



Assignments for Cloud and Devops

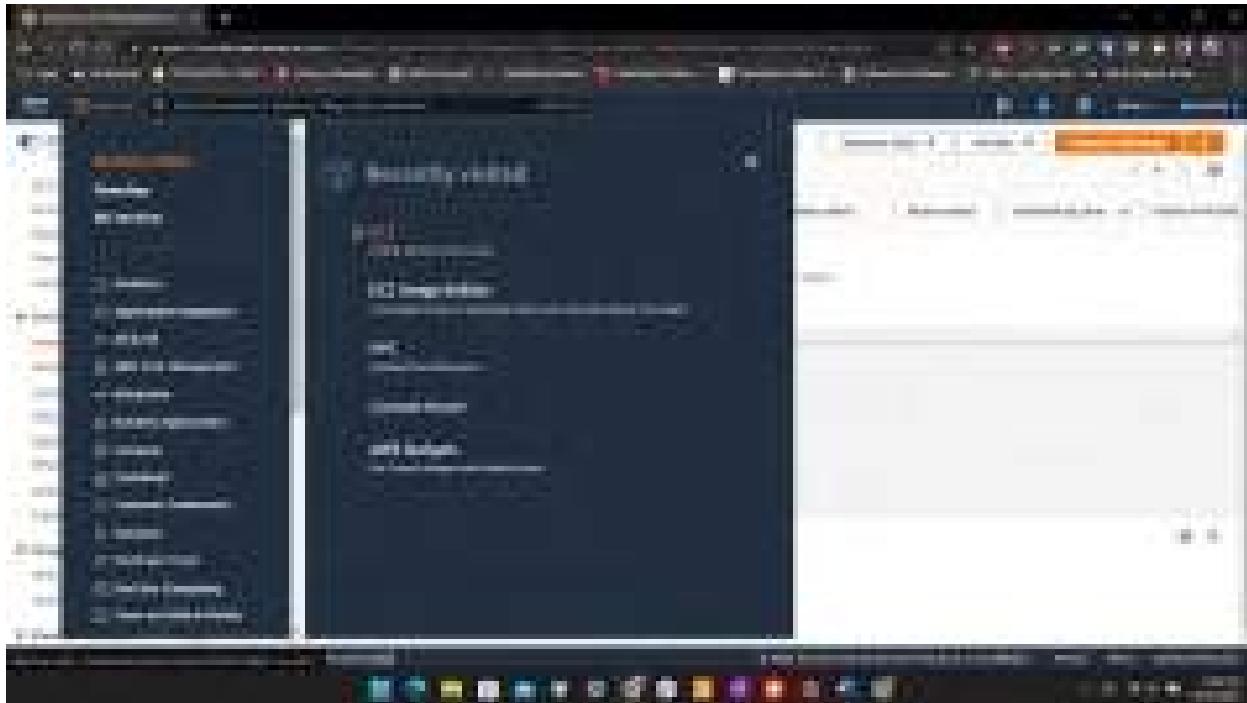
AWS Assignment

Table Of Content

1.	Create EC2.....	2
2.	Create Elastic Block Store	10
3.	Snapshot screenshot creation.....	13
4.	Create AMI	14
5.	Load Balancer Creation.....	15
6.	Create VPC	29
7.	VPC Peering	46

1. Create EC2

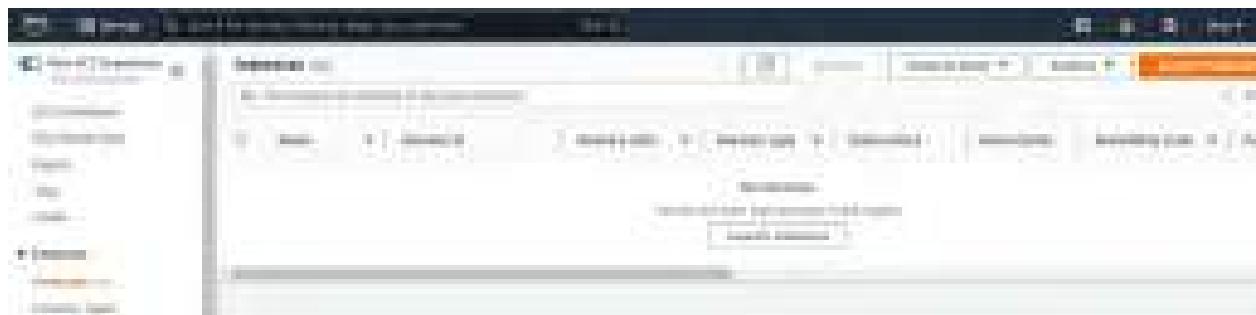
1. Login to AWS
2. Select Services → EC2



3. Click on Instances (running)

A screenshot of the AWS EC2 Instances page. The left sidebar shows 'EC2 Dashboard' with links for 'EC2 Global View', 'Events', 'Tags', 'Limits', and 'Instances'. Under 'Instances', there are links for 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', and 'Savings Plans'. The main content area has a title 'Resources' and a message: 'You are using the following Amazon EC2 resources in the US East (Ohio) Region.' Below this are four tables showing counts for different resources: Instances (0), Dedicated Hosts (0), Elastic IPs (0), Instances (0), Key pairs (1), Load balancers (0), Placement groups (0), Security groups (6), and Volumes (0). A callout box at the bottom right says: 'Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more!' with a close button 'X'.

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



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

ami-089a545a9ed9893b6 (64-bit (x86)) / ami-04897acc32ef05c (64-bit (Arm))

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

Free tier eligible

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20221004.0 x86_64 HVM gp2

Architecture

AMI ID

64-bit (x86)

ami-089a545a9ed9893b6

Verified provider

▼ Instance type Info

Instance type

t2.micro

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

Free tier eligible

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

Q

Proceed without a key pair (Not recommended) Default value

AWSMADHAV

Type: rsa

Create new key pair

Edit

▼ Network settings Info

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 Anywhere
Helps you connect to your instance
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

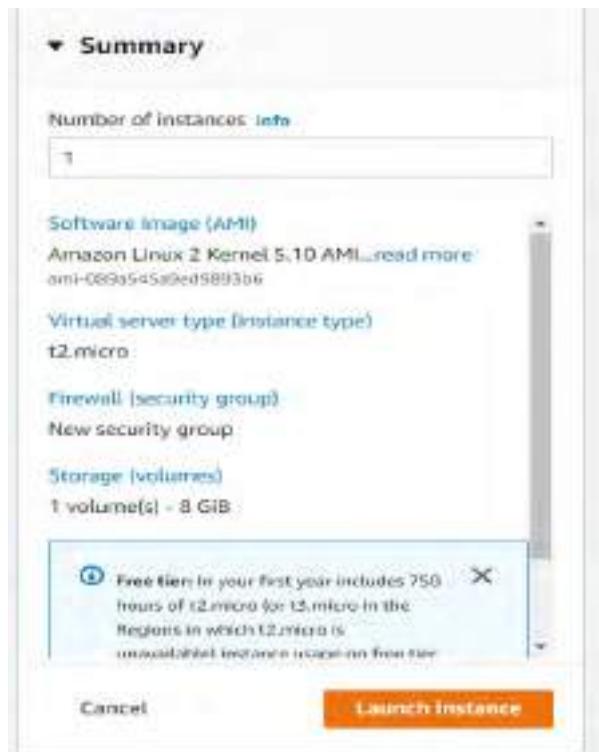
1x GiB Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

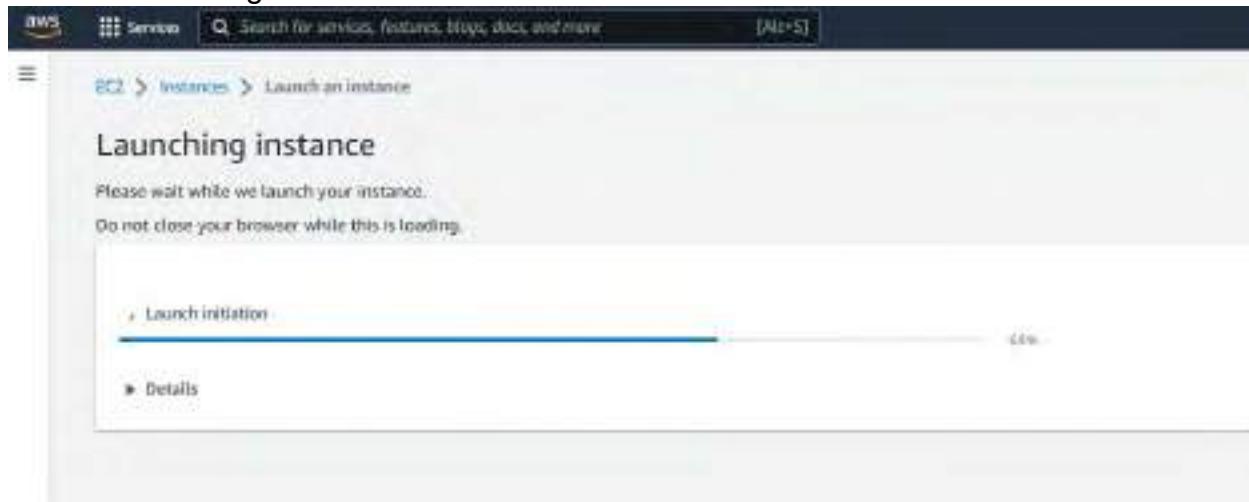
Add new volume

Ox File systems Edit

Click on Launch Instance



7. Launching instance



8. Instance created successfully

The screenshot shows the AWS RDS Instances page. At the top, there is a success message: "Successfully created database instance AWS-Madhav [View details]". Below this, there is a link to "Learn more". The main area is titled "Next Steps" and contains three sections:

- Create billing and usage storage alerts**: A note about managing costs and a button to "Create alert items".
- Connect to your instance**: A note about connecting to the instance using MySQL client tools, a "Download MySQL client" button, and two "Connect" buttons.
- Configure an RDS database**: A note about configuring the database, a "Configure RDS database" button, and two "Launch instance" buttons.

9. Instance created with name AWS-Madhav

The screenshot shows the AWS RDS Instances page with a list of instances. One instance is highlighted in blue, showing its details:

Instance identifier	DB instance class	Engine	Storage type	Storage capacity	Allocated storage	Port	Master username	Character set	Collation	Region	Endpoint	Publicly accessible	Storage type	Storage capacity	Allocated storage	Port	Master username	Character set	Collation	Region	Endpoint	Publicly accessible
AWS-Madhav	db.t2.micro	MySQL	Amazon S3	5 GB	5 GB	3306	aws-madhav	utf8	latin1	us-east-1	aws-madhav.*****.amazonaws.com	No	Amazon S3	5 GB	5 GB	3306	aws-madhav	utf8	latin1	us-east-1	aws-madhav.*****.amazonaws.com	No



10. Connecting to the instance using SSH Client

EC2 > Instances > i-03112de87fba22c7b > Connect to Instance

Connect to instance Info

Connect to your instance i-03112de87fba22c7b (AWS-Madhav) using any of these options:

[EC2 Instance Connect](#) [Session Manager](#) [SSH client](#) [EC2 serial console](#)

