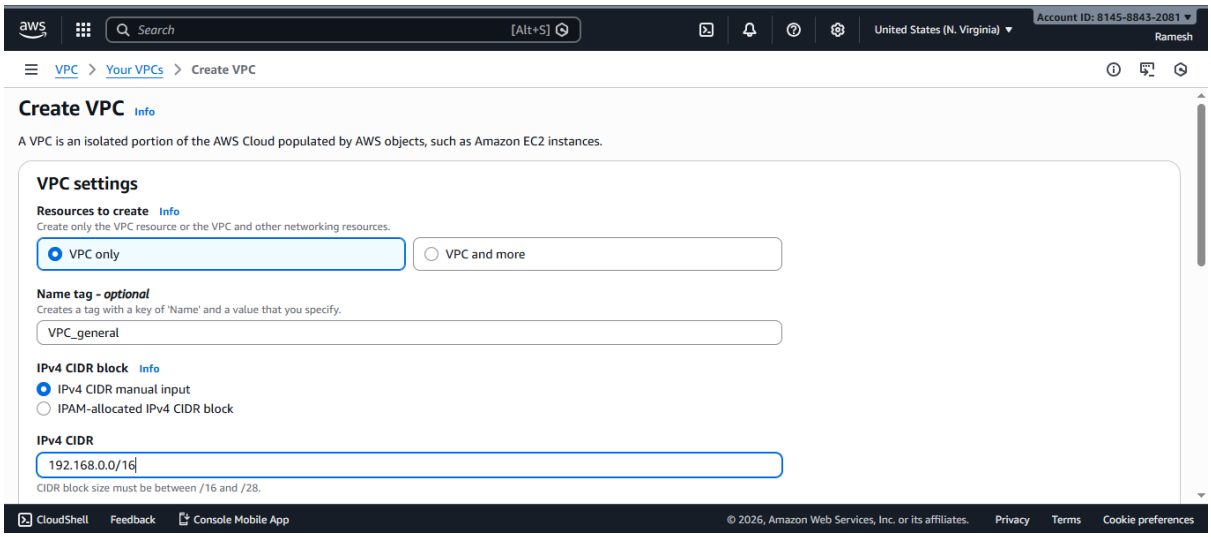
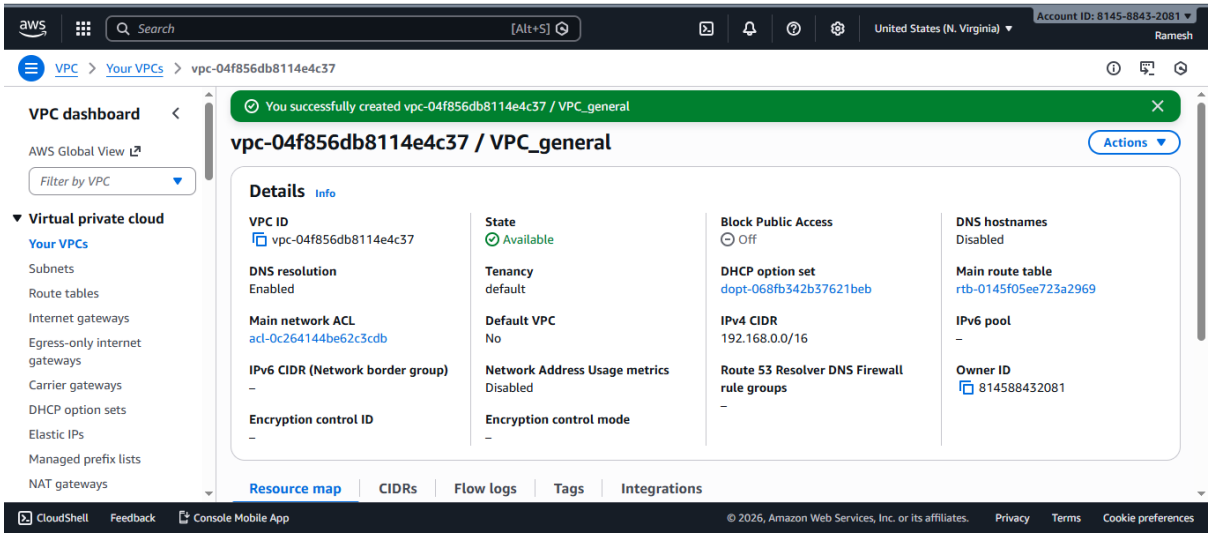
VPC-TASK2

1. Create one VPC, with 1 public subnet and 1 private subnet.

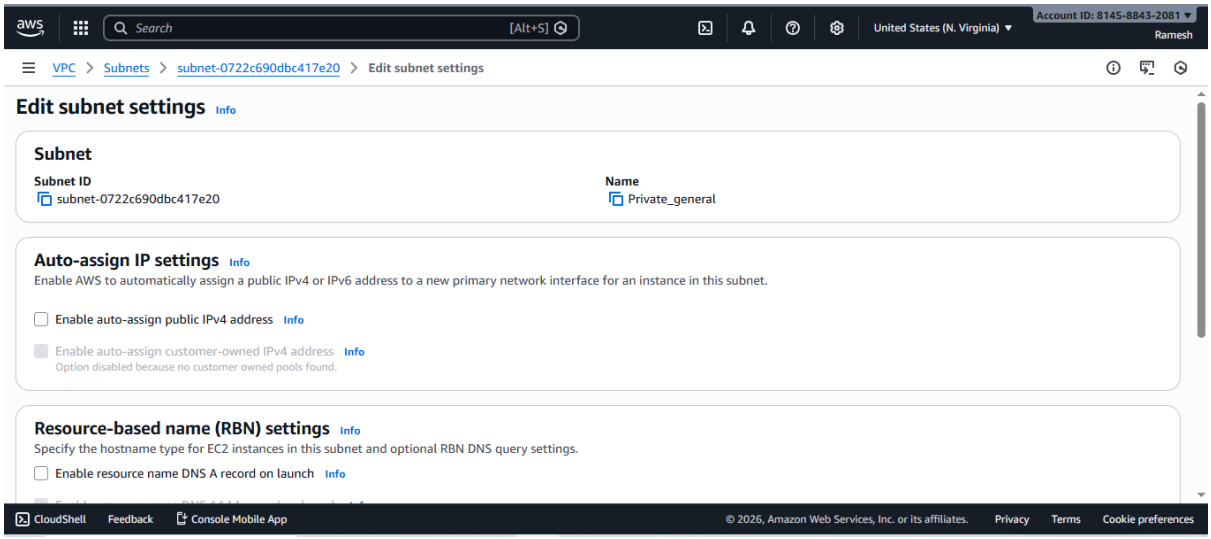


- Search for VPC and create VPC
- VPC setting →VPC only
- Select IPV4 CIDR manual input
- IPV4 CIDR as 192.168.0.0/16
- Click on create CIDR.

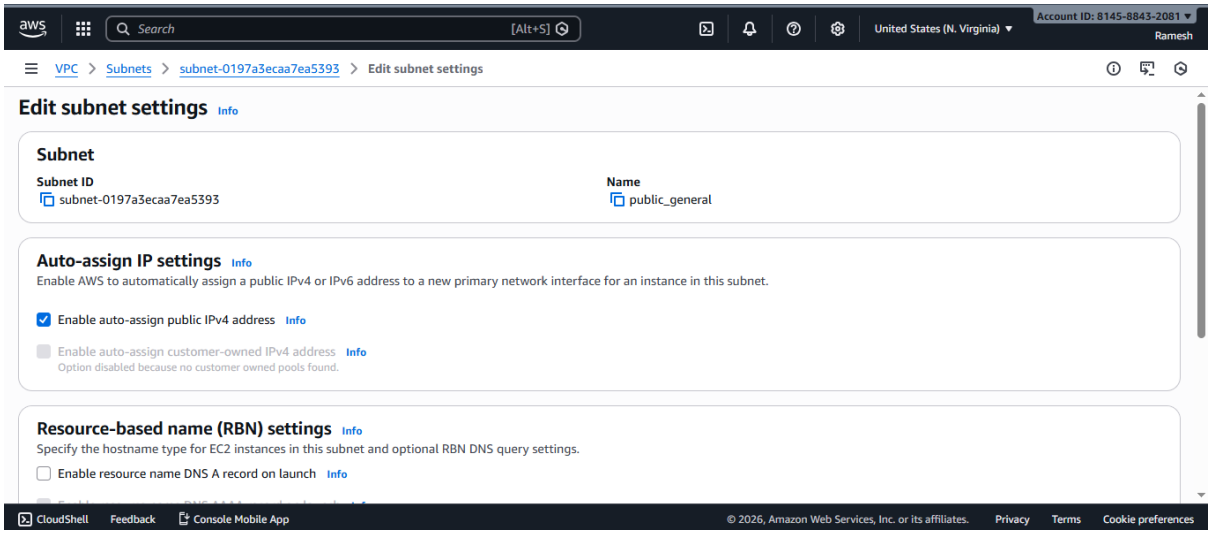


- It shows successfully created VPC.

VPC-TASK2



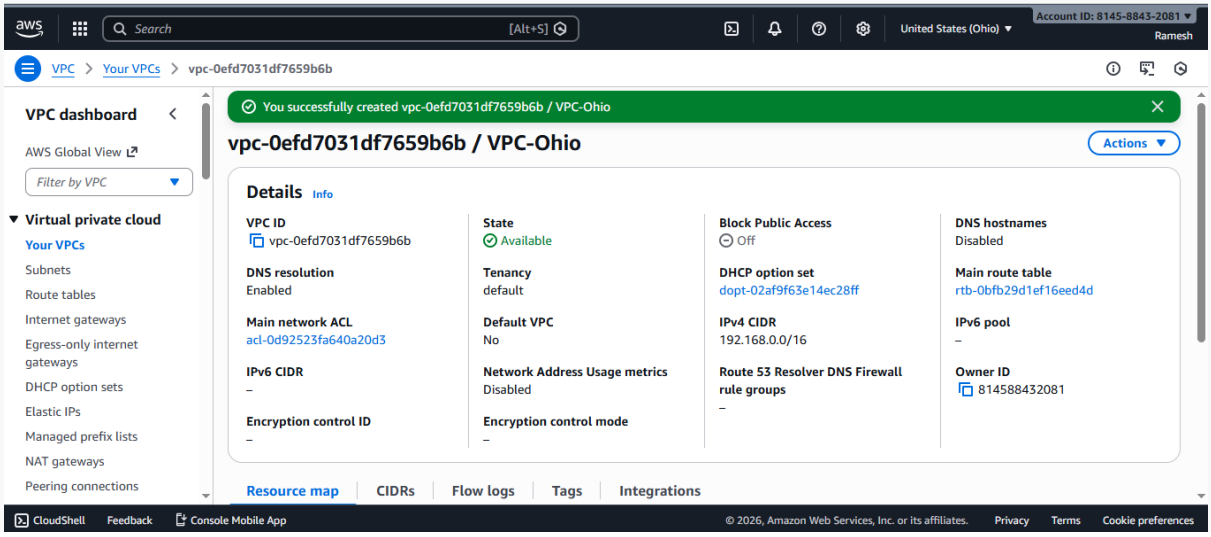
- Click on VPC, select on subnet create subnet as private subnet and public subnet.
- For private subnet we can should not enable auto-assign public IP4 address in edit subnet settings.
- CIDR Range should be within VPC range.



