Assignments for Cloud and Devops

AWS Assignment

Madhav M

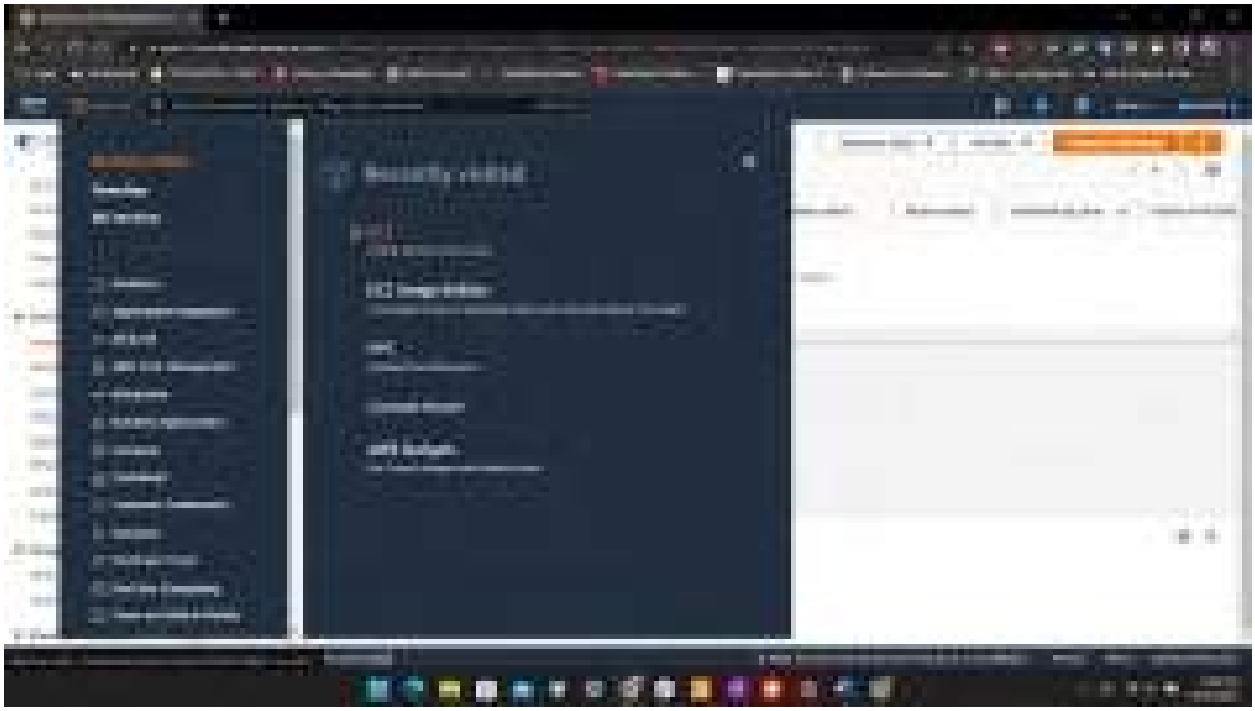
12/1/22

JNTU-CLOUD-DEVOPS

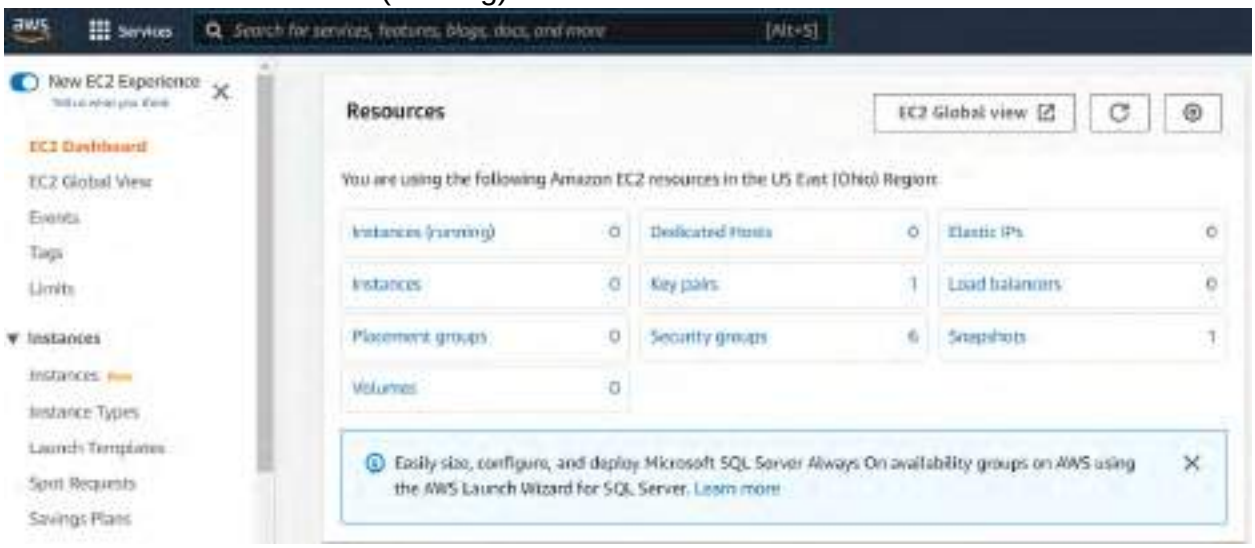
Table Of Content

1. Create EC2

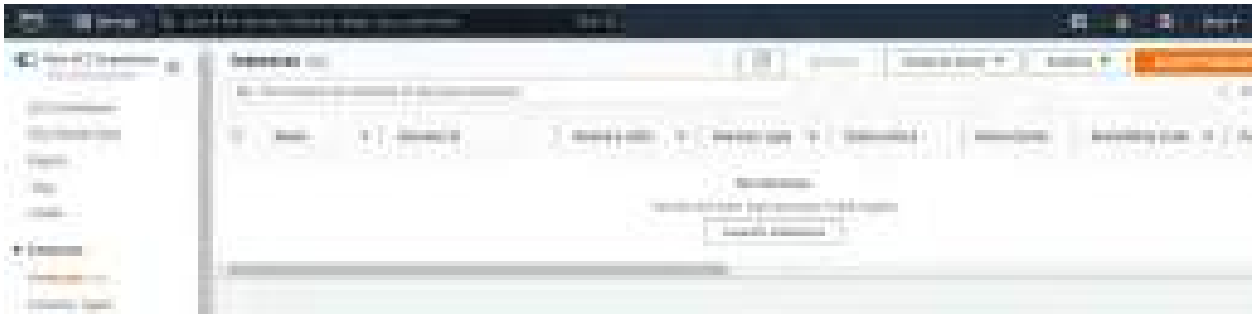
1. Login to AWS
2. Select Services → EC2



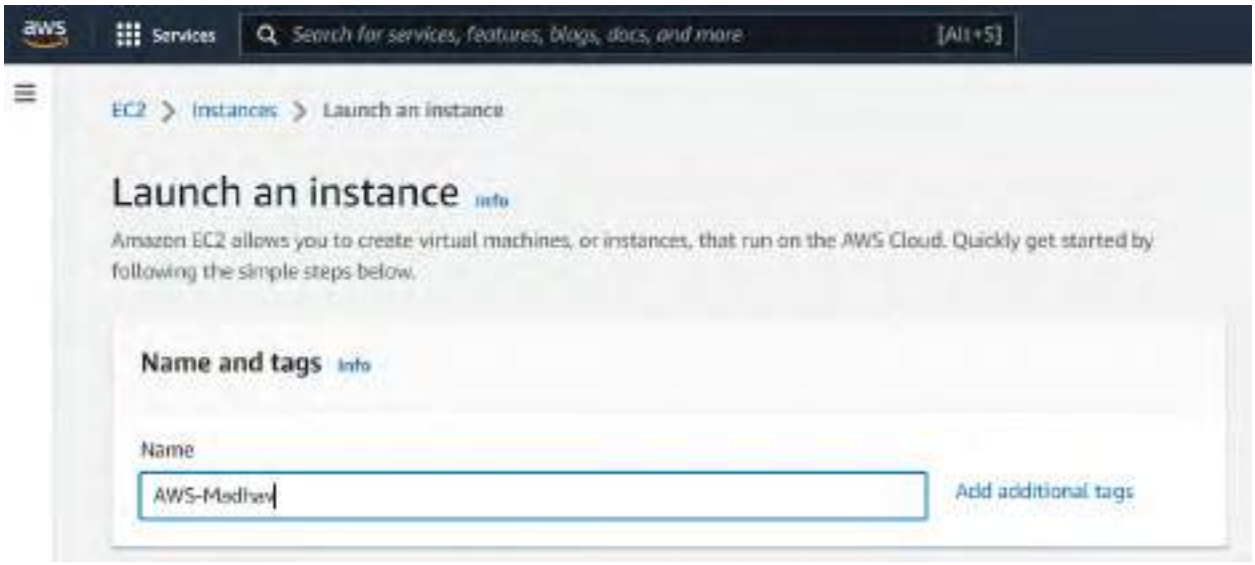
3. Click on Instances (running)



4. Click on Launch Instance



5. Name the instance



6. Select Amazon Linux and keep everything as default

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

My AMIs | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | S

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible

ami-089a545a9ed9893b6 (64-bit (x86)) / ami-04897acca32efd05c (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20221004.0 x86_64 HVM gp2

Architecture: 64-bit (x86)

AMI ID: ami-089a545a9ed9893b6 Verified provider

Instance type Info

Instance type

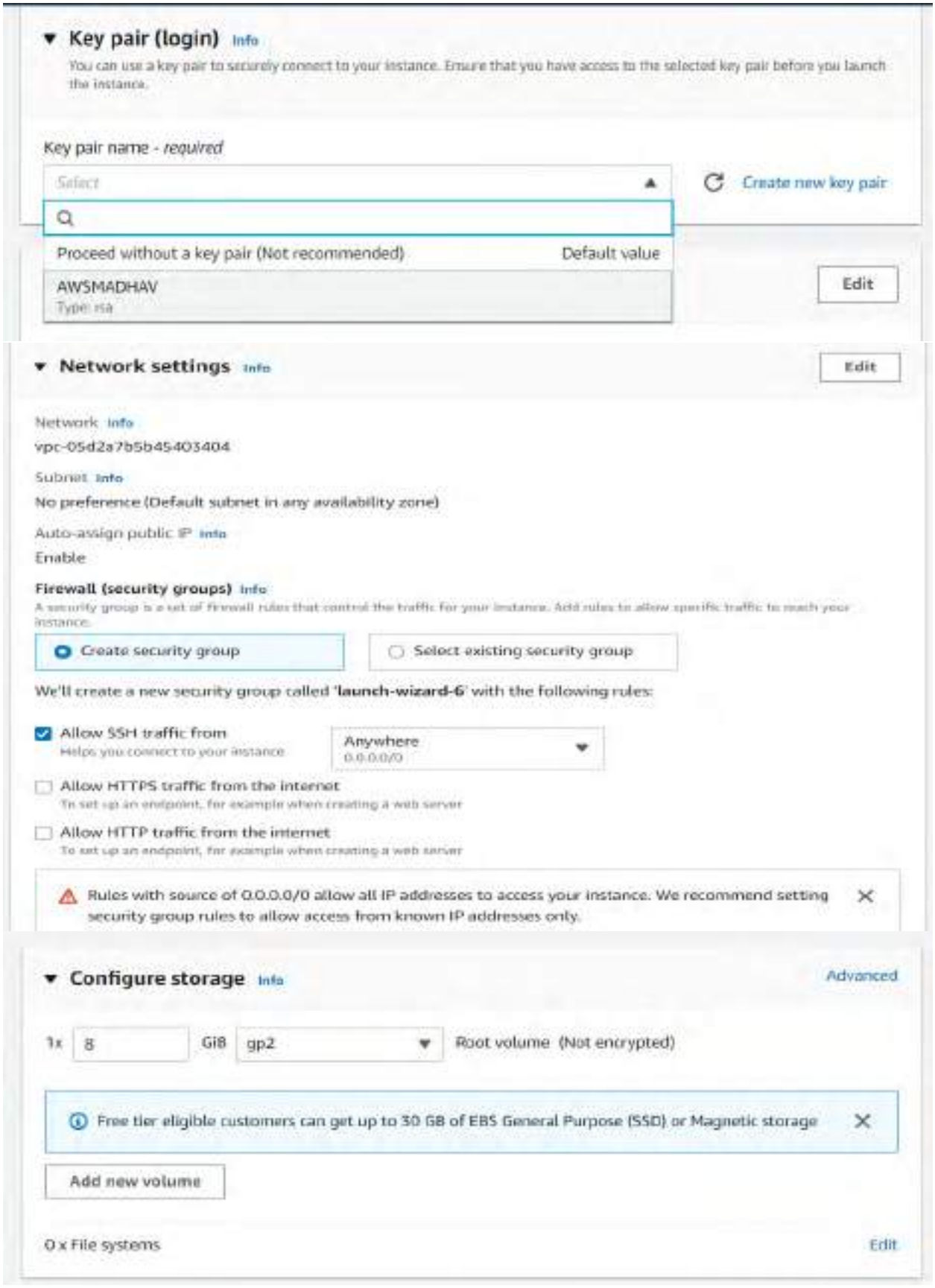
t2.micro — Free tier eligible

Family: t2 1 vCPU 1 GiB Memory

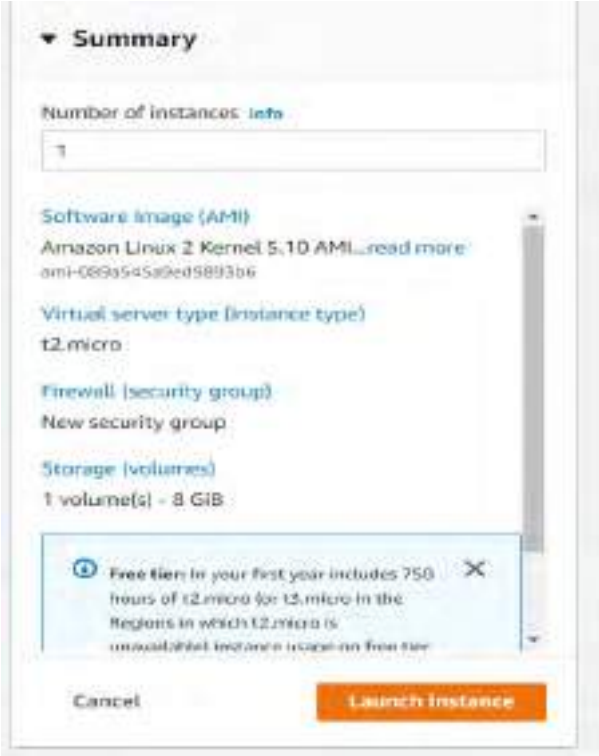
On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

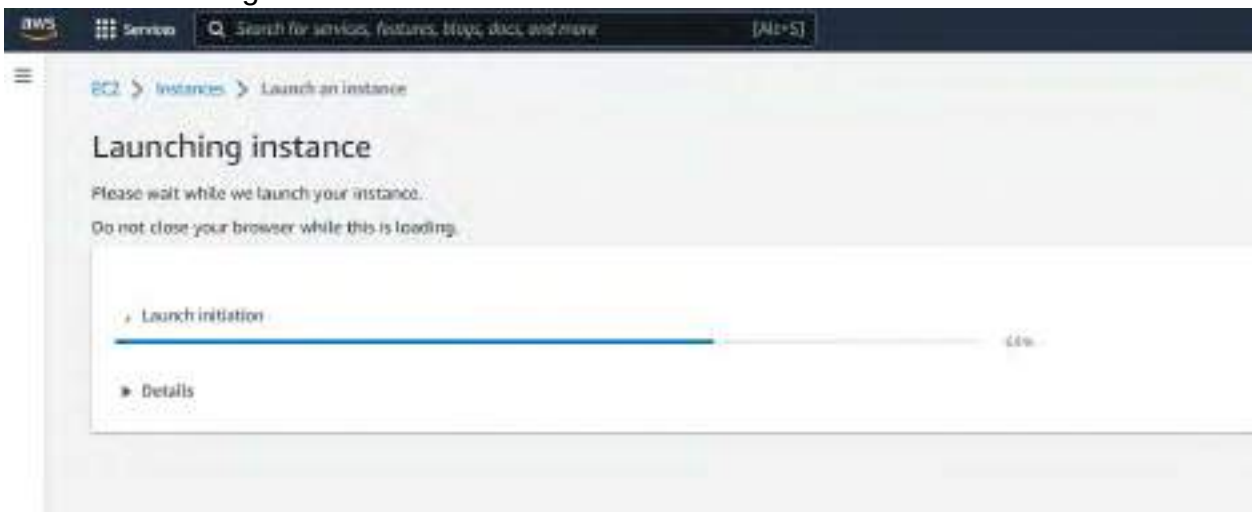
Compare instance types


Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
Select
Create new key pair
Proceed without a key pair (Not recommended)
Default value
AWSMADHAV
Type: rsa
Edit
Network settings Info
Edit
Network Info
vpc-05d2a7b5b45403404
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-6' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Configure storage Info
Advanced
1x
8
GiB
gp2
Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
0 x File systems
Edit

