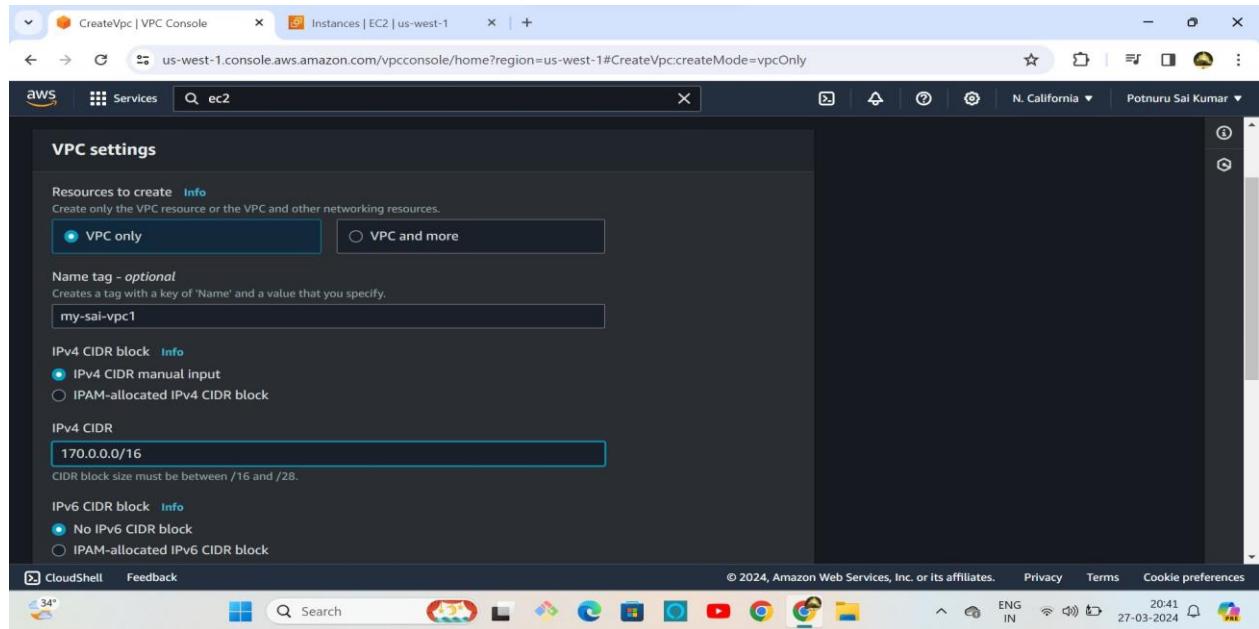


ASSIGNMENT

- Create three VPCS and connect the three VPC'S using transit gateway.
-

- Create a virtual private cloud (VPC1)
- STEP-1: Search for VPC in search space of AWS home page and click onVPC and Now click on create VPC to create our custom VPC1 and give the details and finally click on Create VPC.



Now we have created our custom VPC1 successfully

The screenshot shows the AWS VPC Console interface. At the top, a green banner displays the message: "You successfully created vpc-035b2f7e22fd8fc31 / my-sai-vpc1". Below the banner, the breadcrumb navigation shows "VPC > Your VPCs > vpc-035b2f7e22fd8fc31". The main content area is titled "vpc-035b2f7e22fd8fc31 / my-sai-vpc1". The "Details" tab is selected, showing the following information:

VPC ID	State	DNS hostnames	DNS resolution
vpc-035b2f7e22fd8fc31	Available	Disabled	Enabled
Tenancy	DHCP option set	Main route table	Main network ACL
Default	dopt-005d799a56beaaa14	rtb-0dc5da2b509b4096f	acl-0ed2c1f7f56178c96
Default VPC	IPv4 CIDR	IPv6 pool	IPv6 CIDR
No	170.0.0.0/16	-	-
Network Address Usage metrics	Route 53 Resolver DNS	Owner ID	
Disabled	Firewall rule groups	637423591206	

Below the details table, there are tabs for "Resource map", "CIDRs", "Flow logs", "Tags", and "Integrations". The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

- Now click on subnets to create Subnet to our custom VPC1.

The screenshot shows the "Create subnet" wizard in the AWS EC2 console. The top navigation bar shows "CreateSubnet | VPC Console" and "Instances | EC2 | us-west-1". The main content area is titled "Create subnet" and has the "Info" tab selected. The "VPC" section contains the following fields:

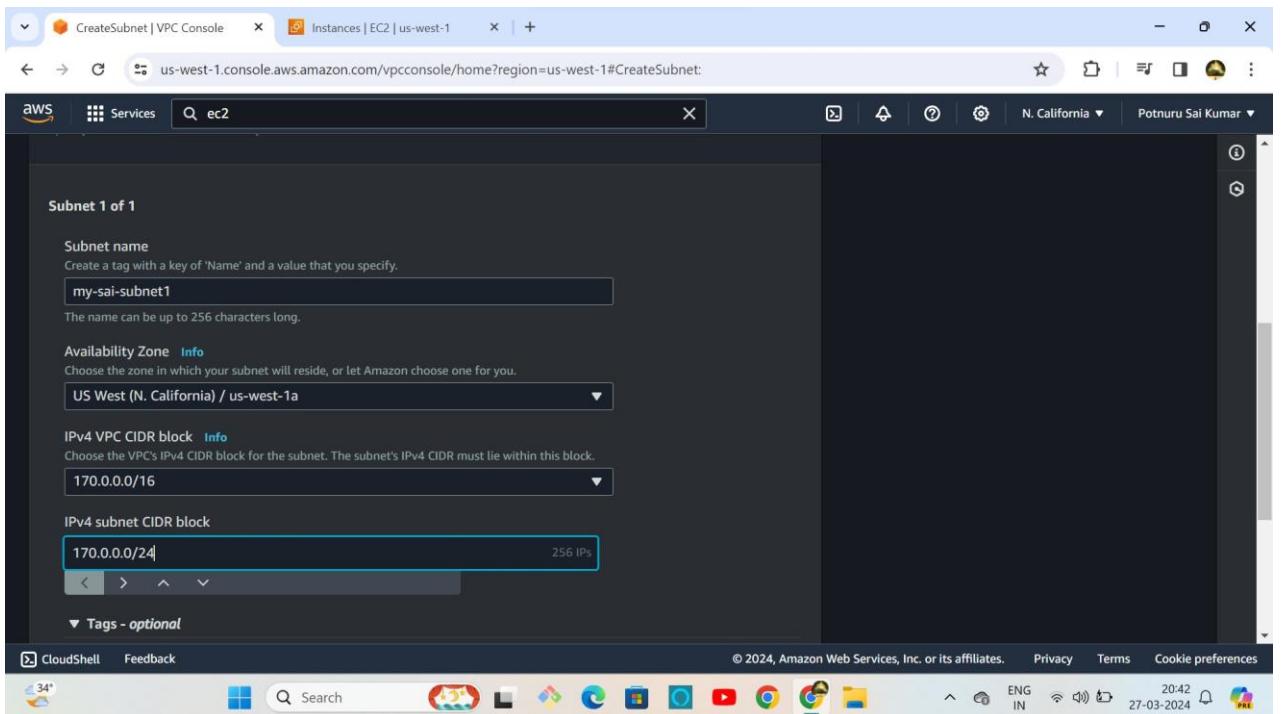
- VPC ID: vpc-035b2f7e22fd8fc31 (my-sai-vpc1)
- Associated VPC CIDRs: IPv4 CIDR: 170.0.0.0/16

The "Subnet settings" section is expanded, showing the following configuration:

- Subnet 1 of 1

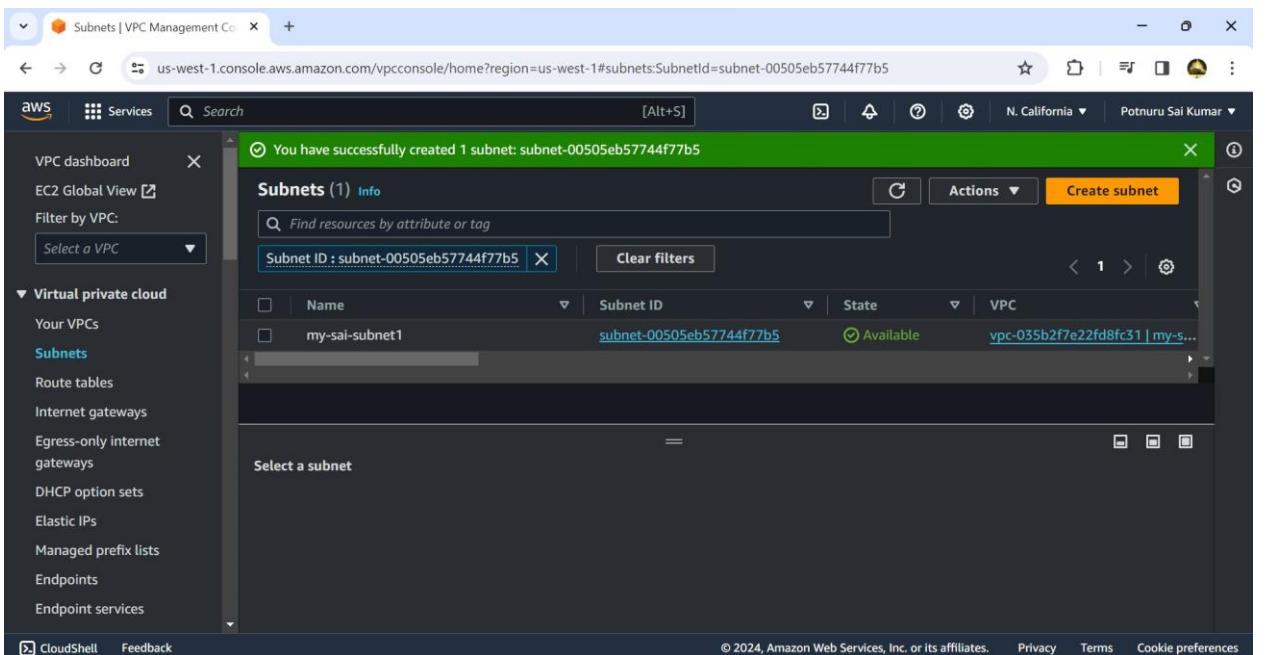
At the bottom of the page, there are standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences. The bottom right corner shows system status icons: 34°, ENG IN, 2042, 27-03-2024, and a battery icon.

- Then create subnets we have given our custom VPC-ID, Subnet name, choose availability Zone, IPv4 subnet CIDR block, then finally create

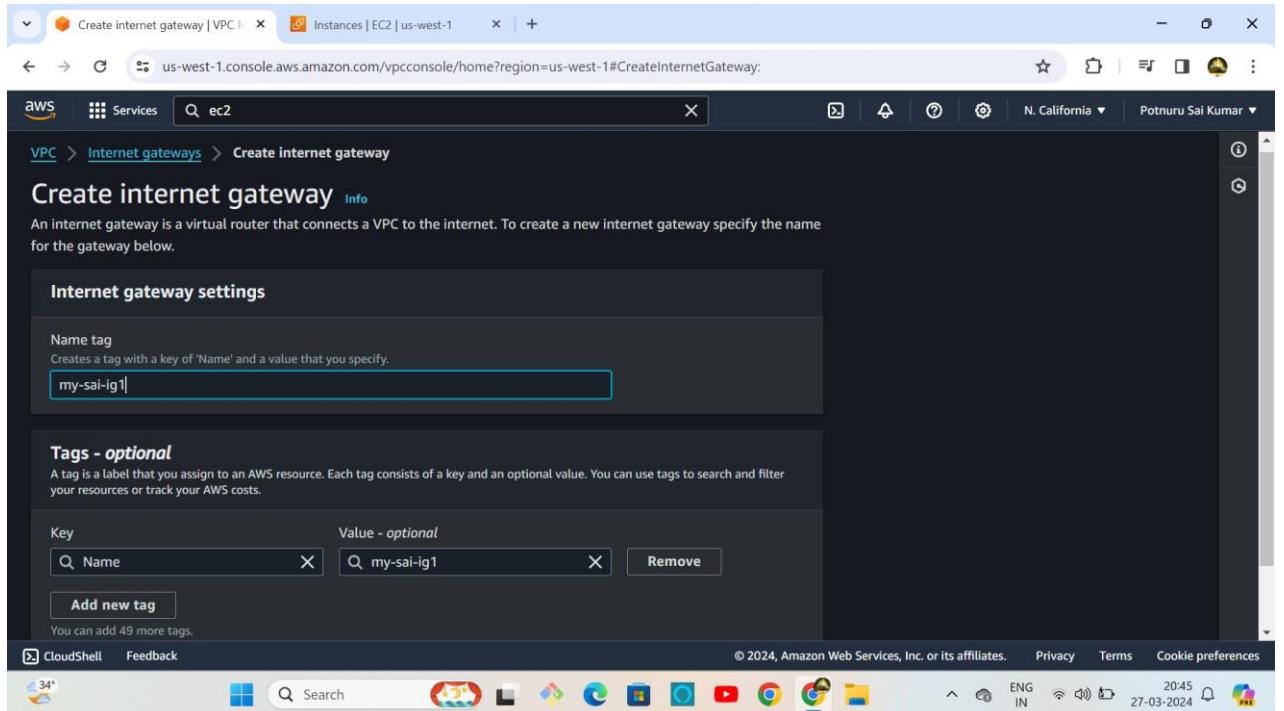


subnet.