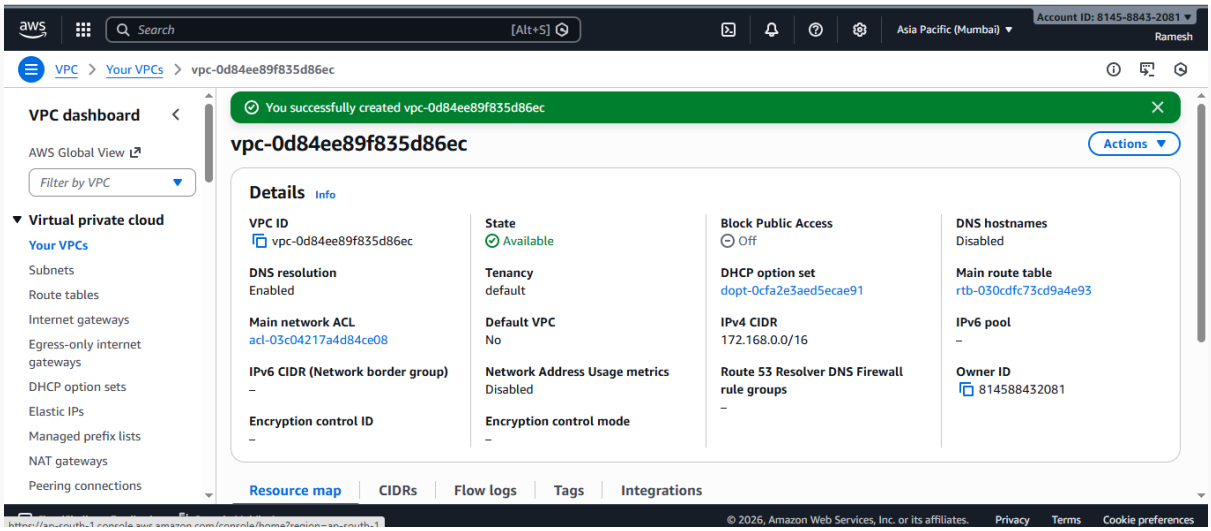
- Click on VPC, select on subnet create subnet as public subnet.
- For public subnet we can should enable auto-assign public IP4 address in edit subnet settings.
- CIDR Range should be within VPC range.

VPC-TASK2

2. Enable VPC peering for cross-region.

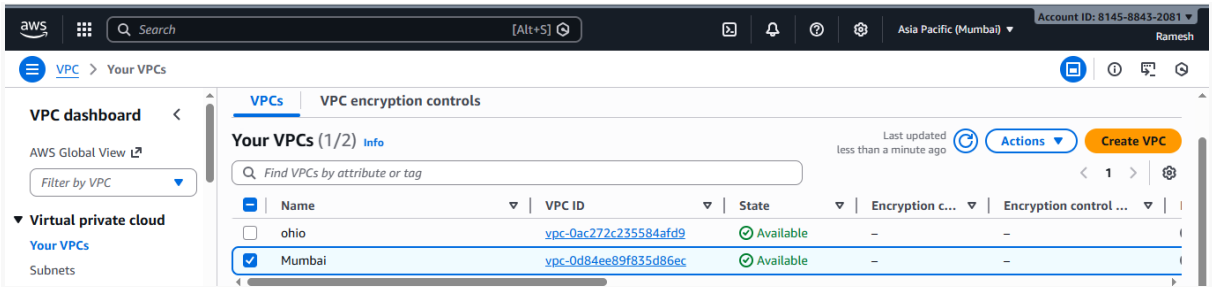


- Search for VPC, click on create VPC for ohio.
- Give IPV4 CIDR as manual input.
- Click on create VPC.

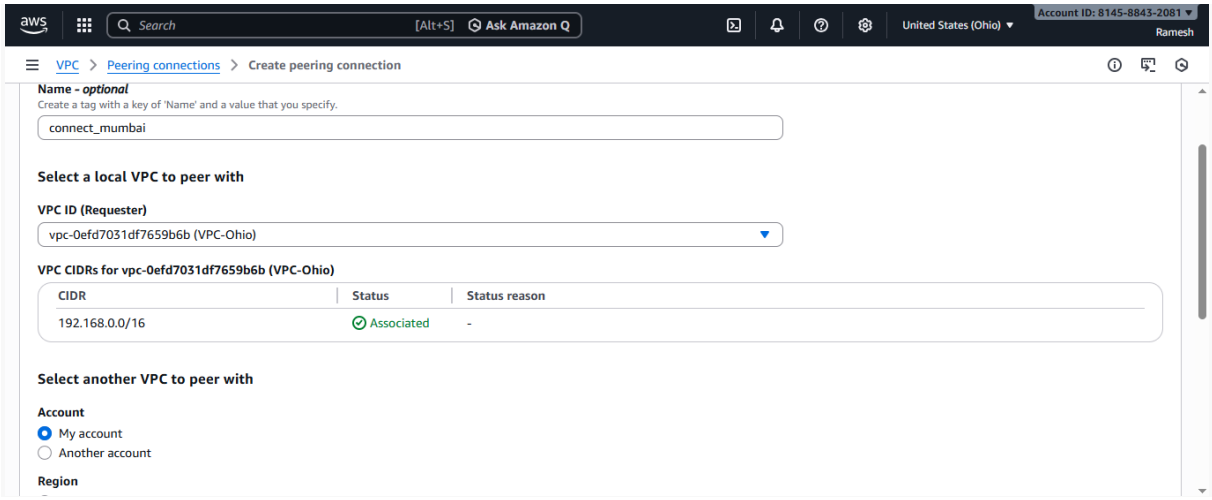


- Search for VPC, click on create VPC for Mumbai Region.
- Give IPV4 CIDR as manual input.
- Click on create VPC.

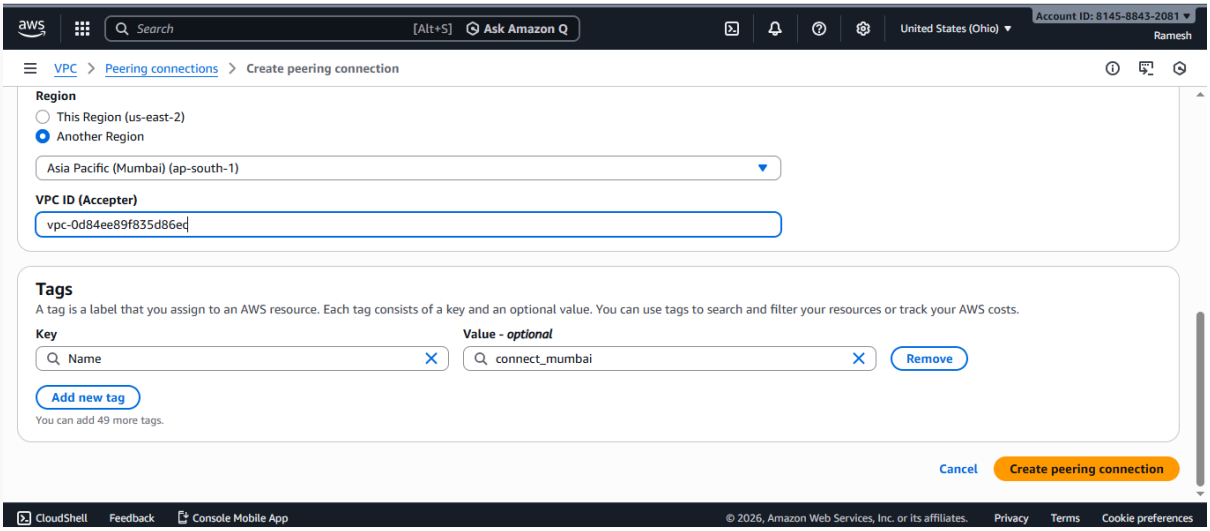
VPC-TASK2



- In Your VPCs we can all VPC what we have created.
- We can create two VPC for VPC cross peering connection

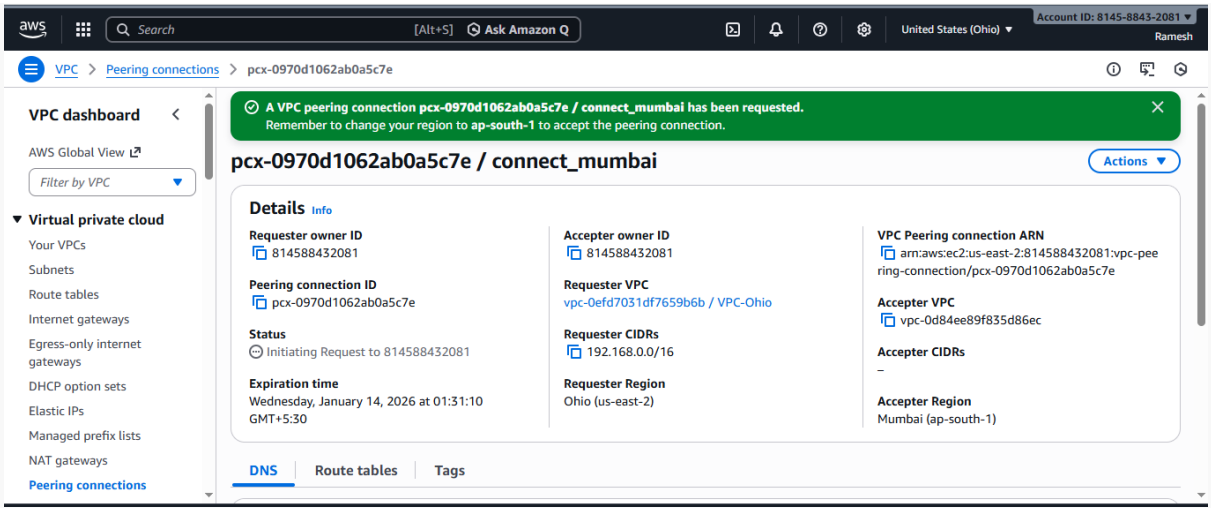


- In VPC we have to search for Peering connections
- Click on Create peering connection
- Select VPC ID (requester)
- Select another Vpc to peer with Account type as my account or another Account.

