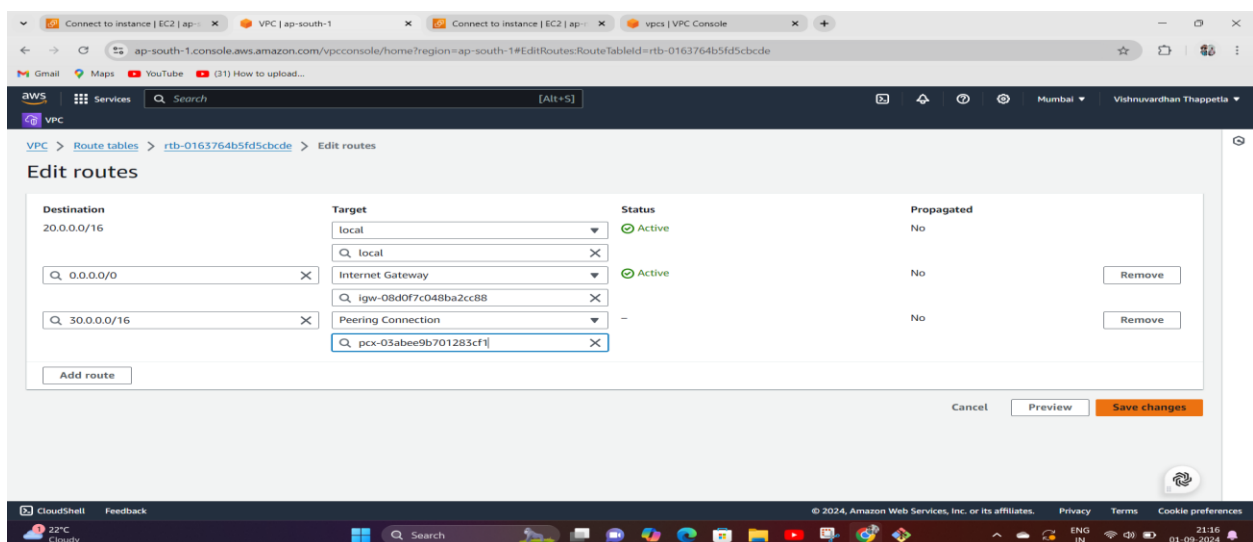


Step 15: Back to TOKYO region VPC and Copy the IPV4 CIDR Address [30.0.0.0/16]