Instance ID

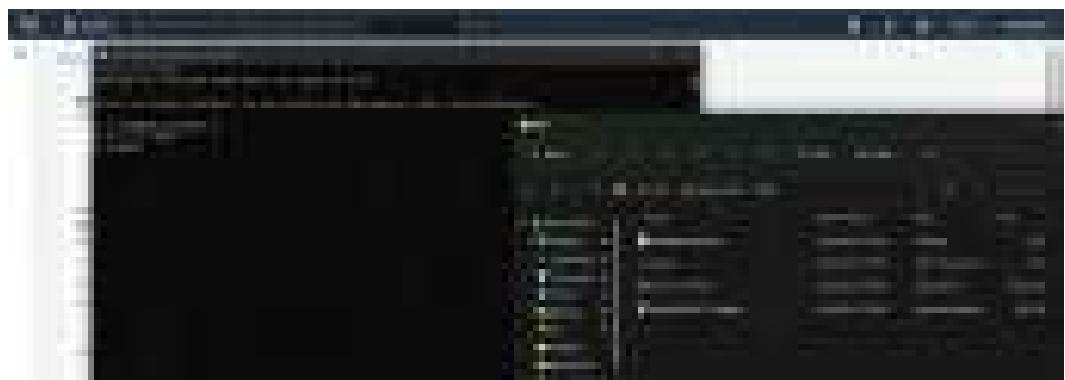
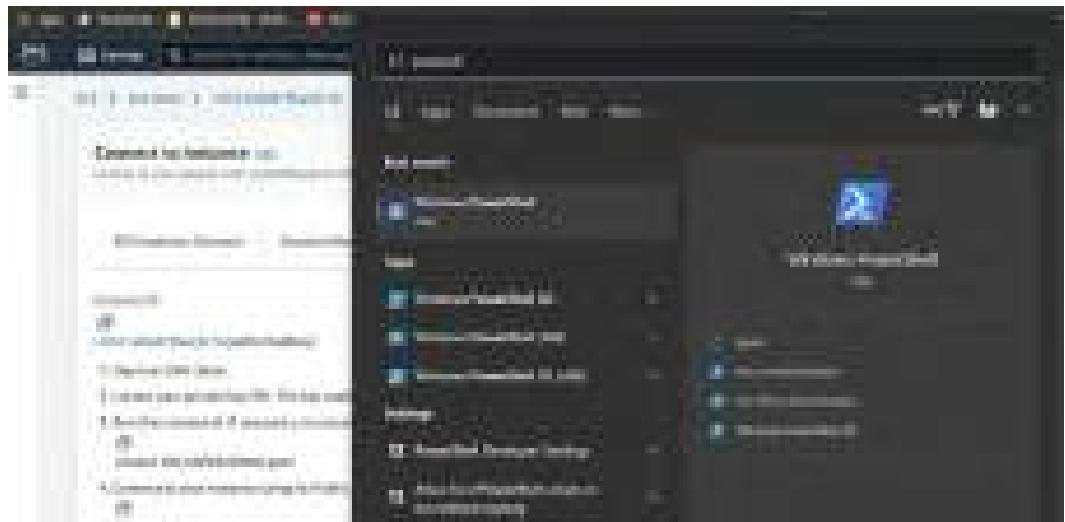
i-03112de87fba22c7b (AWS-Madhav)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is AWSMADHAV.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 AWSMADHAV.pem`
4. Connect to your instance using its Public DNS.
 `ec2-18-190-24-76.us-east-2.compute.amazonaws.com`

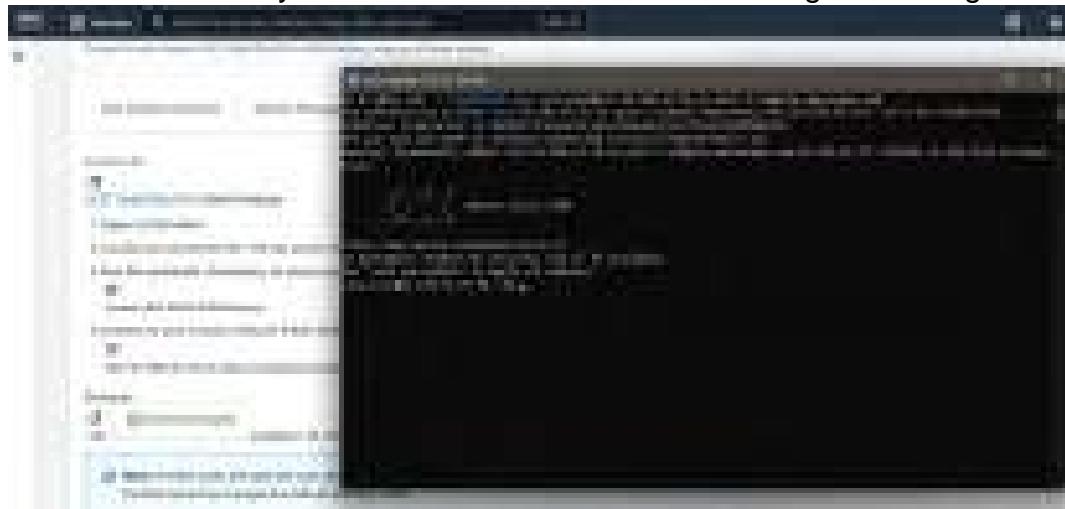
Example:

`ssh -i 'AWSMADHAV.pem' ec2-user@ec2-18-190-24-76.us-east-2.compute.amazonaws.com`

ⓘ Notes In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if



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.

The screenshot shows the 'Create volume' wizard step 1: Volume settings. It includes fields for Volume type (General Purpose SSD gp2), Size (1 GiB), IOPS (1000), Throughput (100 MiB/s), and Availability Zone (us-east-1a). A note says 'This volume must be attached to an instance'. Below is a 'Tags - optional' section with a note about tags being used for search and filtering resources, and a 'No tags associated with this resource' message. At the bottom are 'Add tag' and 'Create volume' buttons.

Volume settings

Volume type: General Purpose SSD gp2

Size (GiB): 1

IOPS: 1000

Throughput: 100 MiB/s

Availability Zone: us-east-1a

Tags - optional

No tags associated with this resource.

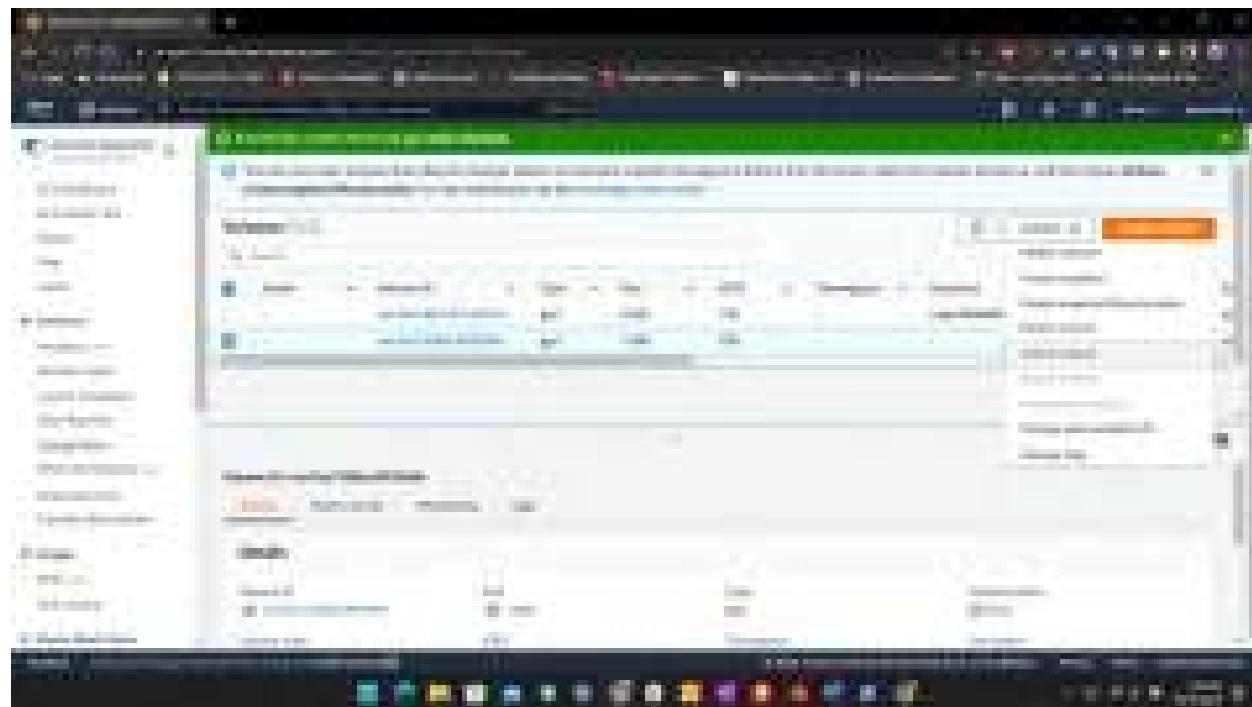
Add tag

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.

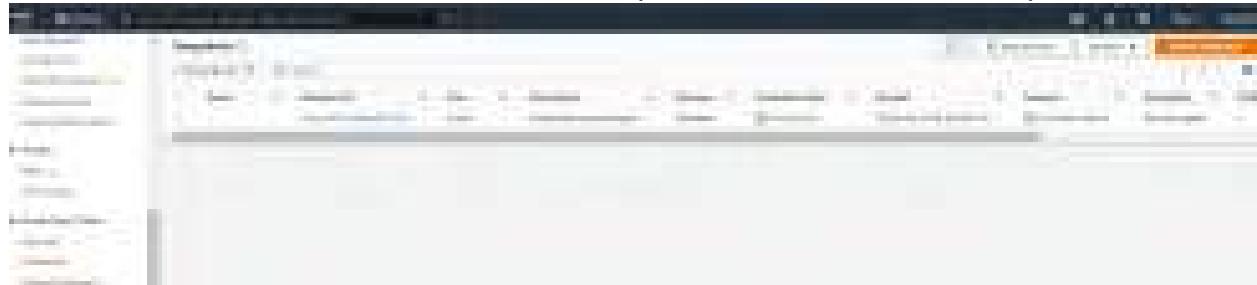


7. Confirmation on the volume is attached successfully

```
aws Services Q: sudo
last login: Mon Oct 31 06:52:05 2022 from ec2-3-16-146-4.us-east-2.compute.amazonaws.com
[ec2-user@ip-172-31-23-98 ~]$ sudo su
[root@ip-172-31-23-98 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda   202:0    0 8G  0 disk
└─xvda1 202:1    0 8G  0 part /
[root@ip-172-31-23-98 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda   202:0    0 8G  0 disk
└─xvda1 202:1    0 8G  0 part /
xvdf   202:80   0 1G  0 disk
[root@ip-172-31-23-98 ec2-user]#
```

3. Snapshot screenshot creation

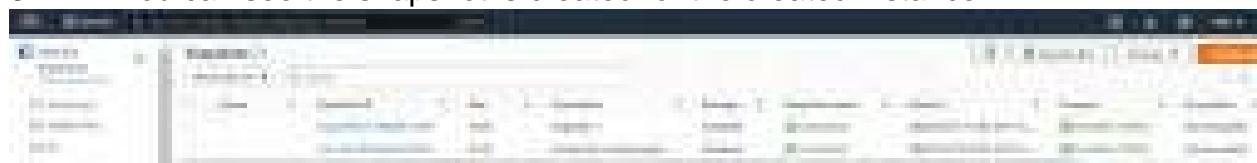
1. Go to Elastic Block Store, select Snapshots, click on Create Snapshot



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

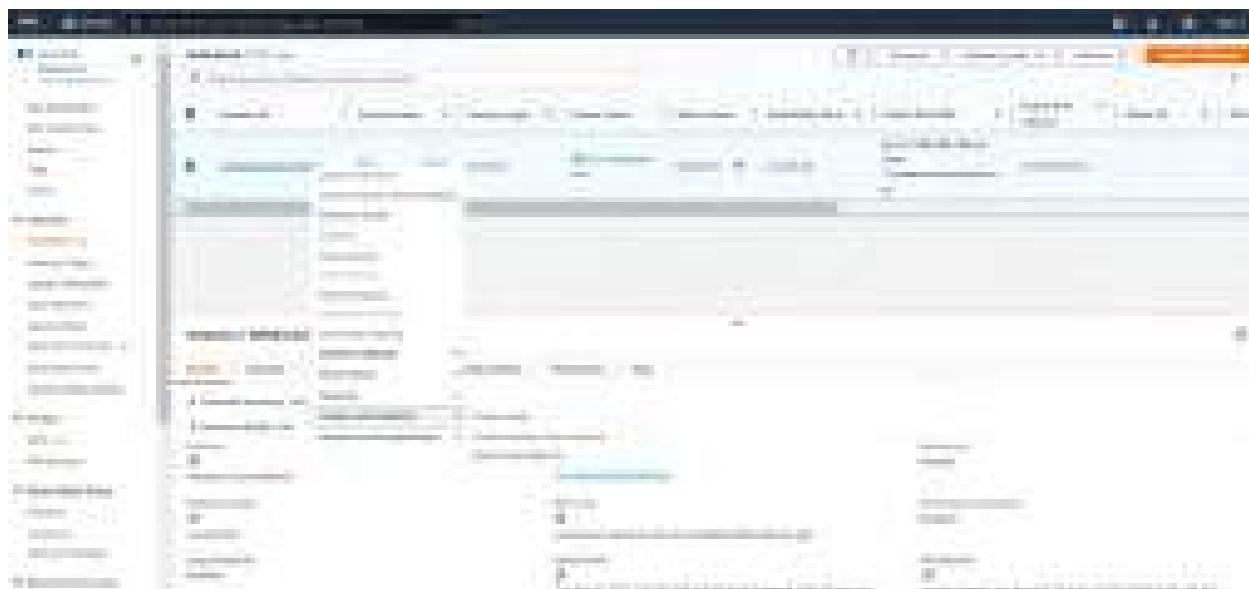
A screenshot of the 'Create snapshot' wizard. At the top, it says 'Create a point-in-time snapshot of an EBS volume and use it as a backup for new volumes or for data backup. You can create snapshots from an individual volume, or you can create multi-volume snapshots from all of the volumes attached to an instance.' Below this is a section titled 'Snapshot settings'. It has two radio button options: 'Volume' (disabled) and 'Instance' (selected). Under 'Instance', there's a dropdown menu labeled 'Select an instance' with the value 'i-03112d607fba22c9b'. At the bottom, there's a 'Tags' section with a note about tags being optional, a 'Add tag' button, and a note about the limit of 50 tags. At the very bottom are 'Cancel' and 'Create Snapshot' buttons.