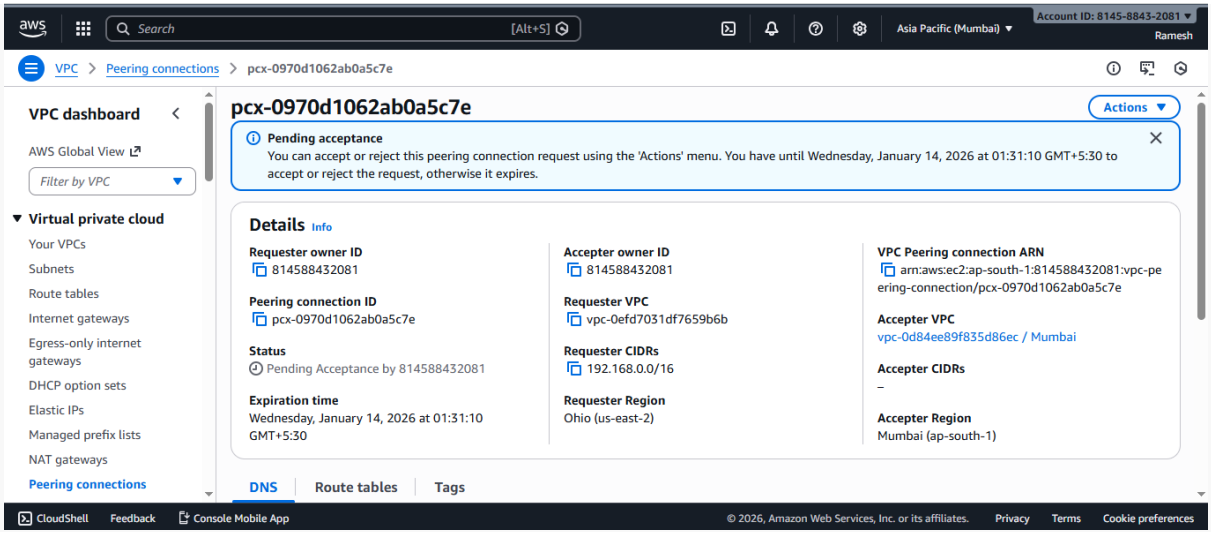


VPC-TASK2

- Select region and
- Acceptor ID-VPC-id
- Click on create on peer connection

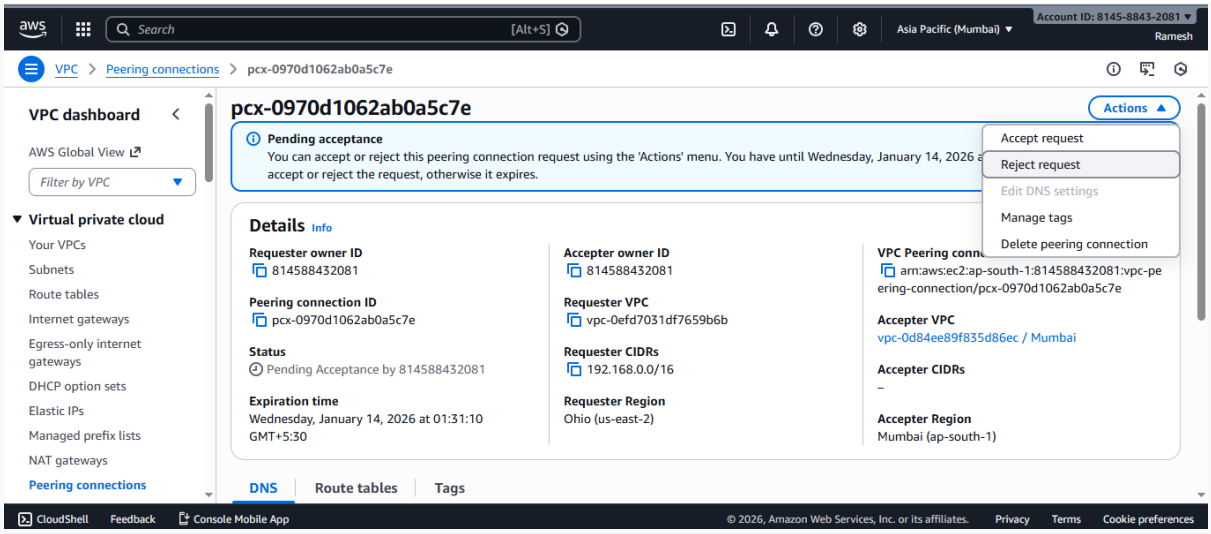


- Above image shows peering requesting send to other region through peering connections.

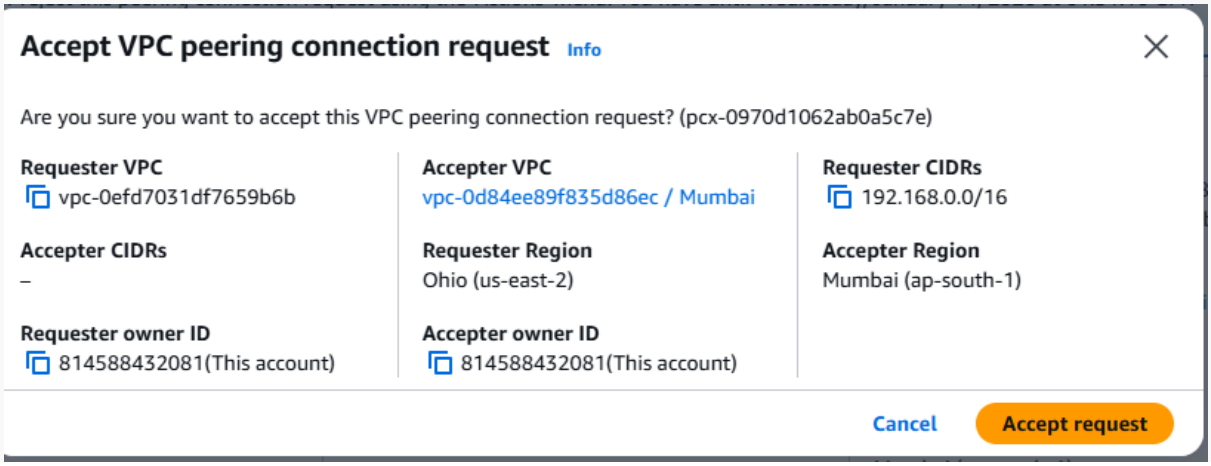


- We can notification of pending acceptance of peering connection as shown above.

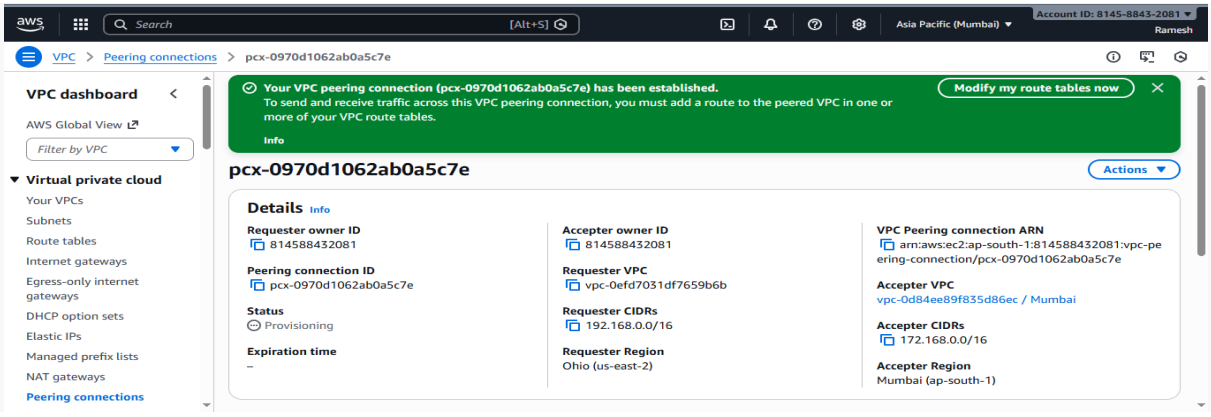
VPC-TASK2



- Click on actions so that we can accept request and reject request.

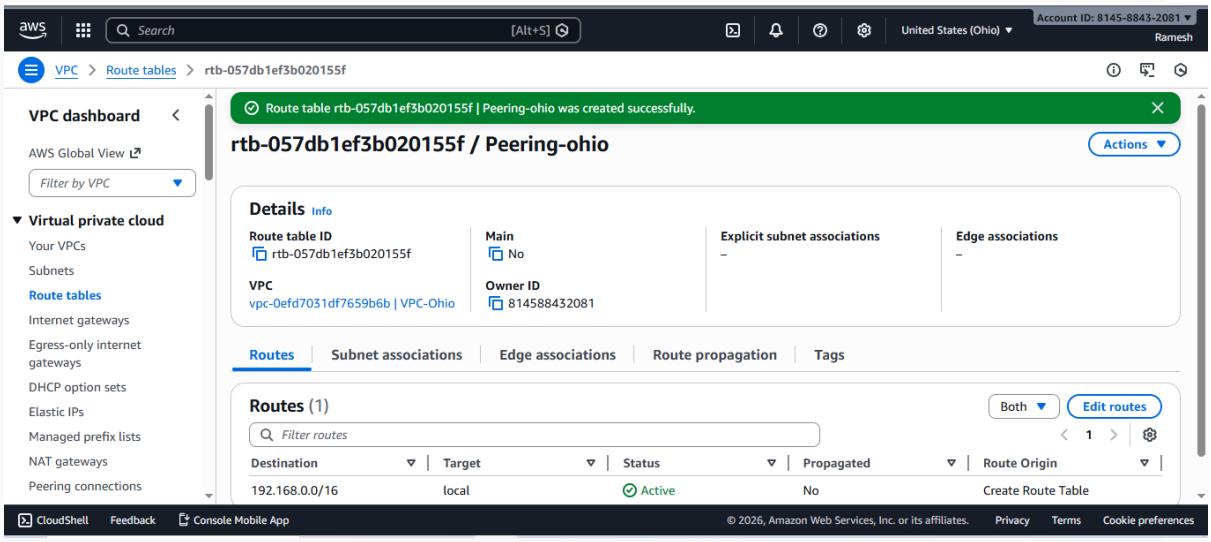


- After clicking on accept request so that we can accept peering connection.

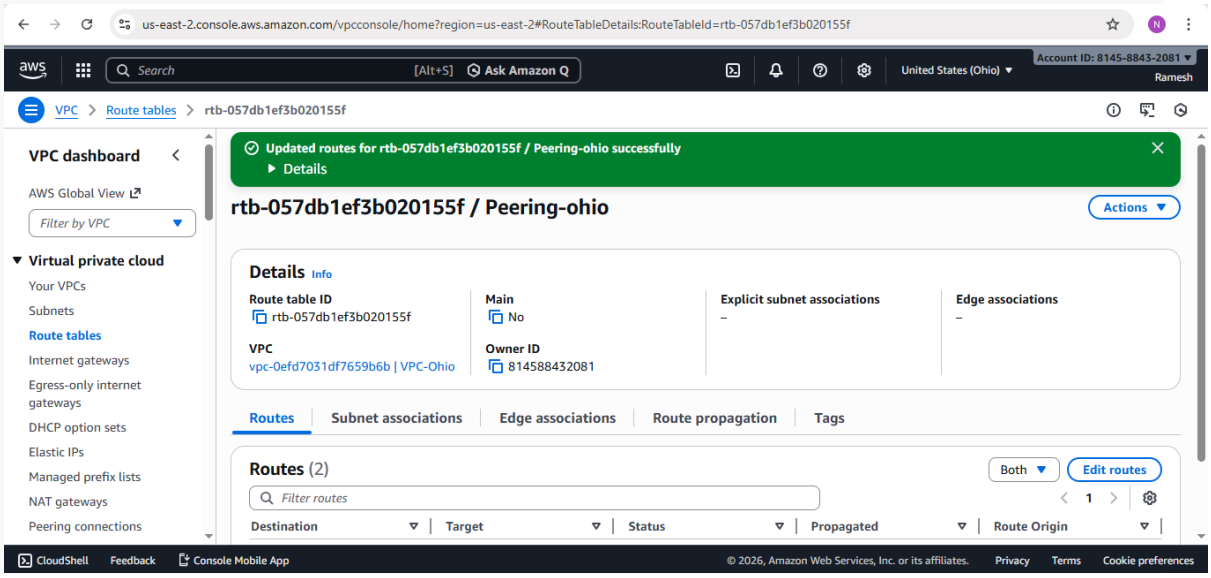


- Above image say successful of peering connection established.