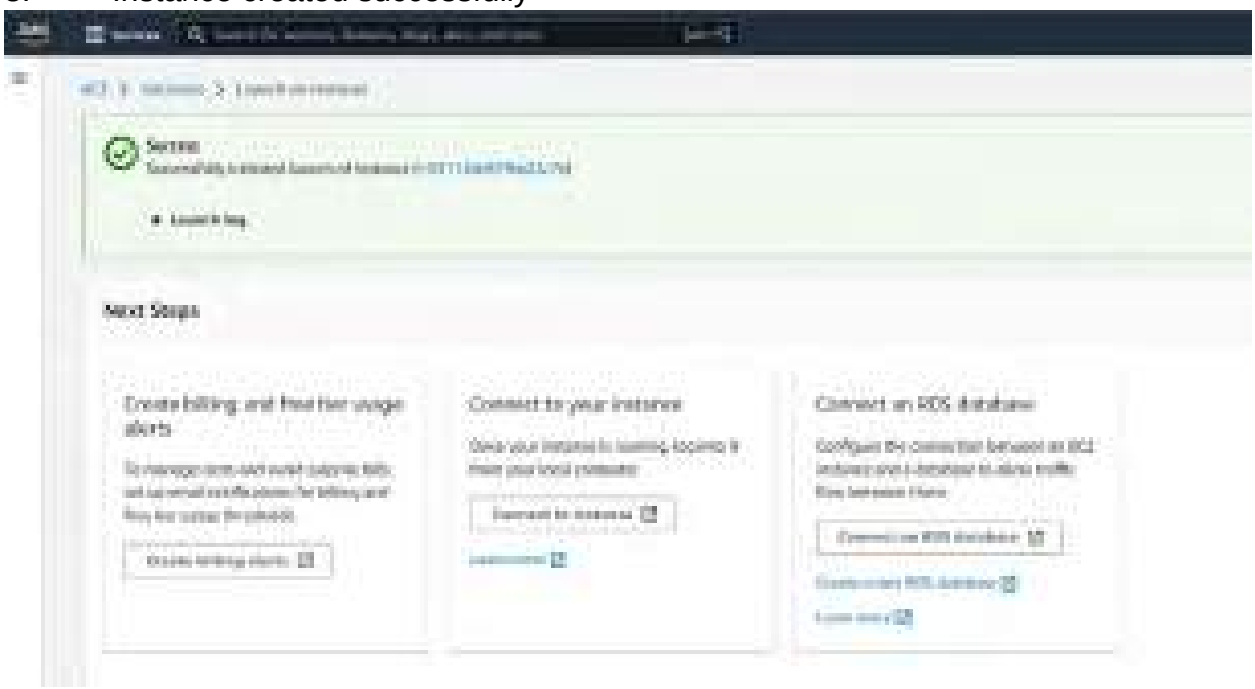
Click on Launch Instance



7. Launching instance

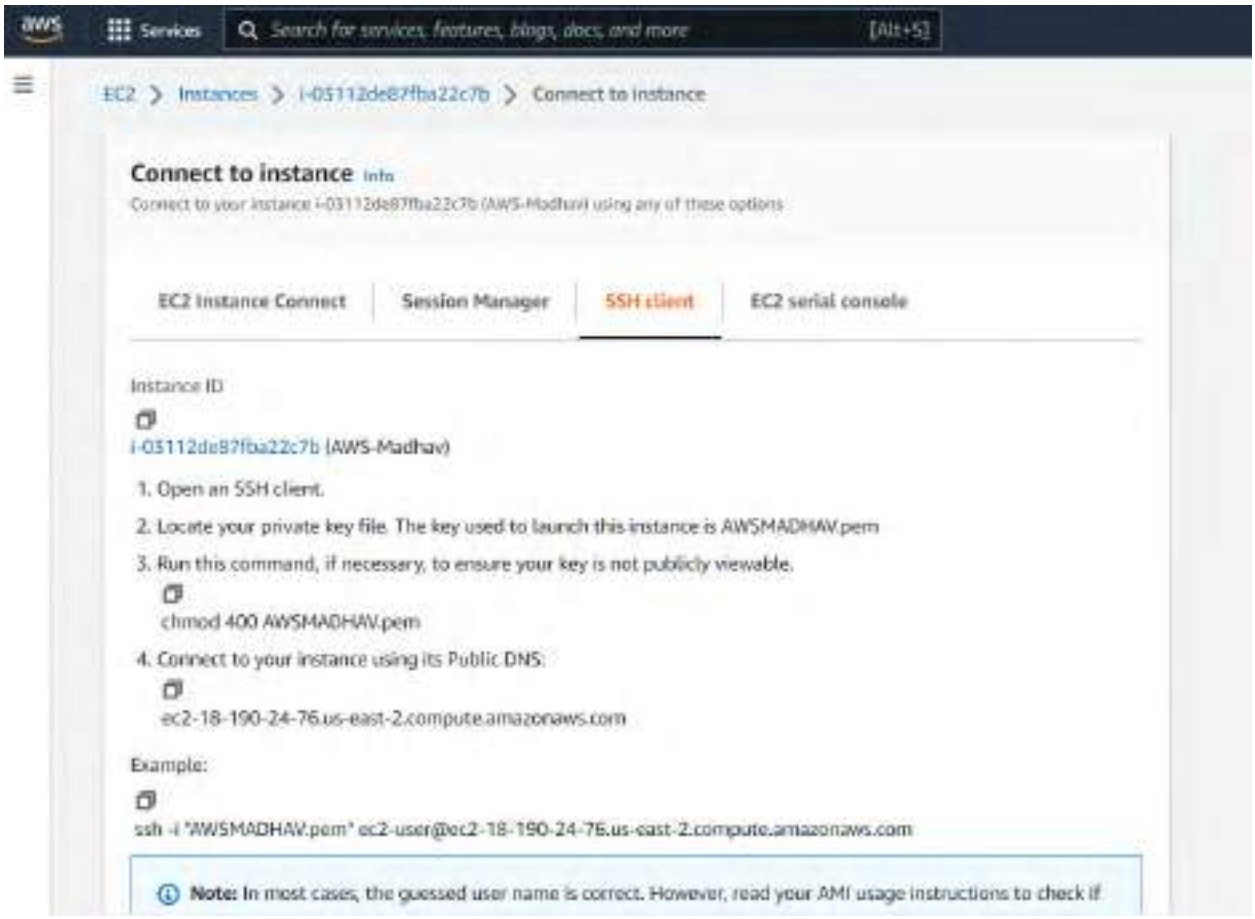


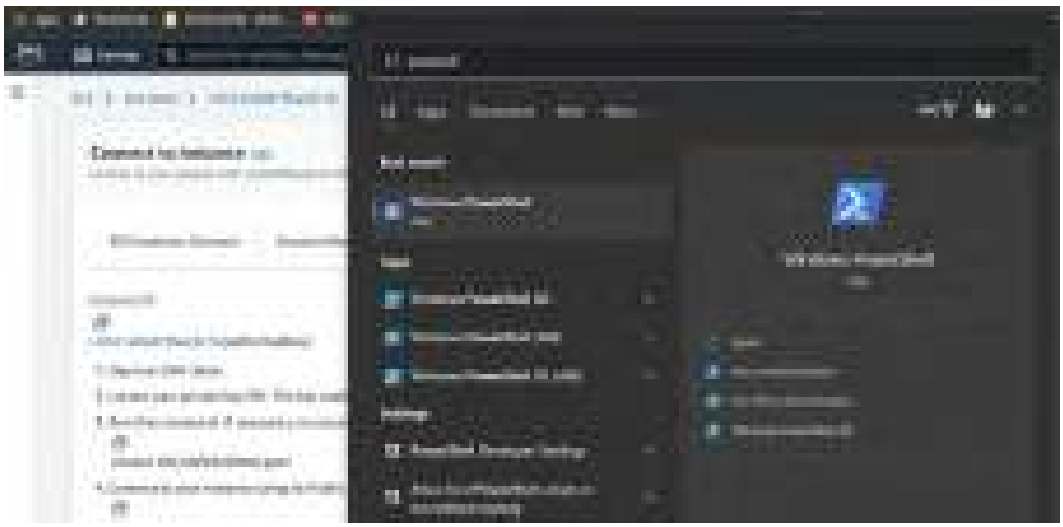
8. Instance created successfully

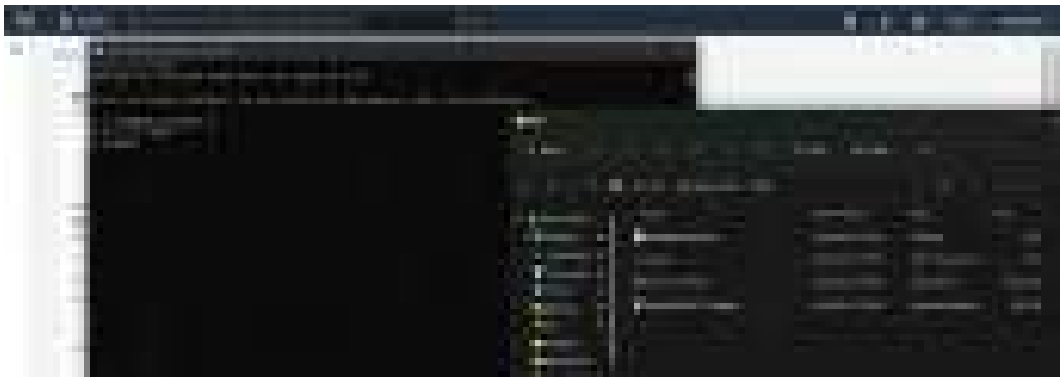


9. Instance created with name AWS-Madhav

10. Connecting to the instance using SSH Client







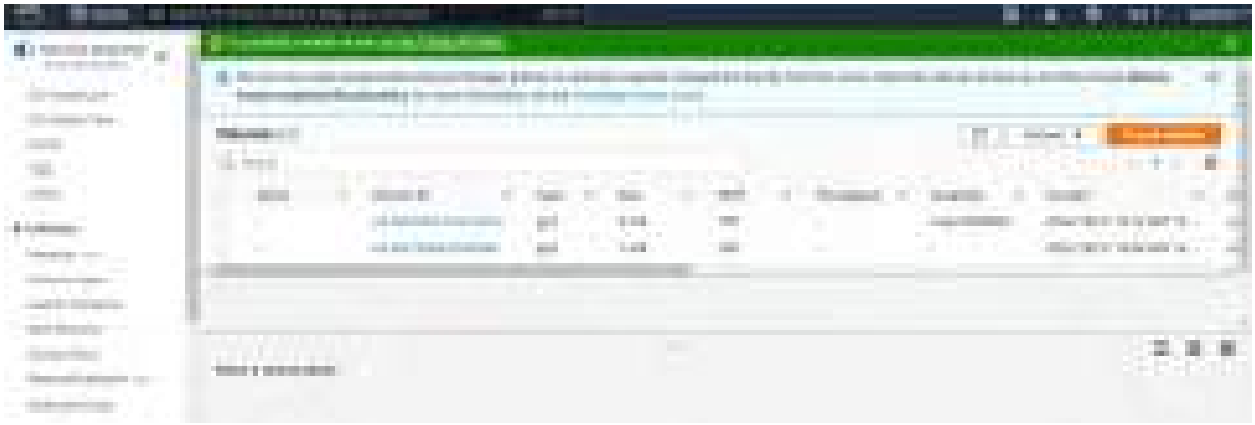
11. Successfully able to connect to the instance using SSH using the key

2. Create Elastic Block Store

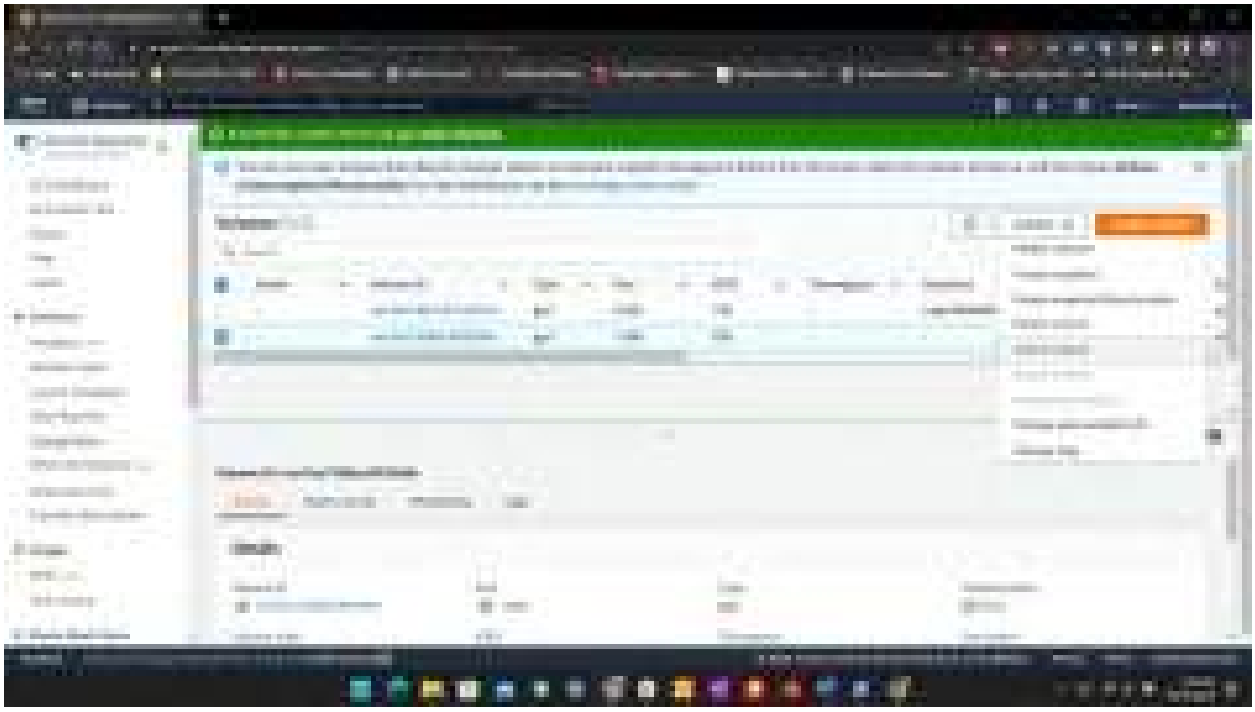
1. Go to Volumes under Elastic Block Store, you can find the volume with size 8 GB for the already created Instance(AWS-Madhav)
2. Click on Create Volume

3. Choose 1GB as the storage size and keep everything default and click on Create Volume.

4. Successfully created the Volume with 1 GB of space.

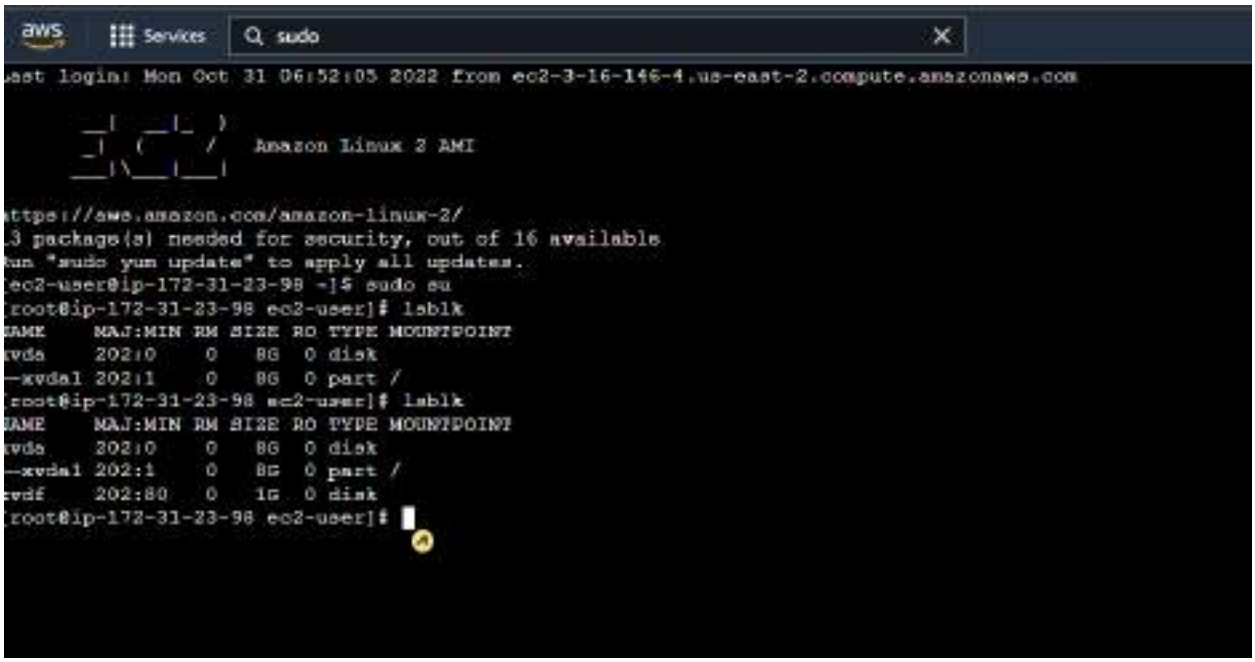


5. Select the created volume, select Actions, and choose Attach Volume



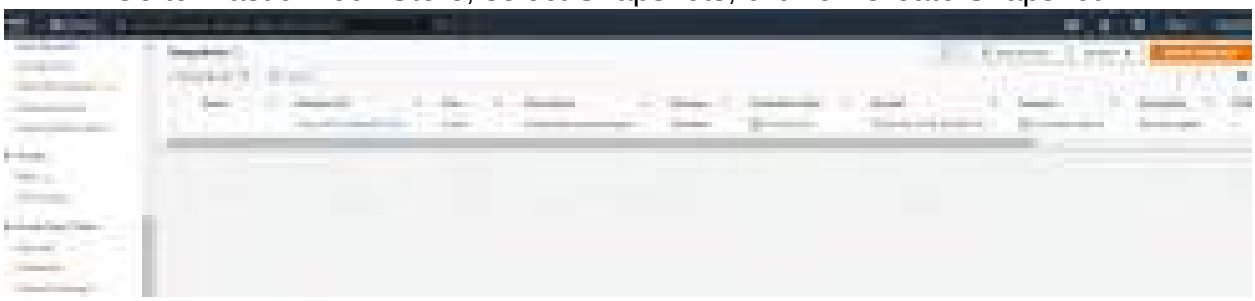
6. Select the instance where you want to attach the volume and check the volume is attached.

7. Confirmation on the volume is attached successfully

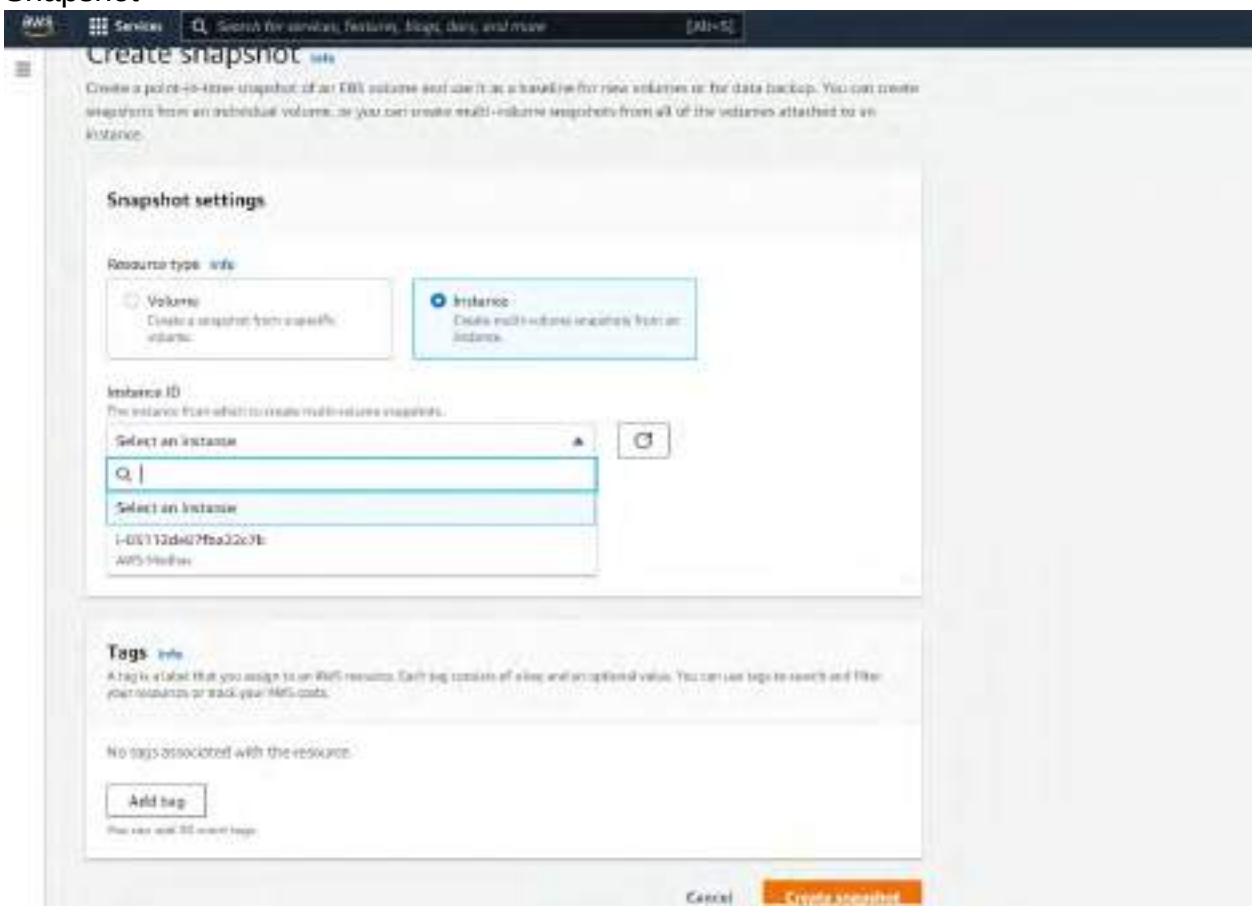


3. Snapshot screenshot creation

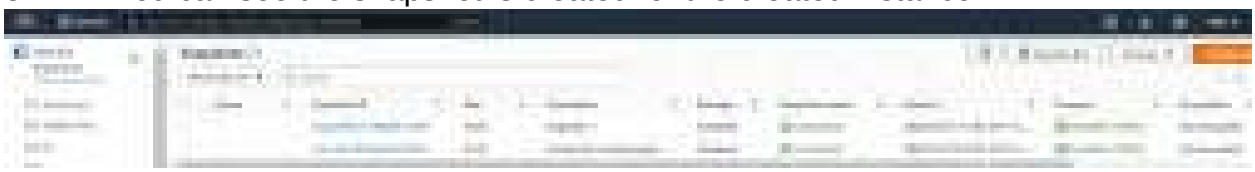
1. Go to Elastic Block Store, select Snapshots, click on Create Snapshot



2. Select Instance, choose the instance ID from the drop down. Click on Create Snapshot



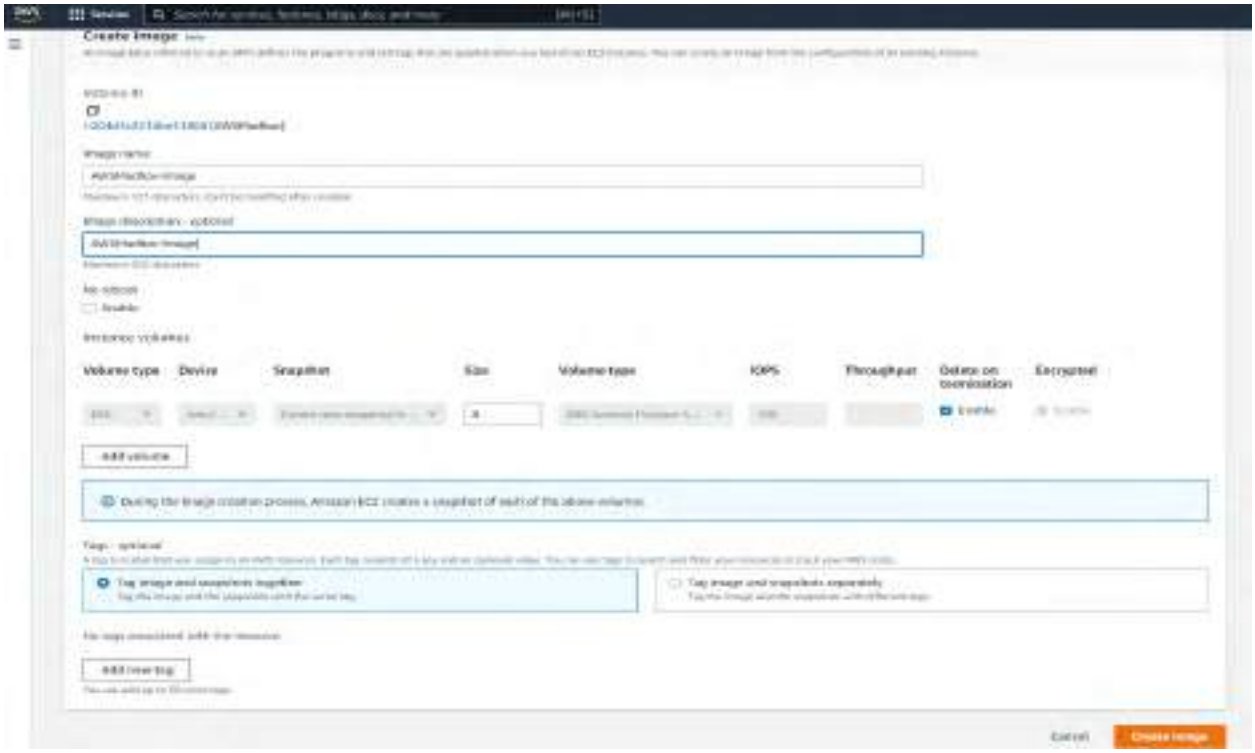
3. You can see the snapshot is created for the created instance.