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**.

Saved Review [] Search for instances, networks, VPCs, roles, and more... (0) (0)

Create Image

ami-01e6-81
 I acknowledge that I understand that this process will stop the instance and create a new Amazon EBS volume. This can result in a loss of configuration or data stored on the instance.

Image name: Maximum 100 characters, including punctuation and whitespace.

Image description: Maximum 255 characters.

Block device mapping - optional: Amazon EBS volumes

Root device: Available

Instance volume(s):

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encryption
EBS	/dev/xvdb	Volume snapshot-100	8	Standard Volume (KLV)	100	100	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Enabled

Add volume

During the image creation process, Amazon EBS creates a snapshot of each of the above volumes.

Tags - optional: A tag is a label that you assign to an AWS resource. Each tag has a key-value pair where both the key and value are strings up to 256 characters long.

Tag image and snapshots together Tag the image and the snapshots it contains together.

Tag image and snapshots separately Tag the image and the snapshots with the same tags.

Use copy command instead of snapshot:

AMI View Tag This can add up to 10 tags per image.

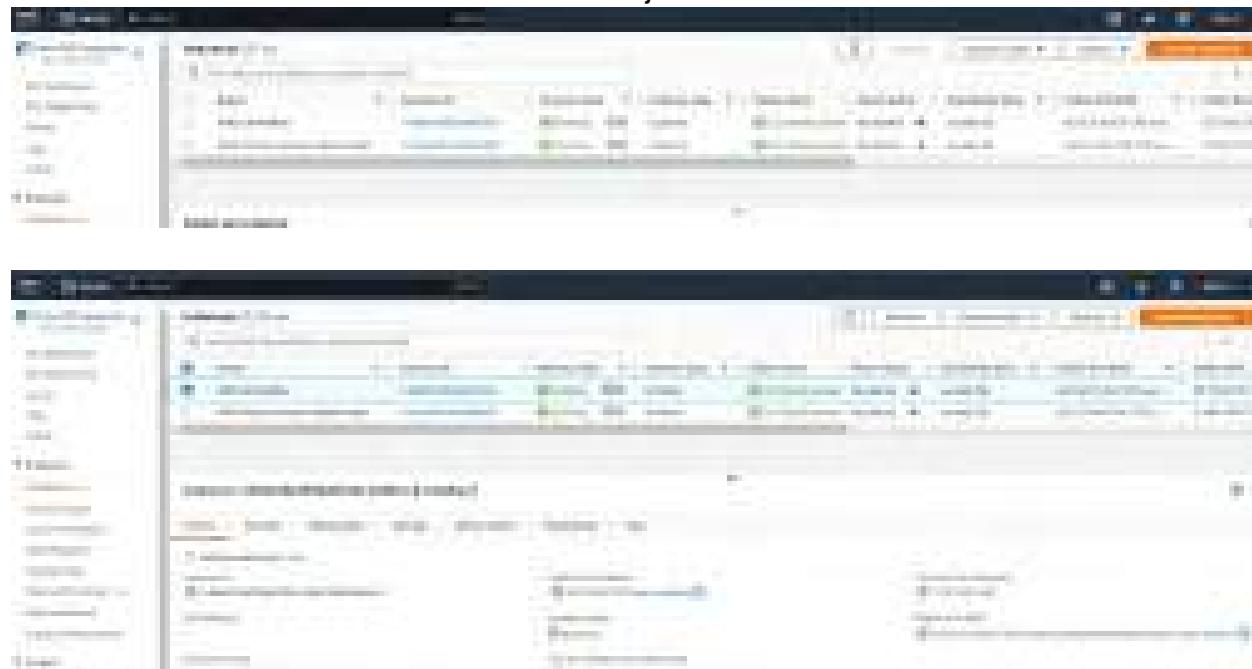
Create **Create Image**

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.

The image contains two screenshots of a web browser. The top screenshot shows the 'Apache2 Default Page' for an Ubuntu system. It features a red 'Ubuntu' logo, the text 'Apache2 Default Page', and a red 'It worked!' button. Below this is a 'Configuration Overview' section with text about the configuration layout. The bottom screenshot shows a 'PHP Version 5.1.2-Ubuntu2.8' page, which is a standard PHP information page displaying various PHP configuration details like 'System', 'Build Date', and 'Loaded Configuration'.

Apache2 Default Page

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this host is working properly. You should [replace this file](#) (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is [fully documented in /usr/share/doc/apache2/README.Debian.gz](#). Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the [manual](#) if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

PHP Version 5.1.2-Ubuntu2.8

System	Linux ip-172-31-148-9 5.15.0-153-generic #23-Ubuntu SMP Wed Aug 17 10:59:11 UTC 2022 x86_64
Build Date	Mon Jul 26 17:08:49
Build System	LTS
Server API	Apache 2.0 Handler
Virtual Directory Support	Enabled
Configuration File Interpreter Path	/etc/php/5.1/cgi/php5
Loaded Configuration File	/etc/php/5.1/cgi/php5
SCM File for Interpreted File	/etc/php/5.1/cgi/php5
Additional .ini Files Loaded	/etc/php/5.1/cgi/conf.d/ctype.ini, /etc/php/5.1/cgi/conf.d/iconv.ini, /etc/php/5.1/cgi/conf.d/intl.ini, /etc/php/5.1/cgi/conf.d/json.ini, /etc/php/5.1/cgi/conf.d/mysqli.ini, /etc/php/5.1/cgi/conf.d/mysqlnd.ini, /etc/php/5.1/cgi/conf.d/openssl.ini, /etc/php/5.1/cgi/conf.d/pdo_dblib.ini, /etc/php/5.1/cgi/conf.d/pdo_firebird.ini, /etc/php/5.1/cgi/conf.d/pdo_mysql.ini, /etc/php/5.1/cgi/conf.d/pdo_oci.ini, /etc/php/5.1/cgi/conf.d/pdo_odbc.ini, /etc/php/5.1/cgi/conf.d/pdo_pgsql.ini, /etc/php/5.1/cgi/conf.d/pdo_sqlite.ini, /etc/php/5.1/cgi/conf.d/pdo_sqlserver.ini, /etc/php/5.1/cgi/conf.d/pdo_udb.ini, /etc/php/5.1/cgi/conf.d/readline.ini, /etc/php/5.1/cgi/conf.d/sqlite3.ini, /etc/php/5.1/cgi/conf.d/tokenizer.ini, /etc/php/5.1/cgi/conf.d/wddx.ini



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



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

Create Image

Amazon Machine Image (AMI) is a pre-configured Amazon Machine Image (AMI) that you can launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID: **aws-madhav7521-AWS-18-Market**

Image name: **AWS-Madhav-APACHE-PHP**

Amazon EBS Optimized: Optimize for sequential read operations.

Image description: **AWS-Madhav-APACHE-PHP**

Volume 223 (Amazon EBS)

Not selected: Create

Attached volumes:

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
Amazon EBS	/dev/sda1	Create new snapshot	8	Amazon EBS Provisioned IOPS (SSD)	100	1000	<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

Add volume

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional

It is included free with your usage of AWS Lambda. Each tag consists of a key and a value. You can use tags to identify and filter your resources across your AWS account.

Try image and snapshots together
Test the image and the snapshots with the instance.

Try image and snapshots separately
Test the image and the snapshots with different tags.

No tags associated with this resource.

Add new tag

Tag key: **aws-madhav7521-AWS-18-Market**

Create **Cancel** **Change tags**

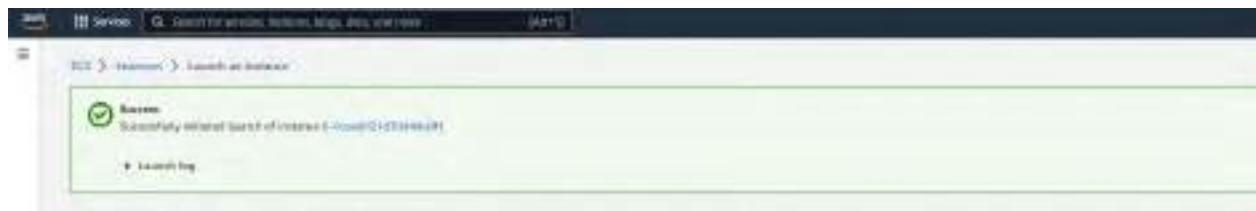
6. Go to AMIs and check the image created with the name (AWS-Madhav-APACHE-PHP)



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



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



The screenshot shows the AWS EC2 'Launch an instance' wizard. The current step is 'Application and OS Images (Amazon Machine Image)'. The page includes fields for 'Name and tags', 'Software stack (AMI)', and 'Virtual private subnet'. A search bar at the top right says 'Search for AMIs, features, blogs, docs, and more...'. On the left, there's a sidebar with 'EC2 > Instances > Launch an instance'.

Name and tags

Name: `dev-lab-test-instance-2022-11-01` [Add additional tags](#)

Application and OS Images (Amazon Machine Image)

An AMI is a temporary file containing the software configuration (operating system, application source, and application) required to launch your instance. Search or browse for AMIs if you don't see what you're looking for below.

Q: Search our AMI catalog including 700k+ of application and OS images

[Get from catalog](#) [Reports](#) [My AMIs](#) [Quick start](#)

Amazon Marketplace Image (AMI)

AWS-Maraju-ARACHE-PHP
ami-01167c5bc724ed7c

Published	Architecture	Virtualization	Root device type	AMI created
2022-11-01 16:45:00	x86_64	hvm	gp2	2022-11-01

[Browse more AMIs including 700k+ from AWS Marketplace and the Community](#)

Summary

Number of instances: New

Software stack (AMI): **AWS-Maraju-ARACHE-PHP** ami-01167c5bc724ed7c

Virtual private subnet: **dev-lab-test-subnet**

Security group: **New security group**

Storage (volumes): **1 volume(s) - 5 GiB**

Free tier usage for your instance: 700 hours of t2.micro for 12 months in the Region in which instance is available

Estimated usage on monthly AMI per month: 30 GB of EBS storage, 2 million IOPS, 1 million requests, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#)