VPC-TASK2

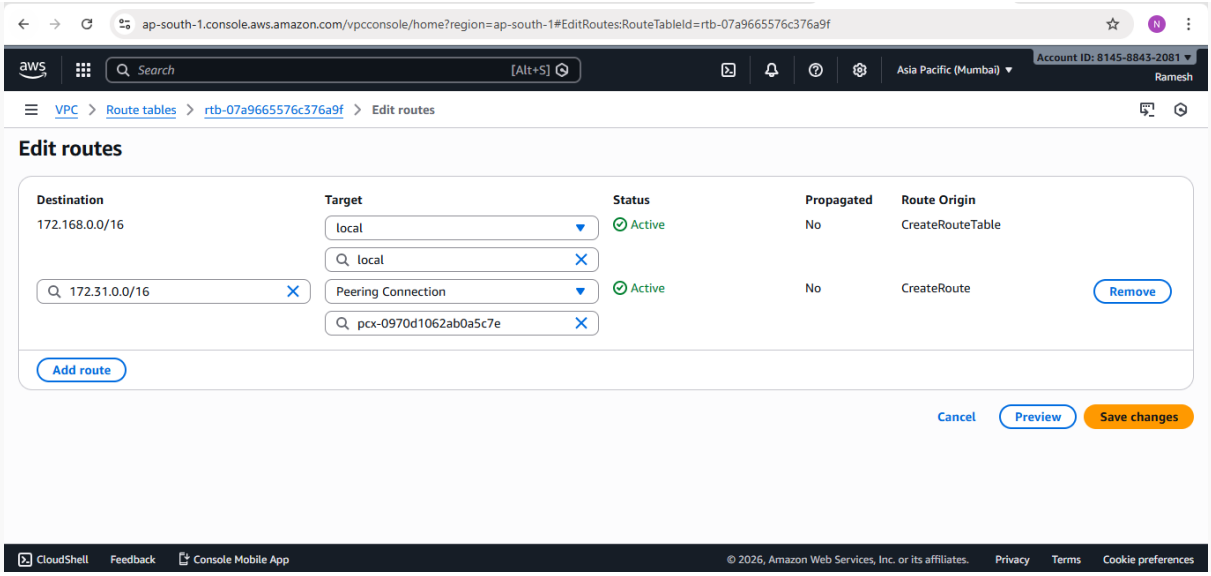


- Click on Route table and create route table.

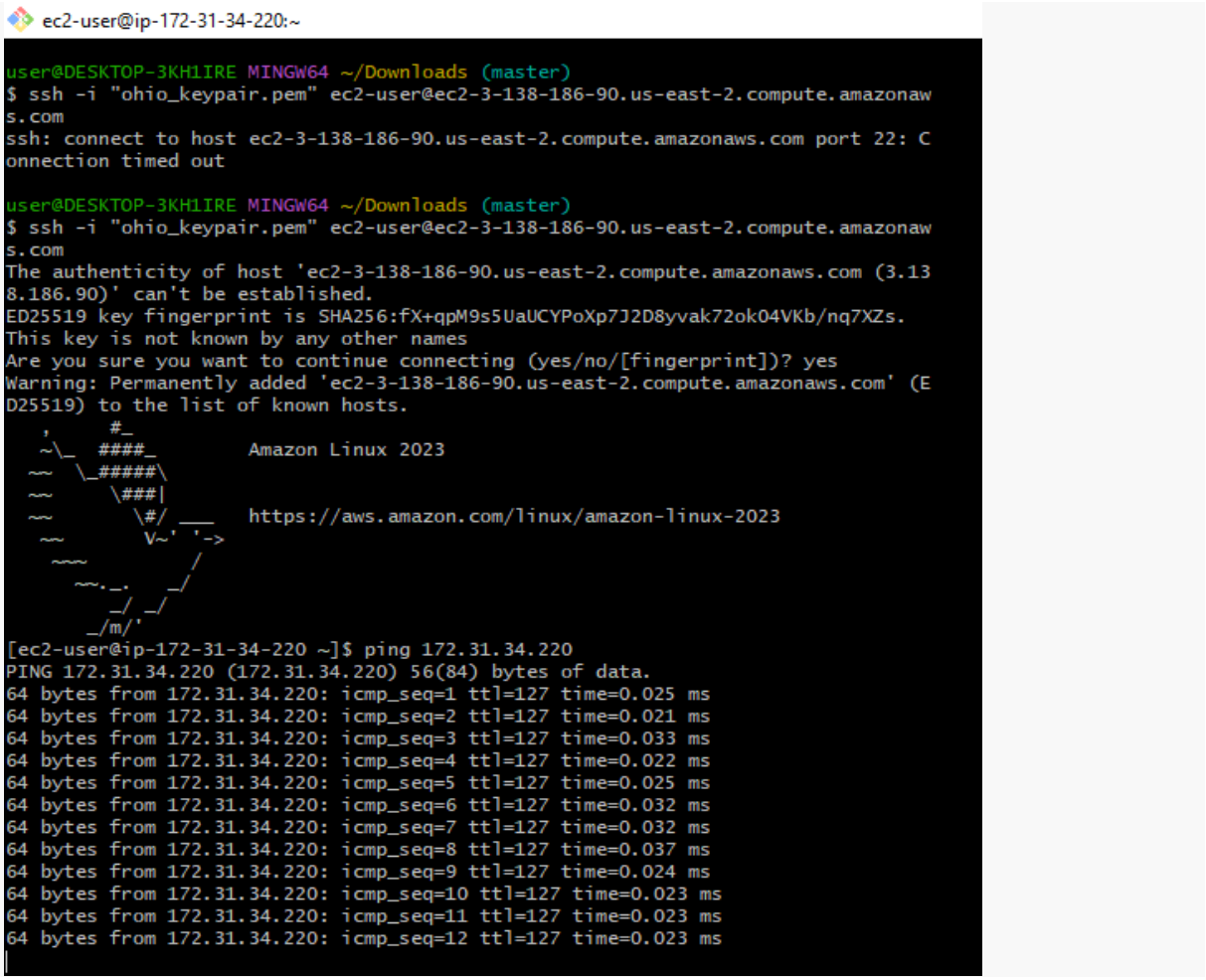


- Click on edit routed and so that we can add peer connection id.

VPC-TASK2



- Here we can add peering connections as shown and save changes.



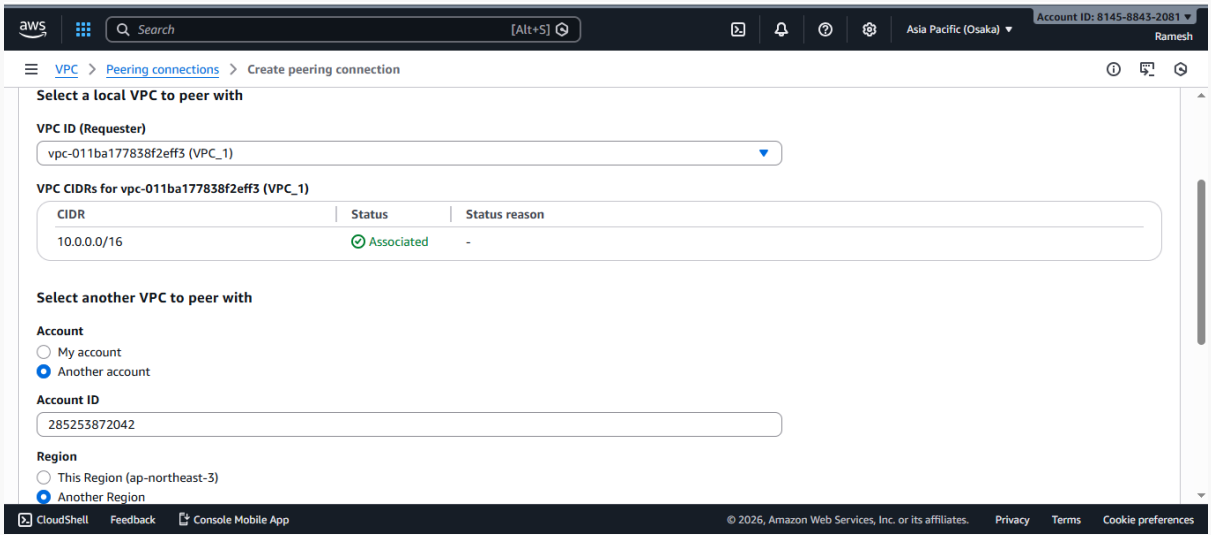
- Connection of EC2 which is created before.
- Checking of ping with private ip.

VPC-TASK2

```
user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
aws.comi "Mumbai_keypair.pem" ec2-user@ec2-65-2-69-128.ap-south-1.compute.amazona
The authenticity of host 'ec2-65-2-69-128.ap-south-1.compute.amazonaws.com (65.2.69.128)' can't be established.
ED25519 key fingerprint is SHA256:KA8sF1hanODAO3kVdg9V8Xa4wzDafpbe1MTEiZOnfLc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-2-69-128.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-4-29 ~]$ ping 172.31.4.29
PING 172.31.4.29 (172.31.4.29) 56(84) bytes of data.
64 bytes from 172.31.4.29: icmp_seq=1 ttl=127 time=0.022 ms
64 bytes from 172.31.4.29: icmp_seq=2 ttl=127 time=0.029 ms
64 bytes from 172.31.4.29: icmp_seq=3 ttl=127 time=0.029 ms
64 bytes from 172.31.4.29: icmp_seq=4 ttl=127 time=0.030 ms
64 bytes from 172.31.4.29: icmp_seq=5 ttl=127 time=0.029 ms
64 bytes from 172.31.4.29: icmp_seq=6 ttl=127 time=0.033 ms
64 bytes from 172.31.4.29: icmp_seq=7 ttl=127 time=0.036 ms
64 bytes from 172.31.4.29: icmp_seq=8 ttl=127 time=0.027 ms
64 bytes from 172.31.4.29: icmp_seq=9 ttl=127 time=0.027 ms
64 bytes from 172.31.4.29: icmp_seq=10 ttl=127 time=0.031 ms
64 bytes from 172.31.4.29: icmp_seq=11 ttl=127 time=0.031 ms
64 bytes from 172.31.4.29: icmp_seq=12 ttl=127 time=0.030 ms
64 bytes from 172.31.4.29: icmp_seq=13 ttl=127 time=0.031 ms
64 bytes from 172.31.4.29: icmp_seq=14 ttl=127 time=0.029 ms
64 bytes from 172.31.4.29: icmp_seq=15 ttl=127 time=0.027 ms
64 bytes from 172.31.4.29: icmp_seq=16 ttl=127 time=0.031 ms
64 bytes from 172.31.4.29: icmp_seq=17 ttl=127 time=0.028 ms
```