4. Create AMI

1. Right-click the instance you want to use as the basis for your AMI and select **Image and Templates** and choose **Create Image** from the context menu.

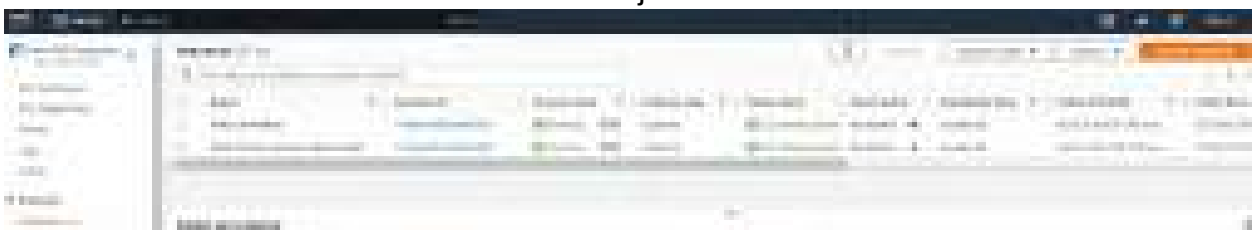
2. In the **Create Image** dialog box, type a unique name and description, and then choose **Create Image.**



3. Under Images, AMIs you will find the created AMI

5. Load Balancer Creation

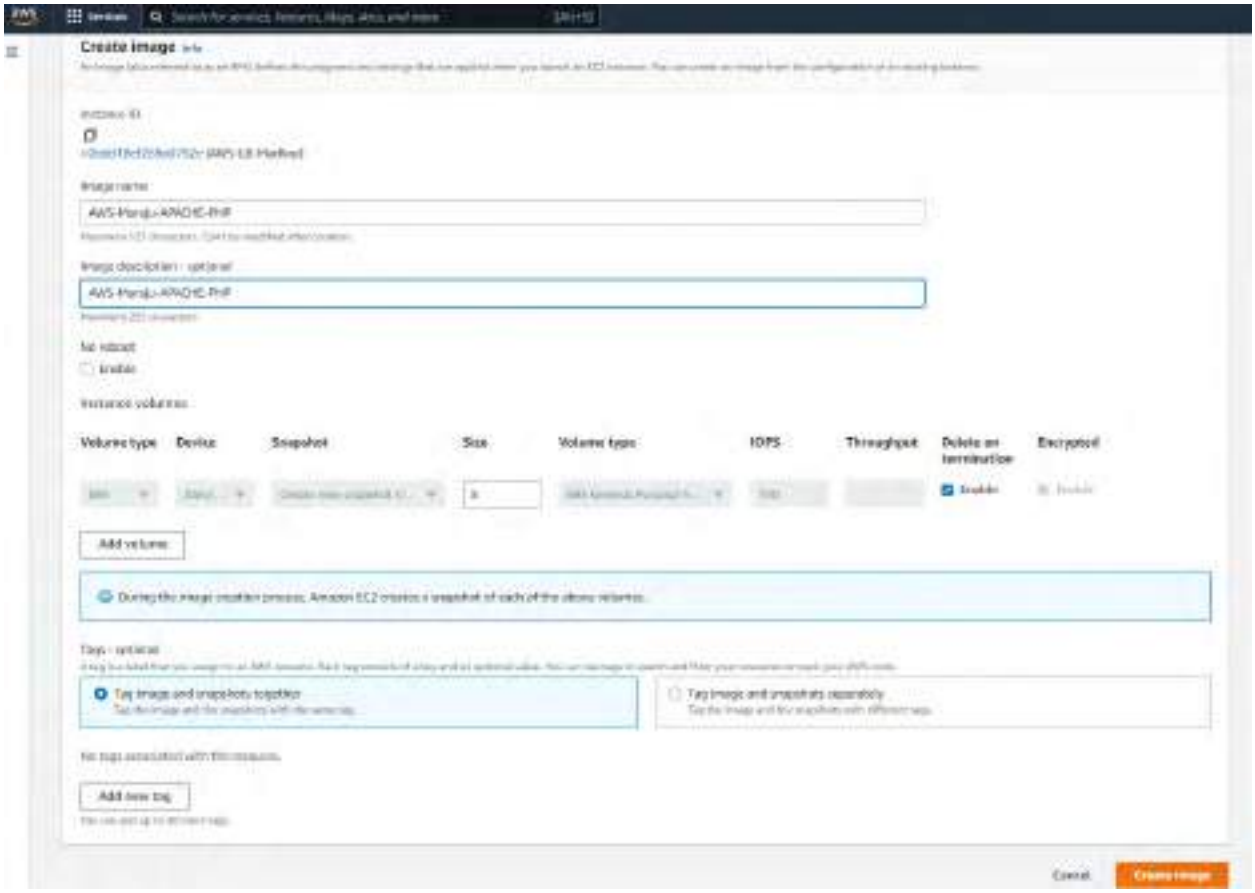
1. To create Application load balancer, we need at least 2 EC2 machines
2. Created AWSMadhav and AWSMaroju – EC2 machines



3. Successfully installed Apache and PHP in the ubuntu machine, both Apache and PHP are working as expected.

4. Create an image from the existing instance which consists of both Apache and PHP version.

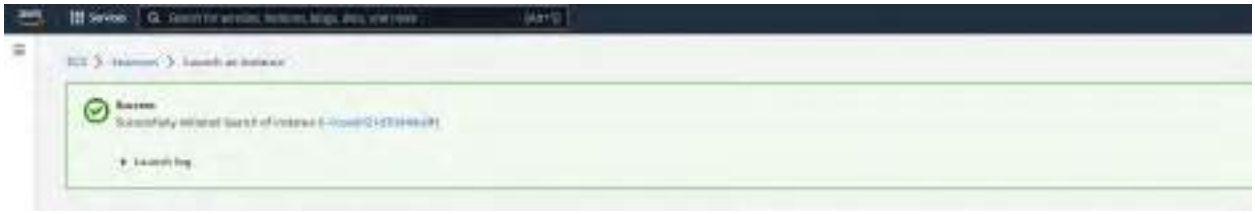
5. Add the image name and description (AWS-Maroju-APACHE-PHP) and click on create image

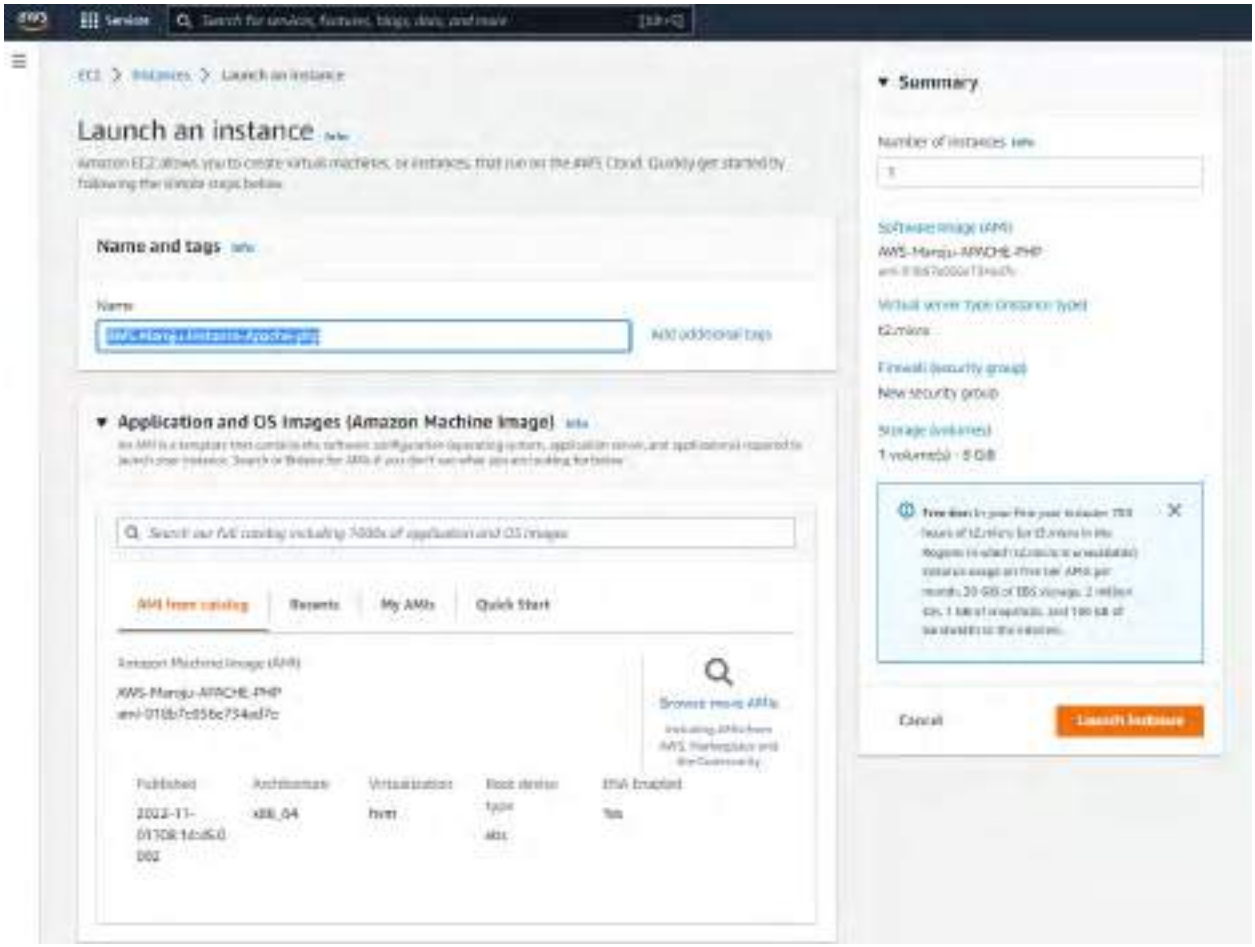


6. Go to AMIs and check the image created with the name (AWS-Madhav-APACHE-PHP)

6. Right click on the AMI and select Launch Instance from AMI

7. Choose the name ‘AWS-Maroju-Instance-Apache-php’ and select MY AMIs and choose the created AMI and set everything as default and click on Launch Instance.

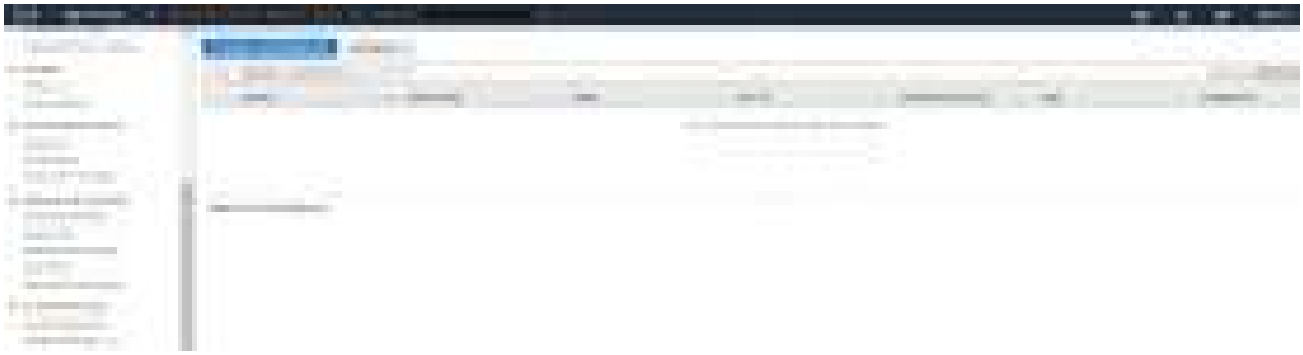




Successfully able to connect to the created instance

```
Unknown command verb apache2.
root@ip-172-31-31-148:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-11-01 08:19:15 UTC; 9min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 517 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 779 (apache2)
      Tasks: 6 (limit: 1143)
     Memory: 22.5M
        CPU: 100ms
     CGroup: /system.slice/apache2.service
             ├─779 /usr/sbin/apache2 -k start
             ├─819 /usr/sbin/apache2 -k start
             ├─820 /usr/sbin/apache2 -k start
             ├─821 /usr/sbin/apache2 -k start
             ├─822 /usr/sbin/apache2 -k start
             └─823 /usr/sbin/apache2 -k start

Nov 01 08:19:13 ip-172-31-31-148 systemd[1]: Starting The Apache HTTP Server...
Nov 01 08:19:15 ip-172-31-31-148 systemd[1]: Started The Apache HTTP Server.
root@ip-172-31-31-148:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-11-01 08:19:15 UTC; 9min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 517 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 779 (apache2)
      Tasks: 6 (limit: 1143)
     Memory: 22.5M
        CPU: 100ms
     CGroup: /system.slice/apache2.service
             ├─779 /usr/sbin/apache2 -k start
             ├─819 /usr/sbin/apache2 -k start
             ├─820 /usr/sbin/apache2 -k start
             ├─821 /usr/sbin/apache2 -k start
             ├─822 /usr/sbin/apache2 -k start
             └─823 /usr/sbin/apache2 -k start

Nov 01 08:19:13 ip-172-31-31-148 systemd[1]: Starting The Apache HTTP Server...
Nov 01 08:19:15 ip-172-31-31-148 systemd[1]: Started The Apache HTTP Server.
root@ip-172-31-31-148:~# php --version
PHP 8.1.2-1ubuntu2.6 (cli) (built: Sep 15 2022 11:30:49) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.6, Copyright (c), by Zend Technologies
root@ip-172-31-31-148:~#
```

Navigate to **Load Balancing, Load Balancers**

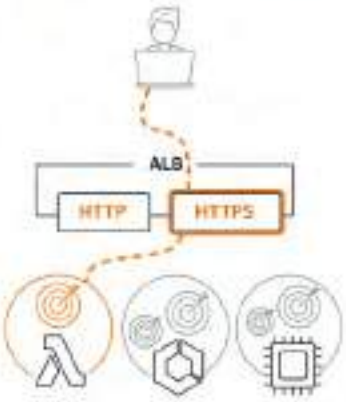


4. Click on Create Load Balancer, select load balancer type as Application Load Balancer

A complete feature-by-feature comparison along with detailed highlights is also available. Learn more

Load balancer types

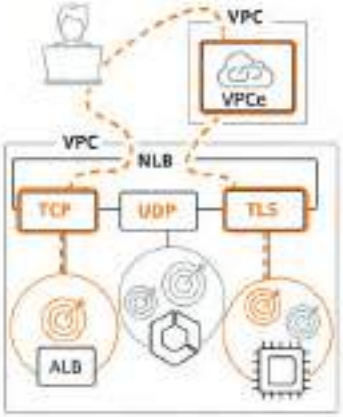
Application Load Balancer Info



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

Create

Network Load Balancer Info



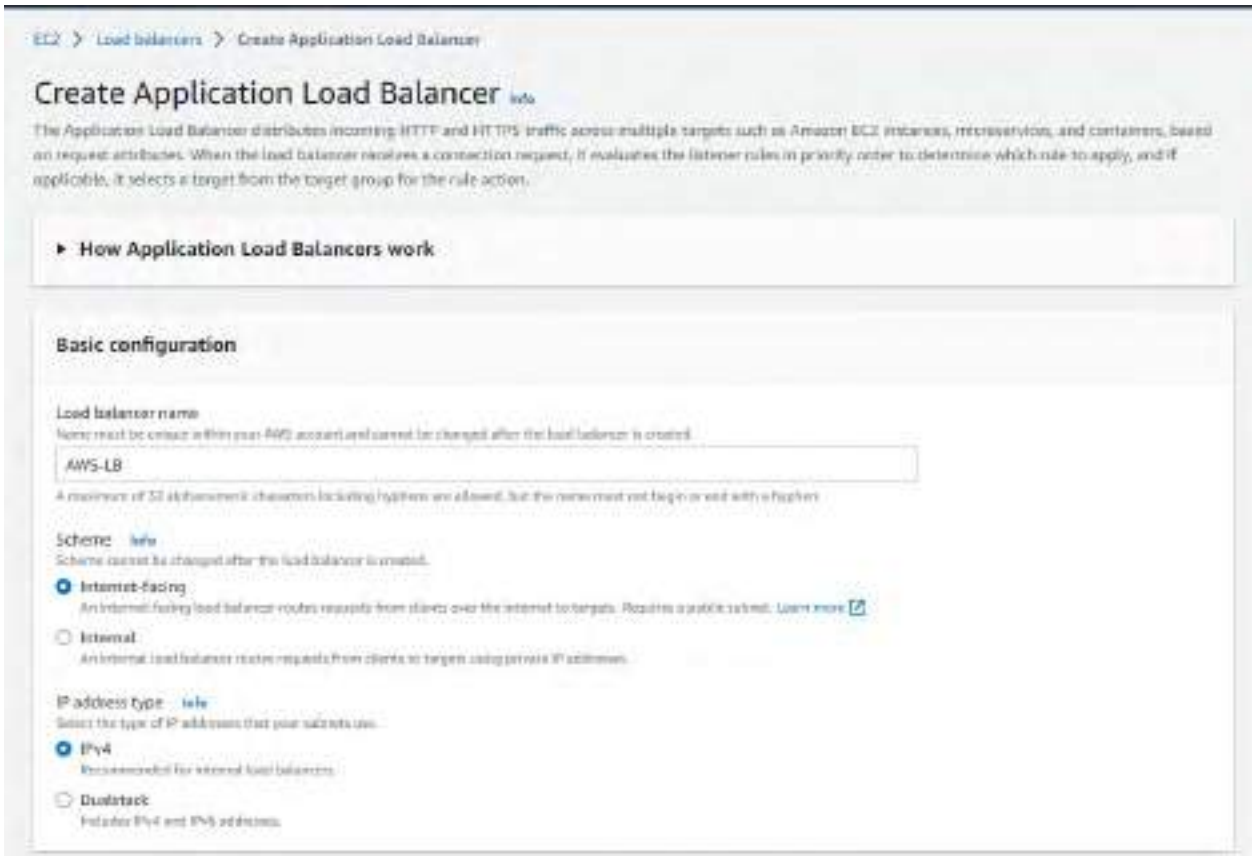
Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