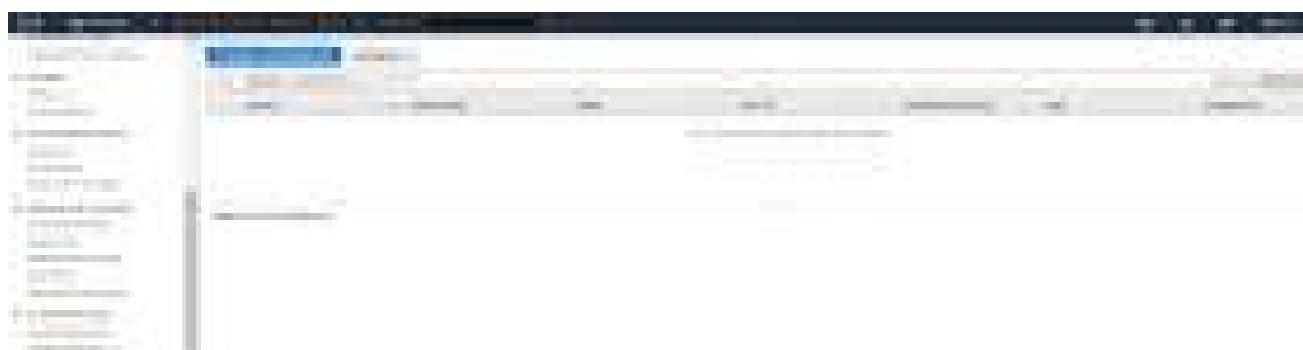
Successfully able to connect to the created instance

```
aws services Q Search for services, features, blogs, docs, and more [Alt+S]
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

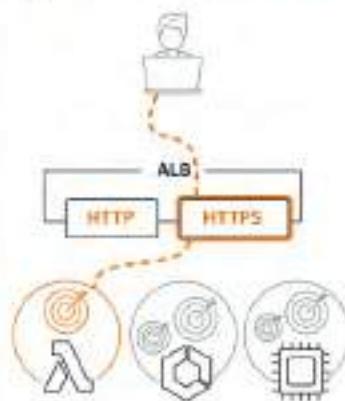


4. Click on Create Load Balancer, select load balancer type as Application Load Balancer

A complete feature-by-feature comparison along with detailed highlights is also available. Learn more [\[Learn more\]](#)

Load balancer types

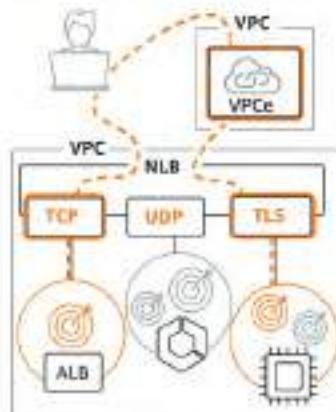
Application Load Balancer [\[Info\]](#)



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](#)

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, AWS Lambda functions, 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

Name must be unique within your AWS account and cannot be changed after the load balancer is created.

AWS-LB

A maximum of 50 alphanumeric characters including hyphens are allowed, but the maximum must begin or end with a letter.

Scheme Info

Scheme cannot be changed after this load balancer is created.

Internet-facing

An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more 

Internal

An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type Info

Select the type of IP address that your subnets use.

IPv4

Recommended for internet load balances.

Dualstack

Includes IPv4 and IPv6 addresses.

Network mapping [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in association with your IP address settings.

VPC [Info](#)

Select the virtual private cloud (VPC) for your targets; only VPCs with an internet gateway are available for selection. The selected VPC cannot be changed after the load balancer is created. To change the VPC for your targets, view your target groups [\[2\]](#).

ip-162-237-150-45.us-east-1.compute.internal IPv4: 172.31.0.0/16	Edit
---	----------------------

Mappings [Info](#)

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that aren't supported by the load balancer or the VPC are not available for selection.

us-east-2a

Subnet: [subnet-045eb04f1f4b0d221](#)

IPv4 settings: Assigned by AWS

us-east-2b

Subnet: [subnet-072c10995eb1d4ad](#)

IPv4 settings: Assigned by AWS

us-east-2c

Subnet: [subnet-09540a7ae00709ef1](#)

IPv4 settings: Assigned by AWS

Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer.

Security groups

Select up to 5 security groups [\[1\]](#)

Create new security group [\[2\]](#)

default sg-0231a52960611x03d X VPC vpc-05d2a7b1d15451433	search-wizard-11 sg-01037763930dc34f5 X VPC vpc-05d2a7b1d15451433
---	--

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	Forward to Select a target group	1-45555 Create target group Delete

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 resource tags.

Add Listener

ECS > Target groups > Create target group

Step 1: Specify group details

After load balancer routes requests to the targets in a target group and performs health checks on the targets.

Step 2: Register 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 non-VPC resources.
- Facilitates scaling to multiple IP addresses and network interfaces as the auto scaling.
- Offers flexibility with microservice-based architectures, simplifying cross-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-Targetgroup

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 resources that you want to include in the target group:

ap-northeast-1a:54.126.23.144:80 172.31.0.2/16

▼ 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 check successes required before considering an unhealthy target healthy.

2-10

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

2-10

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

seconds
2-120

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

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").

Request sample:

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: curl/7.29.0
Accept: */*
```

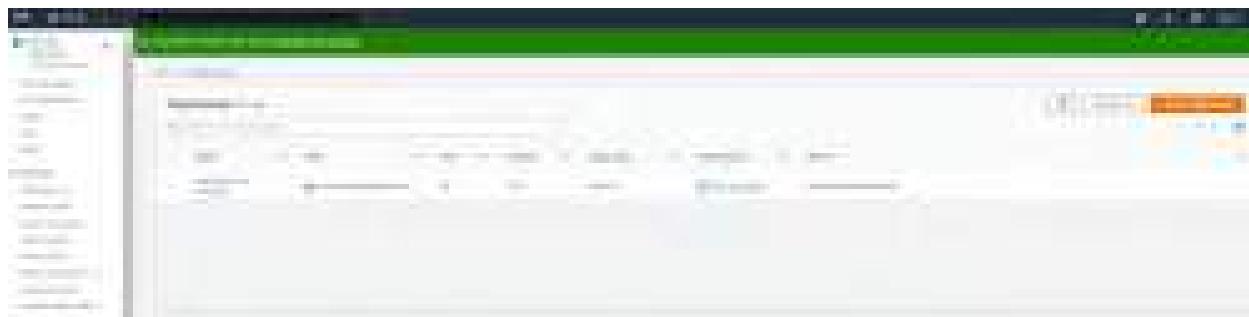
Response sample:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 12345
Connection: keep-alive
Date: Mon, 29 Jul 2019 14:23:37 GMT