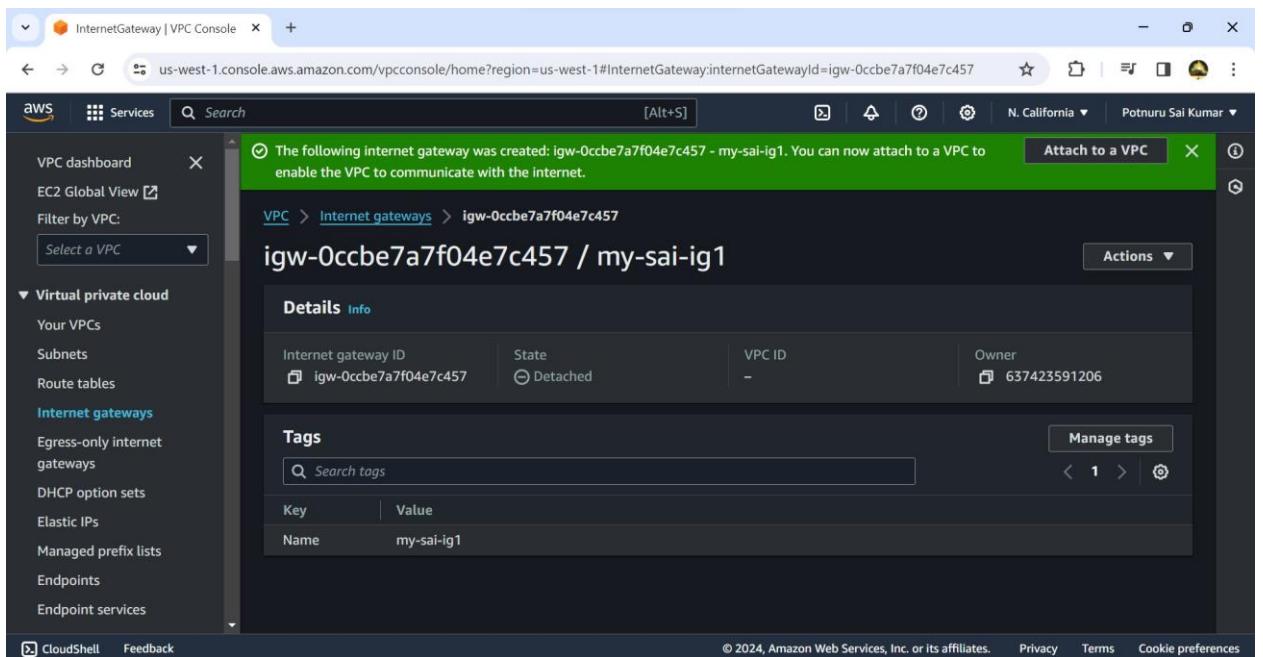
- That we successfully created a subnet 1.



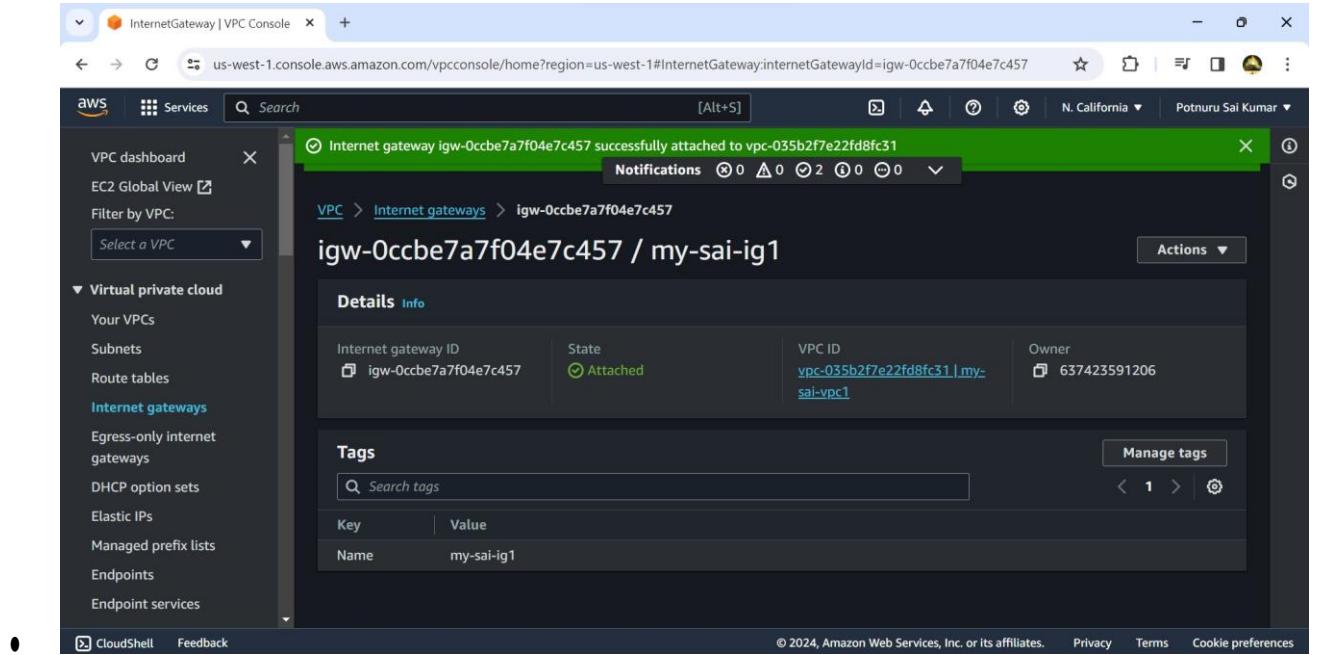
- Now click on Internet gateways from the menu bar and click on create internet gateway.



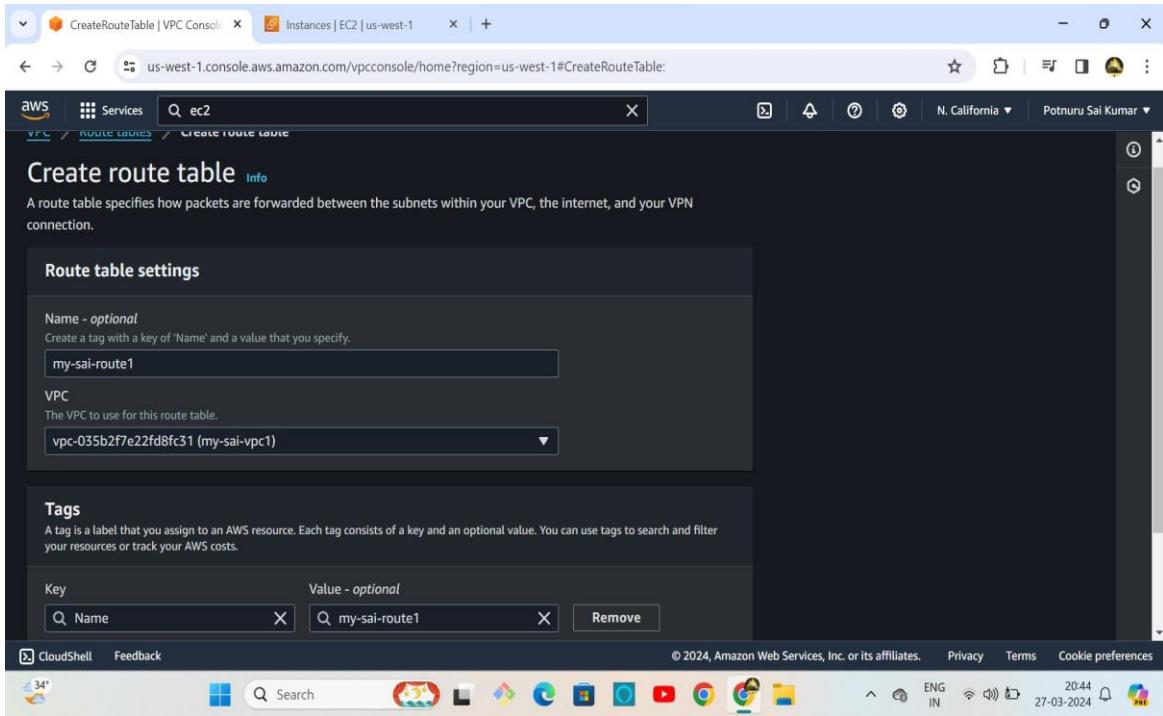
- Click on Create internet gateway.
- We successfully created an internet gateway.



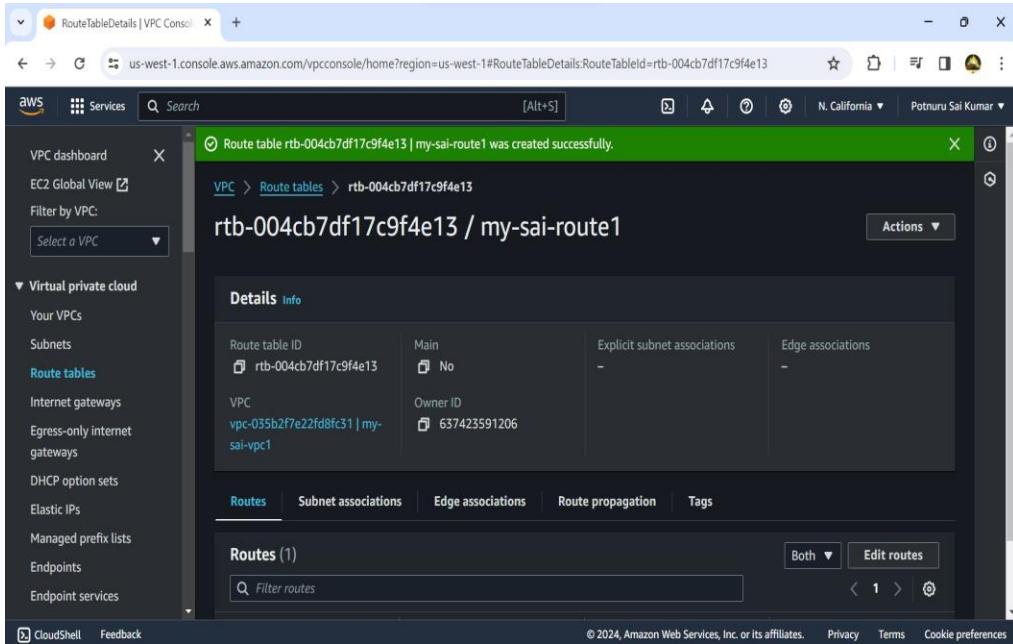
- Then click on actions and attach the internet gateway with VPC1. And now we selected our custom VPC in that available VPCs and finally click on attach internet gateway. We successfully attach internet gateway 1 to VPC1



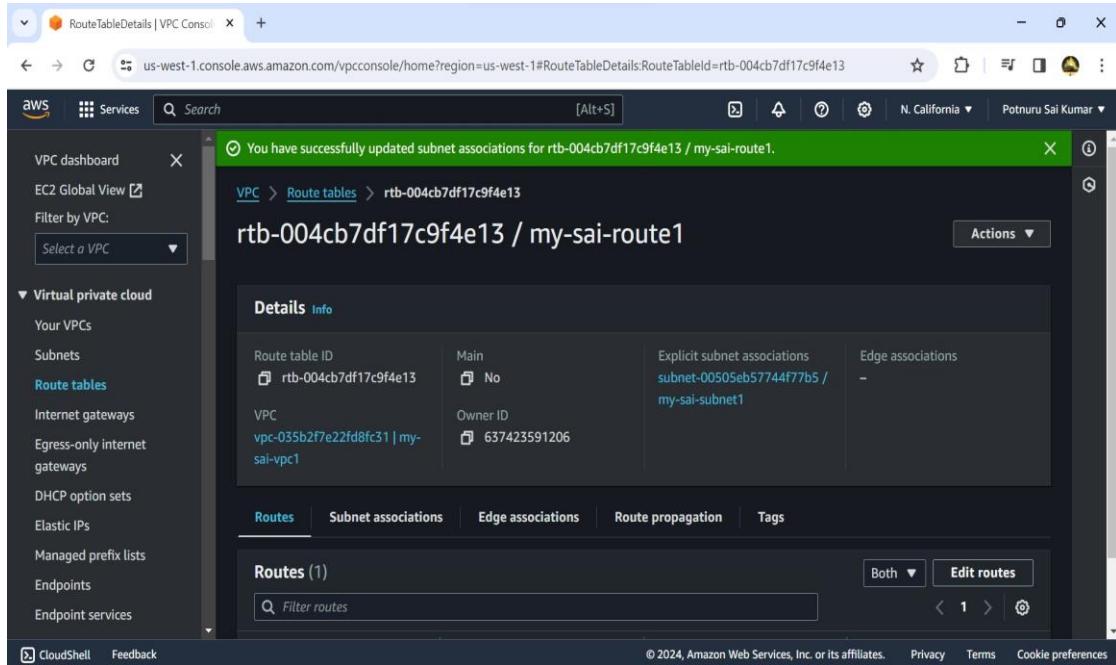
- After that we created an internet gateway. Click on Route tables from the menu bar and click on create route tables.



- Then give the name to the route table and select our custom VPC and finally click on create route table. We have successfully created route table1.



- Now we can edit the subnet associations.



- After that we can create VPC2, Search for VPC in the search space of AWS home page and click on VPC and Now click on create VPC to create our custom VPC2 and give the details and finally click on Create VPC.

The screenshots show the AWS VPC console interface.