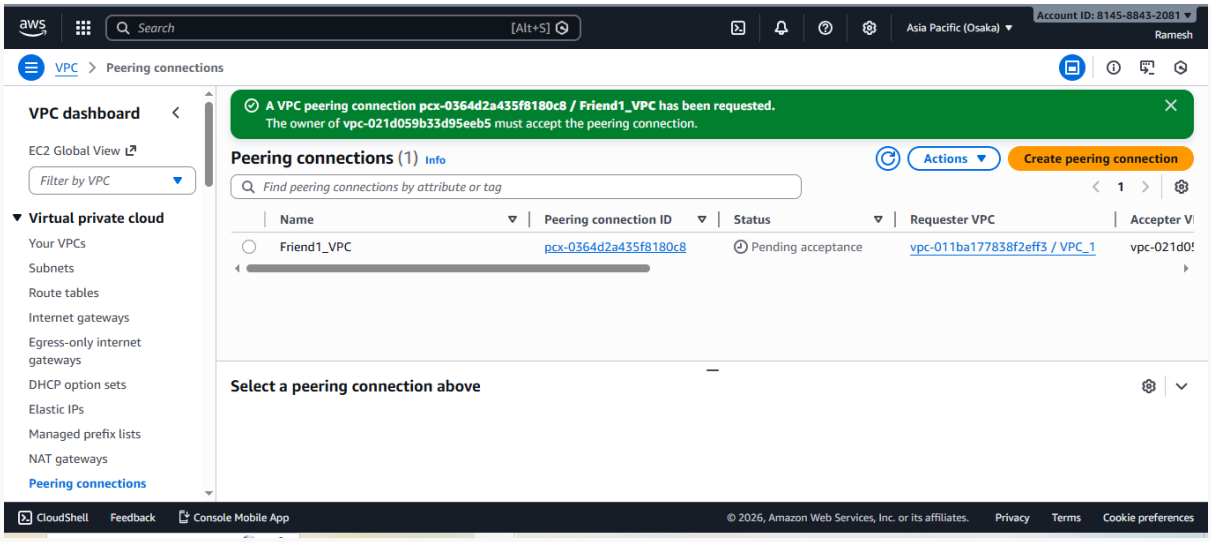
- Checking connection for both EC2.

3. Enable VPC peering for cross-account (you can collaborate with your friend to do this task).

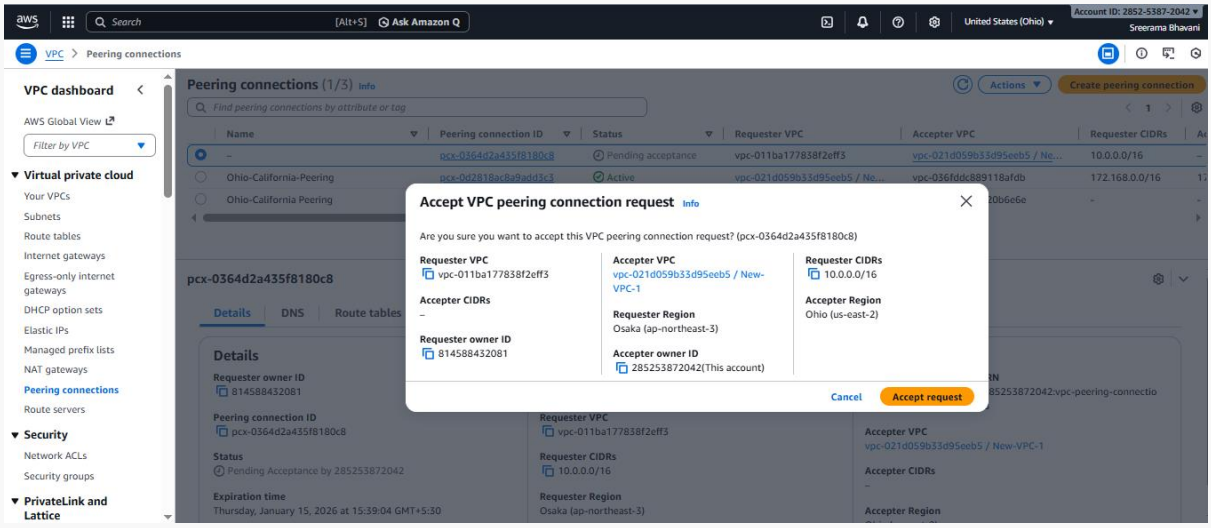


- Select a local VPC to peer with VPC ID (requester).
- Select another VPC to peer with account as another account
- Fill account Id
- Select Region as another region.

VPC-TASK2

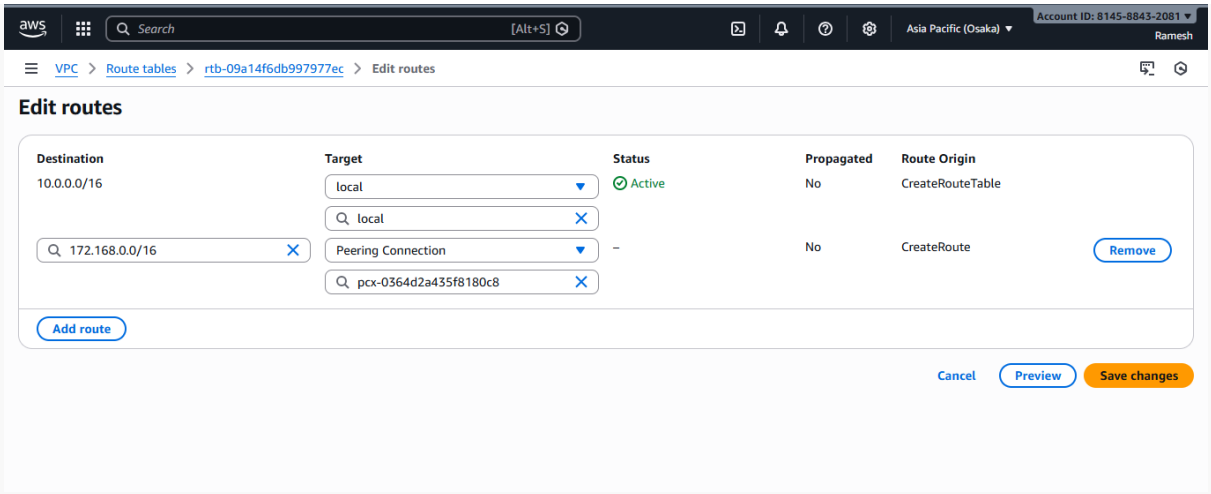


- Click on Create peering connection which is send to acceptor.

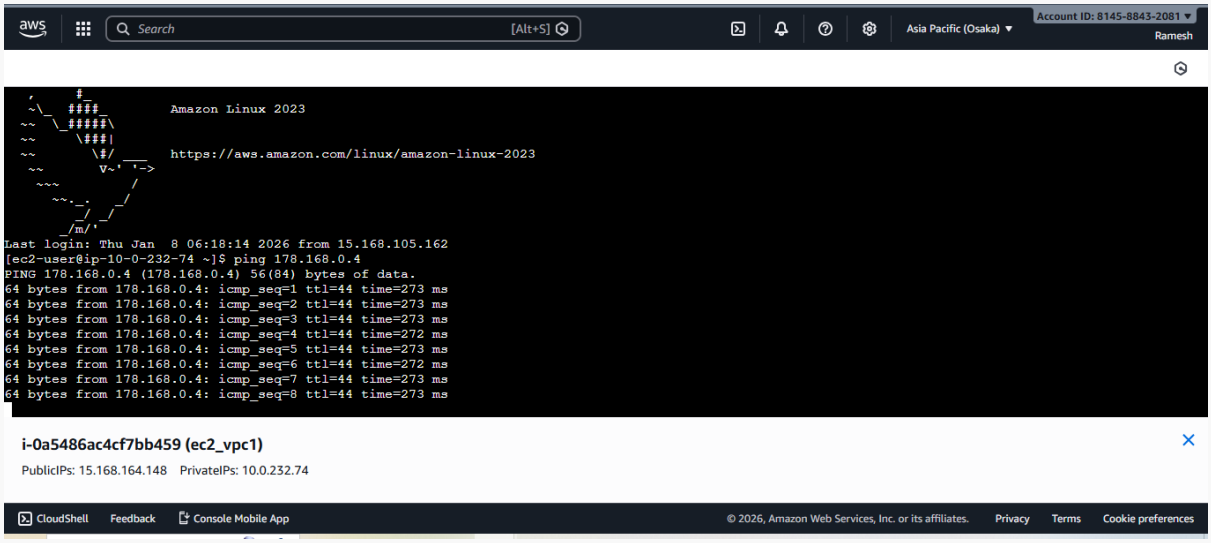


- From other account this shows for accept request

VPC-TASK2



- Click on Route tables and edit route tables and edit routes
- Add friend CIDR and peering connection
- Click on Save changes

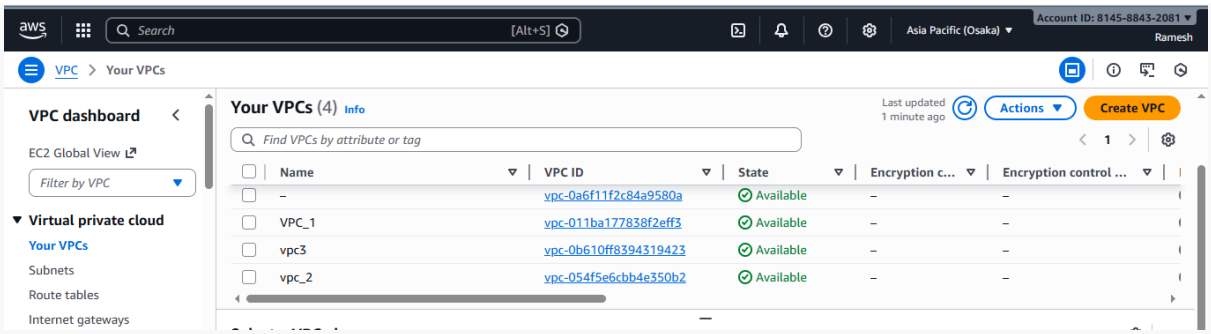


- Check ping ip address with friend ip and ping is connected.