<!DOCTYPE html>
<html>
<head>
</head>
<body>
</body>
</html>
```

Headers:

Name	Type	Value
Content-Type	Text	text/html
Content-Length	Text	12345
Connection	Text	keep-alive
Date	Text	Mon, 29 Jul 2019 14:23:37 GMT



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

Protocol: **HTTP** Port: **80** Default action: **Forward to Application-LB-Targetgrp**

Target type: instances (IPv4) **HTTP** [Edit](#)

[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](#)

You successfully created load balancer - APNSL8

Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

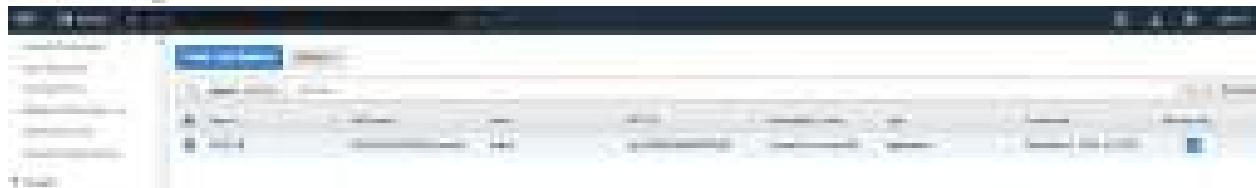
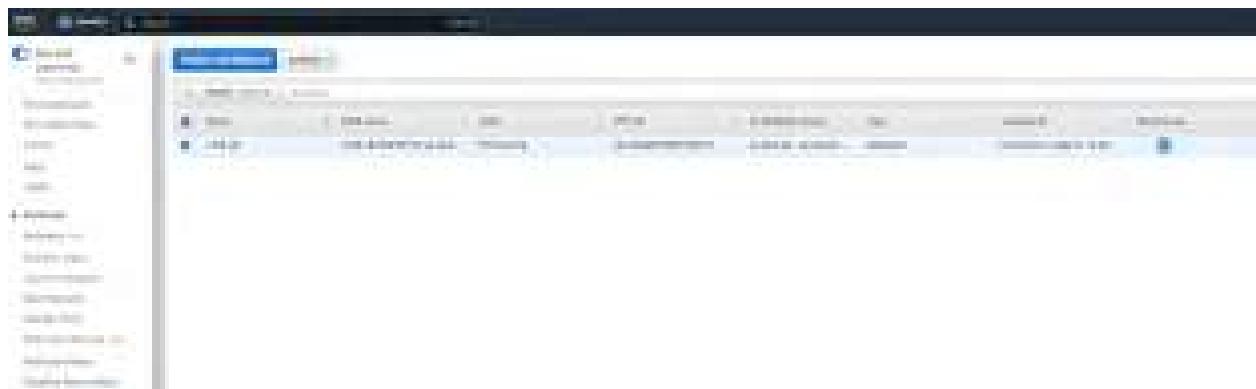
[View](#) [Load Balancers](#) [Create Application Load Balancer](#)

Create Application Load Balancer

Suggested next steps

- Review, customize, or create attributes for your load balancer and listeners using the Description of attributes table in the AWS LB documentation.
- Discover other services that you can integrate with your load balancer. See the integrated services page for more details.

[Create 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

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create: [Info](#)
Create only the VPC resources in the VPC and other networking resources.

VPC only

VPC and more

Name tag - optional:
Create a tag with a key of Name and a value that you specify.

My-VPC-Madhav

IPv4 CIDR block: [Info](#)
 IPv4 CIDR manual input
 IPAM-allocated IPv4 CIDR block

IPv4 CIDR: 192.168.0.0/16

IPv6 CIDR block: [Info](#)
 No IPv6 CIDR block
 IPAM-allocated IPv6 CIDR block
 Amazon-provided IPv6 CIDR block
 IPv6 CIDR owned by me

Tenancy: [Info](#)
Default

Tags:
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an associated value. You can use tags to identify and filter your resources to track your AWS costs.

Key	Value - optional
<input type="text"/> Name	<input type="text"/> My-VPC-Madhav

[Add new tag](#)

You can add 10 more tags.

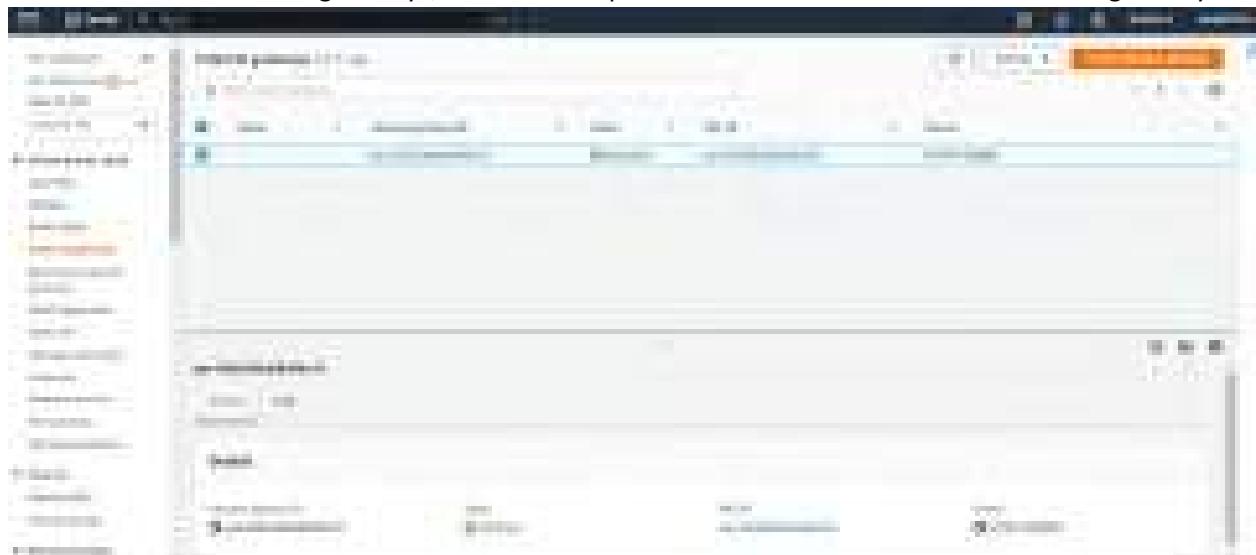
[Cancel](#) [Create VPC](#)



- 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

Services Search [EMR+S]

VPC > Internet gateways > Create internet gateway

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Create a tag with a key of "Name" and a value that you specify.

My-Malhotra-InternetGateway

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
Q Name	Q My-Malhotra-InternetGateway X Remove

Add new tag

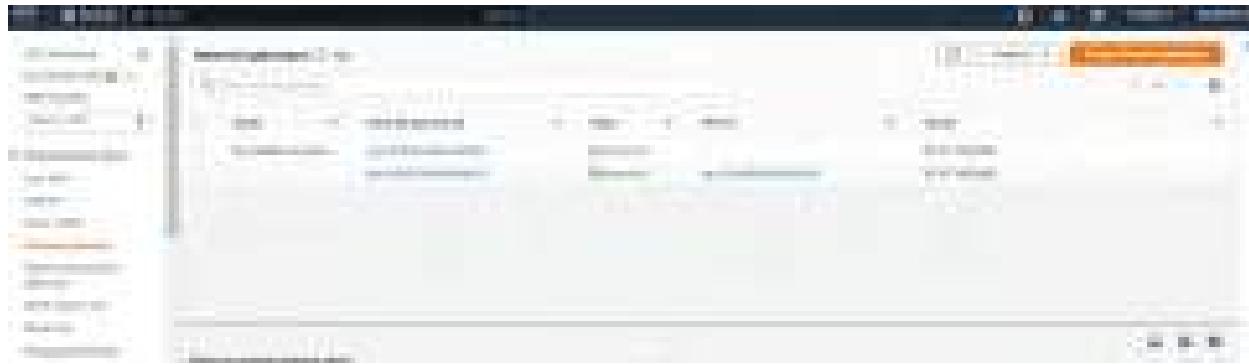
You can add 49 more tags.

Cancel **Create internet gateway**

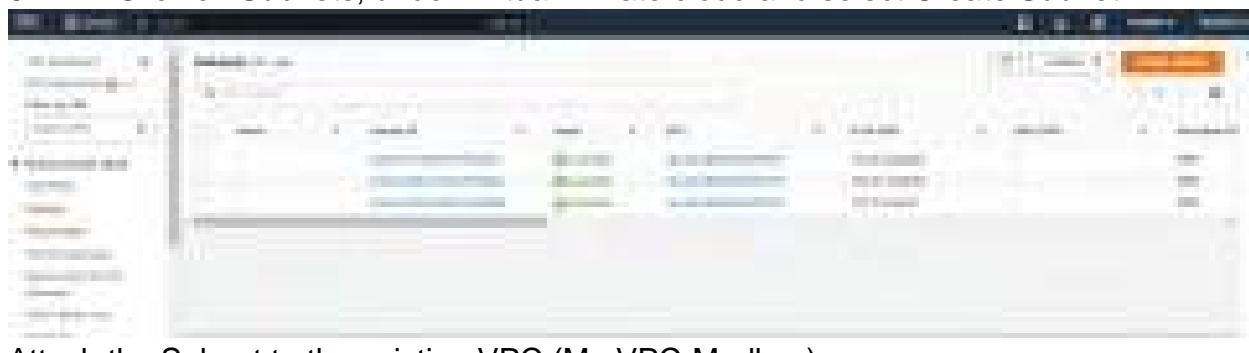
Internet gateway is created

Open OFF VPC Overview (20008) / My-Malhotra-InternetGateway

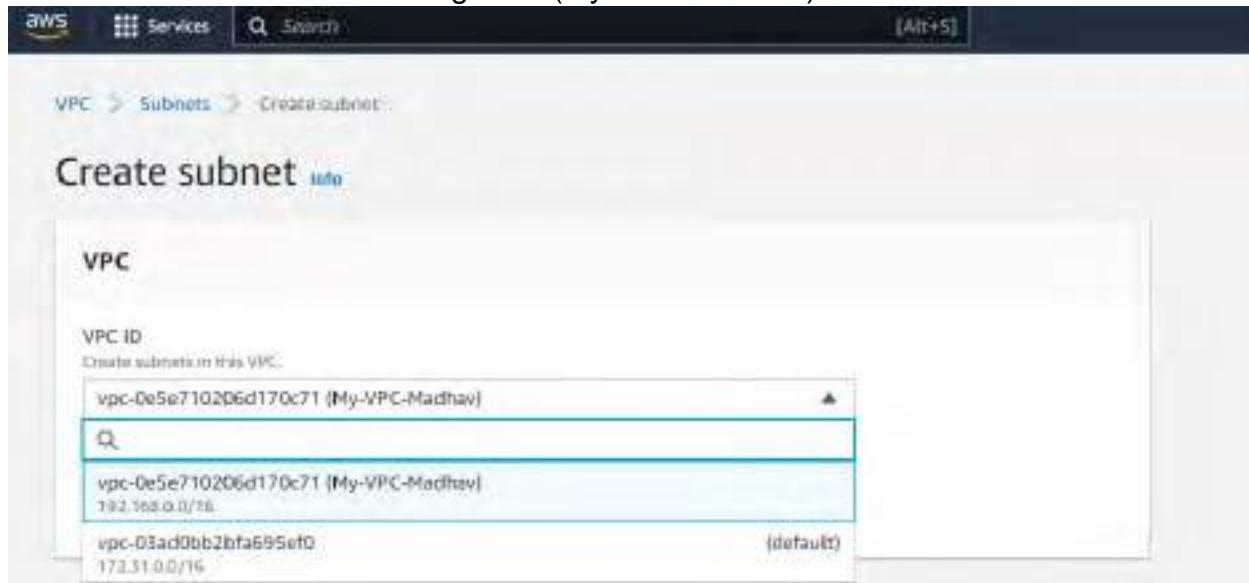
State	Region	Internet gateway ID	Tags
Active	us-east-1	i-gw-00000000000000000000	My-Malhotra-InternetGateway



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

Subnet settings

Specify the CIDR blocks and Availability Zones for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify:

The name can be up to 256 characters long.

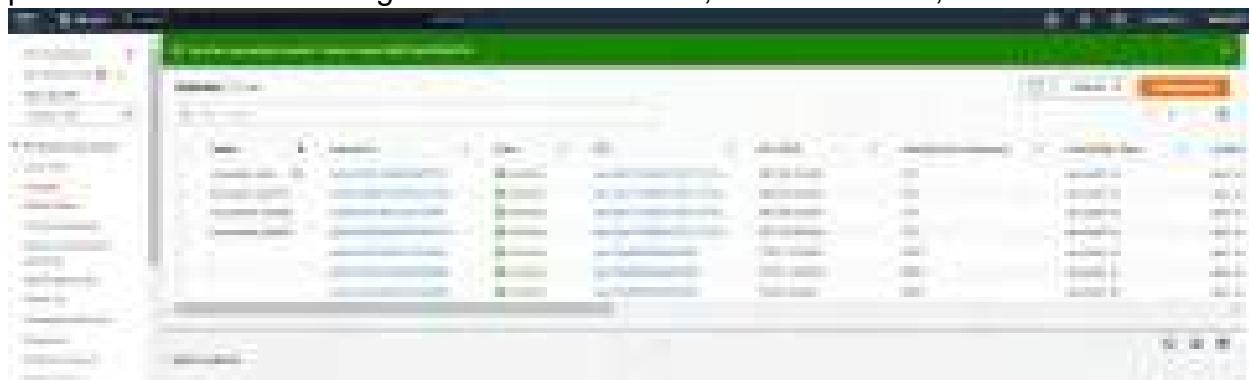
Availability Zone [Info](#)
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 CIDR block [Info](#)

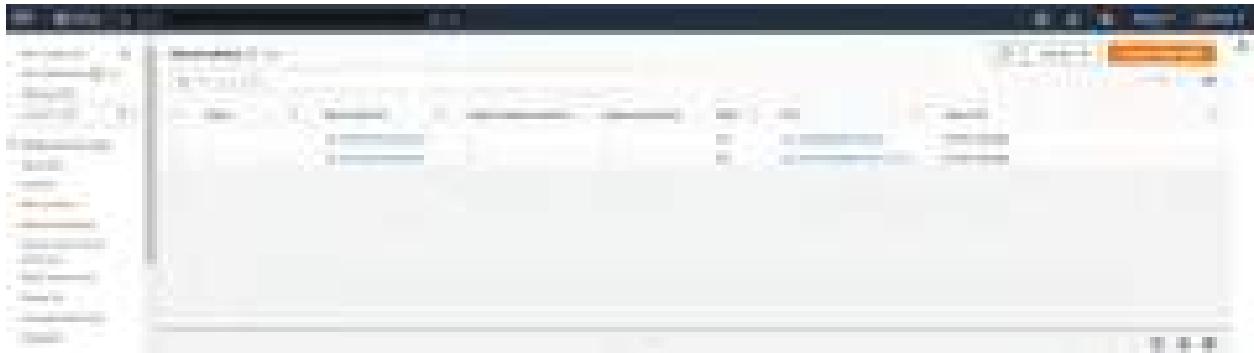
Key **Value - optional**

You can add 49 more tags:

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



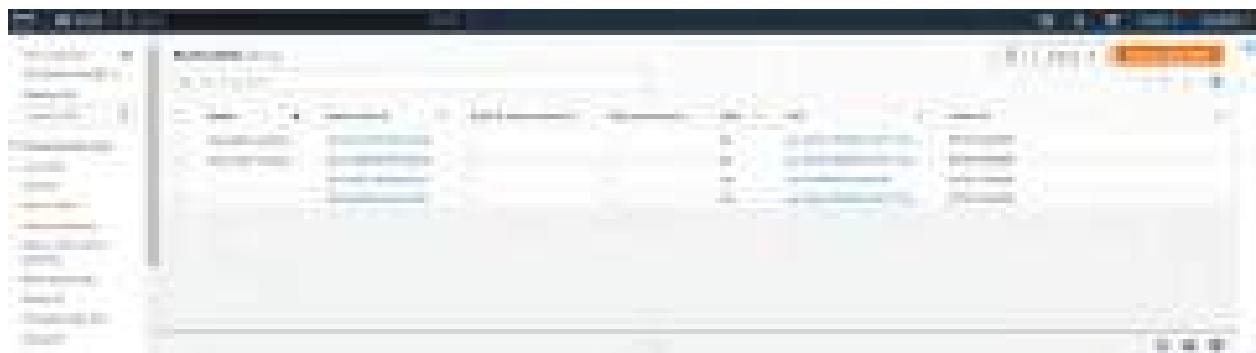
13. Create my-public-routetable-1 and attach it to the existing VPC



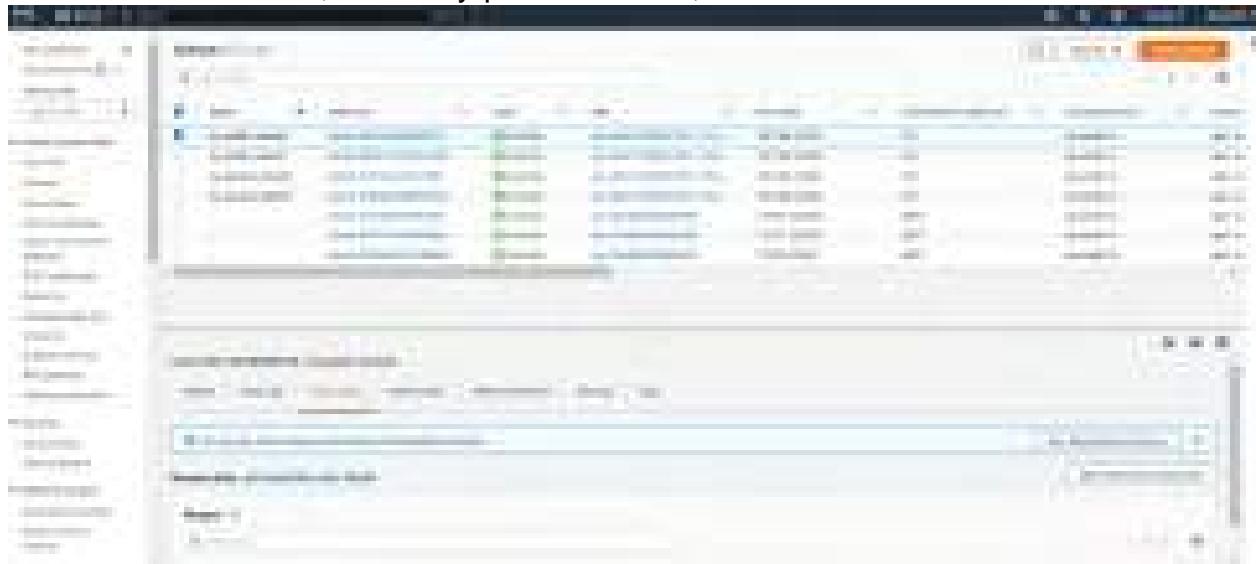
14. Create my-private-routetable-1 and attach it to the existing VPC



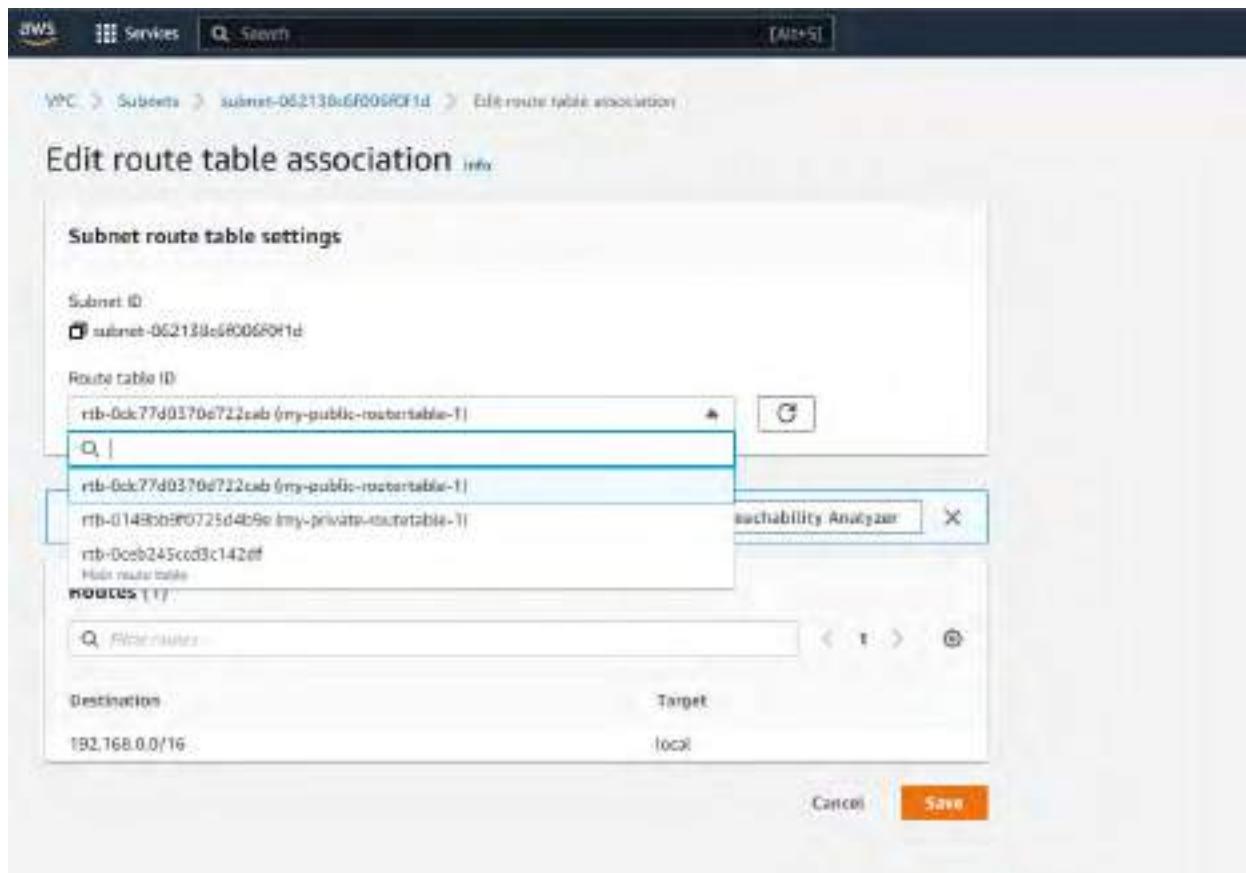
2 route tables – one private and one public route table is created



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



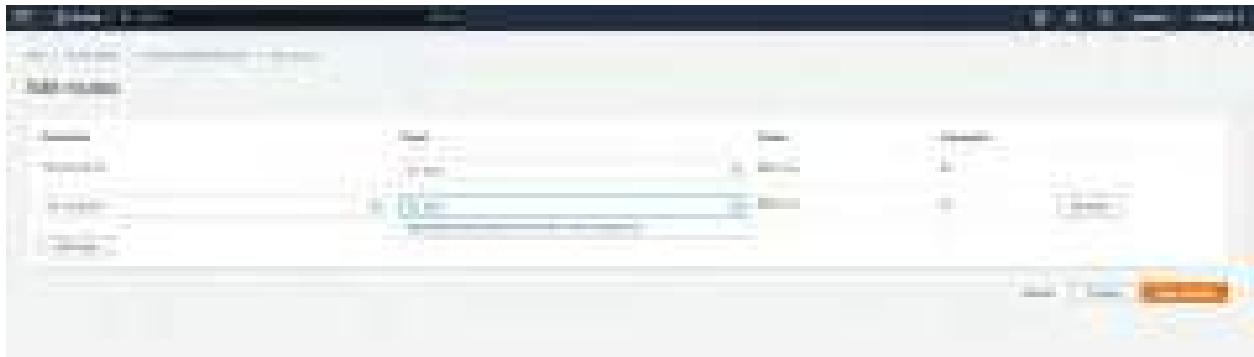
17. Select my-public-routetable1 from the drop down and click on save



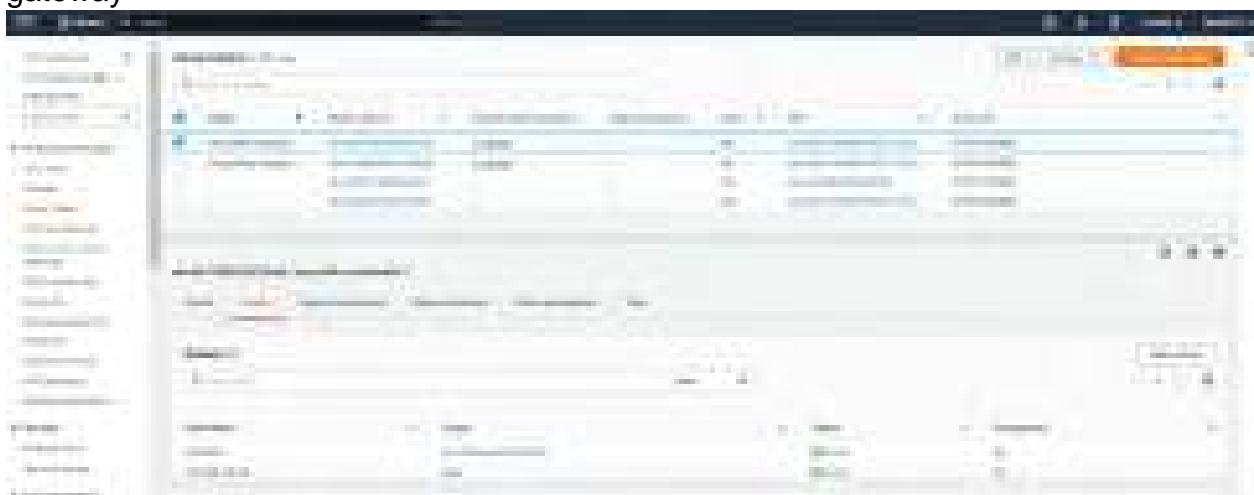
18. Similarly perform the same activity for the remaining association and you can see the below screenshot with all the associations



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)

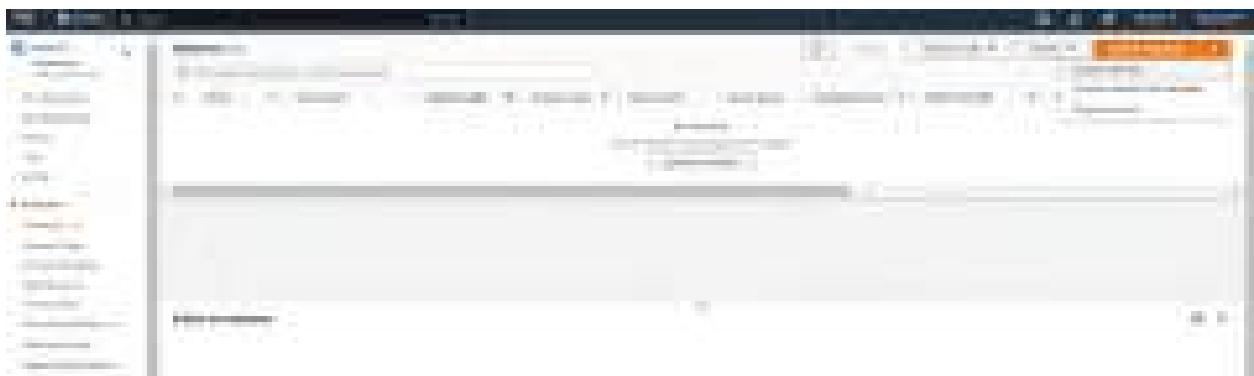


21. Click on Save changes, now you will see the route is associated with Internet gateway



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

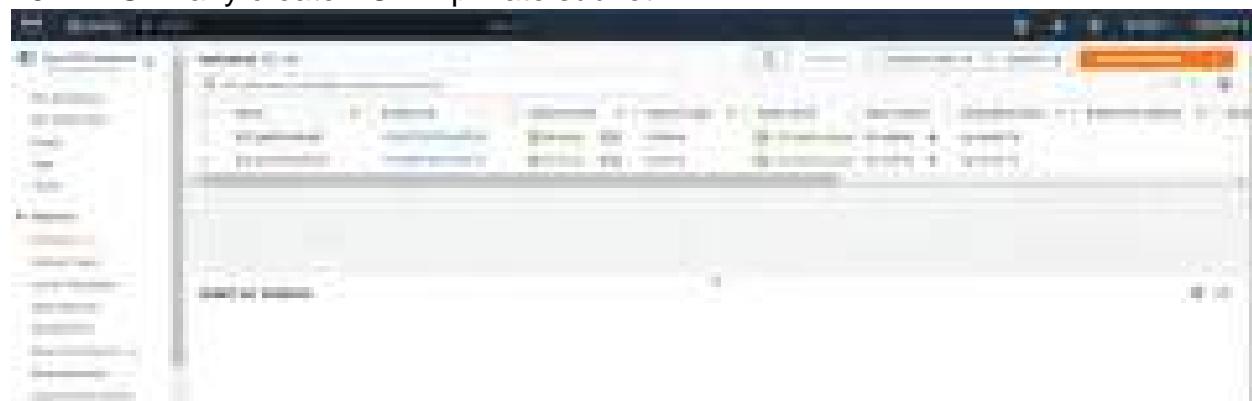


24. During the EC2 instance creation, change the settings under Network Settings
- VPC → My-VPC-Madhav
 - Subnet as → my-public-subnet1(192.168.2.0/24)
 - Create a security Group as → Public-Subnet-Secgroup1 with SSH and anywhere is enabled
 - Click on Launch Instance

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