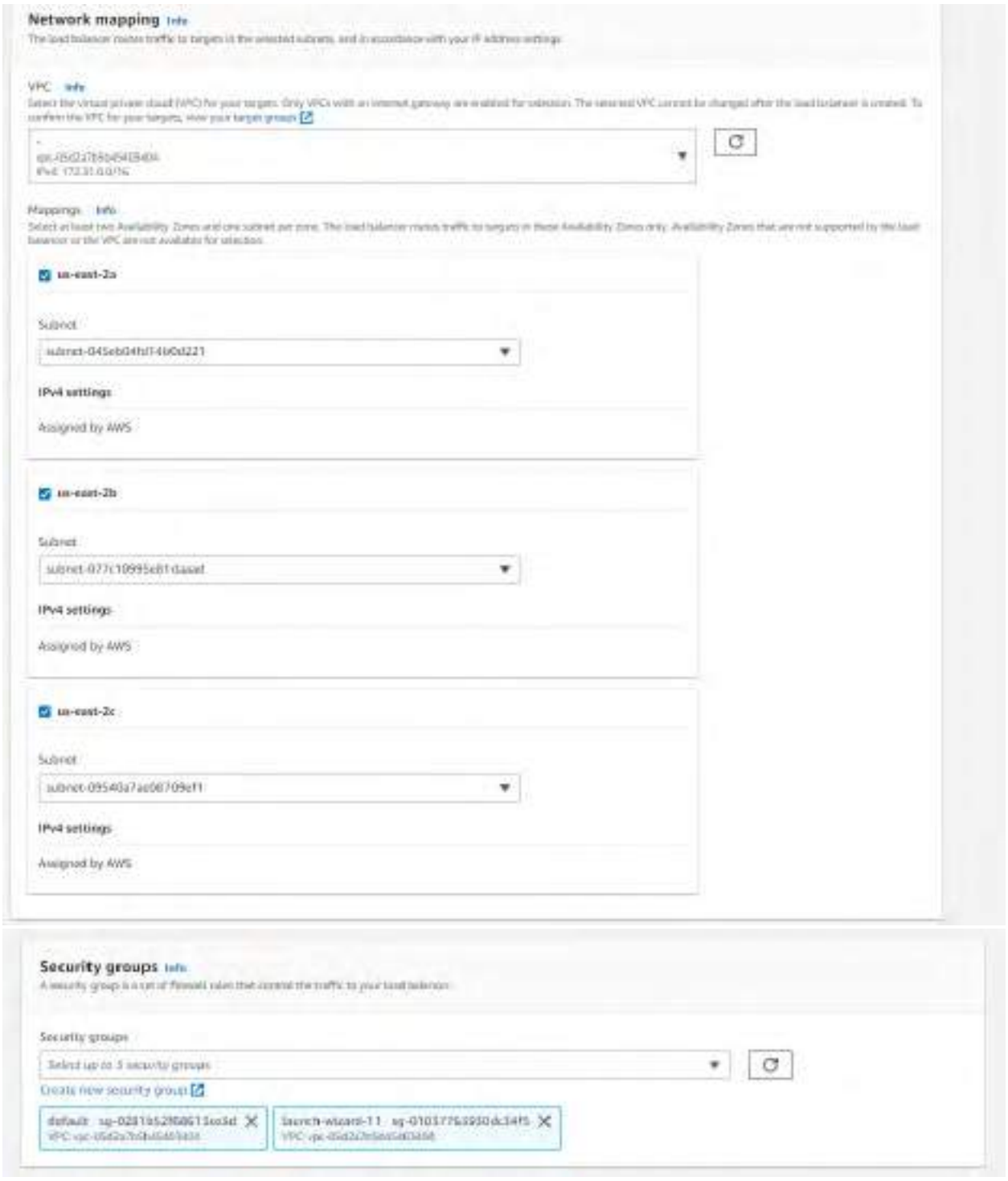
Create

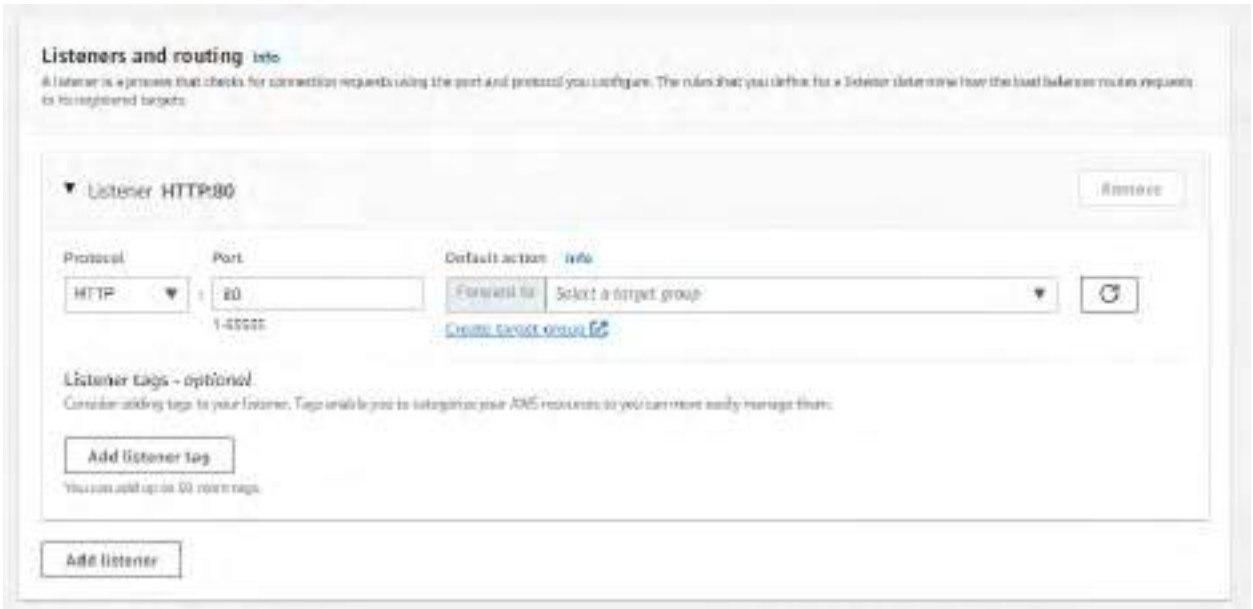
Gateway Load Balancer Info

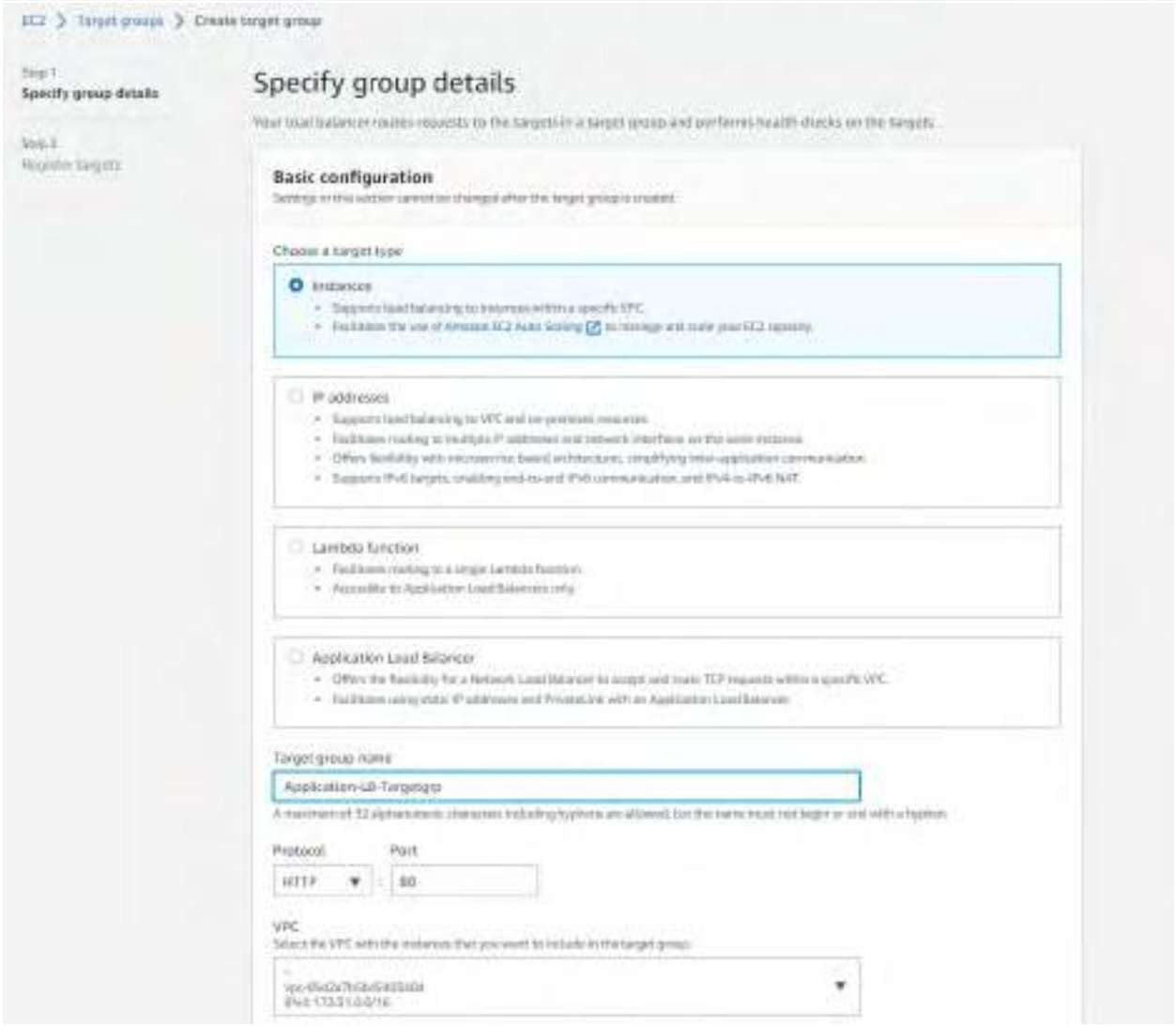
Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

Create


EC2 > Load balancers > Create Application Load Balancer
Create Application Load Balancer info
The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.
How Application Load Balancers work
Basic configuration
Load balancer name
AWS-LB
Scheme info
Internet-facing
Internal
IP address type info
IPv4
Dualstack


Network mapping Info
VPC Info
Mappings Info
us-east-2a
Subnet
subnet-045eb04fd14b0d221
IPv4 settings
Assigned by AWS
us-east-2b
Subnet
subnet-077c10995e81daaat
IPv4 settings
Assigned by AWS
us-east-2c
Subnet
subnet-09540a7ae06709ef1
IPv4 settings
Assigned by AWS
Security groups Info
Security groups
Select up to 5 security groups
Create new security group
default sg-028195280613ce3d
launch-wizard-11 sg-0103776395f0dc34f5


Listeners and routing info
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.
Listener HTTP:80
Remove
Protocol
Port
Default action info
HTTP
80
1-65535
Forward to
Select a target group
Create target group
Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.
Add listener tag
You can add up to 50 more tags.
Add listener


EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
Specify group details
Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
Basic configuration
Settings in this section cannot be changed after the target group is created.
Choose a target type
Instances
Supports load balancing to instances within a specific VPC.
Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.
IP addresses
Supports load balancing to VPC and on-premises resources.
Facilitates routing to multiple IP addresses and network interfaces on the same instance.
Offers flexibility with microservice based architectures, simplifying inter-application communication.
Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
Lambda function
Facilitates routing to a single Lambda function.
Accessible to Application Load Balancers only.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
Application-LB-Targetgp
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Protocol
Port
HTTP
80
VPC
Select the VPC with the instances that you want to include in the target group.

▼ Advanced health check settings

Restore defaults

Port

The port the load balancer uses when performing health checks on targets. The default is the port on which each target receives traffic from the load balancer, but you can specify a different port.

- (•) Traffic port
- ( ) Override

Healthy threshold

The number of consecutive health checks successes required before considering an unhealthy target healthy.

2

2-10

Unhealthy threshold

The number of consecutive health check failures required before considering a target unhealthy.

2

2-10

Timeout

The amount of time, in seconds, during which no response means a failed health check.

2

seconds

2-120

Interval

The approximate amount of time between health checks of an individual target.

5

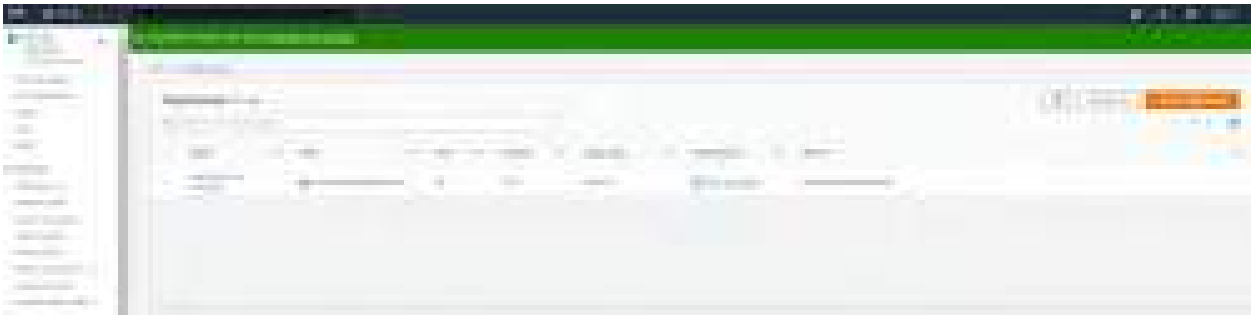
seconds

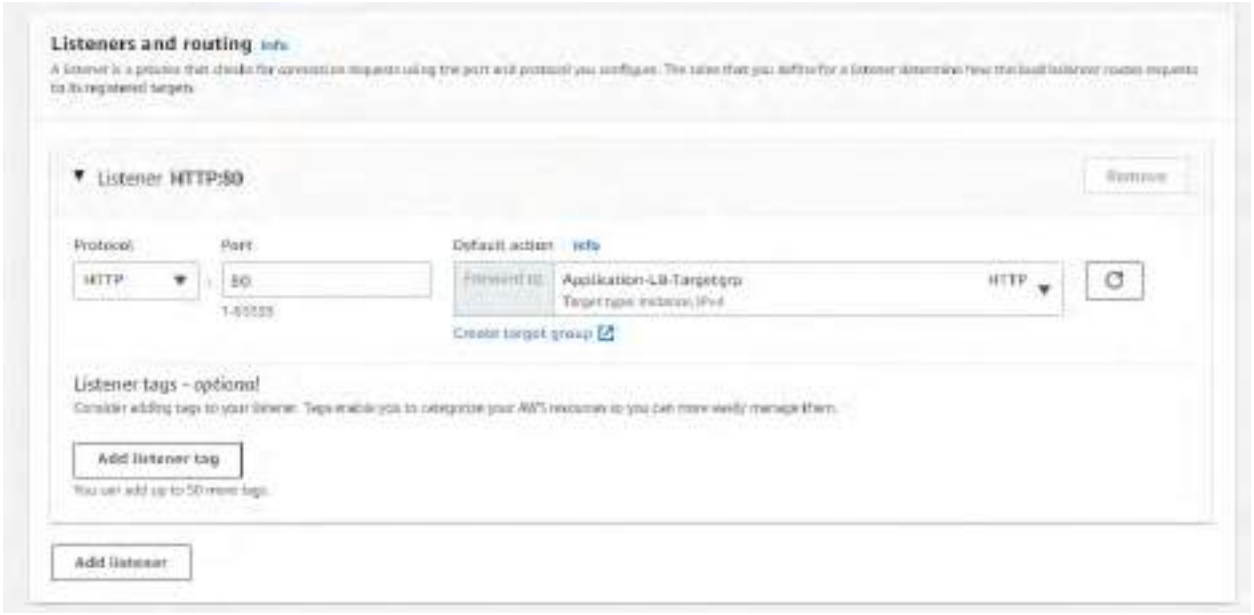
5-300

Success codes

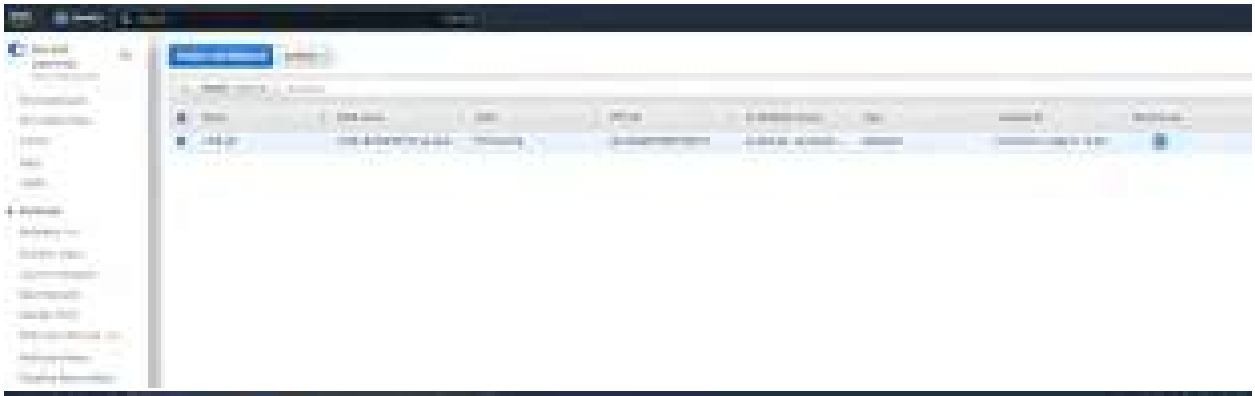
The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").

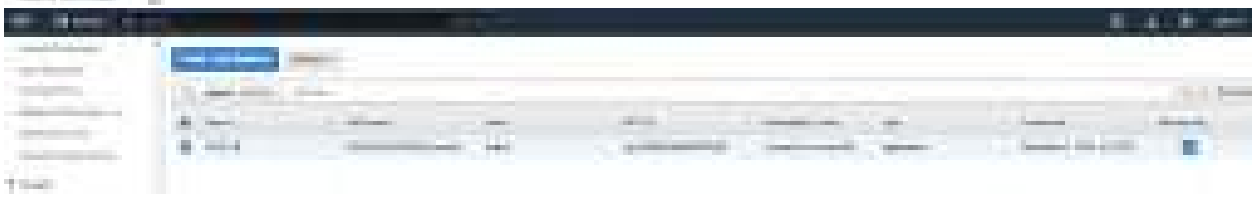
200

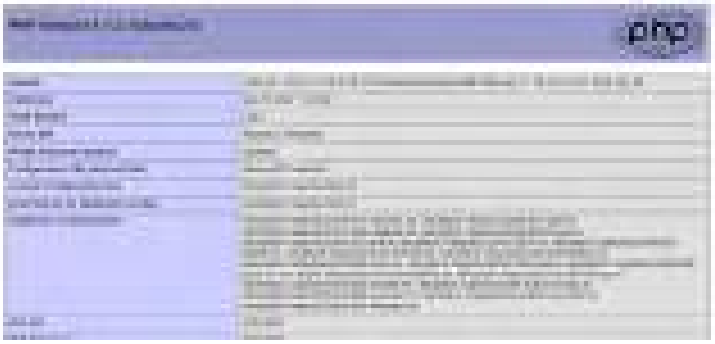



Listeners and routing Info
Listener HTTP:80
Remove
Protocol
Port
Default action Info
HTTP
80
1-65535
Forward to
Application-LB-Targetgrp
HTTP
Target type: Instance, IPv4
Create target group
Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.
Add listener tag
You can add up to 50 more tags.
Add listener


Create Application Load Balancer



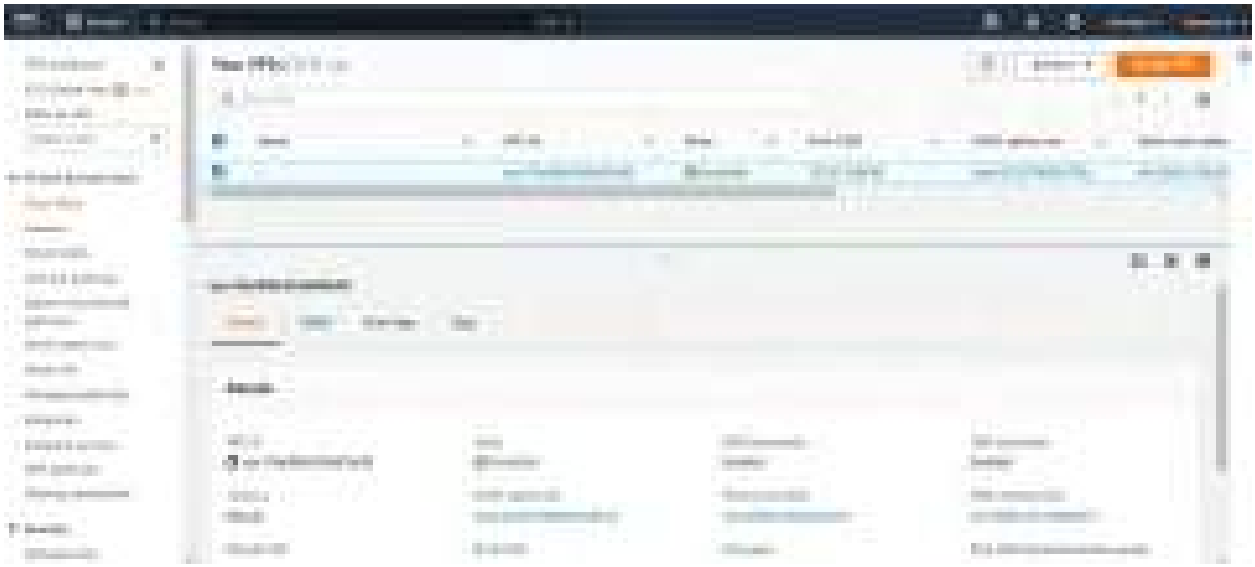




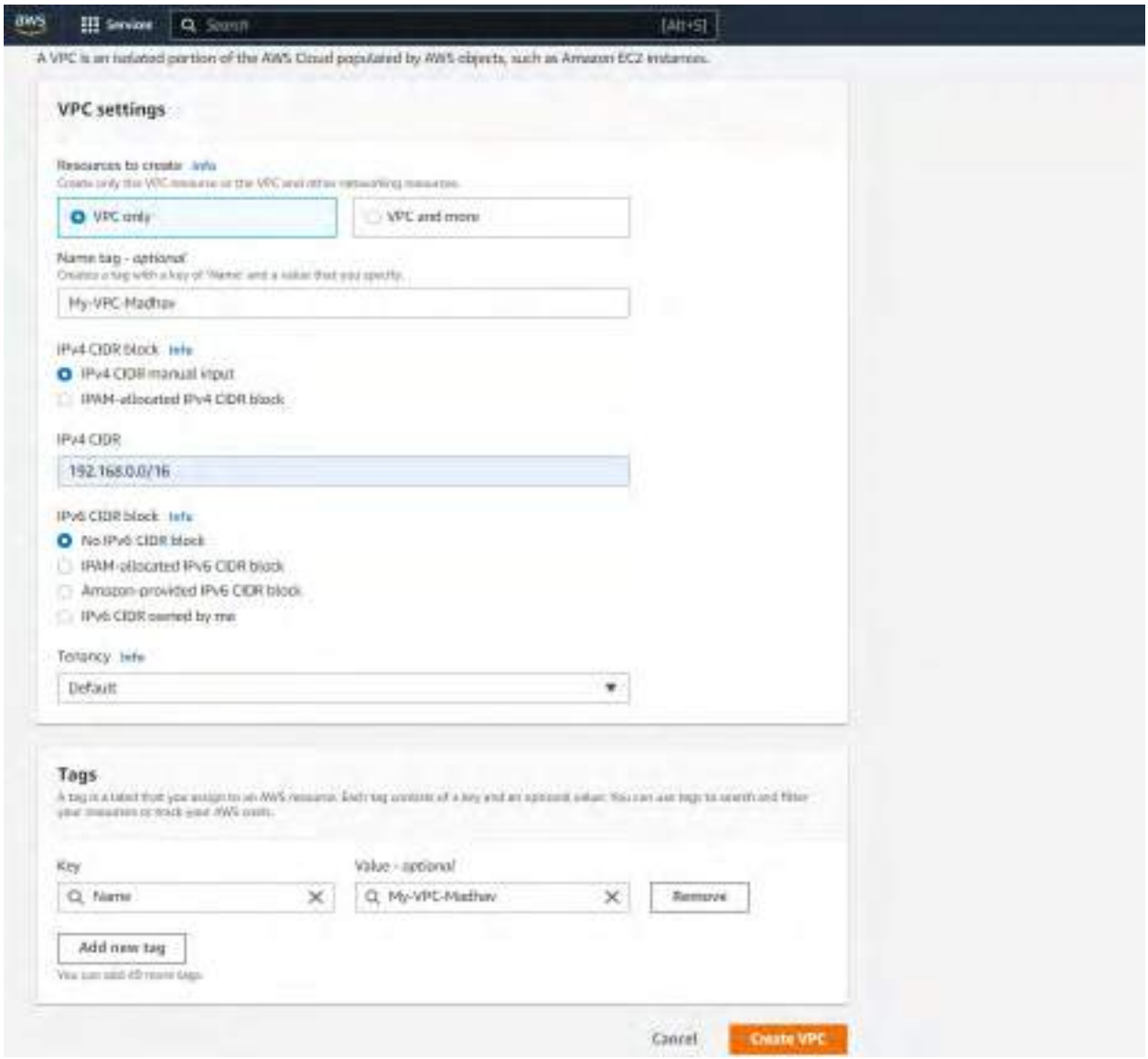
6. Create VPC

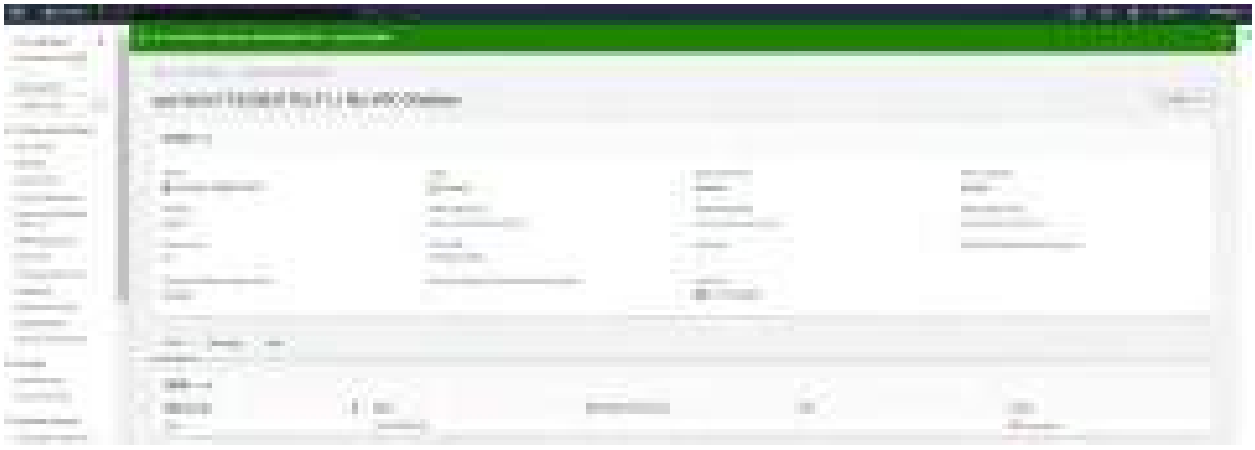
1. Log into AWS console
2. Search for VPC, and select VPC