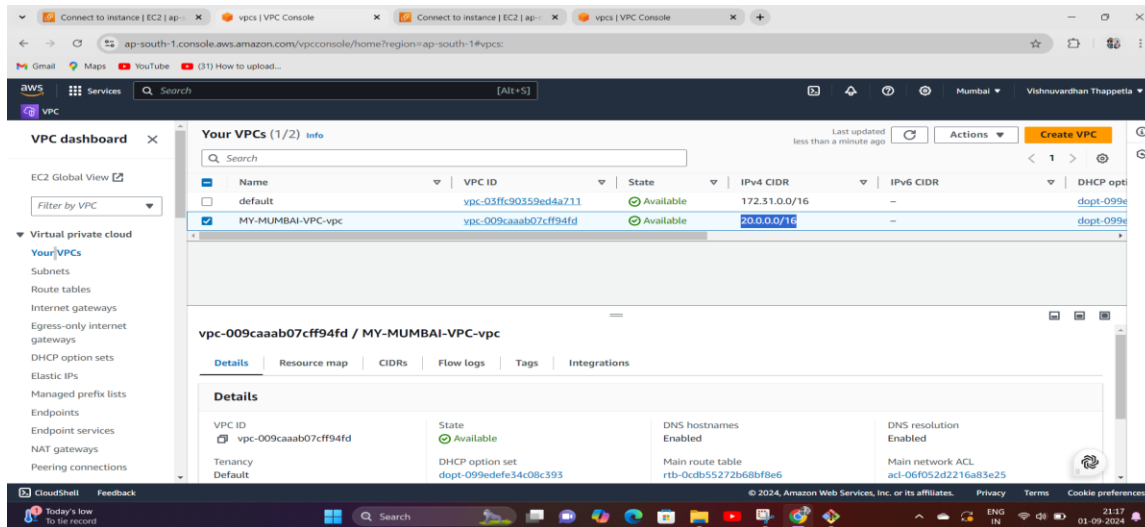


Step 16: Back to Mumbai region public route table ID in edit route

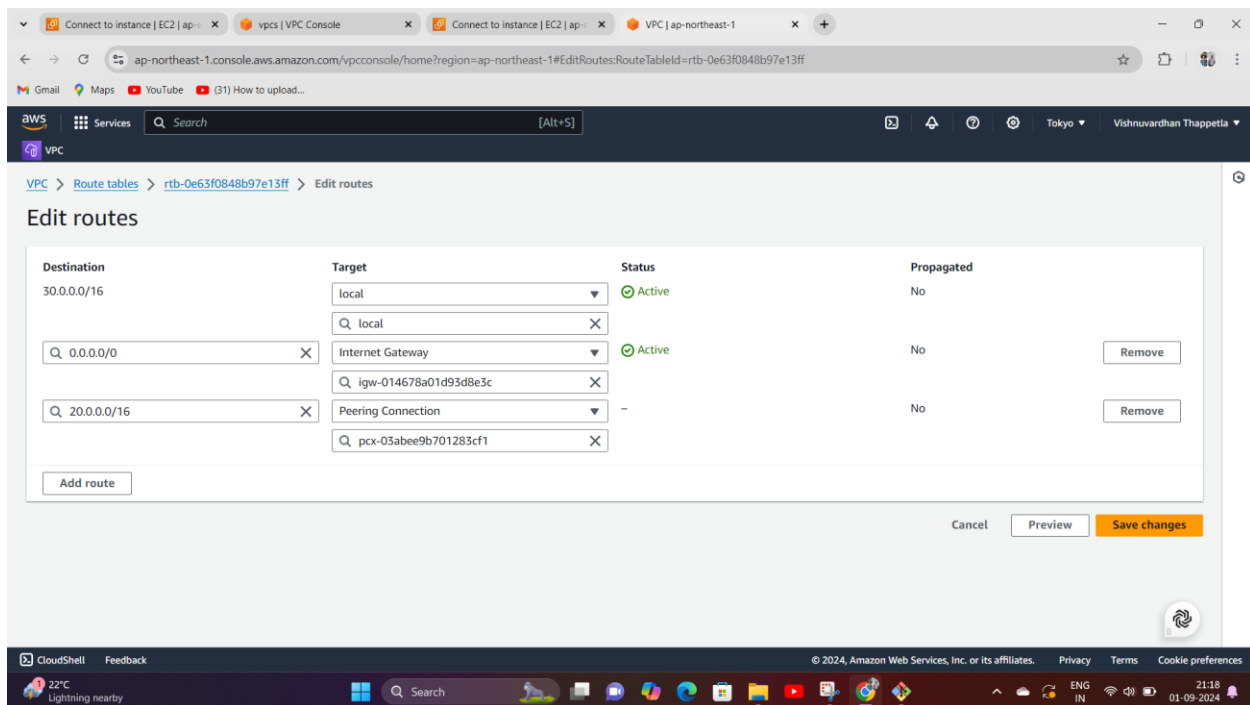
- Add route
- Paste Tokyo IPV4 CIDR Address
- Give Peering Connection and below it gives Peering connection ID
- Save changes



Step 17: Back to Mumbai VPC and Copy the IPV4 CIDR address [20.0.0.0/16]



Step 18: Goto Tokyo region does same as Step 16 but paste the Mumbai copied IPV4 CIDR address [20.0.0.0/16].



- Now we are interchanged IPV4 CIDR Address to establish the connection between two VPC's using Peering connection. [Mumbai & Tokyo]

Step 19: Now goto Git bash of Mumbai server clicks [ctrl + c] to halts the current command.

- Paste the Tokyo IPV4 address

- Using this CURL command

```

root@ip-20-0-14-135 html]# curl 20.0.14.135
This message from Mumbai
root@ip-20-0-14-135 html]# curl 30.0.25.30
This message from Tokyo
root@ip-20-0-14-135 html]#

```

Step 20: Goto gitbash of Tokyo server.

- Click ctrl + c
- Paste the Mumbai IPV4 address
- Using “curl” command

```

root@ip-30-0-25-30:/usr/share/nginx/html
PID USER PR NI VIRT RES
1 root 20 0 105652 16960
2 root 20 0 0 0
3 root 0 -20 0 0
4 root 0 -20 0 0
5 root 0 -20 0 0
6 root 0 -20 0 0
8 root 0 -20 0 0
9 root 20 0 0 0
10 root 0 -20 0 0
11 root 20 0 0 0
12 root 20 0 0 0
13 root 20 0 0 0
14 root 20 0 0 0
15 root 20 0 0 0
16 root rt 0 0 0
18 root 20 0 0 0
20 root 20 0 0 0
21 root 0 -20 0 0
root@ip-30-0-25-30 html]# curl 30.0.25.30
This message from Tokyo
root@ip-30-0-25-30 html]# curl 20.0.14.135
This message from Mumbai
root@ip-30-0-25-30 html]#

```

- From the above steps we have successfully created two VPC`s in different regions and connected the two VPC`s using Peering Connection.
- By creating peering connection between these two VPC`s in different regions they can communicate easily.

Step 21:

- Now delete the peering connection.
- Terminate the EC2 instances.
- Remove the route tables Sub association connection and route connections
- Delete NAT gateway
- Delete Internet Gateway
- Delete Route tables
- Delete subnets
- Delete VPC`s.