```
C:\> ssh -i "AWSNADEHW.pem" ec2-user@3.144.192.243
Microsoft Windows [Version 10.0.22000.1281]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ssh -i "AWSNADEHW.pem" ec2-user@3.144.192.243
Warning: Identity file AWSNADEHW.pem not accessible: No such file or directory.

C:\WINDOWS\system32>cd\

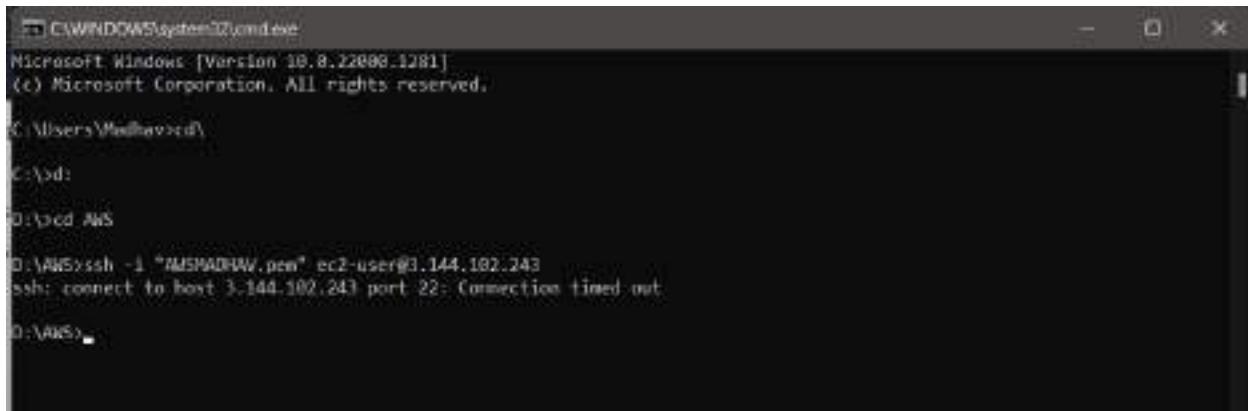
C:\>cd AWS

D:\AWS>ssh -i "AWSNADEHW.pem" ec2-user@3.144.192.243
ssh: connect to host 3.144.192.243 port 22: Connection timed out

D:\AWS>ssh -i "AWSNADEHW.pem" ec2-user@3.142.132.107
The authenticity of host '3.142.132.107 (3.142.132.107)' can't be established.
ECDSA key fingerprint is SHA256:+wAkVm3T17DV6FTdqn0Specz0PE/upX-P009TAISW1.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.142.132.107' (ECDSA) to the list of known hosts.