3. Under ‘Virtual Private Cloud’, select Your VPCs and click on ‘Create VPC’



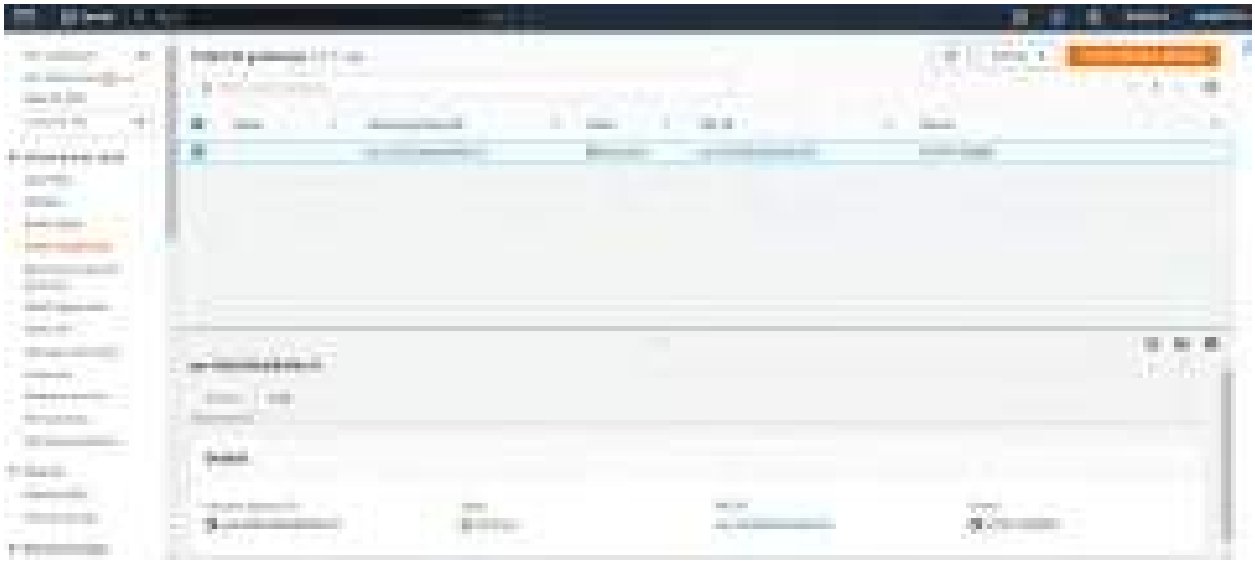
4. Give name as ‘My-VPC-Madhav’ with IP range 192.168.0.0/16 and keep everything as default and click on Create VPC



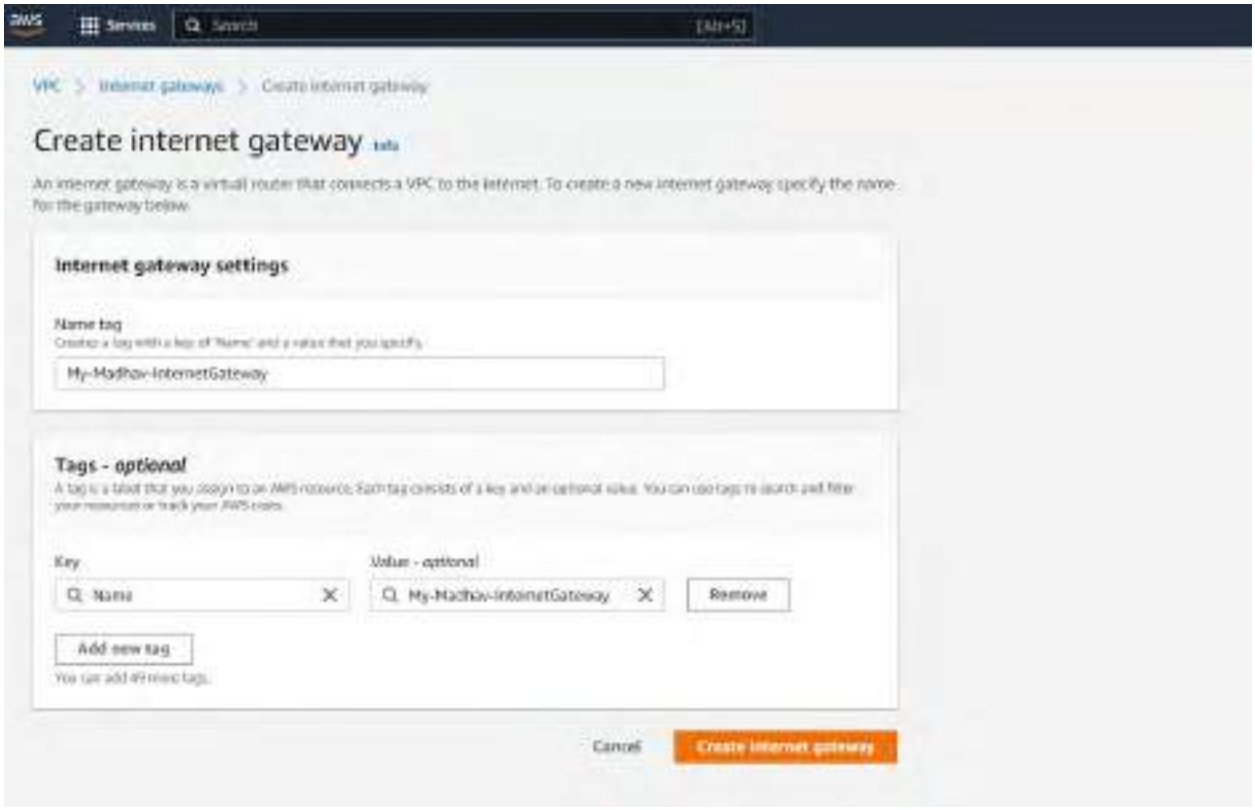


5. Go to Your VPCs and check for the newly created VPC

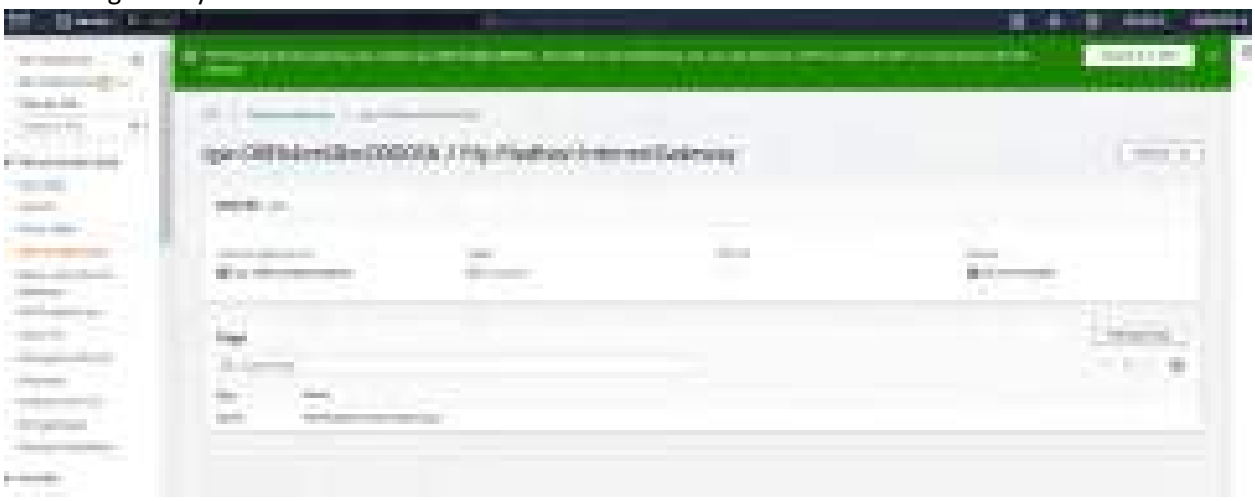
6. Click on Internet gateways, under Virtual private cloud and click on Create Internet gateways



7. Give name as ‘My-Madhav-InternetGateway’, click on Create internet gateway

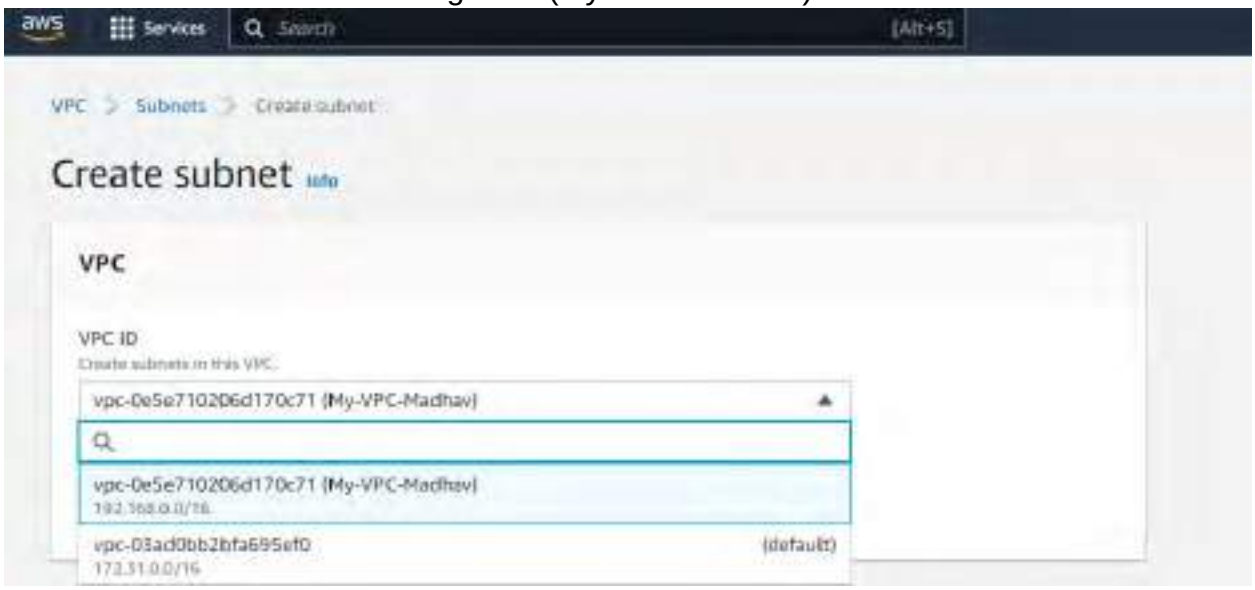


Internet gateway is created

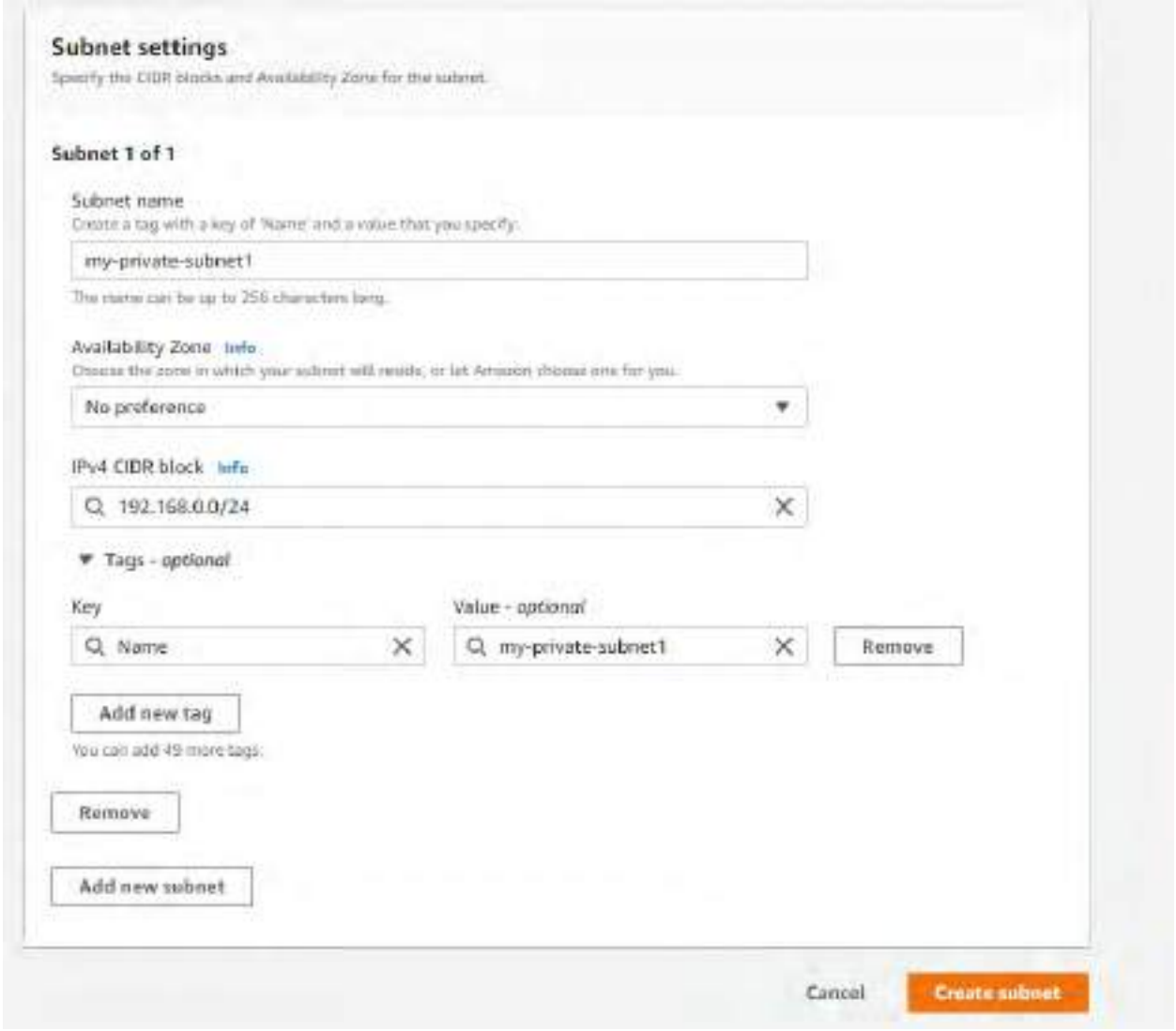


8. Create 4 Subnets of which 2 Subnets are public subnets and 2 subnets are private subnets
9. Click on Subnets, under Virtual Private cloud and select Create Subnet

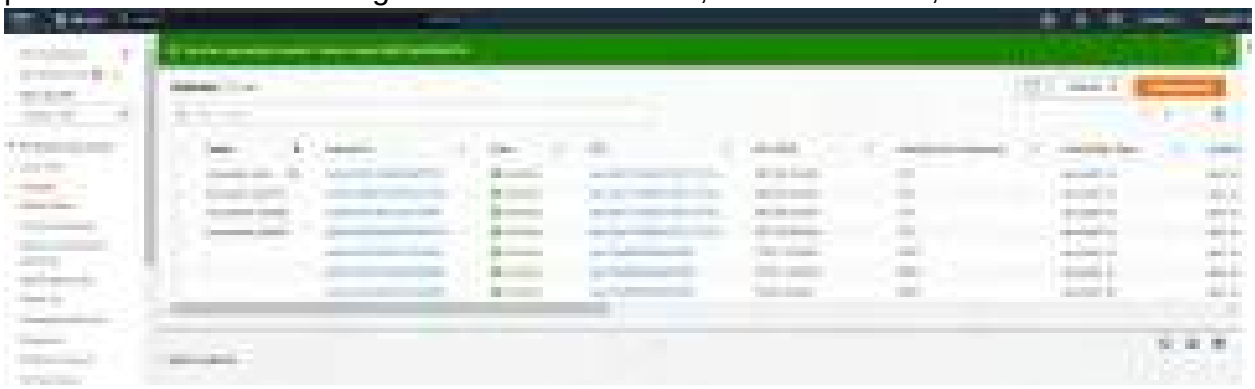
Attach the Subnet to the existing VPC (My-VPC-Madhav)



And create my-private-subnet1 with IP range as 192.168.0.0/24 and click on create subnet



10. Repeat the same process to create my-private-subnet2, my-public-subnet1, my-public-subnet2 with IP range as – 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24

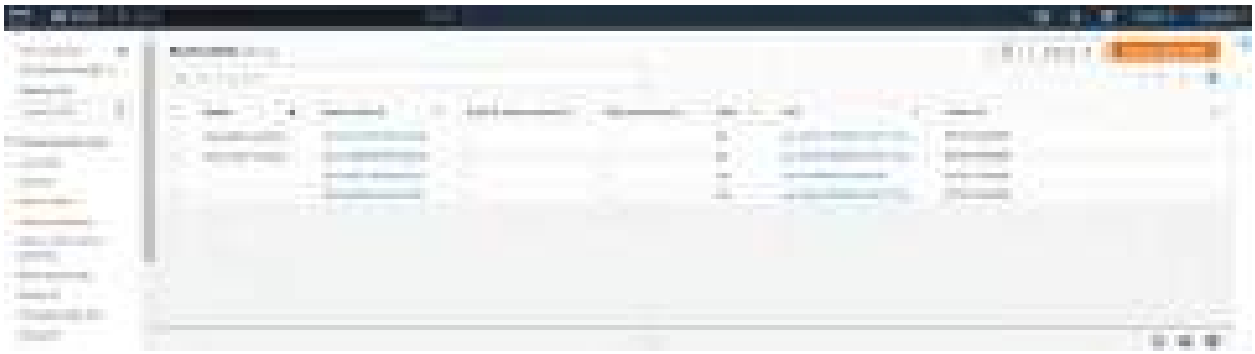


11. Create 2 route tables – One public route table and one private router table
12. Click on Route tables under Virtual Private Cloud, click on create route table

13. Create my-public-routertable-1 and attach it to the existing VPC

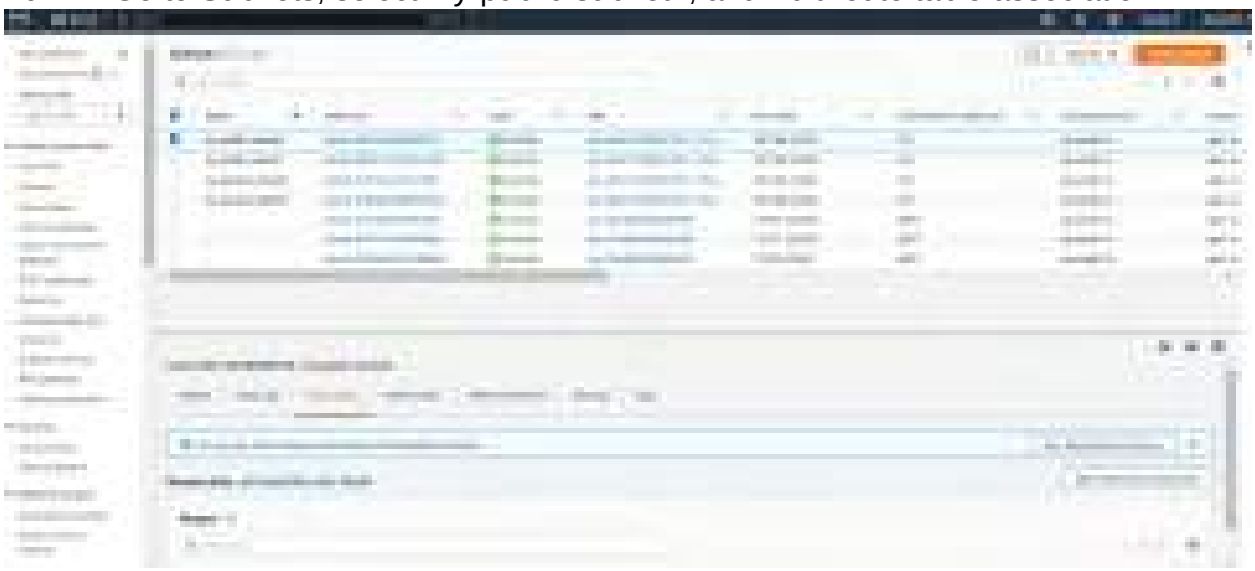
14. Create my-private-routetable-1 and attach it to the existing VPC