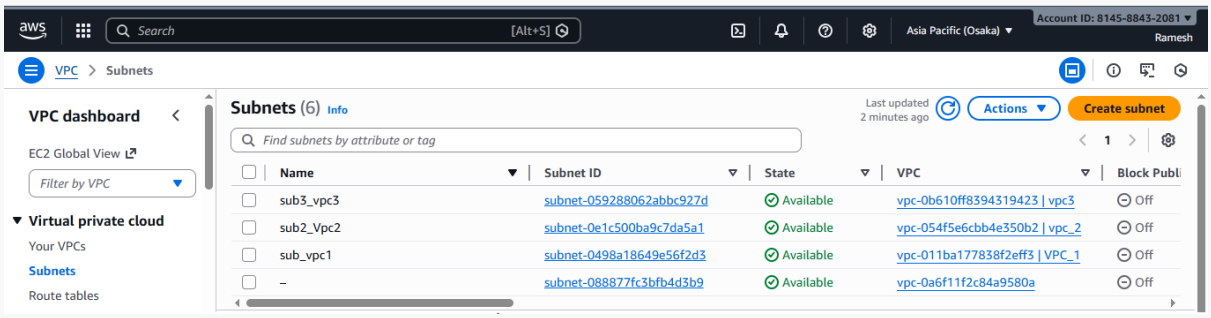
VPC-TASK2

4. Set up a VPC Transit Gateway.

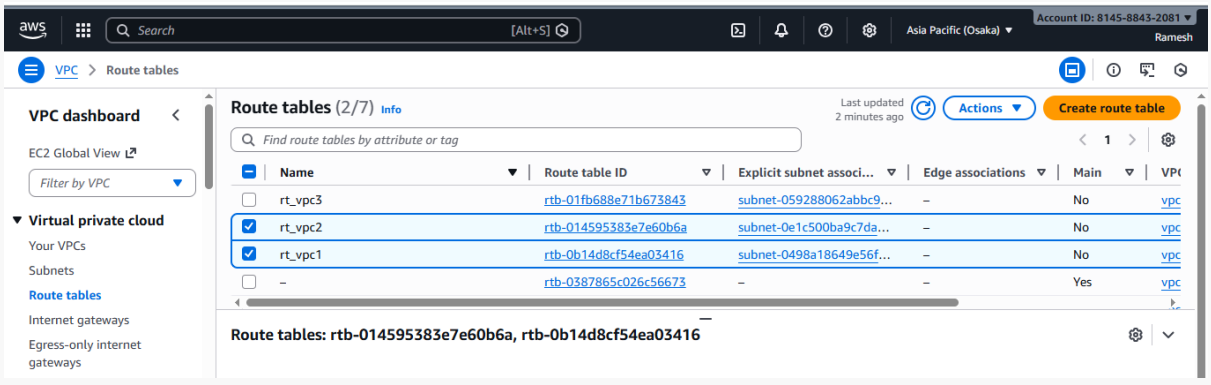
- Transit Gateway allows you to connect many VPCs and on-prem networks through a single gateway instead of managing many VPC peering connections.



- Click on VPC and create VPC as per our requirement with CIDR range
- Click on your VPC and it shows VPCs present in it.

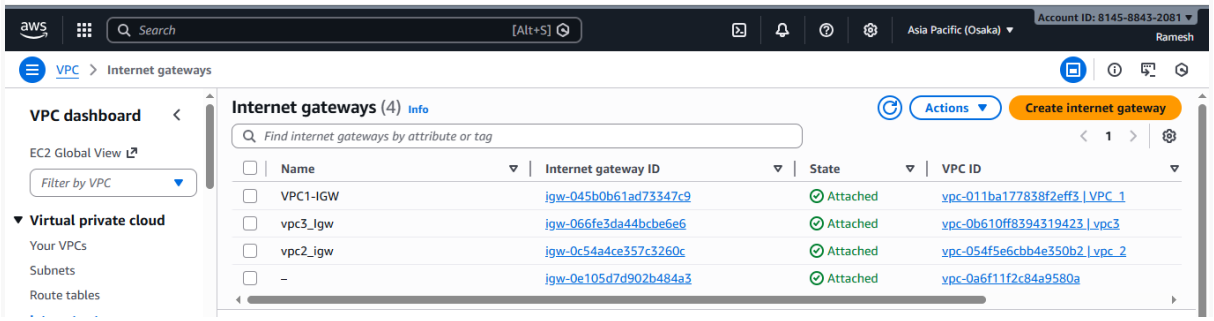


- Click on subnet and create subnet with in VPC CIDR as per our requirement.
- For that subnet we should add corresponding VPC

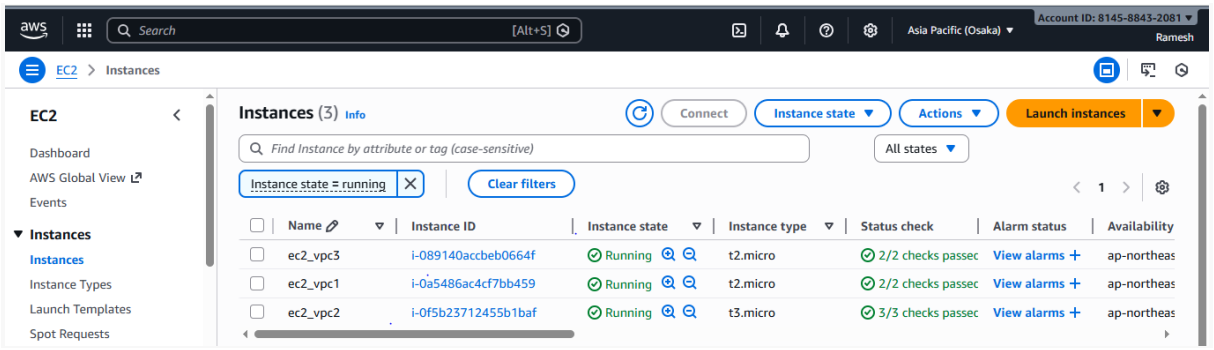


- Click on create on route table and add VPC to them.

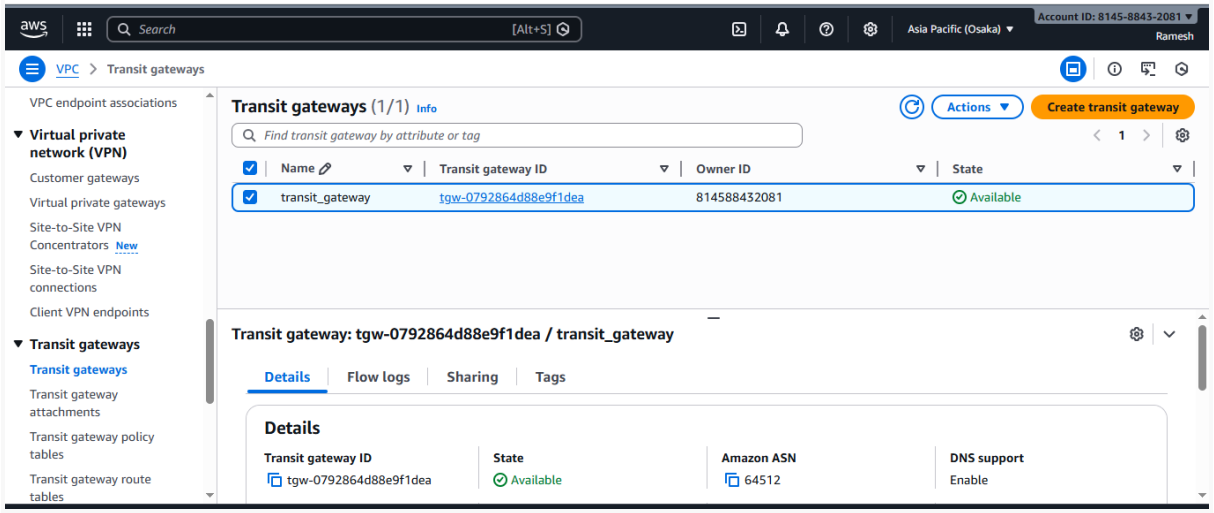
VPC-TASK2



- Click on create Internet gateway for VPC present in our region.

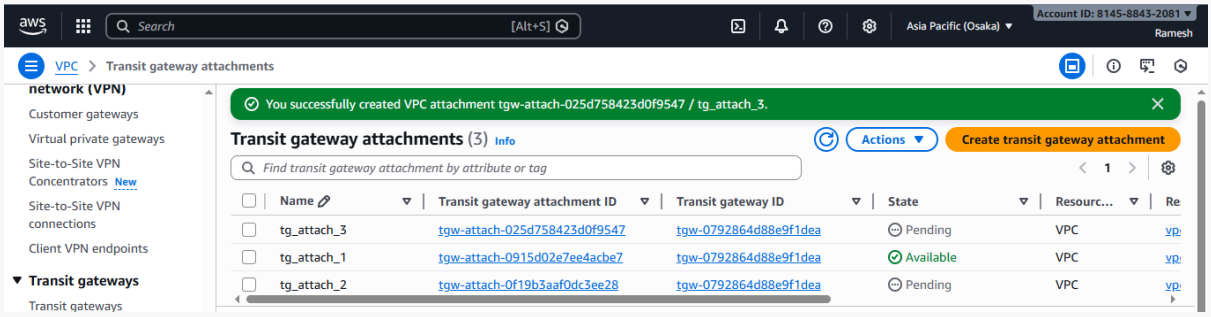


- Click on EC2 and click on instance and launch Instances as required.