[ec2-user@ip-192-168-2-37 ~]$
```

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



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22800.1281]
(c) Microsoft Corporation. All rights reserved.

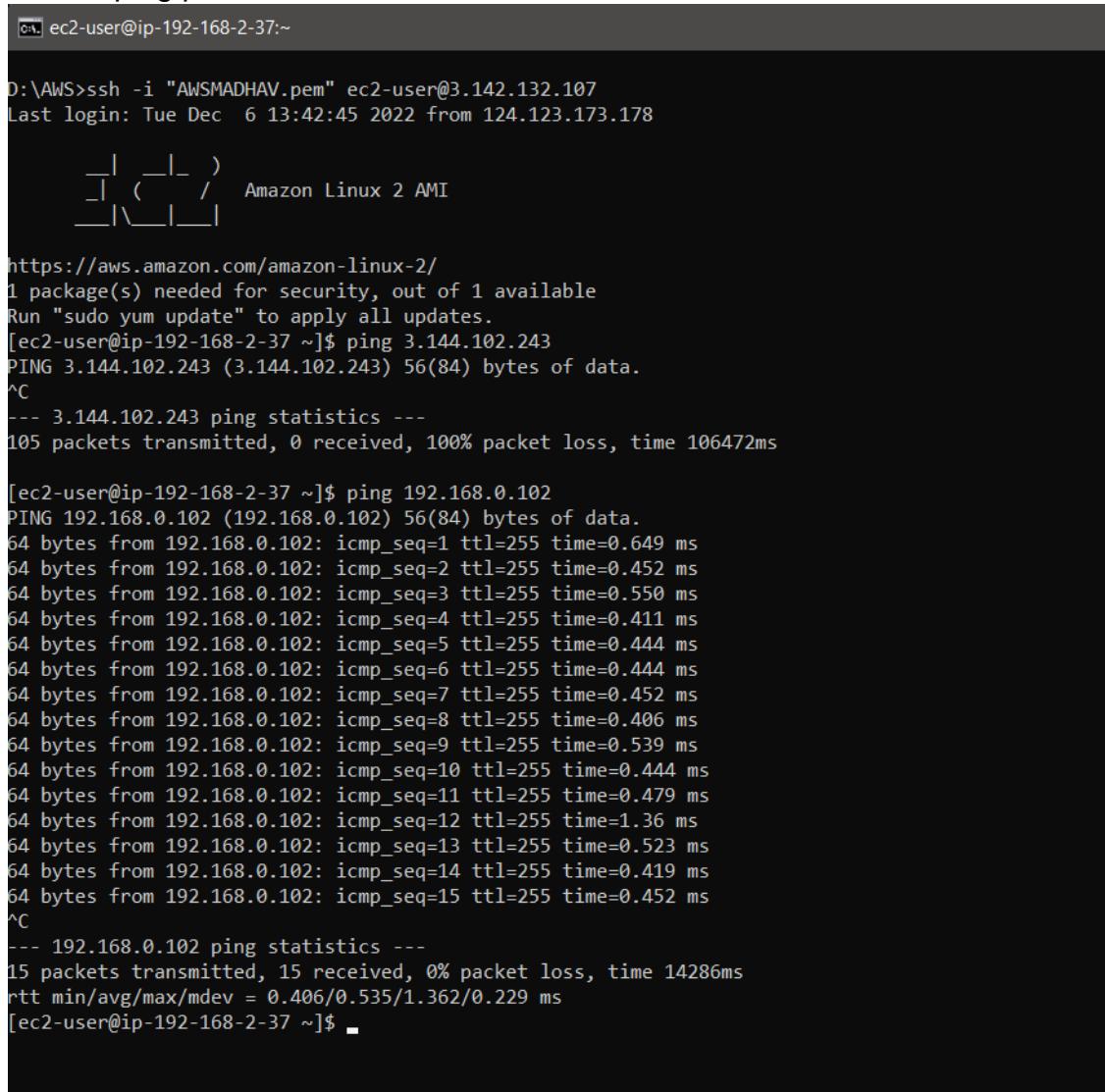
C:\Users\Madhav>cd

C:\>cd AWS

D:\AWS>ssh -i "AWSMADHAV.pem" ec2-user@3.144.102.243
ssh: connect to host 3.144.102.243 port 22: Connection timed out

D:\AWS>
```

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



```
[ec2-user@ip-192-168-2-37:~]

D:\AWS>ssh -i "AWSMADHAV.pem" ec2-user@3.142.132.107
Last login: Tue Dec  6 13:42:45 2022 from 124.123.173.178

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

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-192-168-2-37 ~]$ ping 3.144.102.243
PING 3.144.102.243 (3.144.102.243) 56(84) bytes of data.
^C
--- 3.144.102.243 ping statistics ---
105 packets transmitted, 0 received, 100% packet loss, time 106472ms

[ec2-user@ip-192-168-2-37 ~]$ ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data.
64 bytes from 192.168.0.102: icmp_seq=1 ttl=255 time=0.649 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=255 time=0.452 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=255 time=0.550 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=255 time=0.411 ms
64 bytes from 192.168.0.102: icmp_seq=5 ttl=255 time=0.444 ms
64 bytes from 192.168.0.102: icmp_seq=6 ttl=255 time=0.444 ms
64 bytes from 192.168.0.102: icmp_seq=7 ttl=255 time=0.452 ms
64 bytes from 192.168.0.102: icmp_seq=8 ttl=255 time=0.406 ms
64 bytes from 192.168.0.102: icmp_seq=9 ttl=255 time=0.539 ms
64 bytes from 192.168.0.102: icmp_seq=10 ttl=255 time=0.444 ms
64 bytes from 192.168.0.102: icmp_seq=11 ttl=255 time=0.479 ms
64 bytes from 192.168.0.102: icmp_seq=12 ttl=255 time=1.36 ms
64 bytes from 192.168.0.102: icmp_seq=13 ttl=255 time=0.523 ms
64 bytes from 192.168.0.102: icmp_seq=14 ttl=255 time=0.419 ms
64 bytes from 192.168.0.102: icmp_seq=15 ttl=255 time=0.452 ms
^C
--- 192.168.0.102 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14286ms
rtt min/avg/max/mdev = 0.406/0.535/1.362/0.229 ms
[ec2-user@ip-192-168-2-37 ~]$
```

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

The screenshot shows three terminal windows:

- PublicSubnet-EC2:** Shows a ping test: "3 packets transmitted, 3 received, 0% packet loss, time 2052ms".
- PrivateSubnet-EC2:** Shows an SSH session attempt to a public IP: "ssh -i "AWSMADHAV.pem" ec2-user@144.182.243". It asks for confirmation ("Are you sure you want to continue connecting (yes/no)?") and adds the host to the known hosts.
- Command Prompt:** Shows the same SSH command again, followed by a warning about an unprotected private key file ("WARNING: UNPROTECTED PRIVATE KEY FILE!"), a permission denied message ("Permission denied (publickey,gssapi-keyex,gssapi-with-mic)."), and a cat command output showing the Amazon Linux 2 AMI banner.

At the bottom, the URL <https://aws.amazon.com/amazon-linux-2/> is pasted into the Command Prompt window.

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-gateway 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

The screenshot shows the 'Create NAT gateway' wizard in the AWS VPC service. The top navigation bar includes 'Services' and a search bar. The main title is 'Create NAT gateway'. A descriptive text explains that it's a highly available, managed Network Address Translation (NAT) service. The 'NAT gateway settings' section contains fields for 'Name - optional' (set to 'my-nat-gateway'), 'Subnet' (selected to 'my-private-subnet1'), and 'Allocate Elastic IP' (unchecked). Below the subnet dropdown is a note about using tags to search and filter resources. At the bottom, there's a table for adding tags with a single entry: 'Name' (key) and 'my-nat-gateway' (value).

VPC > NAT gateways > Create NAT gateway

Create NAT gateway [info](#)

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the Internet.

NAT gateway settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

my-nat-gateway

The name can be up to 255 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

Select a subnet [▲](#)

Q

subnet-0595a2e9b3d40df51 (my-private-subnet1)
us-east-2a

subnet-0fbce2e1e30a5cd8a (my-public-subnet2)
us-east-2a

subnet-077c10995e81daaad
us-east-2b

subnet-06d43a2d5f6ef7667 (my-private-subnet2)
us-east-2a

subnet-045eb04fd14b0d221
us-east-2a

subnet-09540a7ae08709ef1
us-east-2c

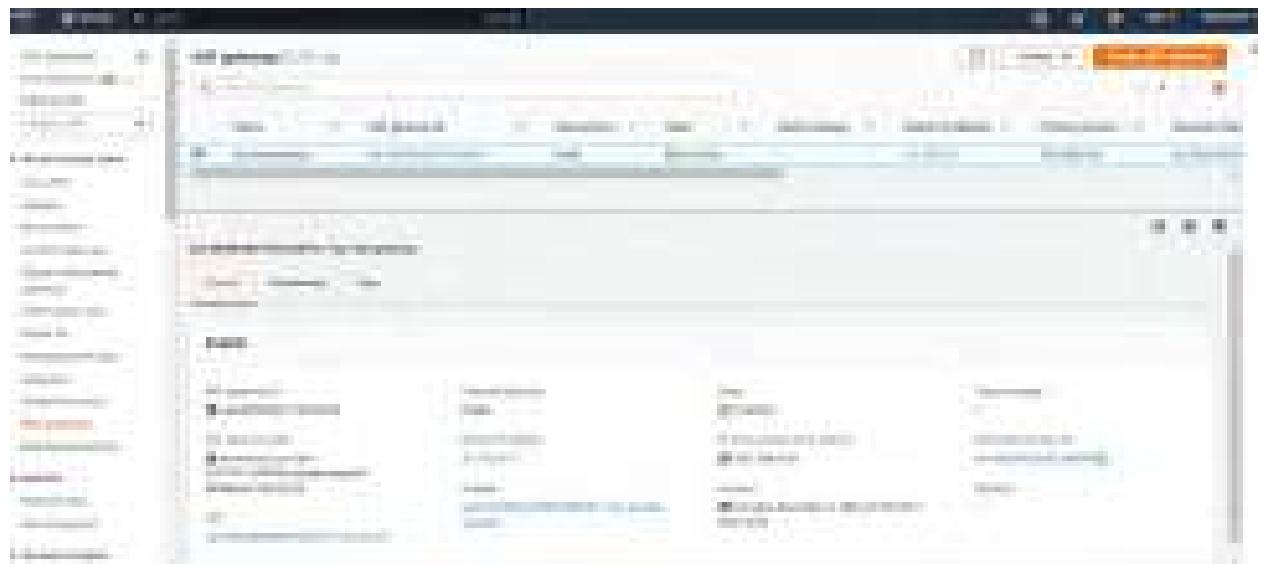
subnet-0afc79f8a75c438e5 (my-public-subnet1)
us-east-2a