2 route tables – one private and one public route table is created

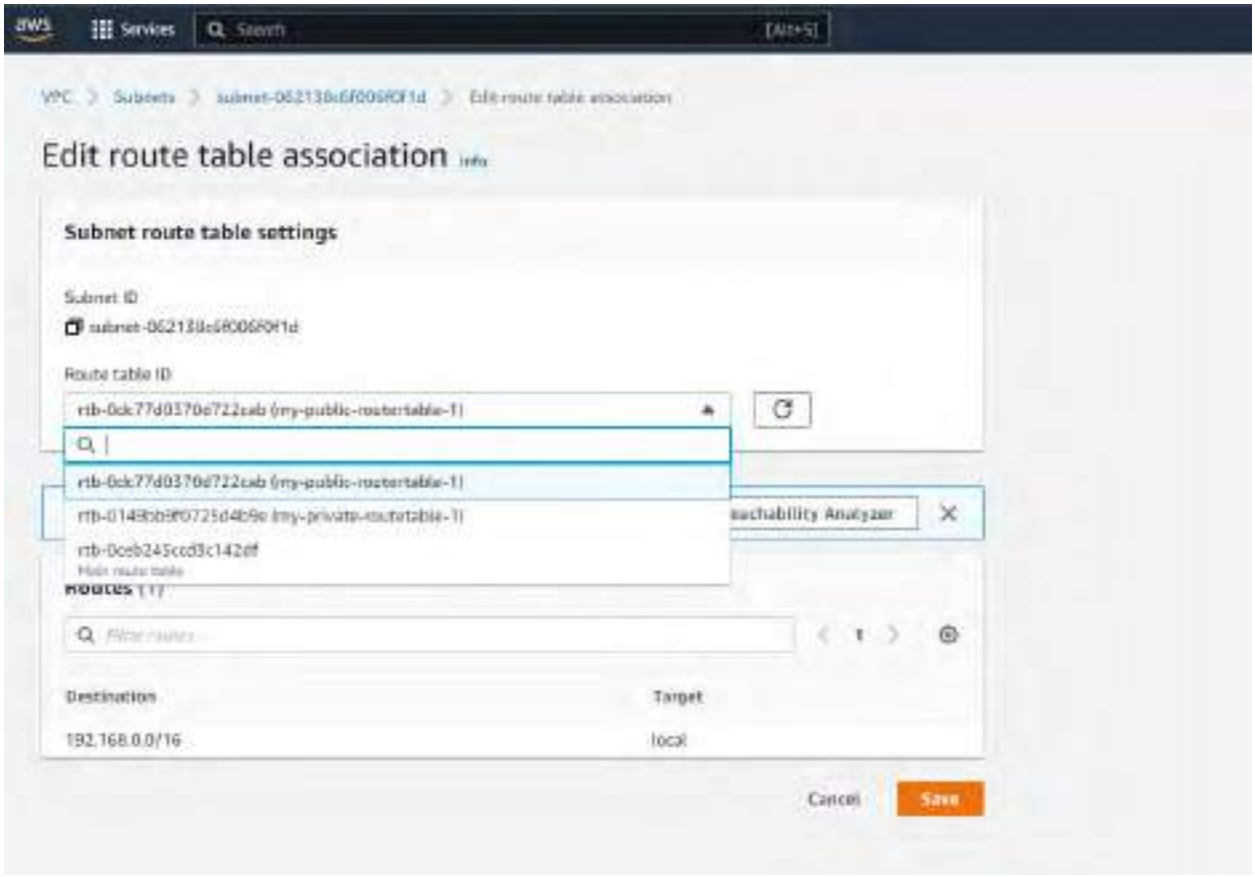


15. Associate public subnets to my-public-routetable1 and Associate private subnets to my-private-routetable1

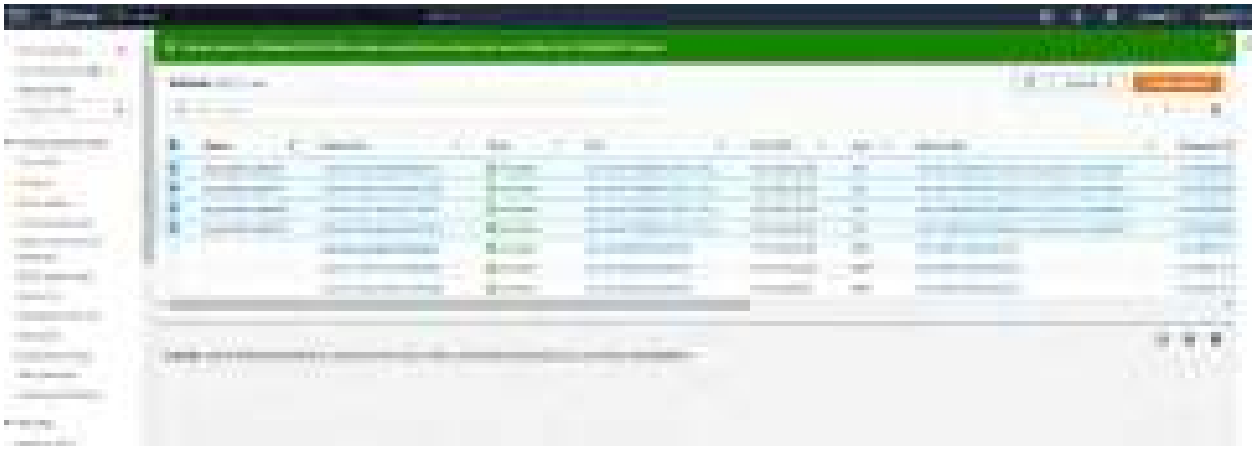
16. Go to Subnets, select my-public-subnet1, and Edit route table association



17. Select my-public-routetable1 from the drop down and click on save



18. Similarly perform the same activity for the remaining association and you can see the below screenshot with all the associations

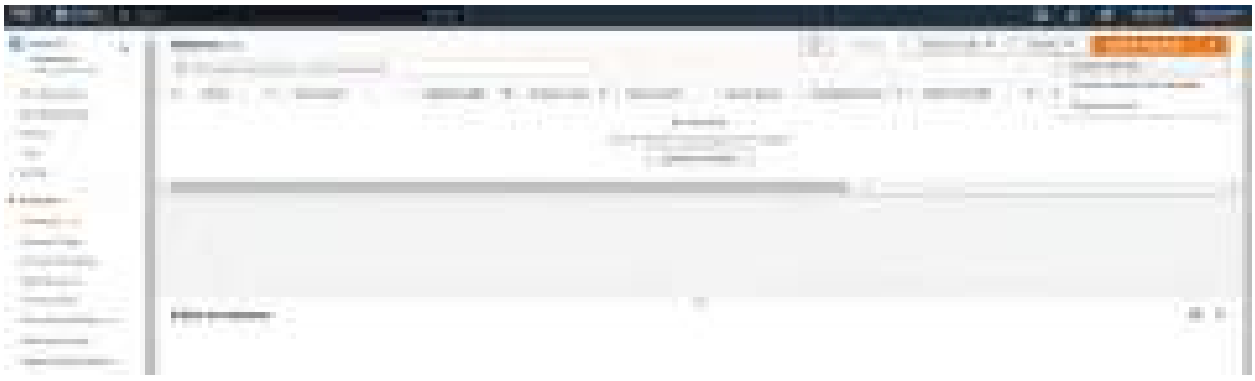


19. Connect the public route table to internet gateway
20. Go to Route tables, select my-public-routetable1 and click on Edit Routes and add 0.0.0.0/0 associated with Internet Gateway (My-Madhav-InternetGateway)

21. Click on Save changes, now you will see the route is associated with Internet gateway

22. Create two instances one in Public Subnet1 and other machine in Private subnet1

23. Search for EC2 and select EC2 and select Launch Instance



24. During the EC2 instance creation, change the settings under Network Settings

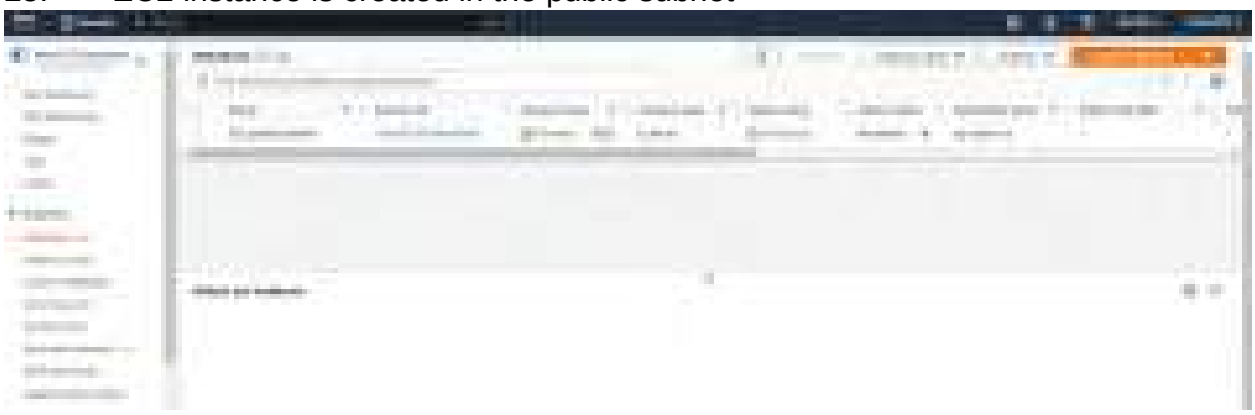
a. VPC –> My-VPC-Madhav

b. Subnet as → my-public-subnet1(192.168.2.0/24)

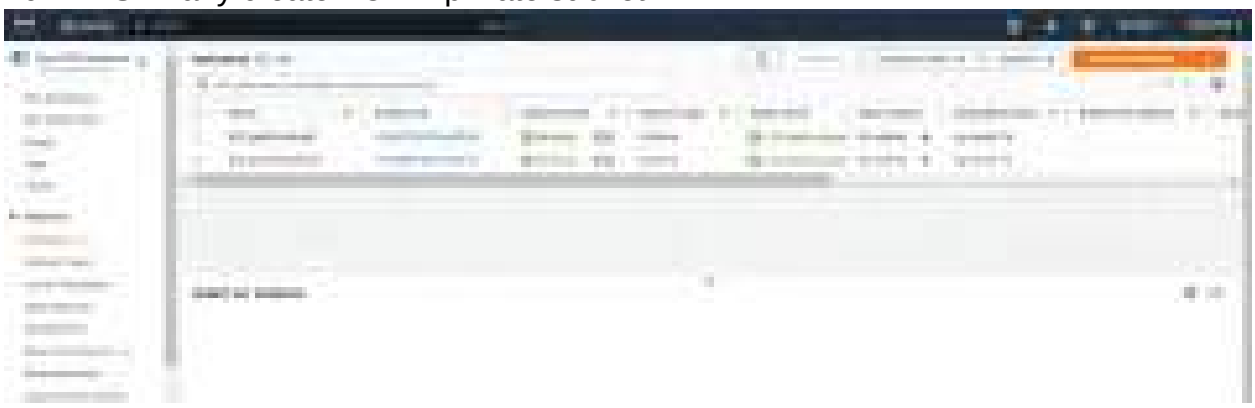
c. Create a security Group as → Public-Subnet-Secgroup1 with SSH and anywhere is enabled

d. Click on Launch Instance

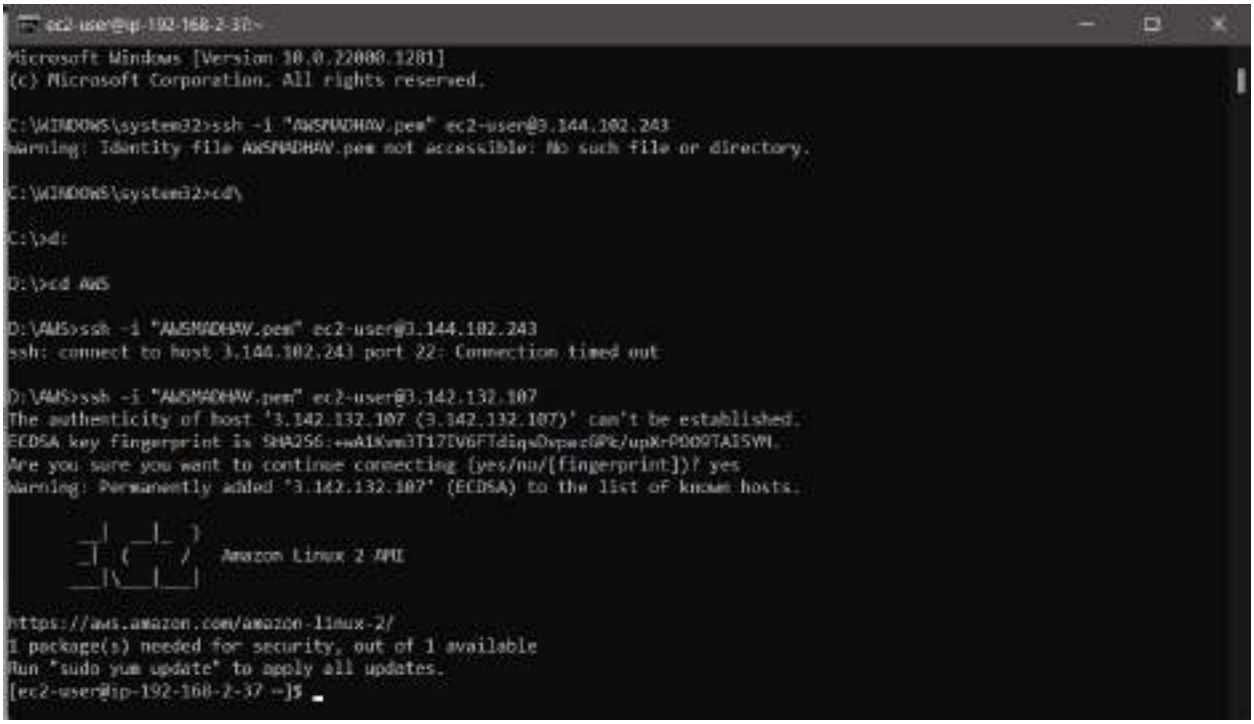
25. EC2 instance is created in the public subnet



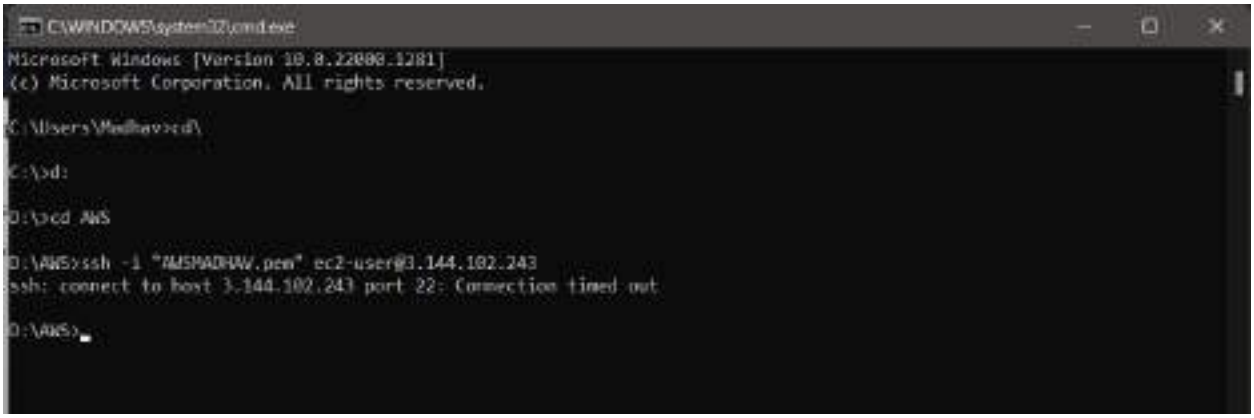
26. Similarly create EC2 in private subnet 1



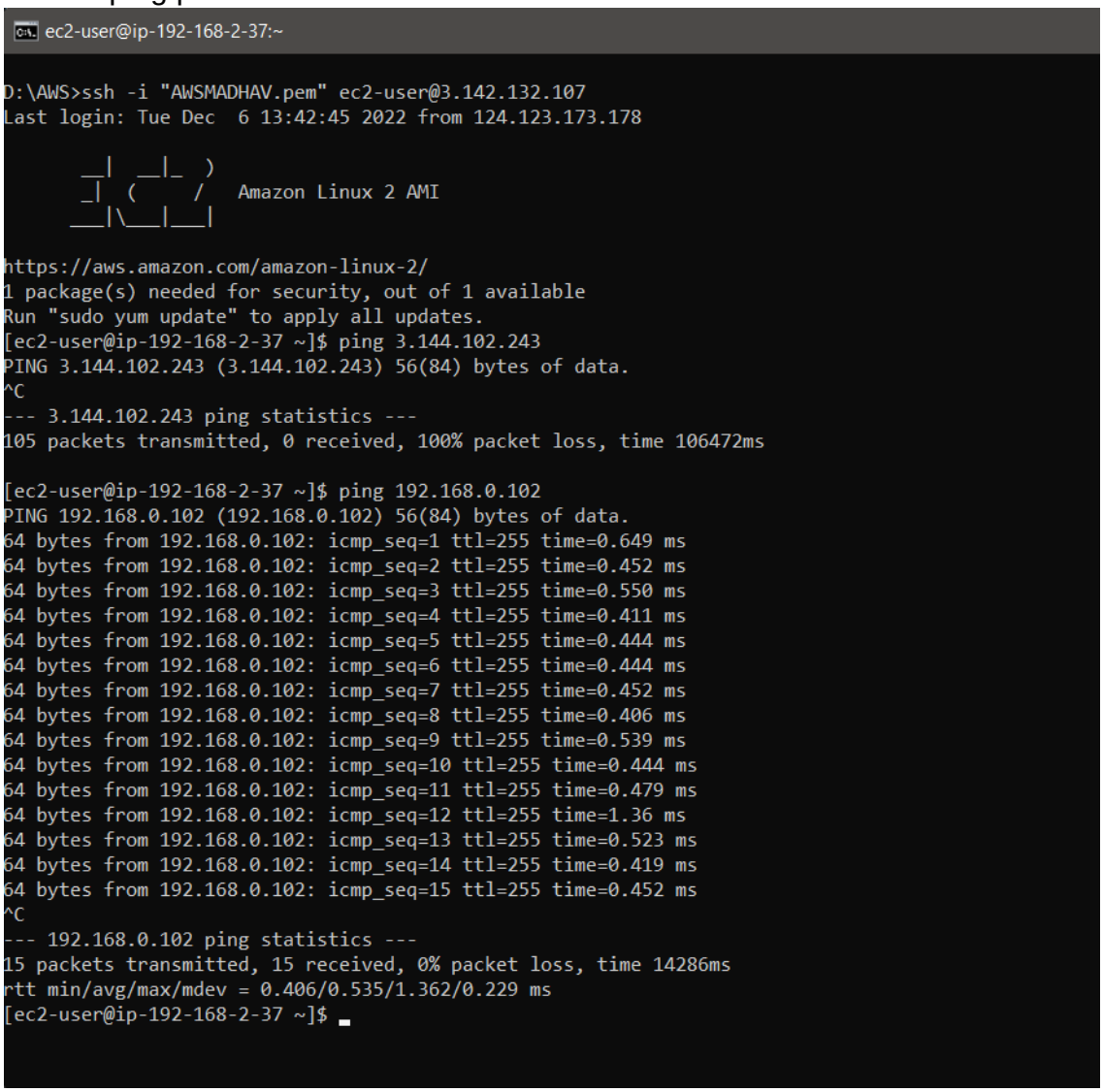
25. Connect to public subnet EC2 machine. Connected successfully