Screenshot 1: CreateVpc | VPC Console

VPC settings

- Resources to create:** VPC only VPC and more
- Name tag - optional:** my-sai-vpc2
- IPv4 CIDR block:** IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block
180.0.0.0/16
- IPv6 CIDR block:** No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block

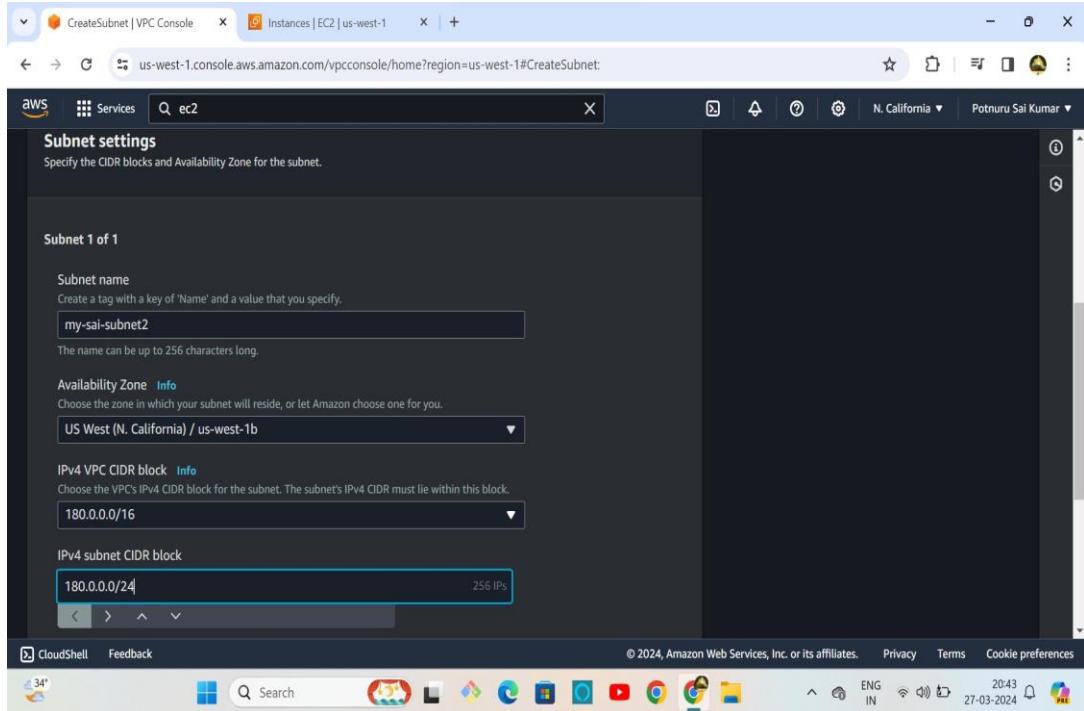
Screenshot 2: VpcDetails | VPC Console

Details

VPC ID	State	DNS hostnames	DNS resolution
vpc-013d24af0ebba2ad5	Available	Disabled	Enabled
Tenancy	DHCP option set	Main route table	Main network ACL
Default	dopt-005d799a56beaaa14	rtb-077f3e0de6a70bc9	act-0a0b7cad50aa9e723
Default VPC	IPv4 CIDR	IPv6 pool	IPv6 CIDR
No	180.0.0.0/16	-	-
Network Address Usage	Route 53 Resolver DNS	Owner ID	
metrics	Firewall rule groups	637423591206	
Disabled	-		

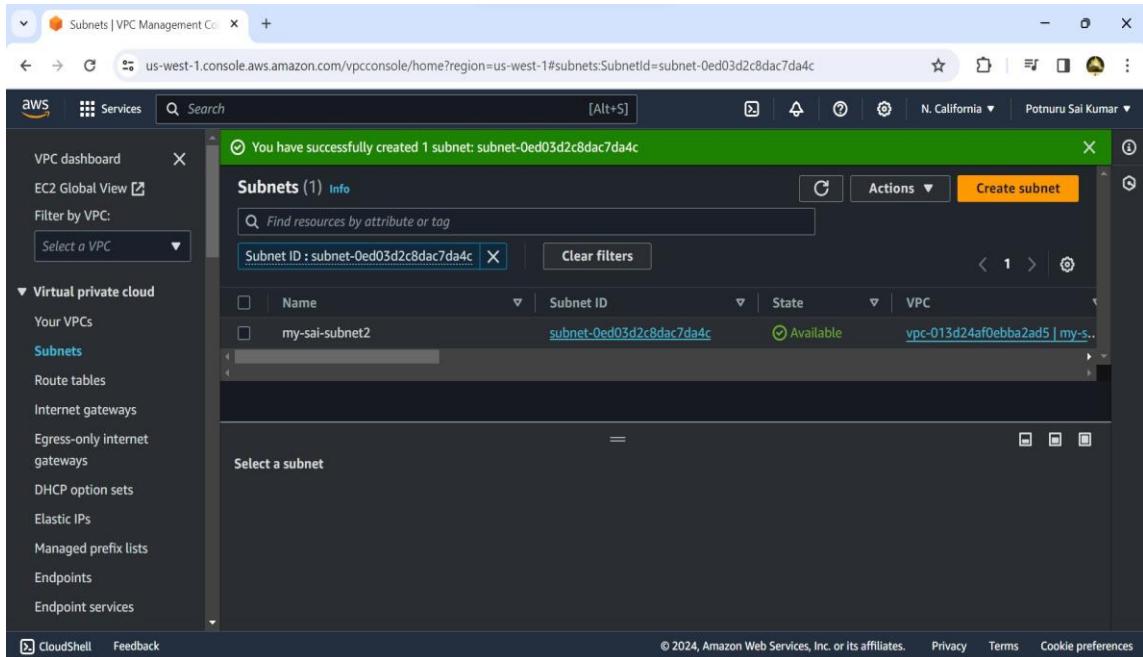
- Now we have created our custom VPC 2 successfully.

- Now click on subnets to create Subnet to our custom VPC2.
- Then create subnets we have given our custom VPC-ID, Subnet name, choose availability Zone, IPv4 subnet CIDR block, then finally create

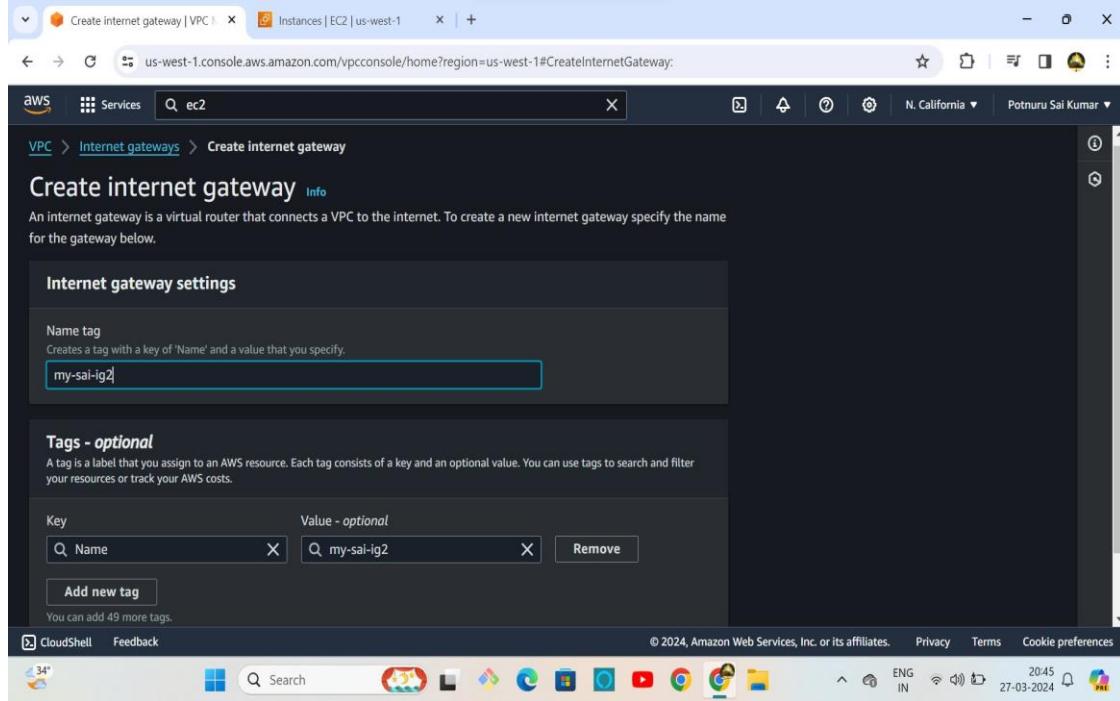


subnet.

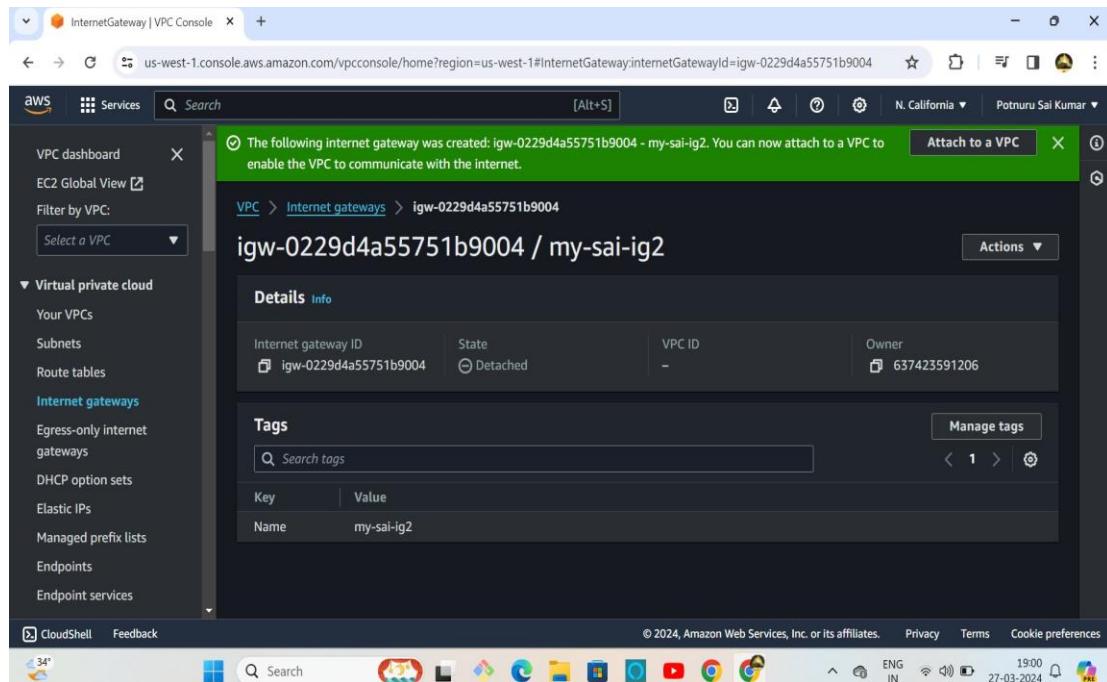
- That we successfully created a subnet 2.



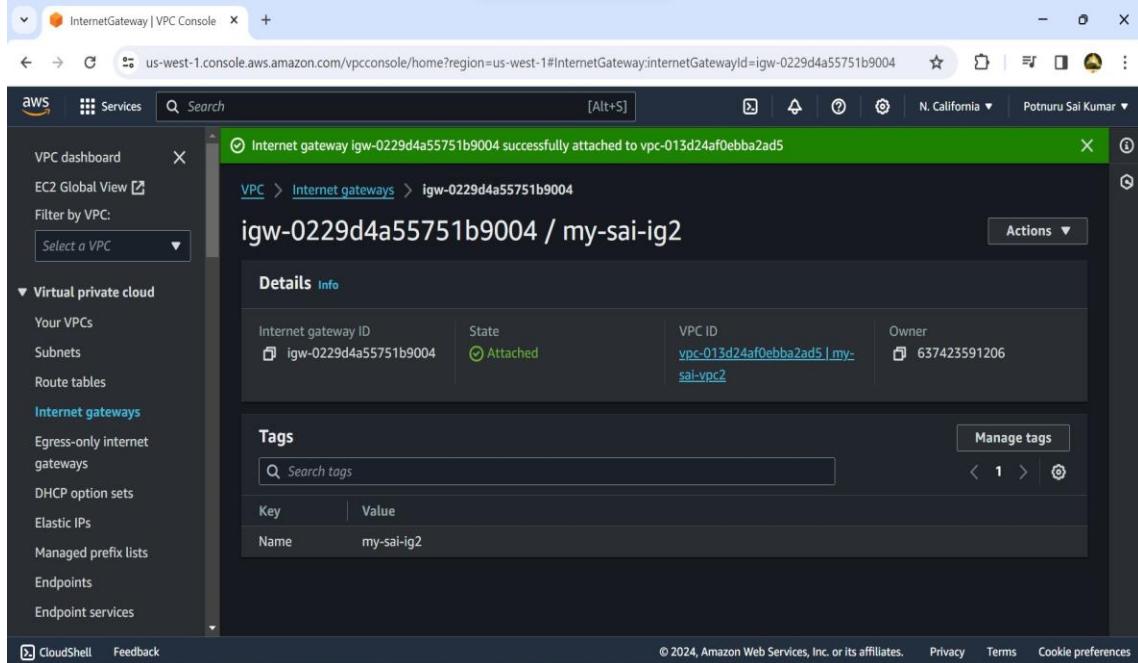
- Now click on Internet gateways from the menu bar and click on create internet gateway.



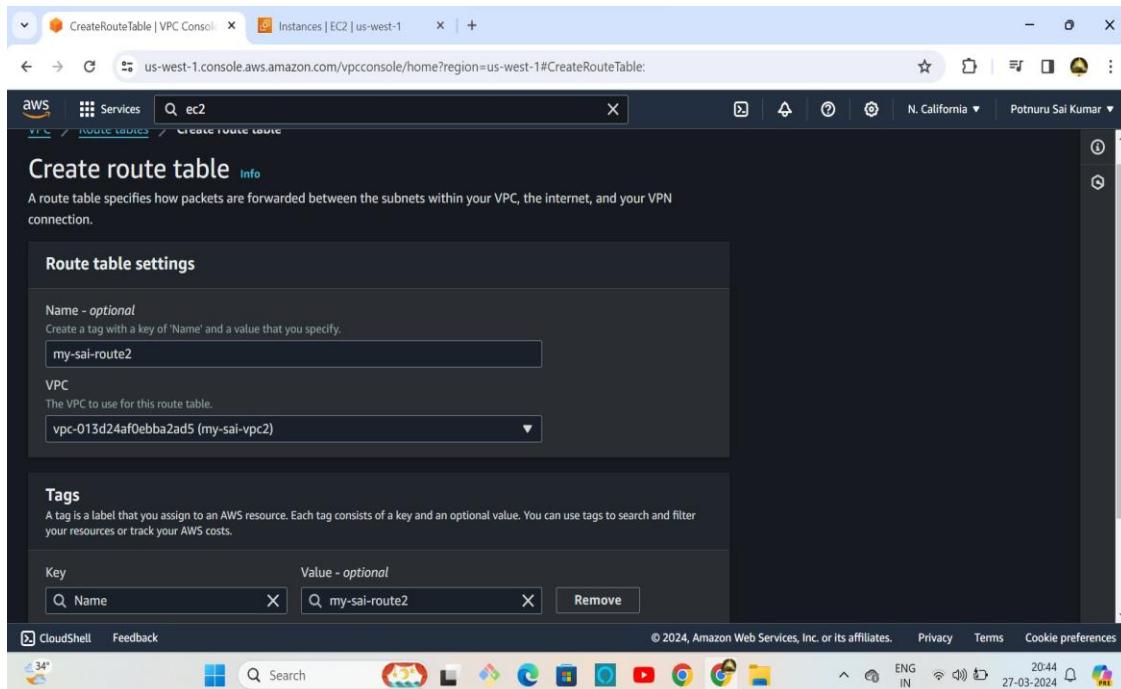
- Click on Create internet gateway.



- Then click on actions and attach the internet gateway with VPC2. And now we selected our custom VPC2 in that available VPCs and finally click on attach internet gateway.