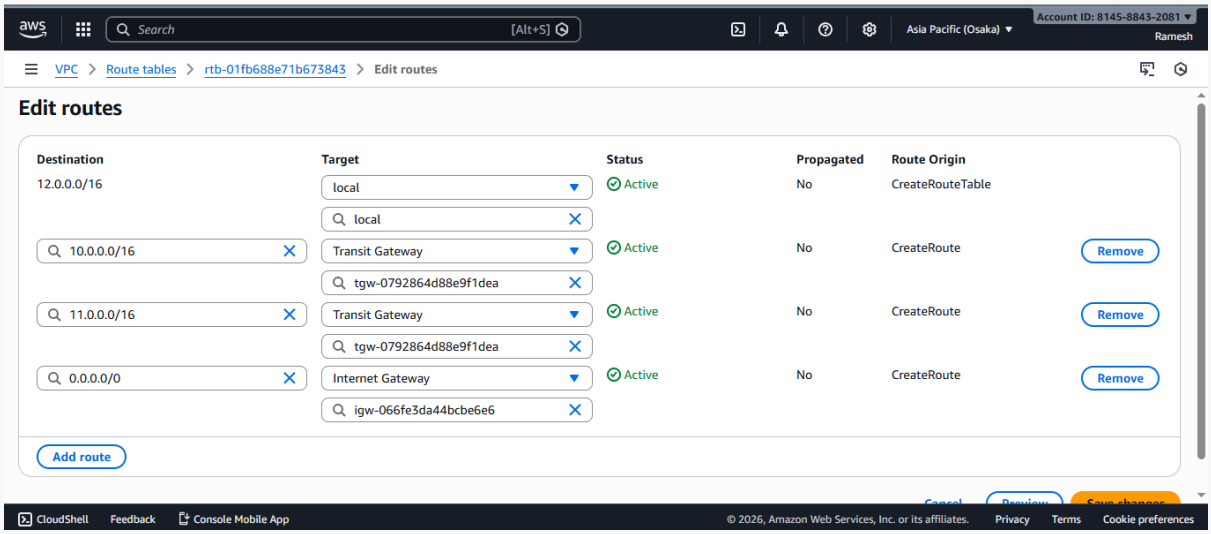


- Search for Transit gateways and create transit gateway

VPC-TASK2

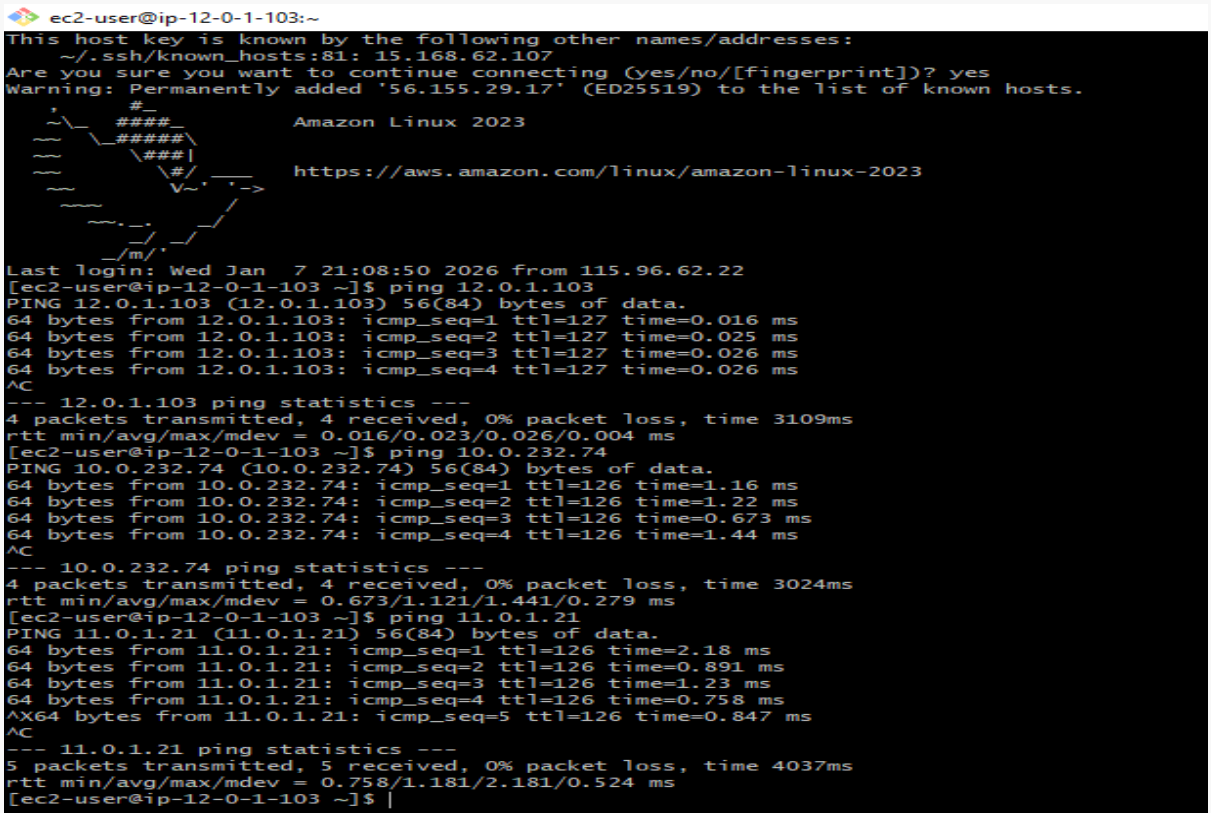


- Click on transit gateway attachments and create transit gateway attachment.



- In VPC and select Route tables and edit routes
- Add transit gateways CIDR and add internet gateway.
- Click on save changes.

VPC-TASK2

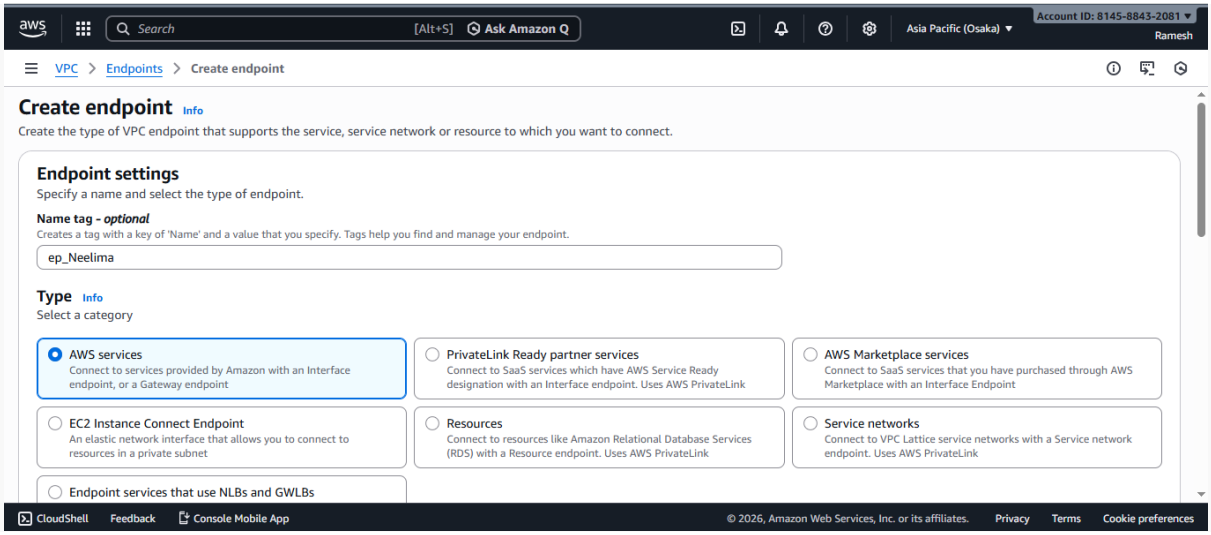


- Above image shows VPC transit gateway among all EC2**.**

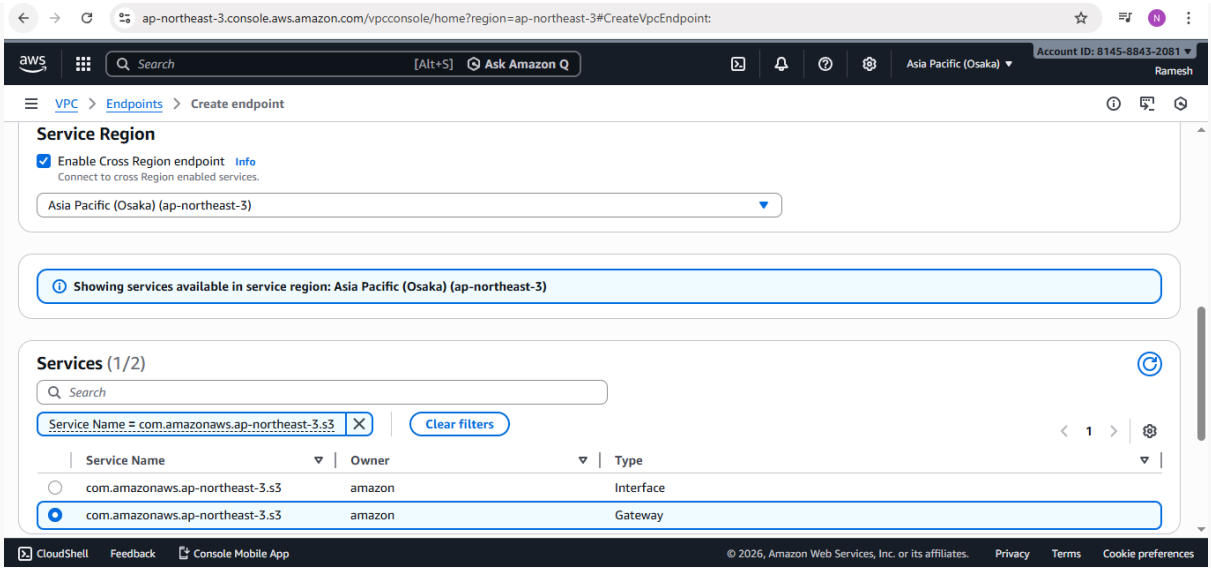
VPC-TASK2

5. Set up a VPC Endpoint.

- An endpoint is a specific entry or exit point used to communicate with a service, application, or network.
  Think of it as an address where requests are sent and responses are received.

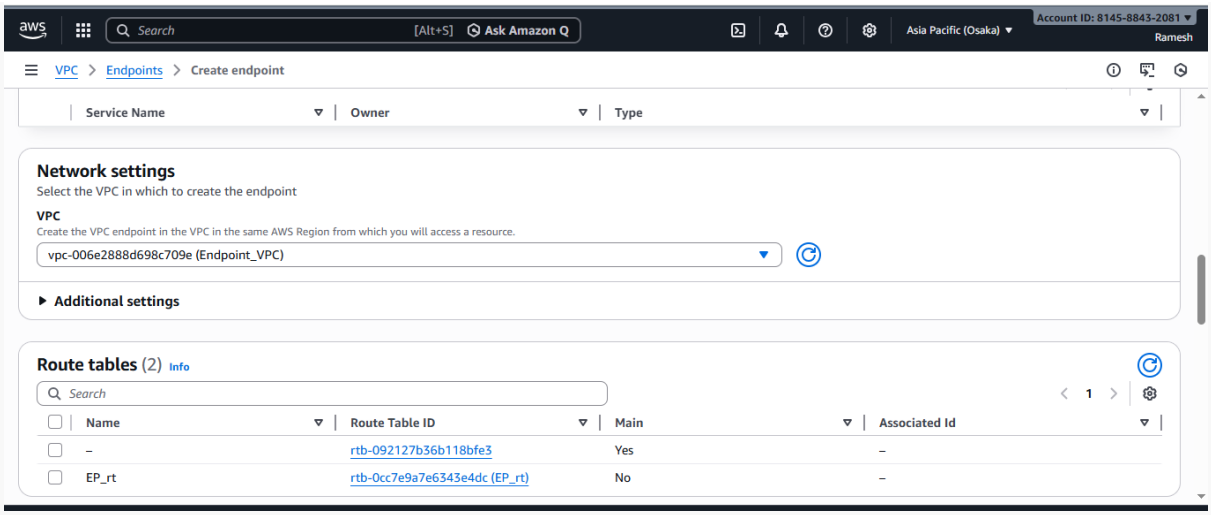


- In VPC select Endpoints and
- Click on create on endpoint.
- Select type as AWS services.

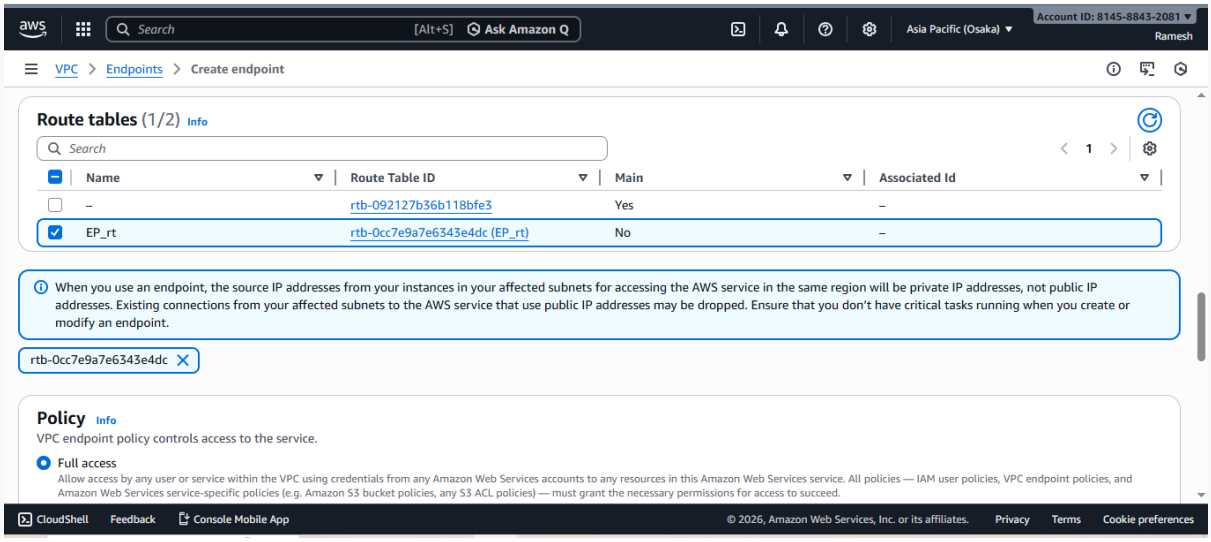


- Select Service Region
- In services select for service which is associate with the region.