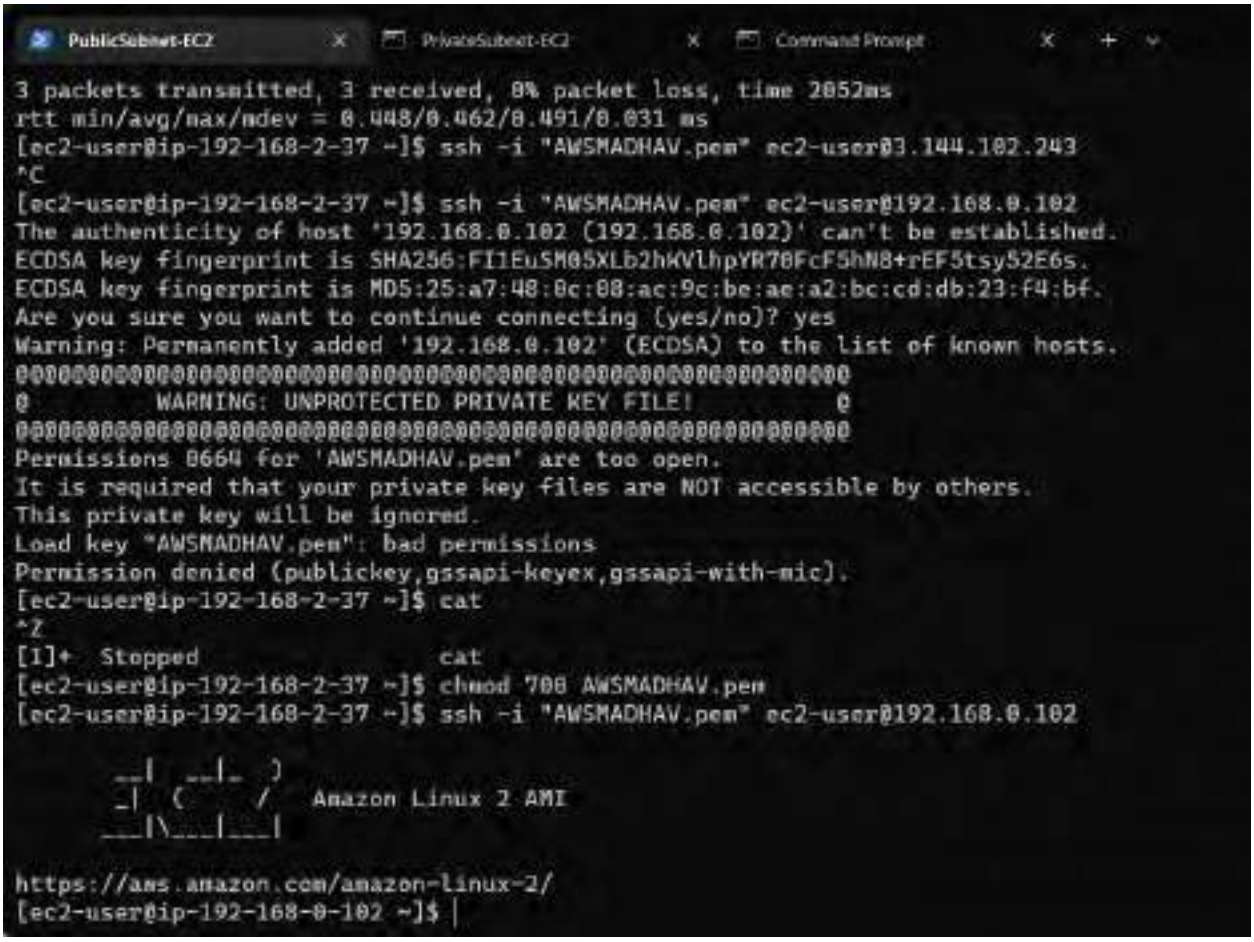
26. Connect to Private subnet EC2 machine, which will not connect



27. Able to ping private IP address 192.168.0.102 from Public Subnet machine



28. Was able to ping but SSH doesn't work, so copied the AWSMADHAV.pem key to public EC2 machine and SSH works using the private IP address

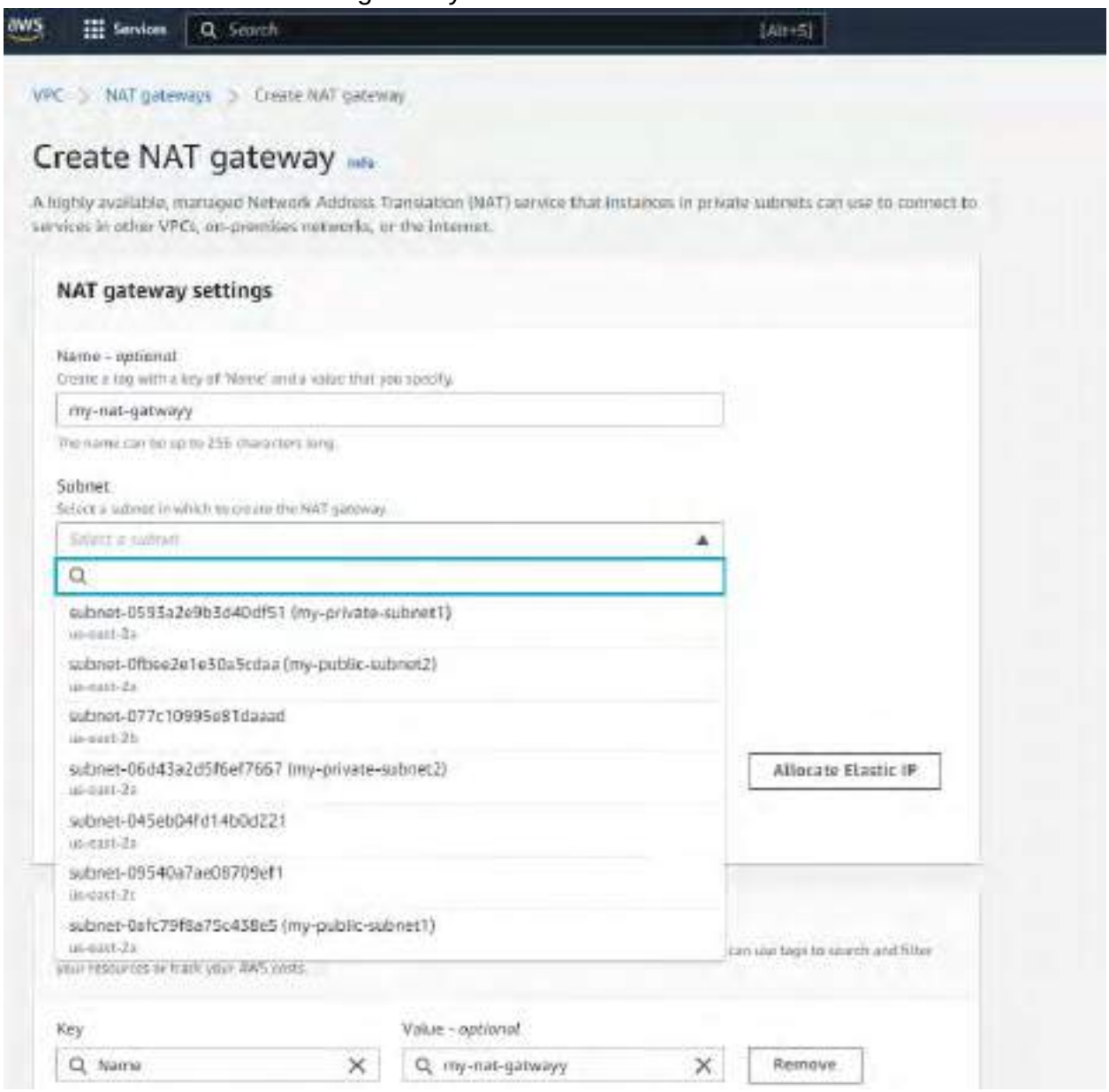


29. Internet works on the public subnet EC2 machine and not Private Subnet EC2 machine because the private subnet EC2 is not connected to the internet.

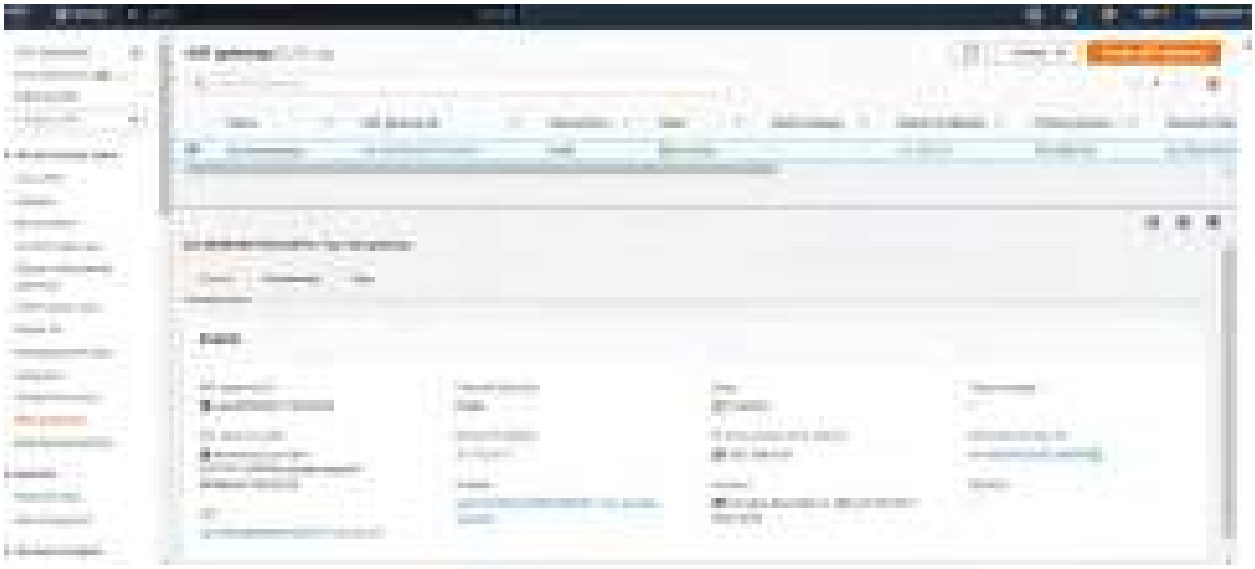
```
[ec2-user@ip-192-168-2-37 ~]$ cat
^C
[ec2-user@ip-192-168-2-37 ~]$ ssh -i "AWSMADHAV.pem" ec2-user@192.168.0.102
Last login: Tue Dec  6 14:10:09 2022 from 192.168.2.37

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-0-102 ~]$ ping google.com
PING google.com (142.250.190.142) 56(84) bytes of data.
^C
--- google.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2036ms

[ec2-user@ip-192-168-0-102 ~]$ exit
logout
Connection to 192.168.0.102 closed.
[ec2-user@ip-192-168-2-37 ~]$ ping google.com
PING google.com (142.251.32.14) 56(84) bytes of data.
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=1 ttl=104 time=17.0 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=2 ttl=104 time=17.0 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=3 ttl=104 time=17.6 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=4 ttl=104 time=17.1 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=5 ttl=104 time=17.0 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=6 ttl=104 time=17.1 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=7 ttl=104 time=17.0 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=8 ttl=104 time=17.0 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=9 ttl=104 time=17.1 ms
64 bytes from ord38s33-in-f14.1e100.net (142.251.32.14): icmp_seq=10 ttl=104 time=17.3 ms
^C
--- google.com ping statistics ---
```

30. To connect internet on Private Subnet EC2 machine, create a NAT Gateway
31. Create Nat Gateway from VPC console, click on NAT Gateway and Create NAT Gateway

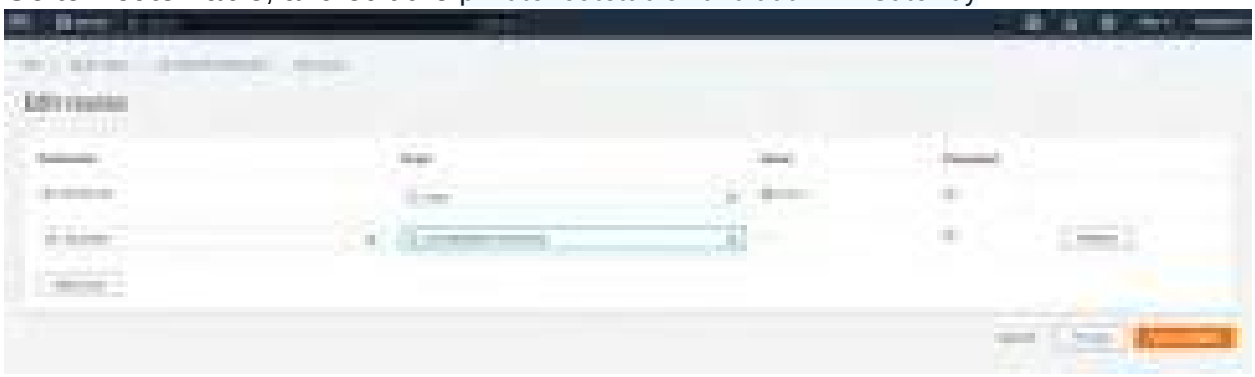
32. Give name as – my-nat-gatway and select my-private-subnet1 and allocate the elastic IP, as we are trying to connect to the internet from the private subnetEC2 machine via nat gateway



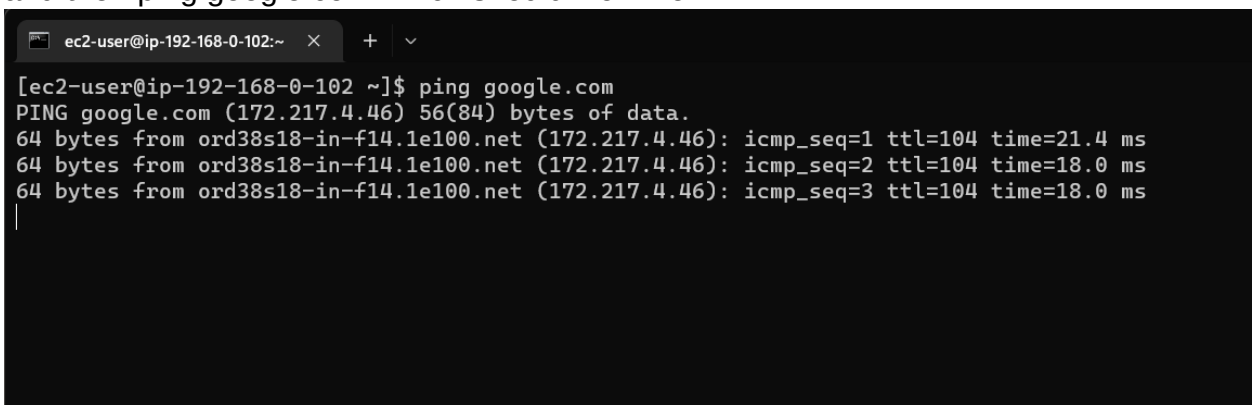
33. NAT Gateway is created



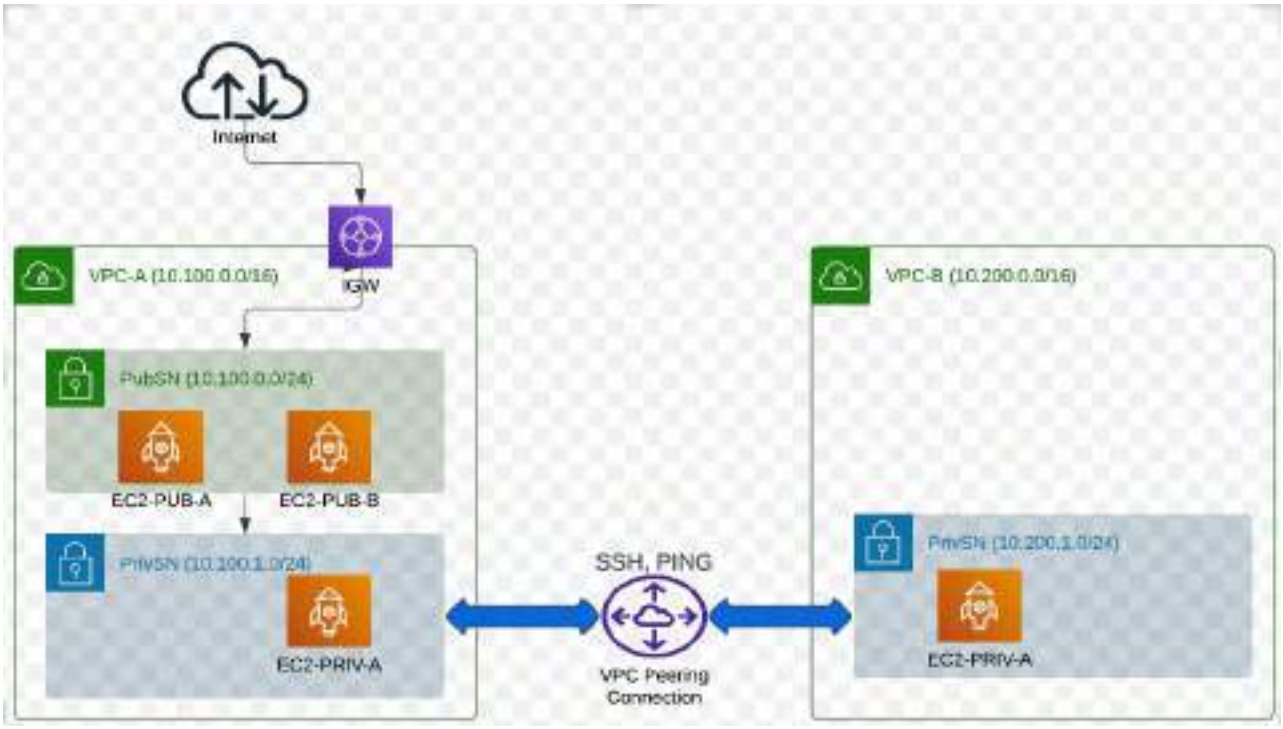
34. Go to Route Table, and edit the private-routetable1 and add NAT Gateway



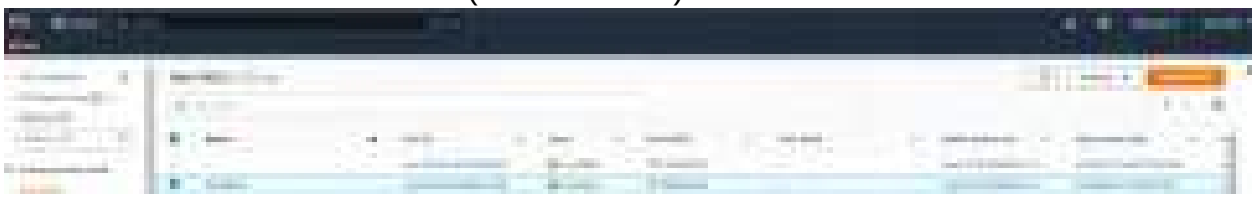
35. Connect private subnet EC2 machine from the Public Subnet EC2 machine and then ping google.com which should work now



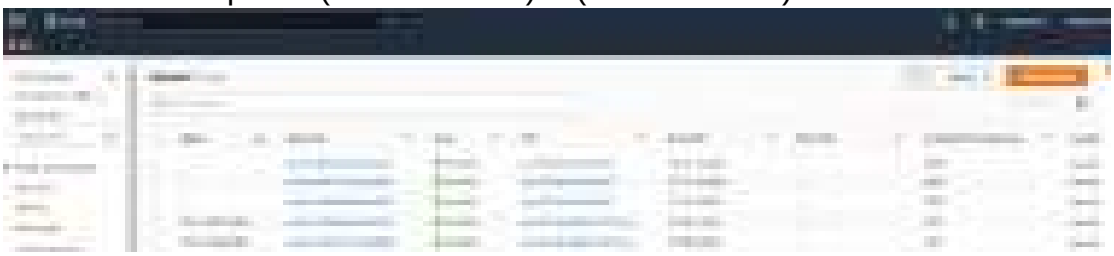
7. VPC Peering



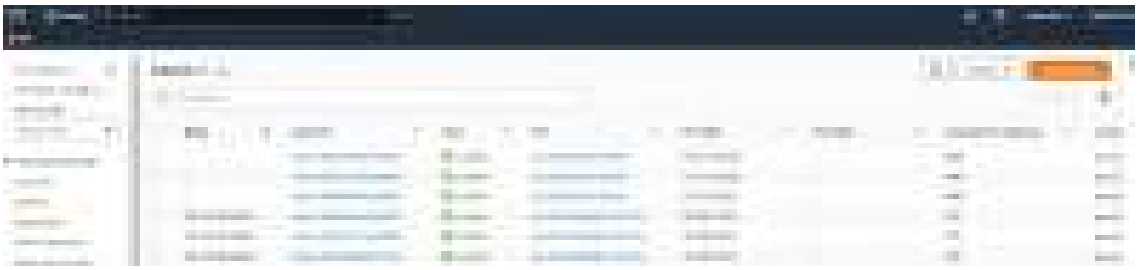
1. To create a VPC peering connection with VPCs in the same account and Region
2. Created “VPC A" with 3 subnets (10.100.0.0/16)



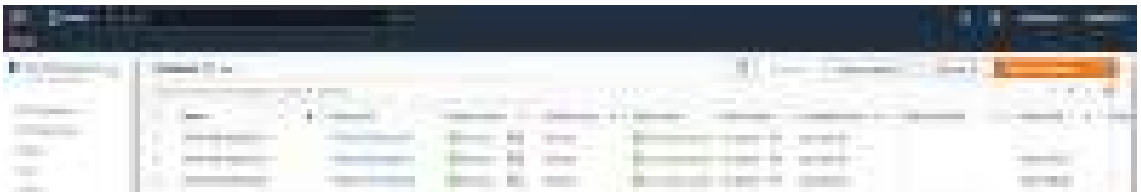
   a. Two subnets in public (10.100.0.0/24) & (10.100.2.0/24)



   b. One subnet in Private (10.100.1.0/24)