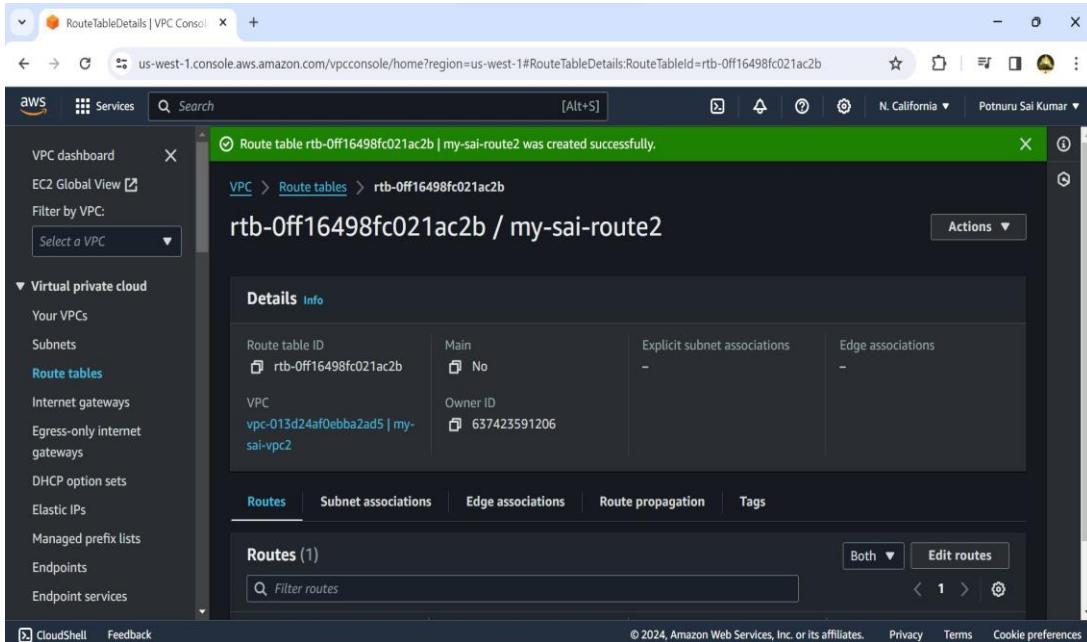


- After that we created an internet gateway. Click on Route tables from the menu bar and click on create route tables.

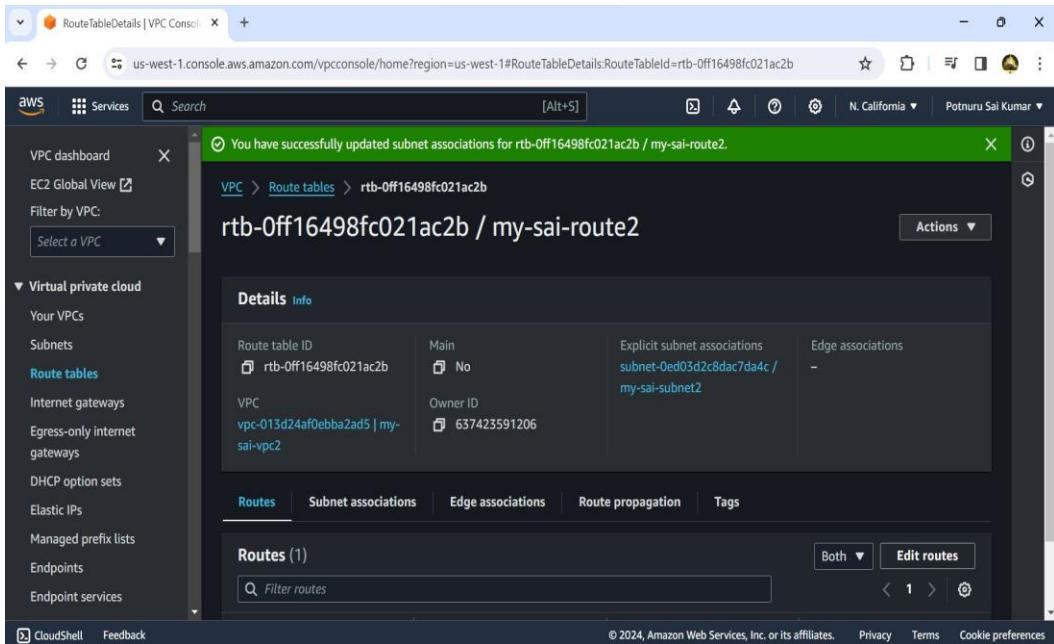


- Then give the name to the route table and select our custom VPC and finally click on create route table. We have successfully created route

table2.

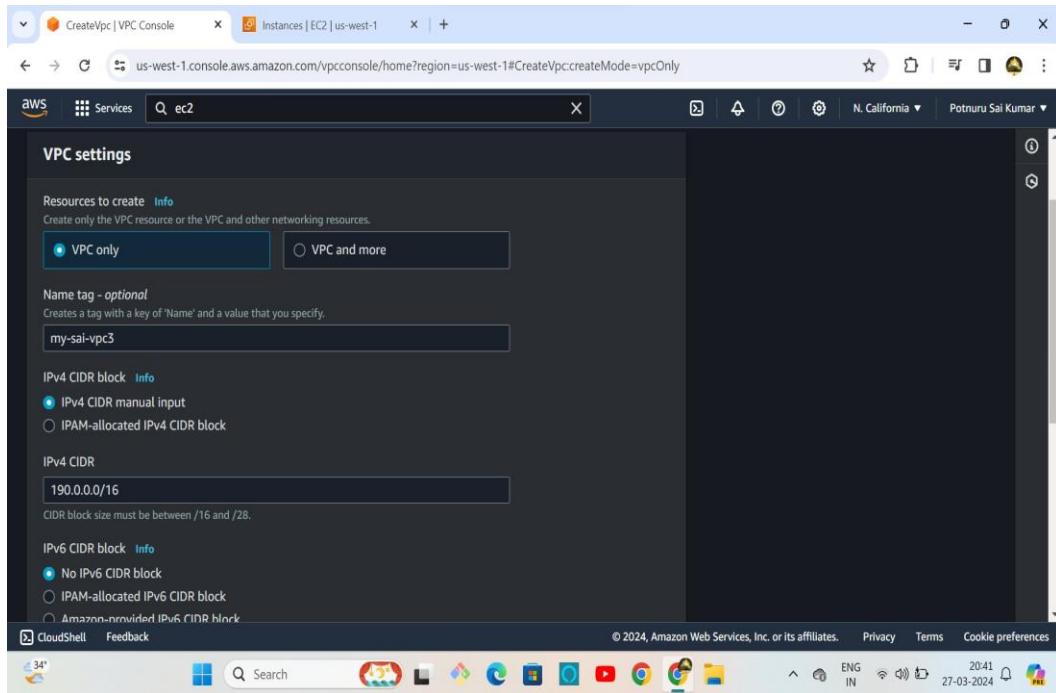


- Now we can edit the subnet associations

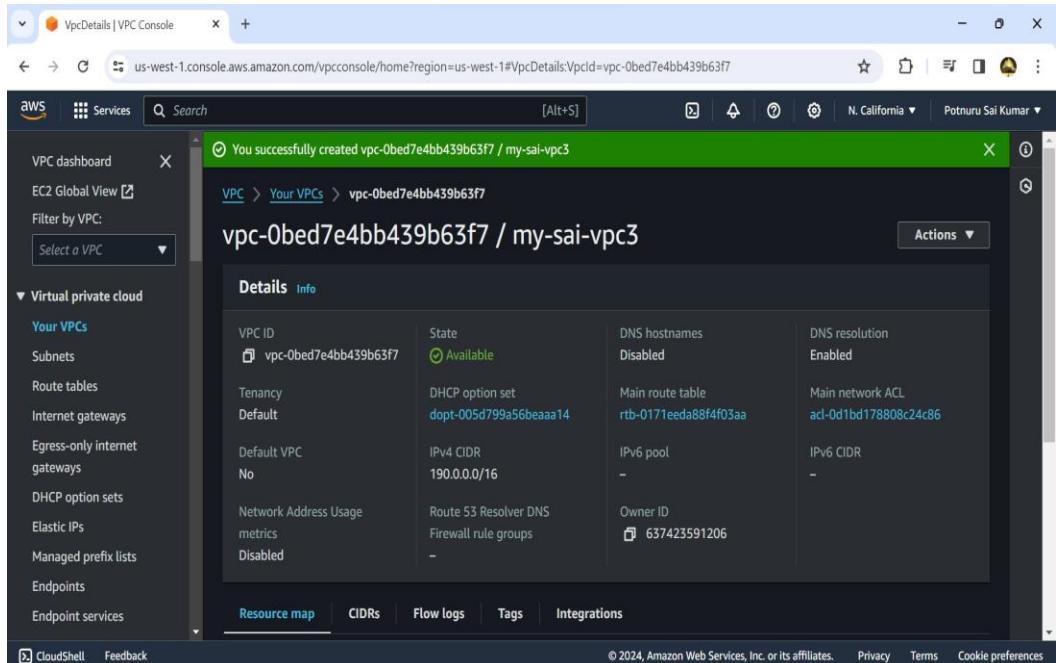


- After that we can create VPC3, Search for VPC in the search space of AWS home page and click on VPC and Now click on create VPC to create

our custom VPC3 and give the details and finally click on Create VPC.

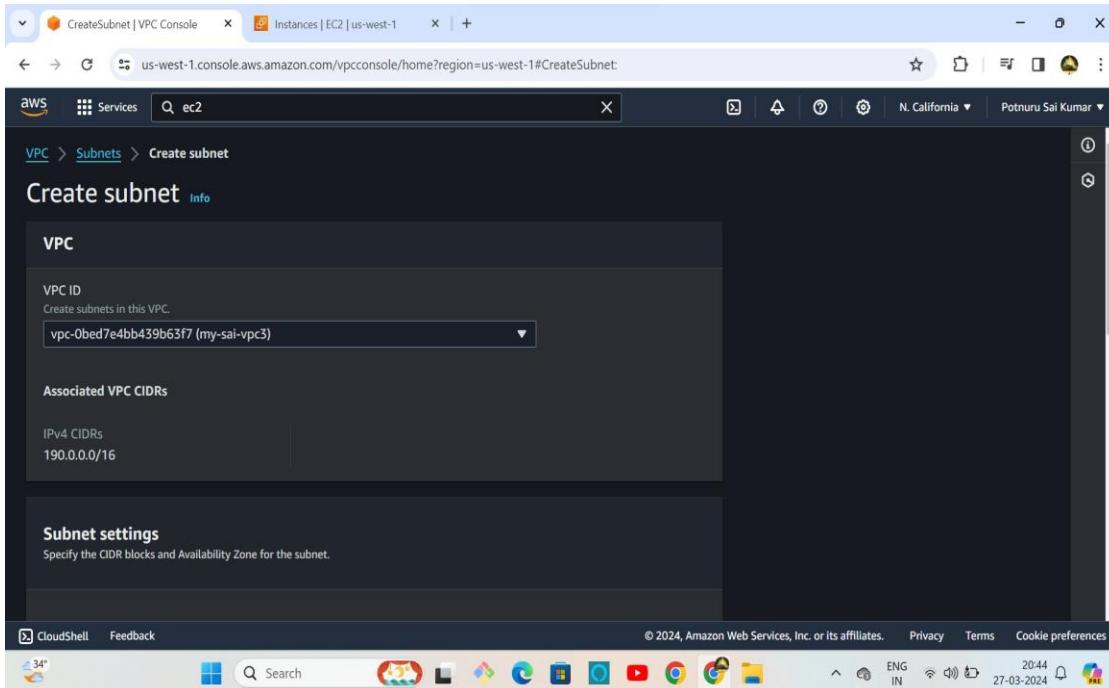


- Now we have created our custom VPC 3 successfully.



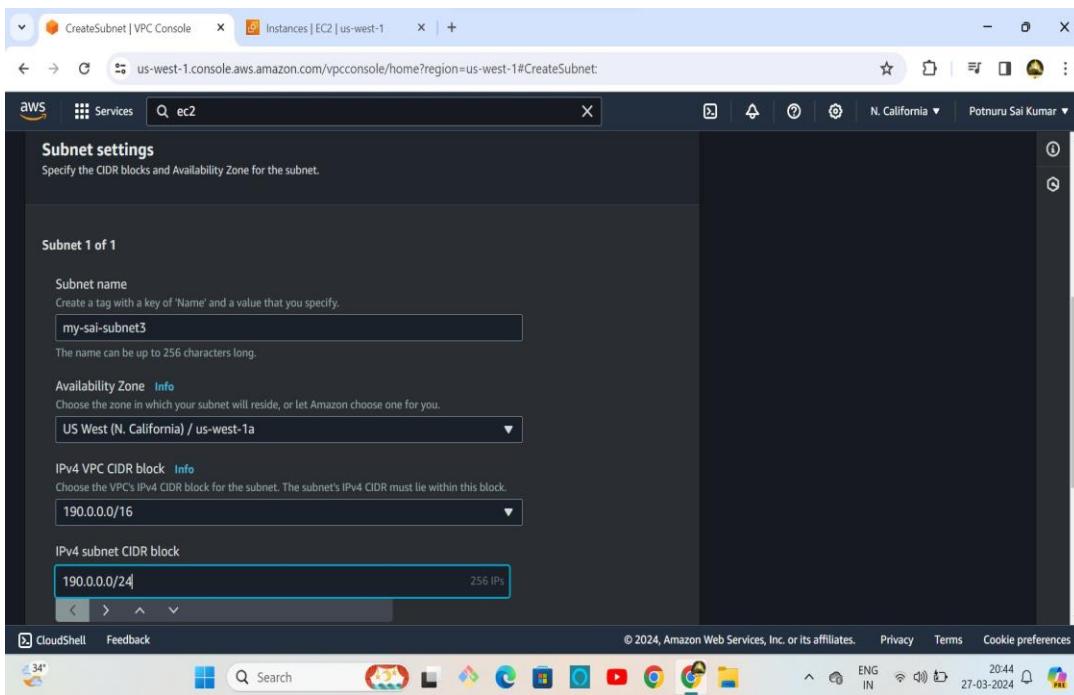
- Now click on subnets to create Subnet to our custom VPC 3.

- Then create subnets we have given our custom VPC-ID, Subnet name, choose availability Zone, IPv4 subnet CIDR block, then finally create



subnet.

- Then create subnets we have given our custom VPC-ID, Subnet name, choose availability Zone, IPv4 subnet CIDR block, then finally create subnet.



- That we successfully created a subnet 3.