VPC-TASK2

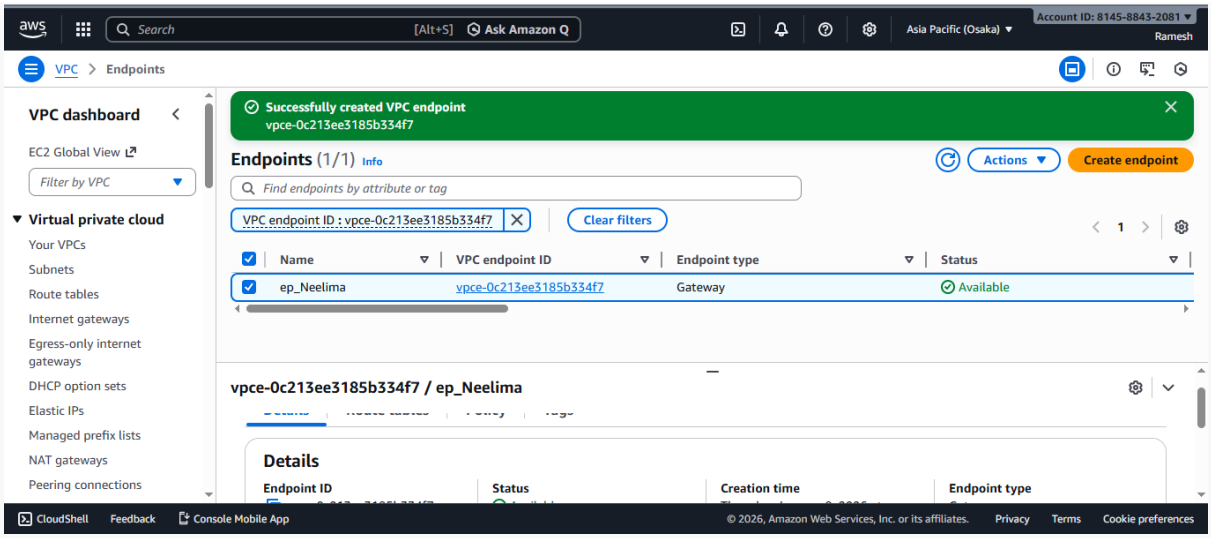


- In Networking setting Select VPC which is created for VPC endpoints.
- Select Route tables which are created for endpoints.

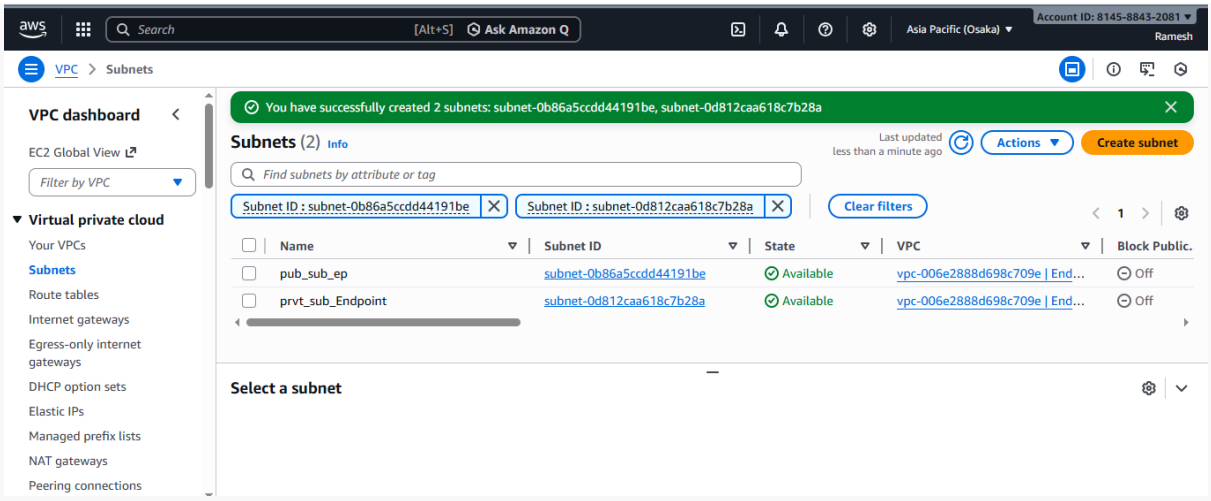


- Click on policy and select full access.
- Save it.

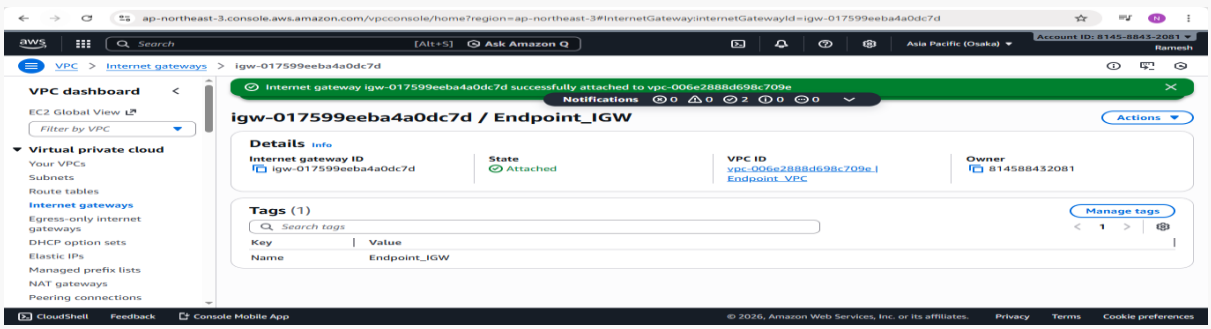
VPC-TASK2



- Above image shows endpoints created successfully



- Click on subnet and create subnets with VPC which is associate with it



- Click on Internet gateway and create IGW and route edit route and internet gateway to public subnet.

VPC-TASK2

```
user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ scp -i osakakeypair.pem osakakeypair.pem ec2-user@56.155.87.114:keys/
osakakeypair.pem                                                      100% 1678     8.0KB/s   00:00

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ ssh -i osakakeypair.pem ec2-user@56.155.87.114
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Jan  8 11:15:42 2026 from 103.203.172.230
[ec2-user@ip-14-0-1-11 ~]$ mkdir -p ~/keys
[ec2-user@ip-14-0-1-11 ~]$ ssh -i osakakeypair.pem ec2-user@56.155.87.114
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0444 for 'osakakeypair.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "osakakeypair.pem": bad permissions
ec2-user@56.155.87.114: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-14-0-1-11 ~]$ chmod 400 ~/keys/osakakeypair.pem
[ec2-user@ip-14-0-1-11 ~]$ ls -l ~/keys/osakakeypair.pem
-r--------. 1 ec2-user ec2-user 1678 Jan  8 11:19 /home/ec2-user/keys/osakakeypair.pem
[ec2-user@ip-14-0-1-11 ~]$ ssh -i ~/keys/osakakeypair.pem ec2-user@10.0.2.4

^C
[ec2-user@ip-14-0-1-11 ~]$ ssh -i ~/keys/osakakeypair.pem ec2-user@10.0.2.4
```

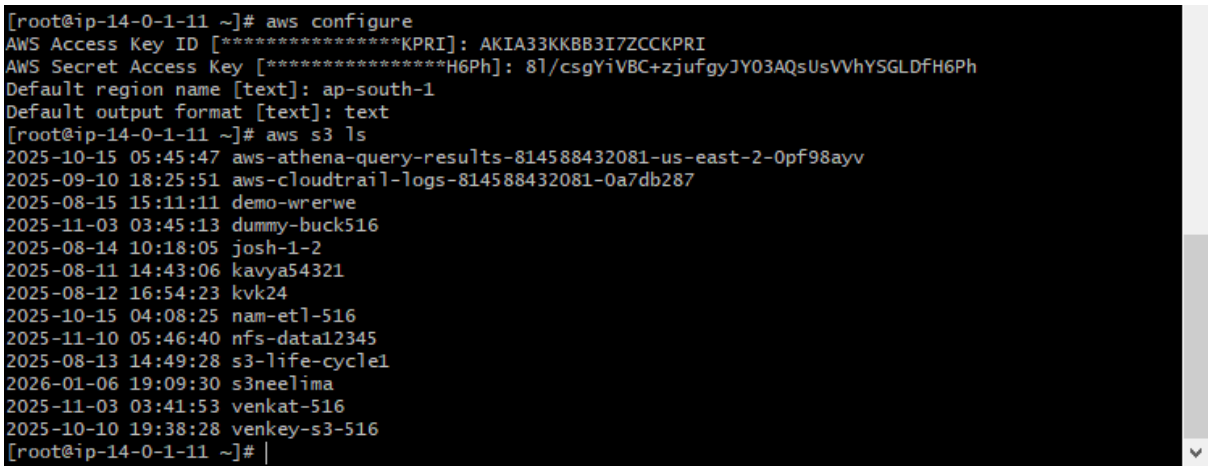
- Connect to ec2 instance which created for endpoints.

root@ip-14-0-1-11:~

```
[ec2-user@ip-14-0-1-11 ~]$ sudo su -
[root@ip-14-0-1-11 ~]# ping 14.0.1.11
PING 14.0.1.11 (14.0.1.11) 56(84) bytes of data.
64 bytes from 14.0.1.11: icmp_seq=1 ttl=127 time=0.015 ms
64 bytes from 14.0.1.11: icmp_seq=2 ttl=127 time=0.027 ms
64 bytes from 14.0.1.11: icmp_seq=3 ttl=127 time=0.025 ms
64 bytes from 14.0.1.11: icmp_seq=4 ttl=127 time=0.026 ms
^C
--- 14.0.1.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3128ms
rtt min/avg/max/mdev = 0.015/0.023/0.027/0.004 ms
[root@ip-14-0-1-11 ~]# ping 14.0.2.4
PING 14.0.2.4 (14.0.2.4) 56(84) bytes of data.
64 bytes from 14.0.2.4: icmp_seq=1 ttl=127 time=0.865 ms
64 bytes from 14.0.2.4: icmp_seq=2 ttl=127 time=0.890 ms
64 bytes from 14.0.2.4: icmp_seq=3 ttl=127 time=1.38 ms
64 bytes from 14.0.2.4: icmp_seq=4 ttl=127 time=1.04 ms
64 bytes from 14.0.2.4: icmp_seq=5 ttl=127 time=1.47 ms
^C
--- 14.0.2.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.865/1.128/1.469/0.249 ms
[root@ip-14-0-1-11 ~]# |
```

VPC-TASK2

- Check ping connection for private ip with public subnet ip.



To access AWS first we have to configure through IAM

- $aws configure
- AWS access key ID:
- AWS Secret Access ID:
- Default region name:
- Default output format:
- Access one of the service like S3 service through it
- AWS s3 ls it shows all files present it.