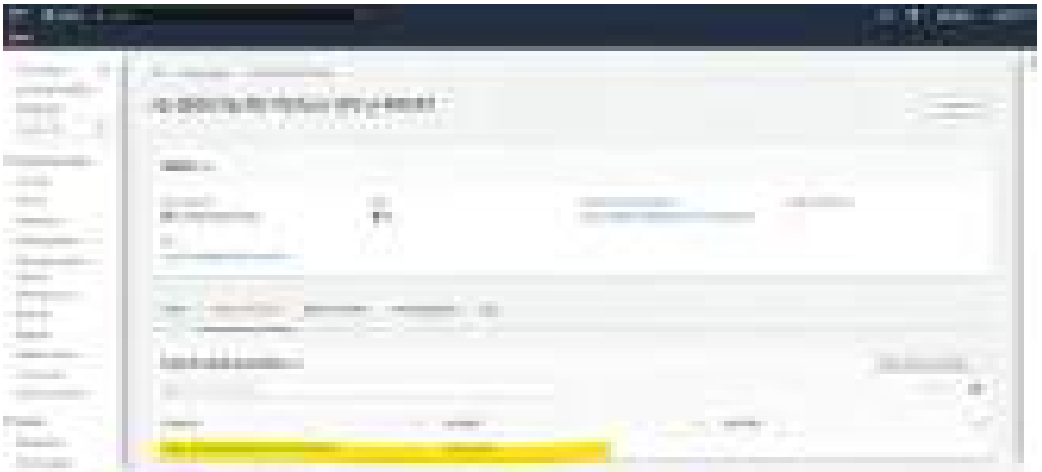


c. Create a EC2 machine in each subnet

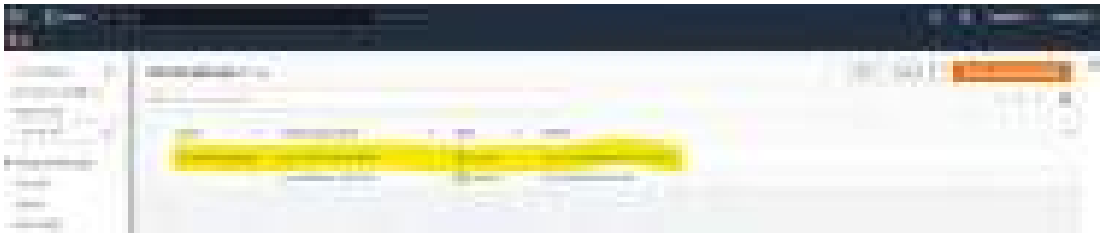


d. Create two route tables VPC-A-PRIVRT and VPC-A-PUBRT
    i. Connect 2 Public EC2 machines to Public route table → VPC-A-PUBRT

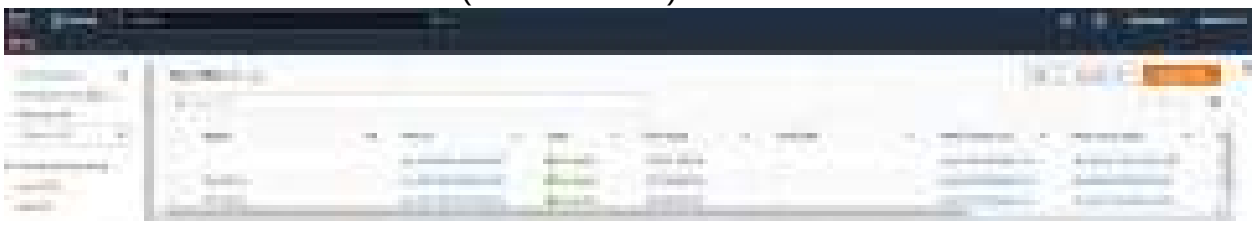
    ii. Connect 1 Private EC2 machine to Private route table → VPC-A-PRIVRT



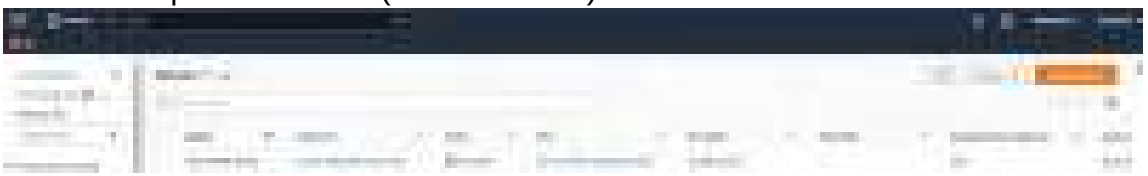
e. Attach Internet Gateway for VPC-A



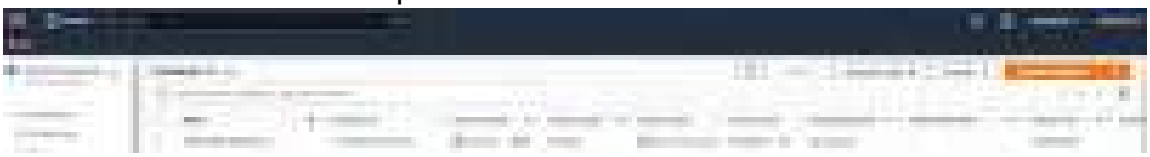
3. Create a "VPC B" with 1 subnet (10.200.0.0/16)



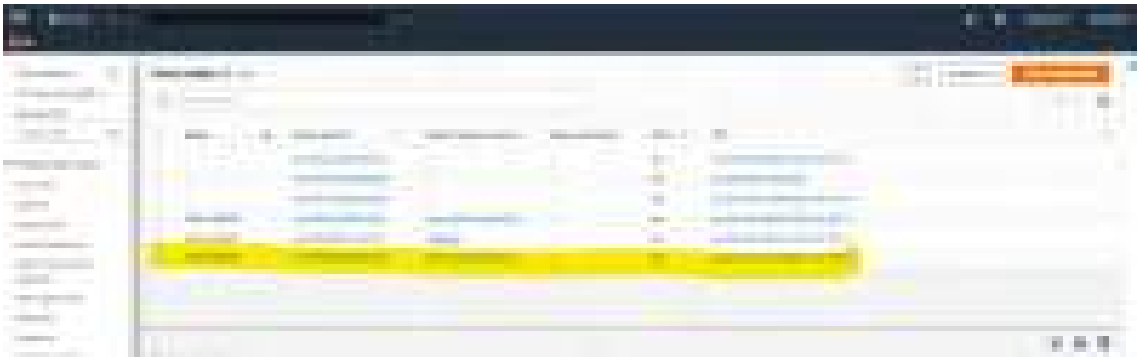
   a. Create one private subnet (10.200.1.0/24)



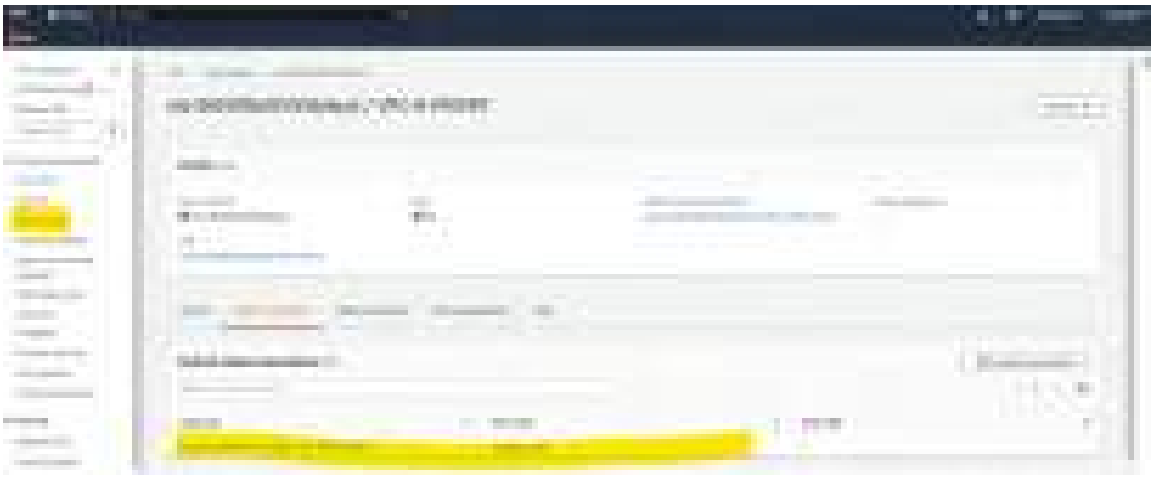
   b. create one EC2 instance in private subnet



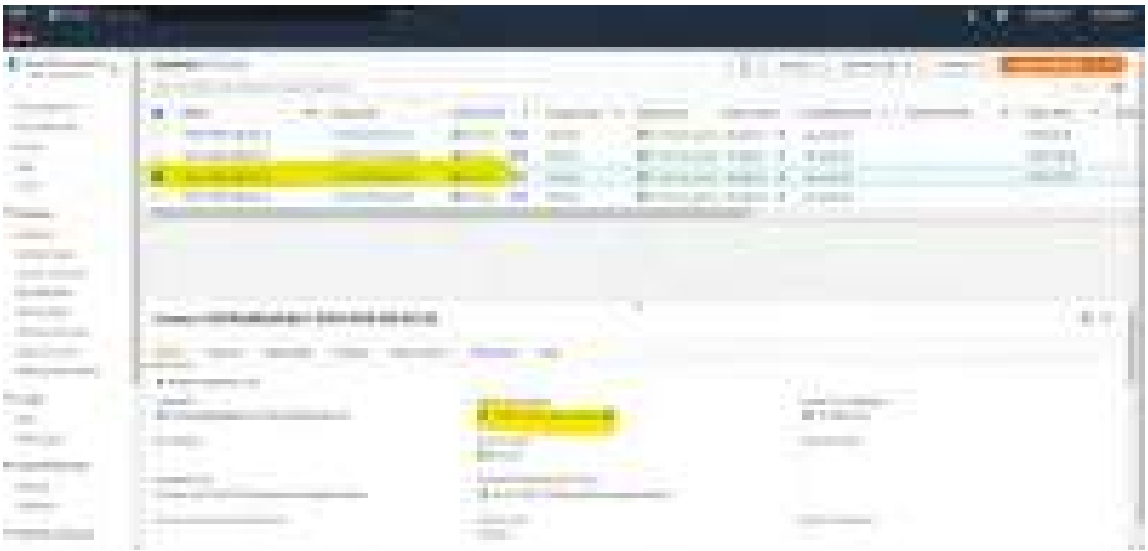
   c. Create one route table VPC-B-PRIVRT



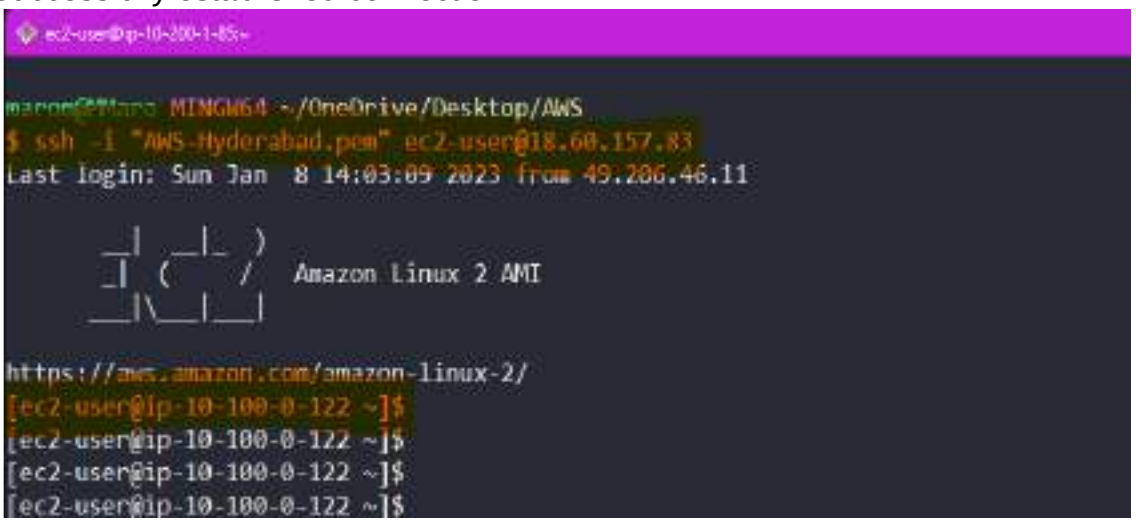
   d. Connect the EC2 machine to Route table



4. Check connectivity
    a. Connect to VPCA-PUB-SUB-EC2-A machine



Successfully established connection



b. Now try connecting to VPCA-PUB-SUB-EC2-A EC2 machine, which will fail because of non-availability of the key
c. Transfer the key to the EC2 machine with the command
    i. SCP -i .\AWS-Hyderabad.pem -r .\AWS-Hyderabad.pem ec2-user@18.60.157.83:/home/ec2-user

```
PS C:\Users\marom\OneDrive\Desktop\AWS> scp -i .\AWS-Hyderabad.pem -r .\AWS-Hyderabad.pe
m ec2-user@18.60.157.83:/home/ec2-user
AWS-Hyderabad.pem                             100% 1674   272.6KB/s   00:00
PS C:\Users\marom\OneDrive\Desktop\AWS>
```

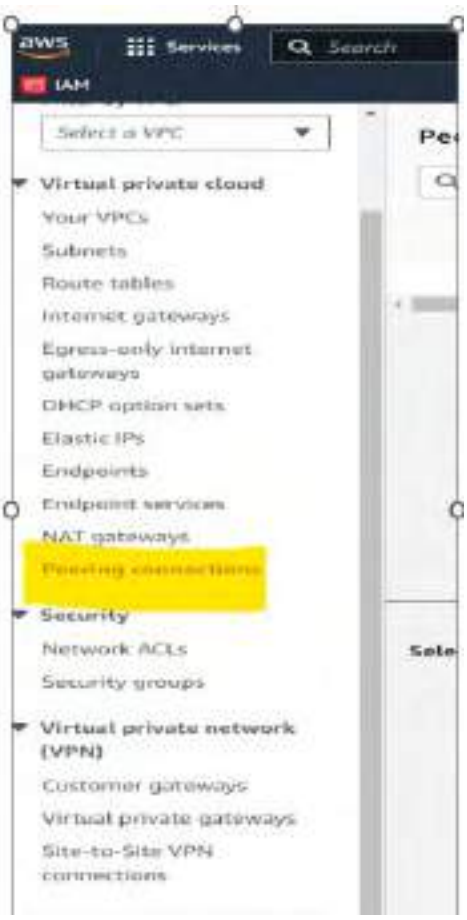
    ii. Once the key is transferred check the SSH connection, as you can see 10.100.0.122 is VPCA-PUB-SUB-EC2-A machine and 10.100.1.242 is VPCA-PRIV-SUB-EC2-A machine which is successful

```
[ec2-user@ip-10-100-0-122 ~]$ ssh -i AWS-Hyderabad.pem ec2-user@10.100.1.242
Last login: Sun Jan  8 14:03:39 2023 from 10.100.0.122

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-100-1-242 ~]$
[ec2-user@ip-10-100-1-242 ~]$
[ec2-user@ip-10-100-1-242 ~]$
```

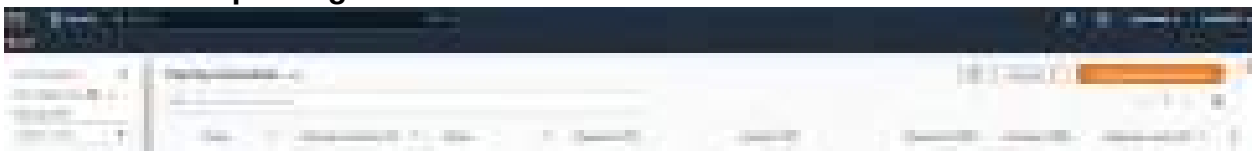
    iii. Now try to SSH to 10.200.1.85 which is a VPCB-PRIV-SUB-EC2-A machine from 10.100.1.242 which is VPCA-PRIV-SUB-EC2-A machine

As this doesn't work because there is connectivity from VPC-A to VPC-B, so we need to create a VPC peering

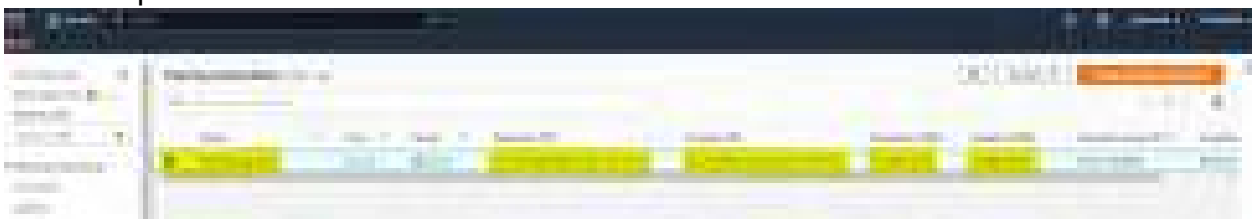
5. Setup VPC peering



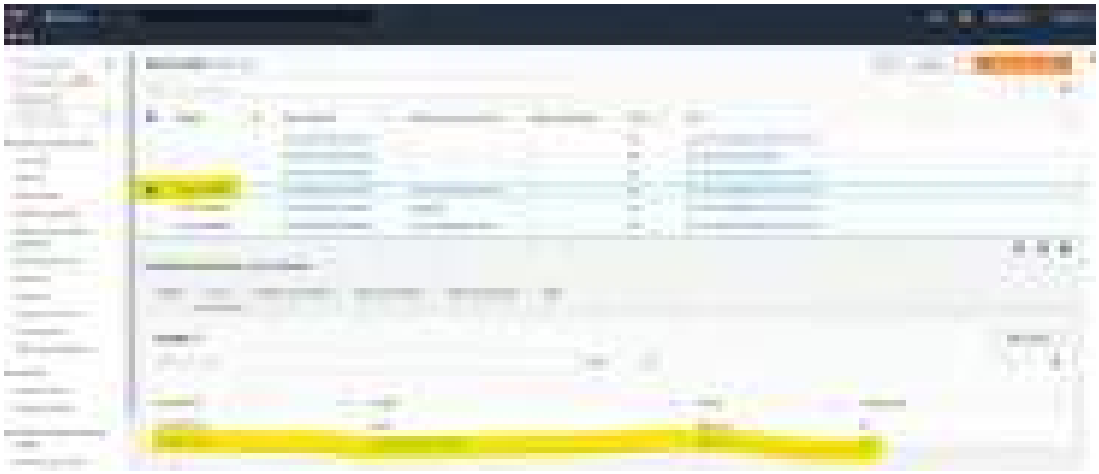
6. Choose **Create peering connection.**

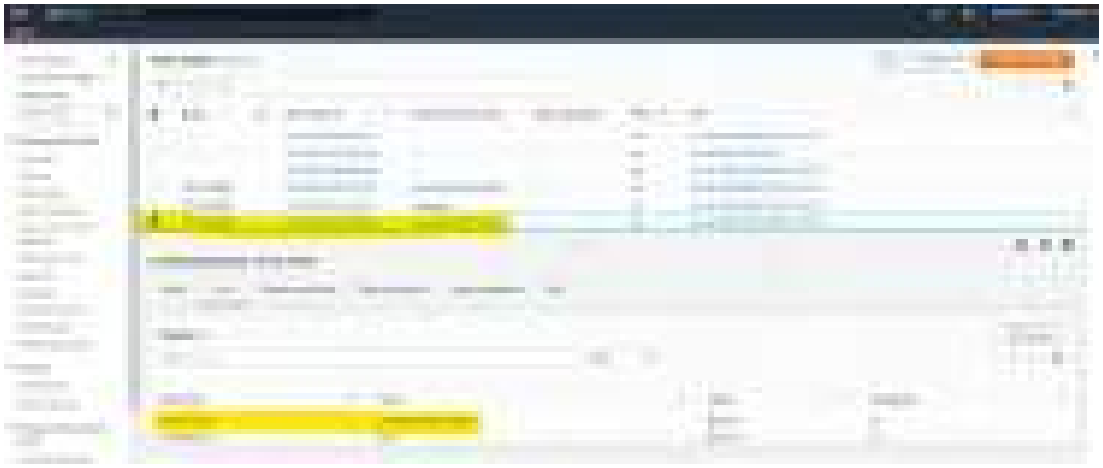


7. Create VPC peering between VPC-A and VPC-B and associate the requester and Acceptor VPC



8. Once this is done, we need to also establish the routing between VPC-A PRIVSUBNT-EC2-B machine and VPC-B PRIVSUBNT-EC2-A





9. Once this is done, to perform SSH connection we need to transfer the key from VPCA-PUB-SUB-EC2-A to VPCA-PRIV-SUB-EC2-A machine.
10. The SSH connection is established now. Now we can establish the connection from VPC-B-PRIV-SUBNT-EC2A to VPCA-PRIV-SUBNT-EC2B machine successfully.

```
[ec2-user@ip-10-100-0-122 ~]$ scp -i AWS-Hyderabad.pem -r AWS-Hyderabad.pem ec2-
user@10.100.1.242:/home/ec2-user
AWS-Hyderabad.pem                             100% 1674     2.0MB/s   00:00
[ec2-user@ip-10-100-0-122 ~]$ ssh -i AWS-Hyderabad.pem ec2-user@10.100.1.242
Last login: Sun Jan  8 14:18:19 2023 from 10.100.0.122

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-100-1-242 ~]$
[ec2-user@ip-10-100-1-242 ~]$
[ec2-user@ip-10-100-1-242 ~]$ sudo su
[root@ip-10-100-1-242 ec2-user]# ls
AWS-Hyderabad.pem
[root@ip-10-100-1-242 ec2-user]# ssh -i AWS-Hyderabad.pem ec2-user@10.200.1.85

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-200-1-85 ~]$
[ec2-user@ip-10-200-1-85 ~]$
[ec2-user@ip-10-200-1-85 ~]$
```