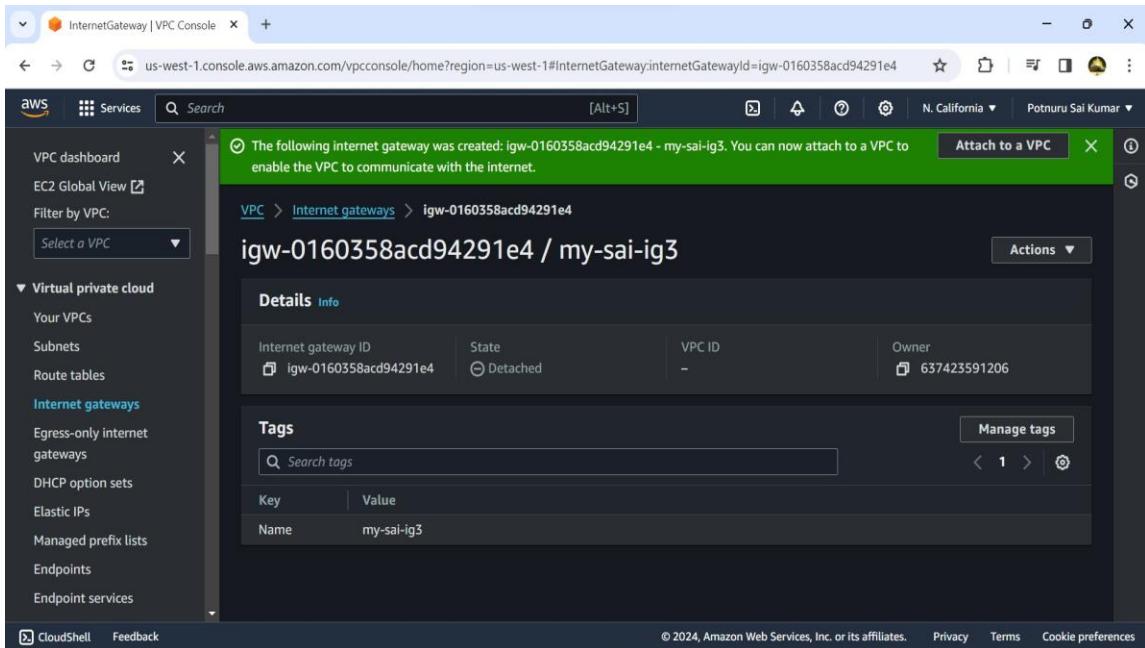
The screenshot shows the AWS VPC Management Console with the URL us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#subnets. A green success message at the top says "You have successfully created 1 subnet: subnet-088bfc6daf6890b94". The main pane displays a table titled "Subnets (1) Info" with one row. The row contains the following data:

Name	Subnet ID	State	VPC
my-sai-subnet3	subnet-088bfc6daf6890b94	Available	vpc-0bed7e4bb439b63f7 my-vpc

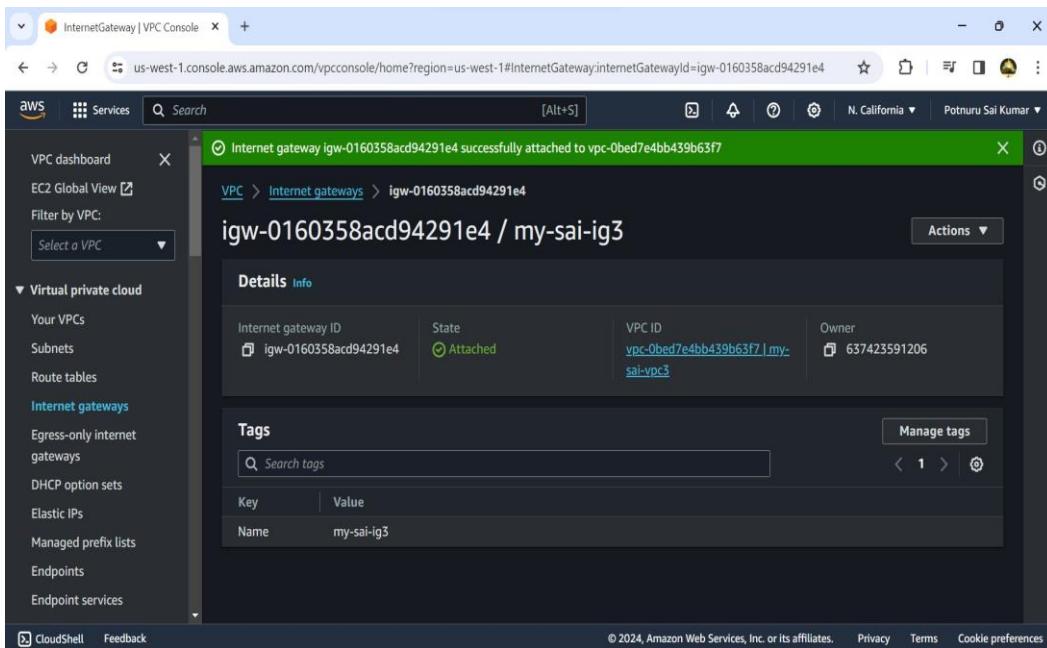
- Now click on Internet gateways from the menu bar and click on create internet gateway.

The screenshot shows the AWS EC2 Instances console with the URL us-west-1.console.aws.amazon.com/vpcconsole/home?region=us-west-1#CreateRouteTable. The page is titled "Create route table Info". It includes sections for "Route table settings" and "Tags". In the "Route table settings" section, the "Name - optional" field contains "my-sai-route3" and the "VPC" dropdown is set to "vpc-0bed7e4bb439b63f7 (my-sai-vpc3)". In the "Tags" section, there is a single tag with the key "Name" and value "my-sai-route3". The bottom of the screen shows the Windows taskbar with various icons.

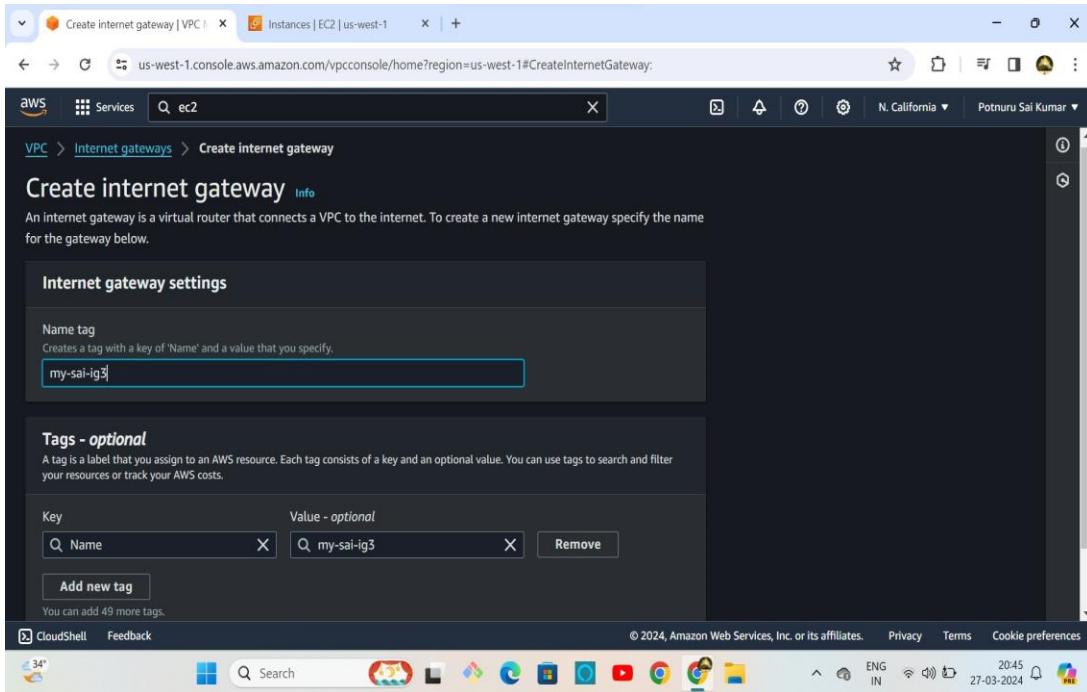
- Click on Create internet gateway.



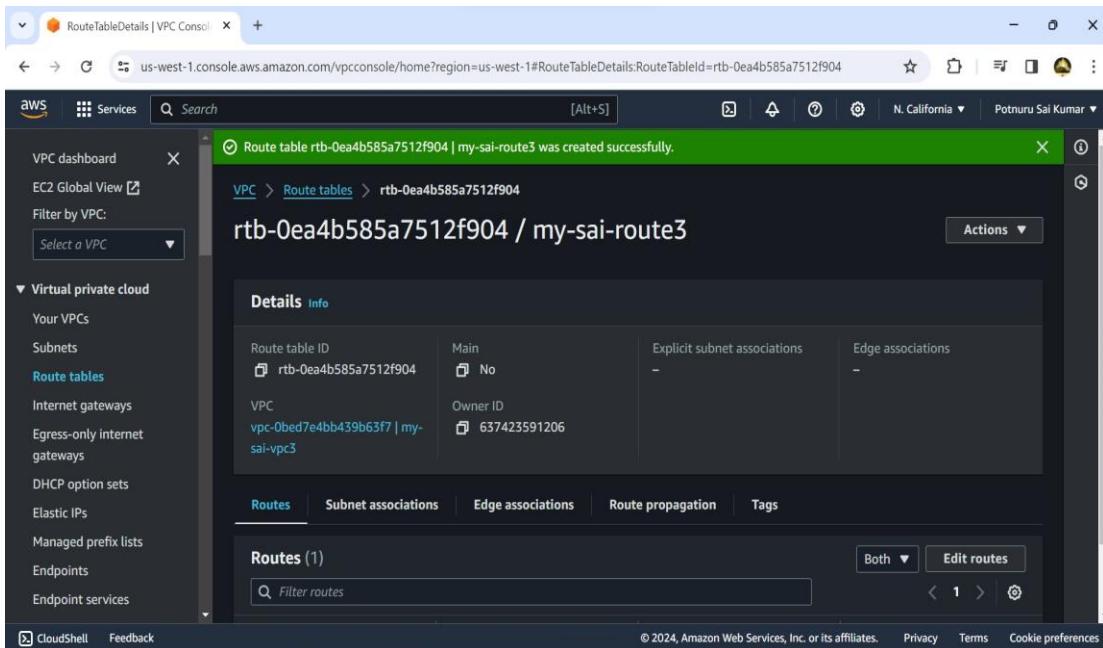
- Then click on actions and attach the internet gateway with VPC 3. And now we selected our custom VPC 3 in that available VPCs and finally click on attach internet gateway.



- After that we created an internet gateway. Click on Route tables from the menu bar and click on create route tables.



- Then give the name to the route table and select our custom VPC and finally click on create route table. We have successfully created route table3.



- Now we can edit the subnet associations.

The screenshot shows the AWS VPC Route Table Details page. A green banner at the top indicates: "You have successfully updated subnet associations for rtb-0ea4b585a7512f904 / my-sai-route3." The main panel displays route table details for "rtb-0ea4b585a7512f904 / my-sai-route3". The "Details" tab is selected, showing the following information:

Route table ID	rtb-0ea4b585a7512f904	Main	No	Explicit subnet associations	subnet-088bfcc6daff6890b94 / my-sai-subnet3	Edge associations	-
VPC	vpc-0bed7e4bb439b63f7 my-sai-vpc3	Owner ID	637423591206				

Below the details, there are tabs for Routes, Subnet associations, Edge associations, Route propagation, and Tags. The Routes tab shows one route entry: "Routes (1)". At the bottom right of the main panel, there are buttons for Both, Edit routes, and a search bar. The left sidebar shows the VPC navigation menu with "Route tables" selected. The bottom of the screen includes standard AWS footer links and a CloudShell button.

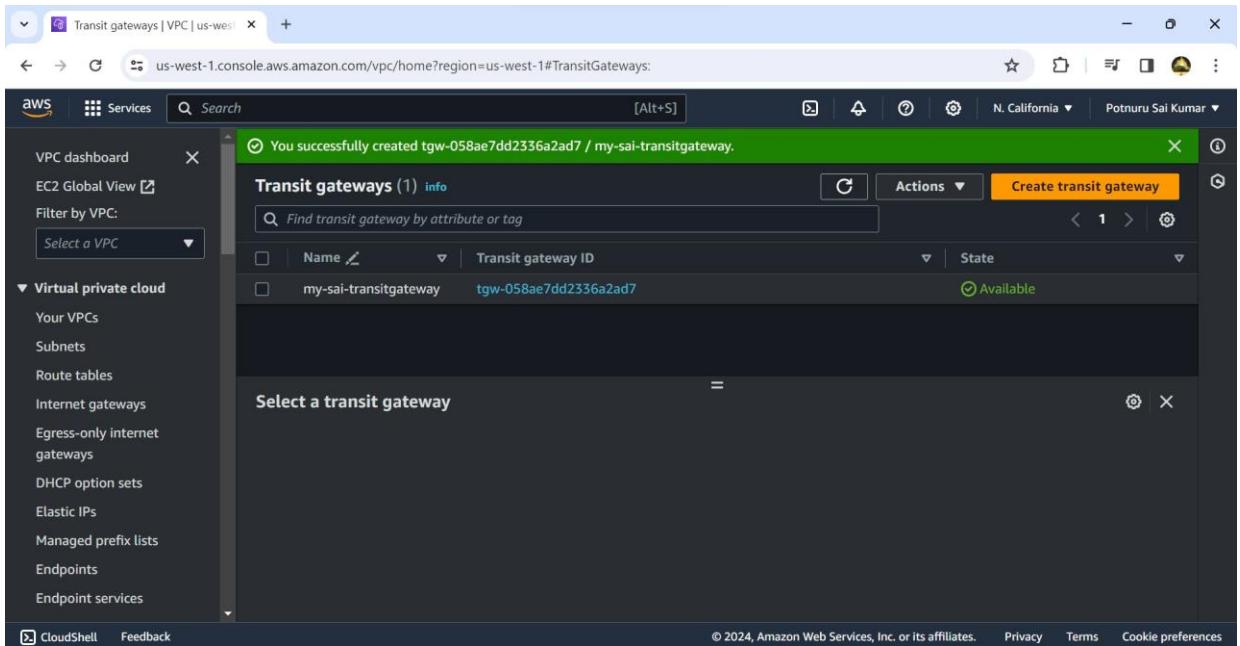
- TRANSIT GATEWAY
- We can create a transit gateway.
- Create transit gateway details.

The screenshot shows the AWS VPC Create Transit Gateway page. The URL is "us-west-1.console.aws.amazon.com/vpc/home?region=us-west-1#CreateTransitGateway:". The main panel is titled "Create transit gateway" and includes the following sections:

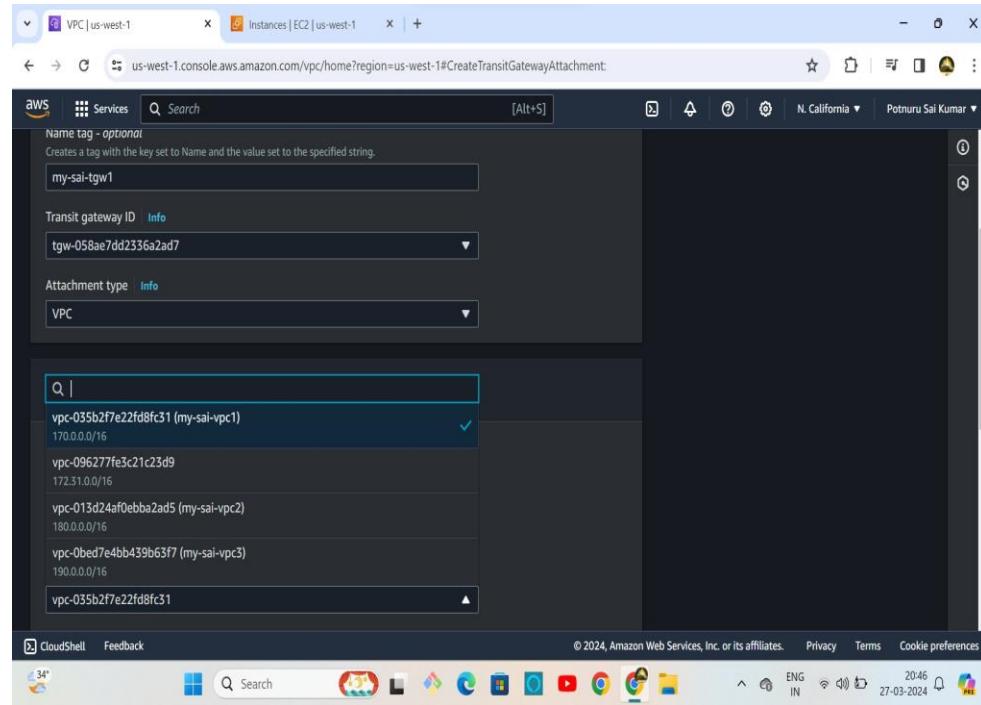
- Details - optional**: Fields for Name tag (containing "my-sai-transitgateway") and Description (containing "description").
- Configure the transit gateway**: Field for Amazon side Autonomous System Number (ASN) (containing "ASN").

At the bottom, there are standard AWS footer links and a CloudShell button. The left sidebar shows the VPC navigation menu with "Transit gateways" selected. The bottom of the screen includes a taskbar with various icons and system status information.

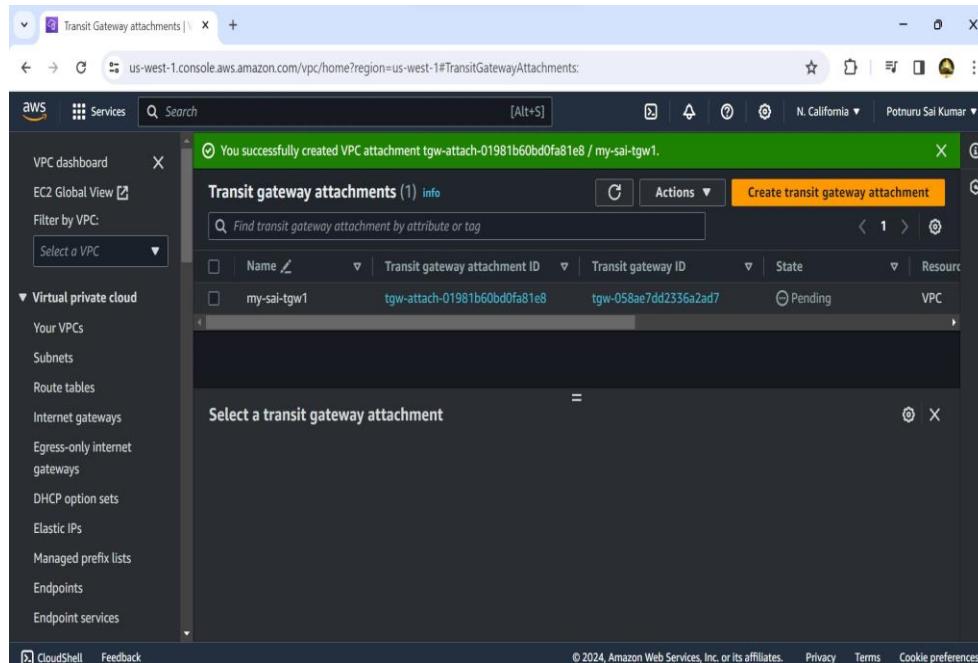
- We successfully created the transit gateway.



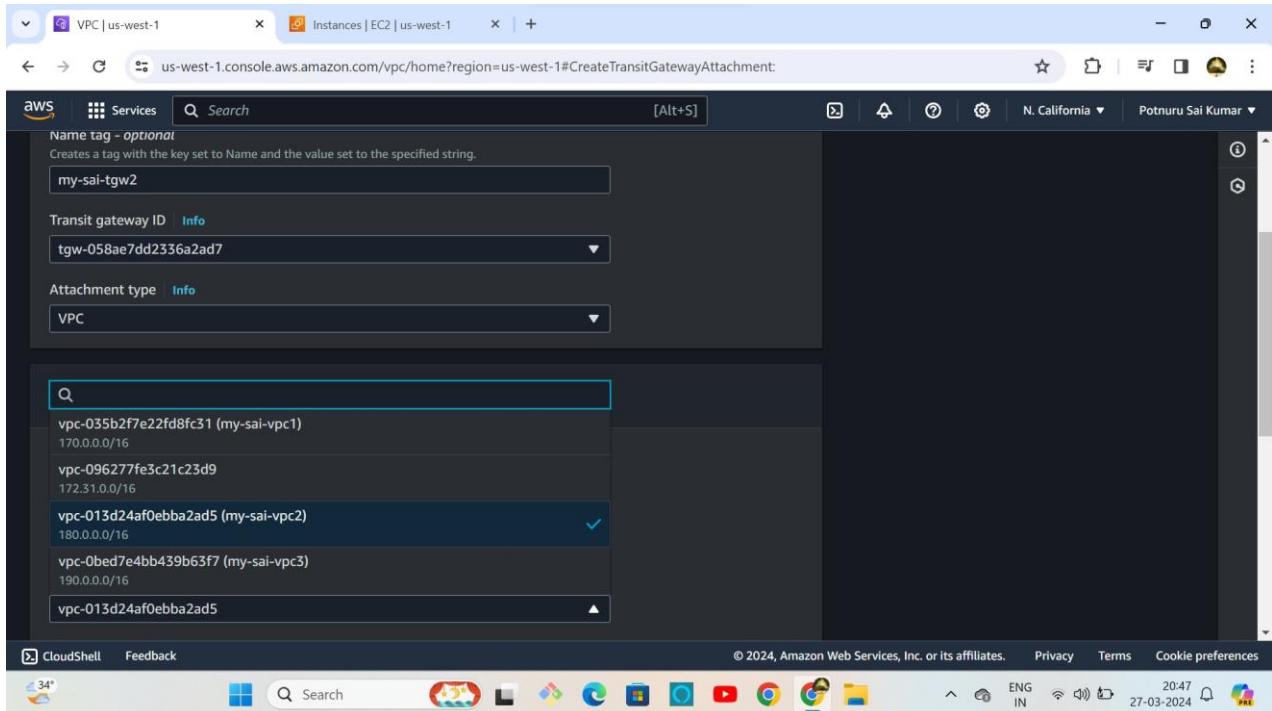
- Now we can create three transit gateway attachments, to attach the EC2 instances.
- First, we create a first transit gateway. And create a transit gateway attach name, connect with transit gateway and attach with VPC 1 to transit gateway attachment 1.



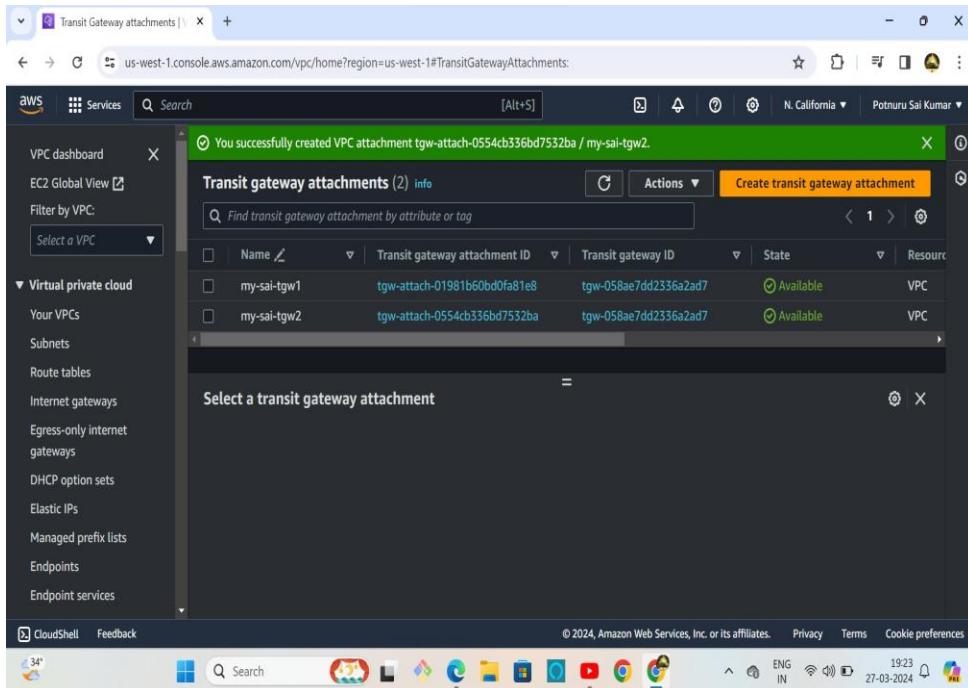
- Click on the create attachment that we successfully created the transit gateway attachment 1.



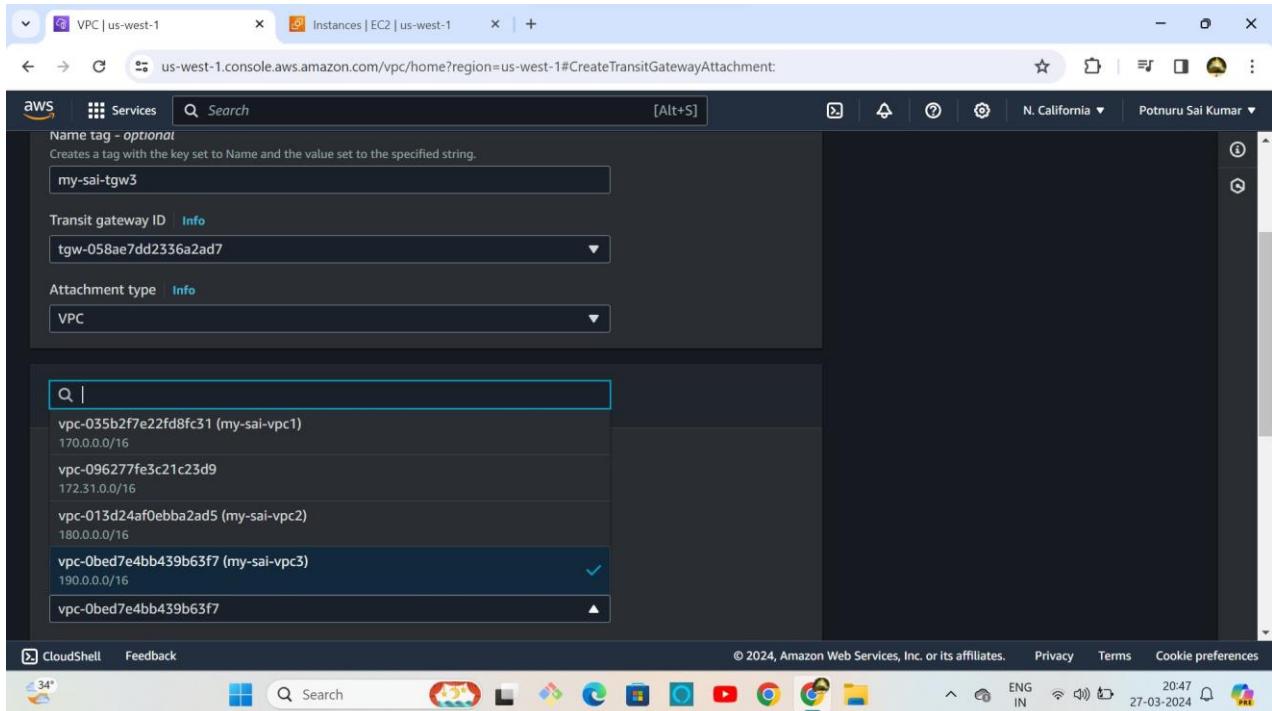
- First, we create a first transit gateway. And create a transit gateway attach name, connect with transit gateway and attach with VPC 2 to transit gateway attachment 2.



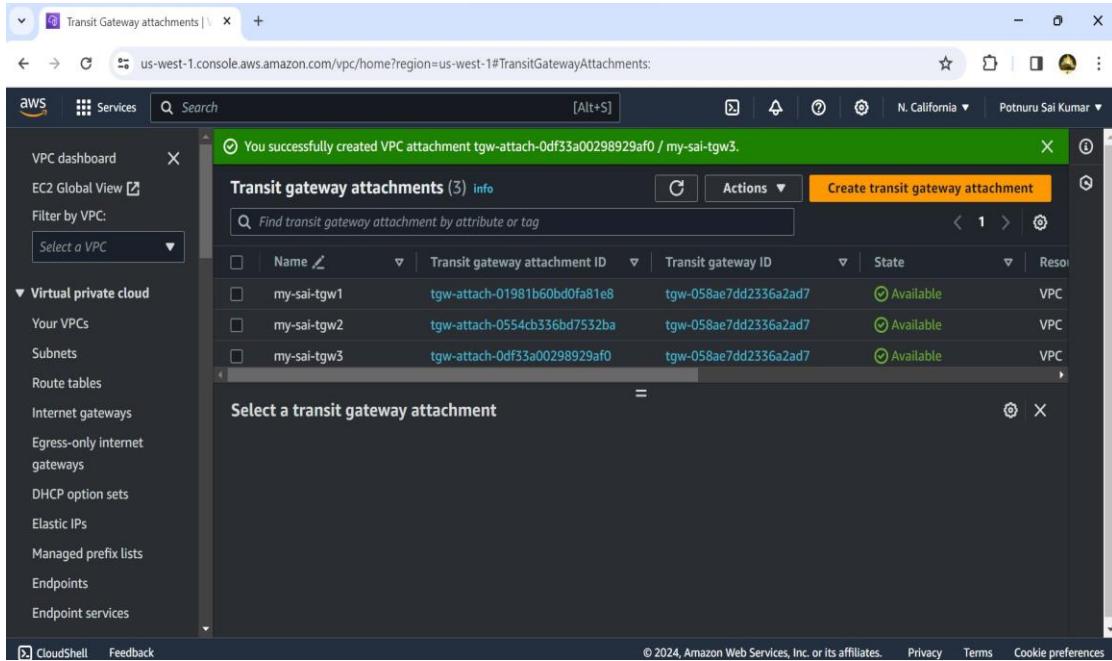
- Click on the create attachment that we successfully created the transit gateway attachment 2.



- Then we created a first transit gateway. And create a transit gateway attach name, connect with transit gateway and attach with VPC 3 to transit gateway attachment 3.



- Click on the create attachment that we successfully created the transit gateway attachment 3.



- After creating the transit gateway now, we can connect the transit gateway to route tables 1. We can edit routes to connect the transit gateway to connect the VPCs.

- We have successfully added the transit gateway to the routes table 1.

- After creating the transit gateway now, we can connect the transit gateway to route 2. We can edit routes to connect the transit gateway to connect the VPCs.

Destination	Target	Status	Propagated
180.0.0.0/16	local	Active	No
170.0.0.0/16	Transit Gateway	-	No
190.0.0.0/16	Transit Gateway	-	No
0.0.0.0/0	Internet Gateway	-	No

Destination	Target	Status	Propagated
170.0.0.0/16	local	Active	No
0.0.0.0/0	Internet Gateway	-	No
180.0.0.0/16	Transit Gateway	-	No
190.0.0.0/16	Transit Gateway	-	No

- We have successfully added the transit gateway to the routes table 2.
- After creating the transit gateway now, we can connect the transit gateway to route 3. We can edit routes to connect the transit gateway to connect the VPCs.

The screenshot shows the 'Edit routes' page in the AWS VPC Console. The table lists the following routes:

Destination	Target	Status	Propagated
190.0.0.0/16	local	Active	No
Q 0.0.0.0/0	Internet Gateway	-	No
Q 170.0.0.0/16	Transit Gateway	-	No
Q 180.0.0.0/16	Transit Gateway	-	No

Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

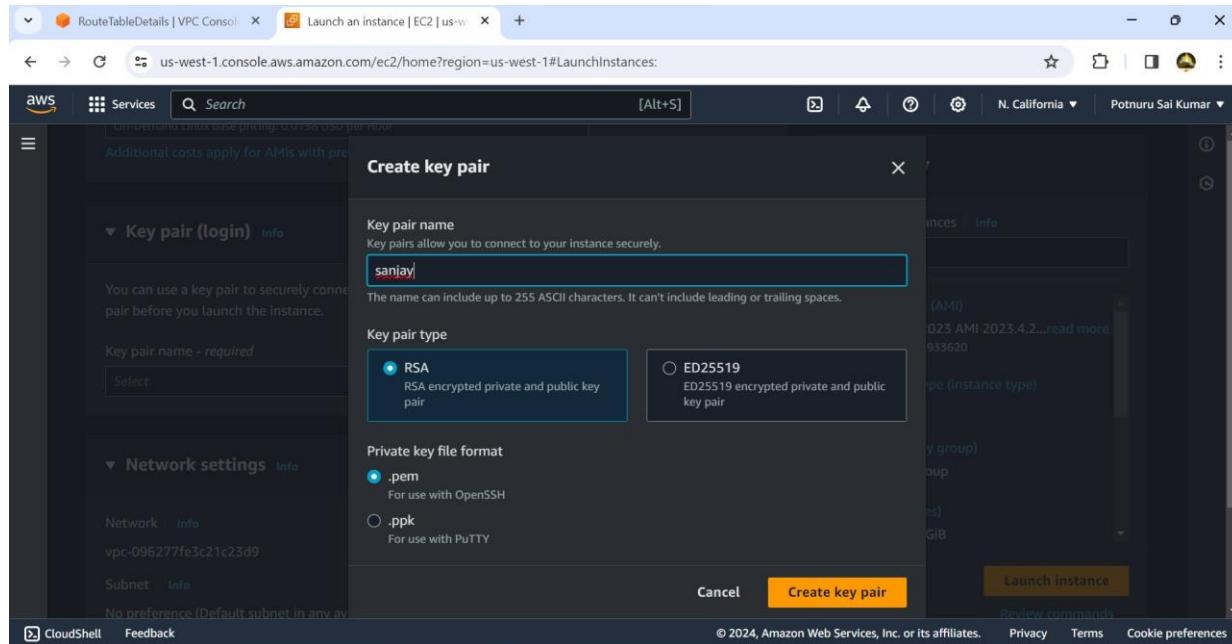
- We have successfully added the transit gateway to the routes table 3.

The screenshot shows the 'RouteTableDetails' page for route table `rtb-0ea4b585a7512f904 / my-sai-route3`. A success message indicates that routes were updated successfully. The 'Details' tab displays the following information:

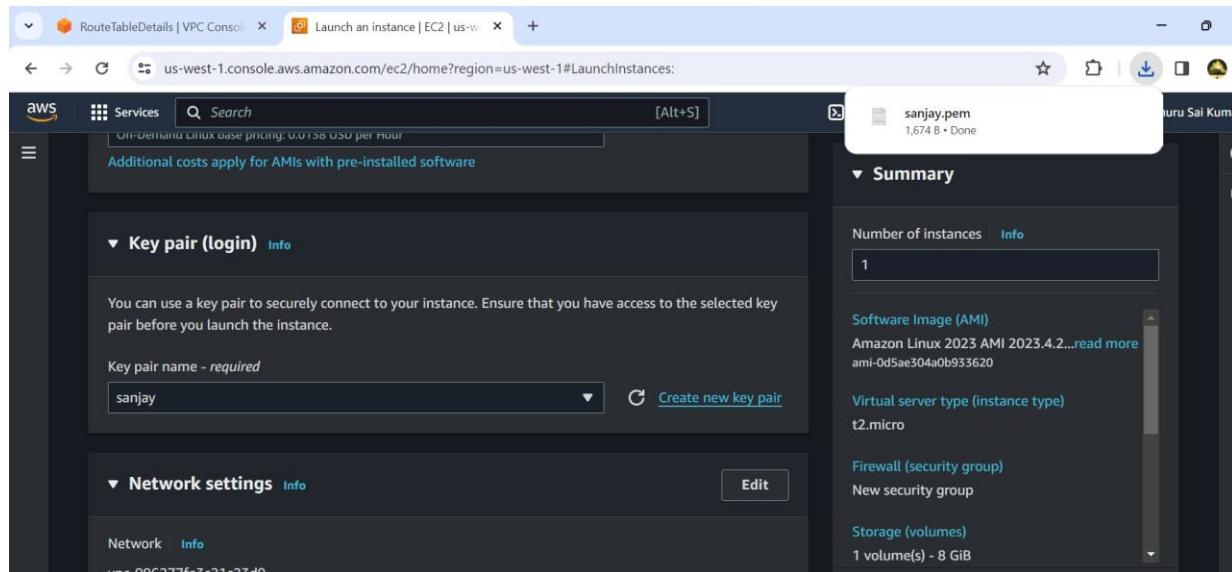
Route table ID: <code>rtb-0ea4b585a7512f904</code>	Main: <code>No</code>	Explicit subnet associations: <code>subnet-088bfc6daff6890b94 / my-sai-subnet3</code>	Edge associations: -
VPC: <code>vpc-0bed7e4bb439b63f7 my-sai-vpc3</code>	Owner ID: <code>637423591206</code>		

The 'Routes' tab shows 4 routes. Buttons at the bottom include 'Actions', 'Edit routes', and 'Filter routes'.

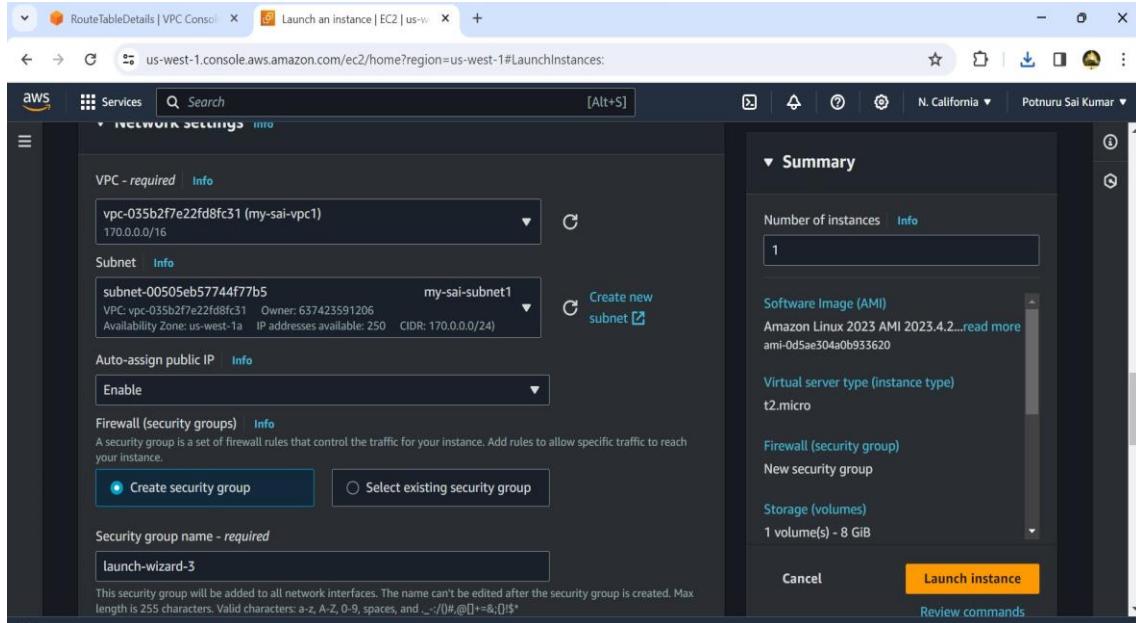
- ★ Now we can launch the three EC2 instances.
- ★ First we launch the instance, create the name of the instances and create the key pair and the type of the key pair is RSA and the file format is .Pem.
- ★ After clicking on the create key pair.



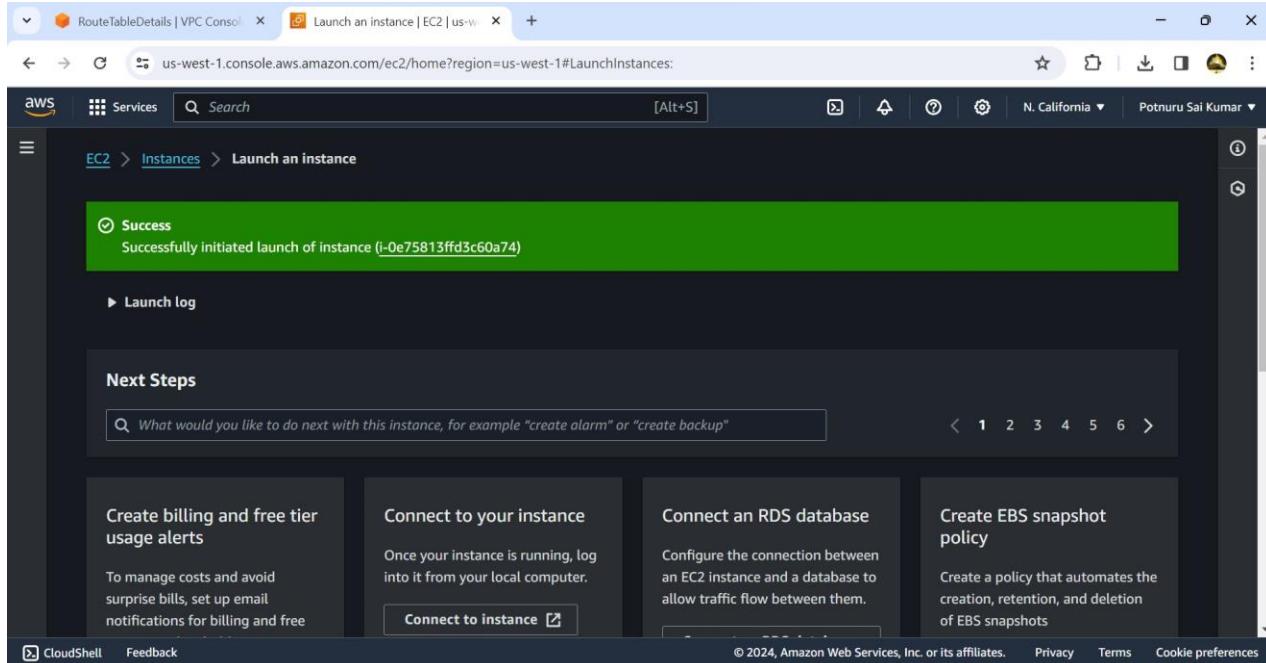
- Now the key pair will download to the desktop. The sanjay.Pem file was downloaded at top of the desktop.



- After downloading the key pair now, we can edit the network settings and select VPC 1 and subnet 1 and finally enable the instance. That click on the launch instance.



- Now the first EC2 instance was successfully initiated.



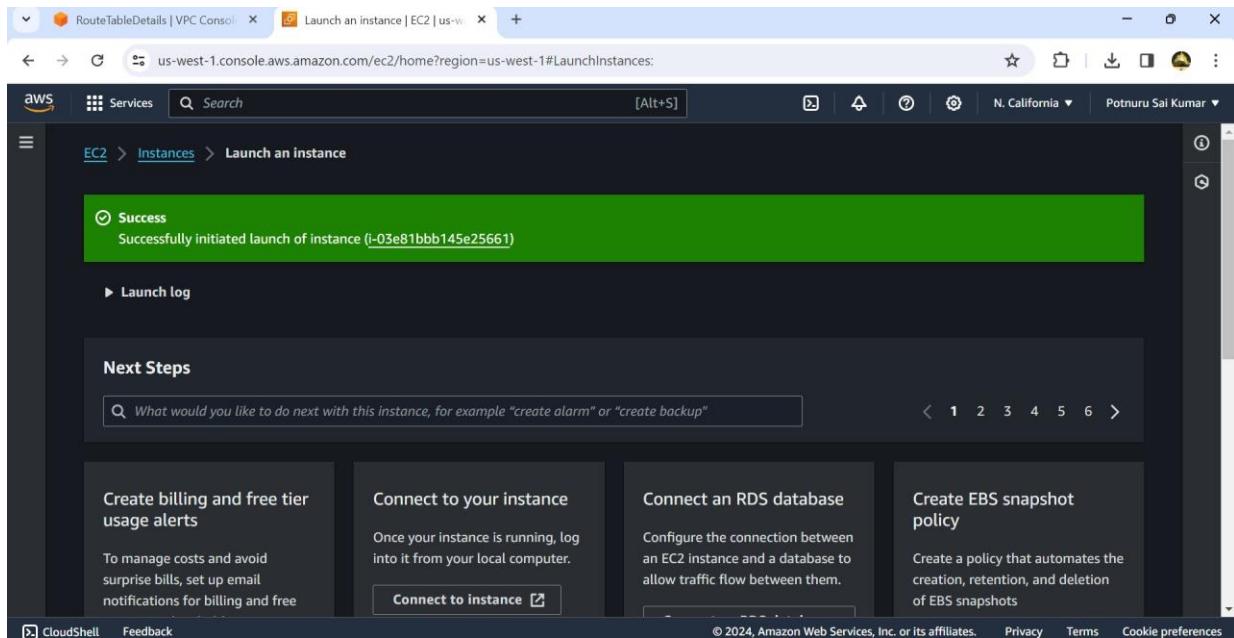
- Now we create a second instance after downloading the key pair now we can edit the network settings and select VPC 1 and subnet 1 and finally enable the instance. That click on the launch instance.

The screenshot shows the AWS VPC console with the 'Launch an instance' wizard. In the 'Network settings' section, a VPC ('my-sai-vpc2') and a subnet ('my-sai-subnet2') are selected. Under 'Auto-assign public IP', 'Enable' is chosen. A security group ('Create security group') is selected. The 'Resource tags' sidebar defines a tag key ('Key') as 'Up to 128 Unicode characters in UTF-8' and a tag value ('Value') as 'Optional tag value up to 256 characters in UTF-8'. The 'Resource types' sidebar lists the resource type as 'The resource type on which the tag is applied'. The status bar at the bottom indicates the instance is launching.

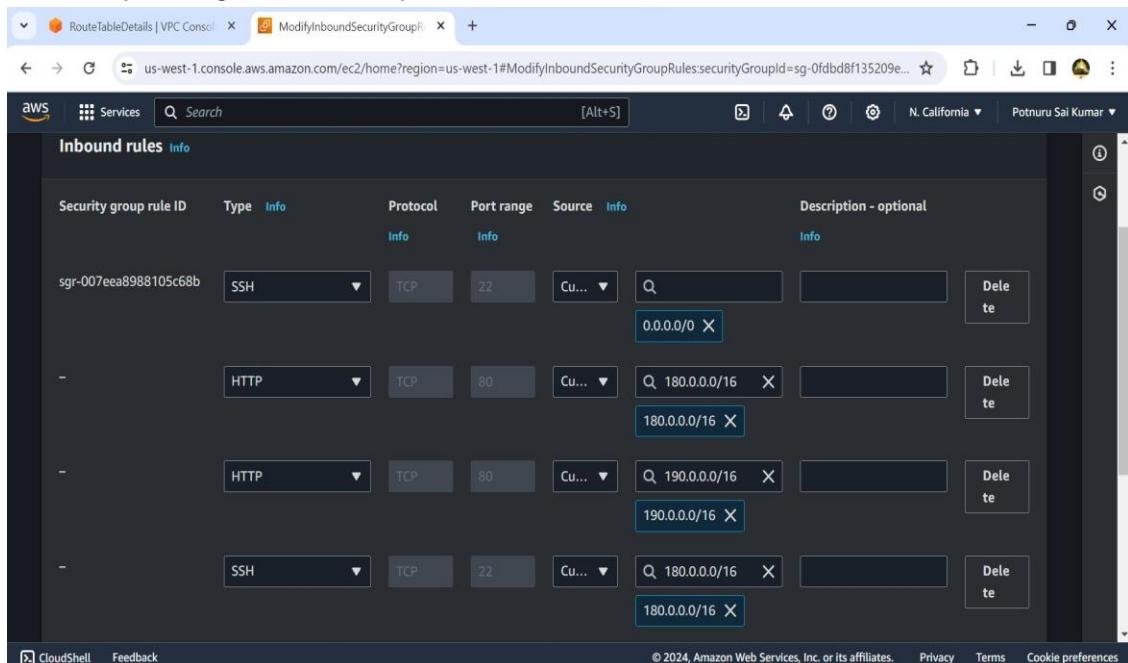
- Now the second EC2 instance was successfully initiated.

The screenshot shows the AWS EC2 Instances page. A green success message states 'Successfully initiated launch of instance i-079db7e9c5244f7a3'. Below it, a 'Launch log' button is visible. The 'Next Steps' section includes links for 'Create billing and free tier usage alerts', 'Connect to your instance' (with a 'Connect to instance' button), 'Connect an RDS database', and 'Create EBS snapshot policy'. The status bar at the bottom indicates the instance has been successfully launched.

- Finally, we can third instance successfully initiated.



- By using the security id, we can edit the inbound rules of the ec1 instance.



Finally, the rules update.

The screenshot shows the AWS EC2 Security Groups page. A green banner at the top states: "Inbound security group rules successfully modified on security group (sg-0fdbd8f135209e8bf | launch-wizard-3)". Below this, the security group details are shown: name "sg-0fdbd8f135209e8bf - launch-wizard-3", owner "637423591206", security group ID "sg-0fdbd8f135209e8bf", description "launch-wizard-3 created 2024-03-27T14:01:46.858Z", and VPC ID "vpc-035b2f7e22fd8fc31". The "Inbound rules" tab is selected. At the bottom, there are links for CloudShell, Feedback, and Copyright information.

- By using the security id, we can edit the inbound rules of the ec2 instance.

The screenshot shows the "Inbound rules" configuration page for a specific security group. It lists four existing rules:

- Rule 1: Type SSH, Protocol TCP, Port range 22, Source 0.0.0.0/0
- Rule 2: Type SSH, Protocol TCP, Port range 22, Source 170.0.0.0/16
- Rule 3: Type SSH, Protocol TCP, Port range 22, Source 190.0.0.0/16
- Rule 4: Type HTTP, Protocol TCP, Port range 80, Source 170.0.0.0/16

Each rule has a "Delete" button to its right. The page includes tabs for "Info" and "Description - optional". At the bottom, there are links for CloudShell, Feedback, and Copyright information.

- By using the security id, we can edit the inbound rules of the ec3 instance.

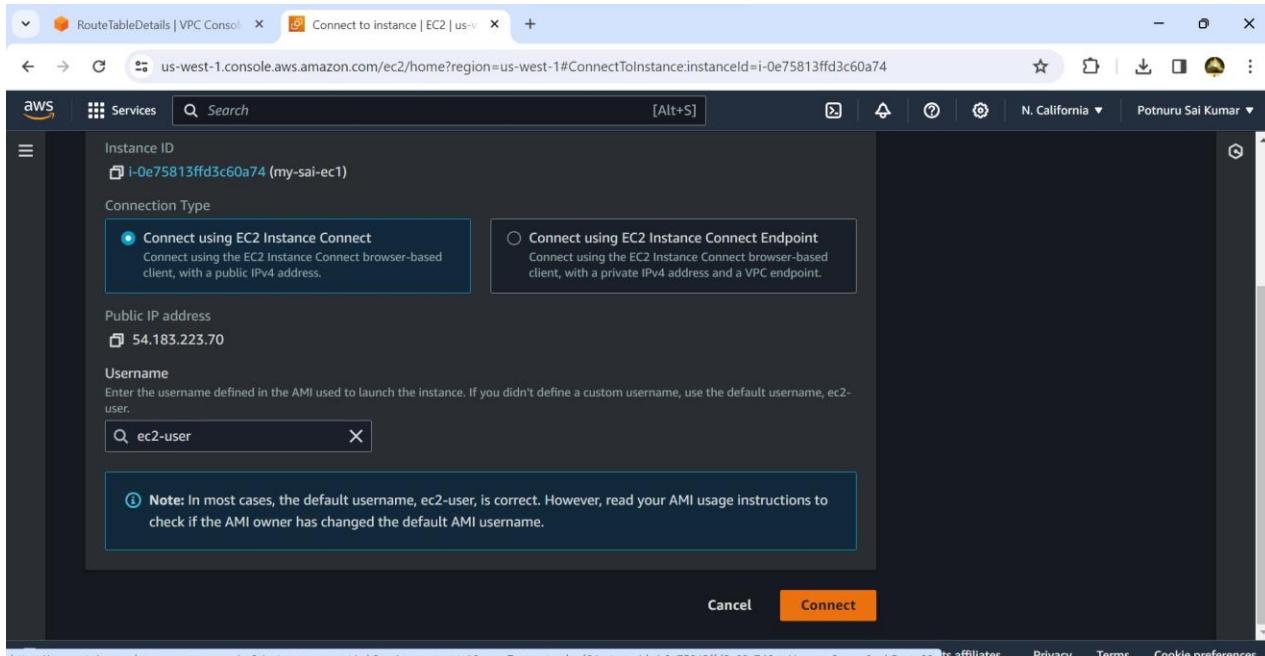
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0bed87a1eaa9006ee	SSH	TCP	22	Cu... ▾	Q 0.0.0.0/0 X
-	SSH	TCP	22	Cu... ▾	Q 170.0.0.0/16 X
-	SSH	TCP	22	Cu... ▾	Q 180.0.0.0/16 X
-	HTTP	TCP	80	Cu... ▾	Q 170.0.0.0/16 X

We are successful added.

Inbound security group rules successfully modified on security group (sg-0ad47a7be4e485a08 | launch-wizard-5)

Details			
Security group name: launch-wizard-5	Security group ID: sg-0ad47a7be4e485a08	Description: launch-wizard-5 created 2024-03-27T14:07:00.634Z	VPC ID: vpc-0bed7e4bb439b63f7
Owner: 637423591206	Inbound rules count: 5 Permission entries	Outbound rules count: 1 Permission entry	

- Finally, we connect the first ec1 instances. Click on the connect to connect to the server.



- Then we connect to the ec1 instances.

```
Complete!
[root@ip-170-0-0-44 ~]# cd /usr/share/nginx/html
[root@ip-170-0-0-44 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-170-0-0-44 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-170-0-0-44 html]# vi index.html
[root@ip-170-0-0-44 html]# systemctl restart nginx
[root@ip-170-0-0-44 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-170-0-0-44 html]# curl 54.183.223.70:80
curl: (28) Failed to connect to 54.183.223.70 port 80 after 129180 ms: Couldn't connect to server
[root@ip-170-0-0-44 html]# curl 54.183.223.70
curl: (28) Failed to connect to 54.183.223.70 port 80 after 130855 ms: Couldn't connect to server
[root@ip-170-0-0-44 html]# curl 54.183.223.70
this is from vpc1
[root@ip-170-0-0-44 html]# curl 190.0.0.45:80
this is from vpc3
[root@ip-170-0-0-44 html]#
```

i-0e75813ffd3c60a74 (my-sai-ec1)

PublicIPs: 54.183.223.70 PrivateIPs: 170.0.0.44

Ec2

```

aws Services Search [Alt+S] N. California Potnuru Sai Kumar
nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[root@ip-180-0-0-114 ~]# cd /usr/share/nginx/html
[root@ip-180-0-0-114 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-180-0-0-114 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-180-0-0-114 html]# vi index.html
[root@ip-180-0-0-114 html]# systemctl restart nginx
[root@ip-180-0-0-114 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-180-0-0-114 html]# curl 54.177.163.18:80
curl: (28) Failed to connect to 54.177.163.18 port 80 after 133726 ms: Couldn't connect to server
[root@ip-180-0-0-114 html]# curl 54.177.163.18
this is from vpc2
[root@ip-180-0-0-114 html]# curl 190.0.0.45:80
this is from vpc3
[root@ip-180-0-0-114 html]# 

```

i-079db7e9c5244f7a3 (my-sai-ec2)

Public IPs: 54.177.163.18 Private IPs: 180.0.0.114

Ec3

```

aws Services Search [Alt+S] N. California Potnuru Sai Kumar
Verifying : libunwind-1.4.0-5.amzn2023.0.2.x86_64          4/7
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch  5/7
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 6/7
Verifying : nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch 7/7

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

Complete!
[root@ip-190-0-0-45 ~]# cd /usr/share/nginx/html
[root@ip-190-0-0-45 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-190-0-0-45 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-190-0-0-45 html]# vi index.html
[root@ip-190-0-0-45 html]# systemctl restart nginx
[root@ip-190-0-0-45 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-190-0-0-45 html]# curl 13.52.177.250
this is from vpc3
[root@ip-190-0-0-45 html]# curl 170.0.0.44:80
this is from vpc1
[root@ip-190-0-0-45 html]# 

```

- Finally, we can connect from one instance to another instance by using the transit gateway.

```
Complete!
[root@ip-170-0-0-44 ~]# cd /usr/share/nginx/html
[root@ip-170-0-0-44 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-170-0-0-44 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-170-0-0-44 html]# vi index.html
[root@ip-170-0-0-44 html]# systemctl restart nginx
[root@ip-170-0-0-44 html]# ls
404.html 50x.html icons index.html nginx-logo.png poweredby.png
[root@ip-170-0-0-44 html]# curl 54.183.223.70:80
curl: (28) Failed to connect to 54.183.223.70 port 80 after 129180 ms: Couldn't connect to server
[root@ip-170-0-0-44 html]# curl 54.183.223.70
curl: (28) Failed to connect to 54.183.223.70 port 80 after 130855 ms: Couldn't connect to server
[root@ip-170-0-0-44 html]# curl 54.183.223.70
this is from vpc1
[root@ip-170-0-0-44 html]# curl 190.0.0.45:80
this is from vpc3
[root@ip-170-0-0-44 html]# 
```

i-0e75813ffd3c60a74 (my-sai-ec1)

PublicIPs: 54.183.223.70 PrivateIPs: 170.0.0.44