Allocate Elastic IP

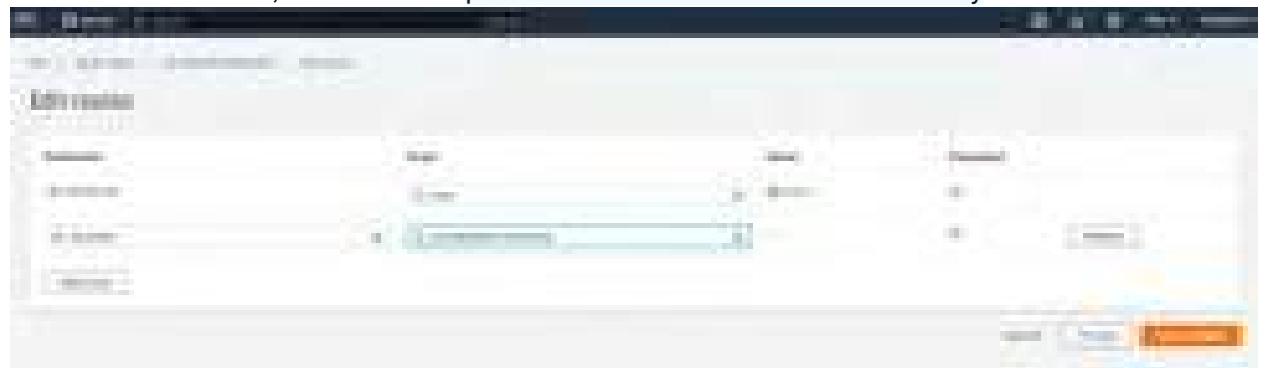
can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
Q Name	X my-nat-gateway Remove

33. NAT Gateway is created



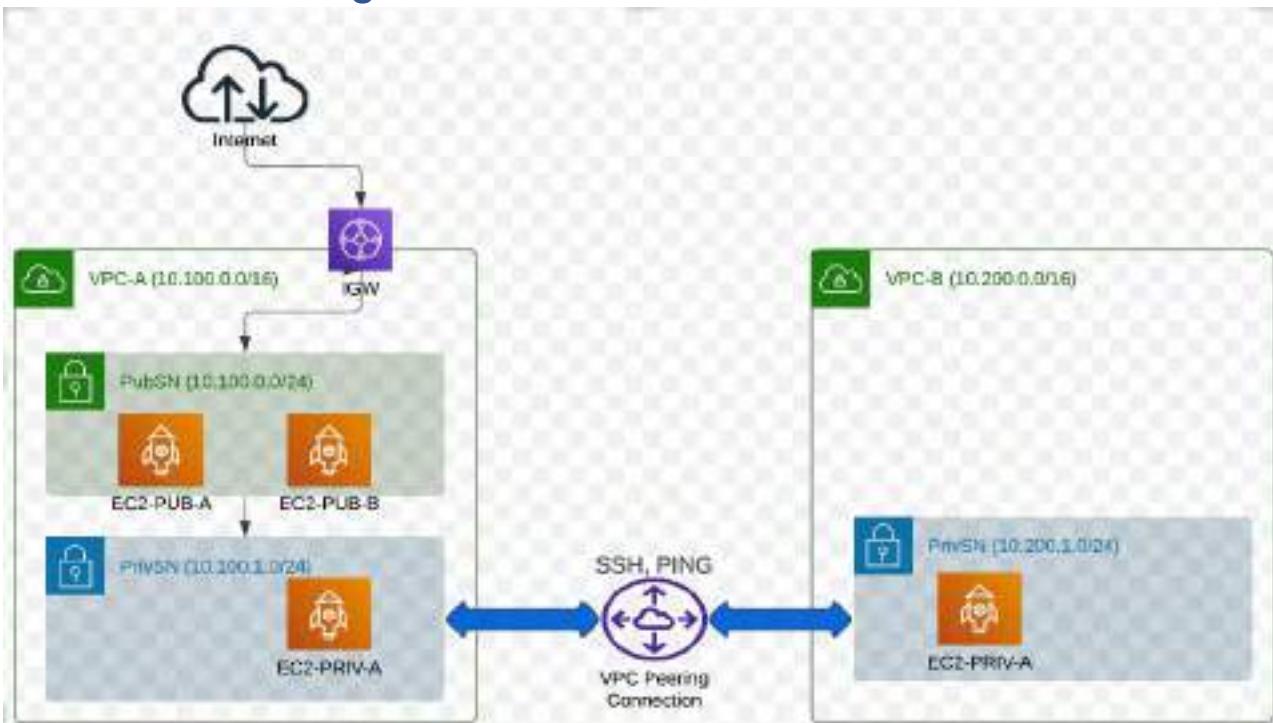
34. Go to Route Table, and edit the private-routetable1 and add NAT Gateway



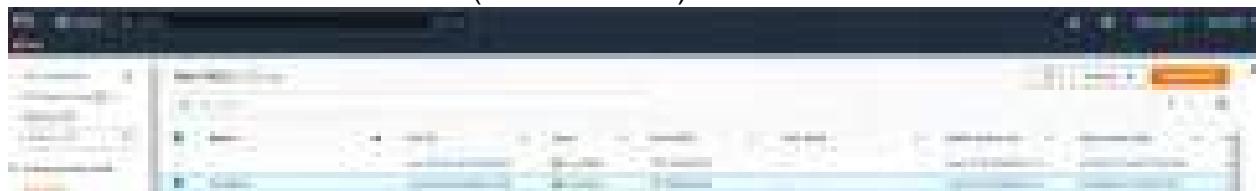
35. Connect private subnet EC2 machine from the Public Subnet EC2 machine and then ping google.com which should work now

```
[ec2-user@ip-192-168-0-102 ~]$ ping google.com
PING google.com (172.217.4.46) 56(84) bytes of data.
64 bytes from ord38s18-in-f14.1e100.net (172.217.4.46): icmp_seq=1 ttl=104 time=21.4 ms
64 bytes from ord38s18-in-f14.1e100.net (172.217.4.46): icmp_seq=2 ttl=104 time=18.0 ms
64 bytes from ord38s18-in-f14.1e100.net (172.217.4.46): icmp_seq=3 ttl=104 time=18.0 ms
```

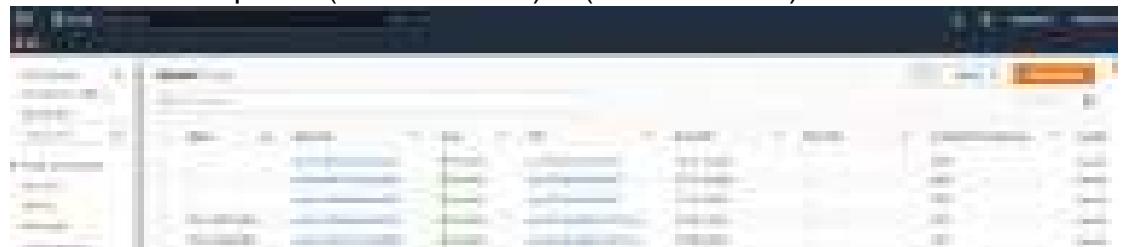
7. VPC Peering



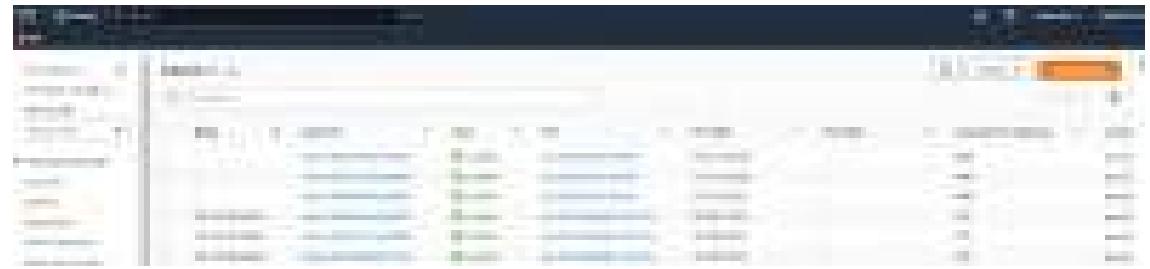
1. To create a VPC peering connection with VPCs in the same account and Region
2. Created "VPC A" with 3 subnets (10.100.0.0/16)



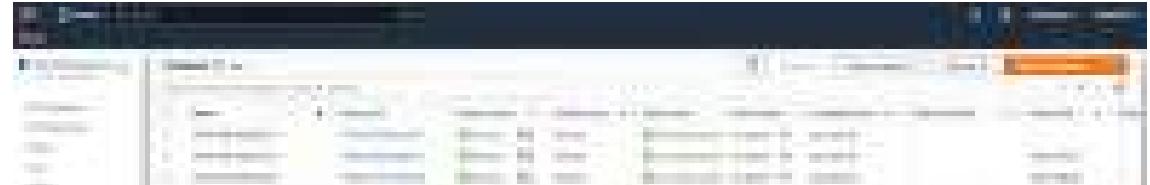
- a. Two subnets in public (10.100.0.0/24) & (10.100.2.0/24)



- b. One subnet in Private (10.100.1.0/24)



- c. Create a EC2 machine in each subnet

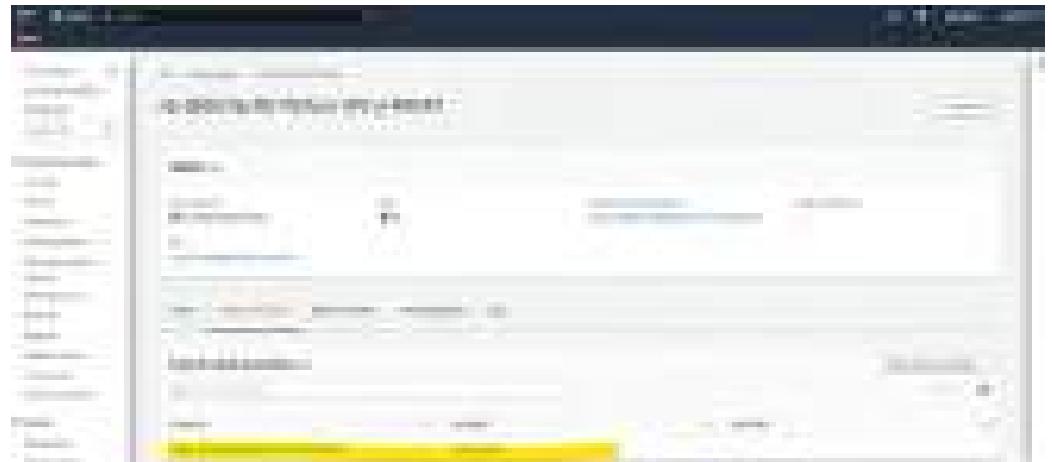


- d. Create two route tables VPC-A-PRIVRT and VPC-A-PUBRT

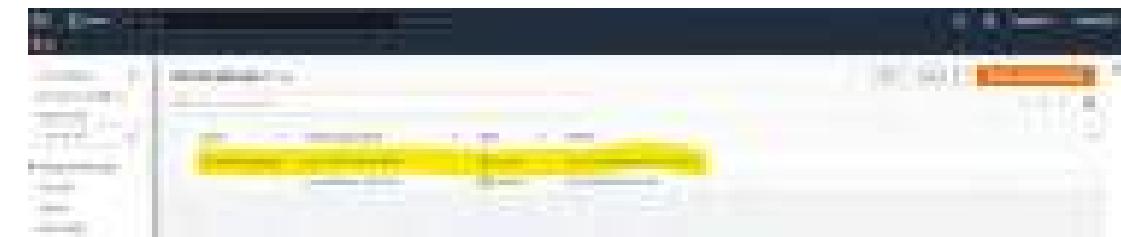
- Connect 2 Public EC2 machines to Public route table → VPC-A-PUBRT



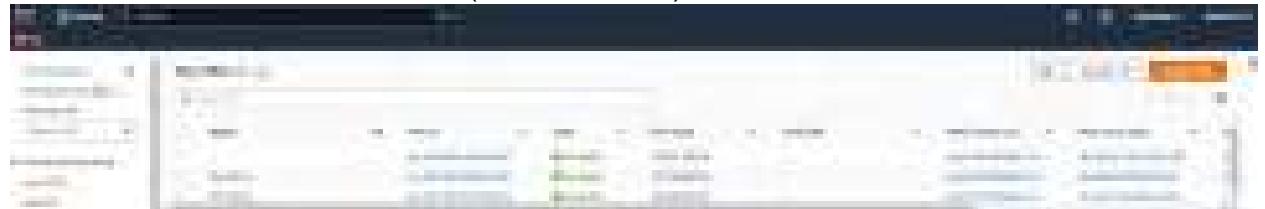
- Connect 1 Private EC2 machine to Private route table → VPC-A-PRIVRT



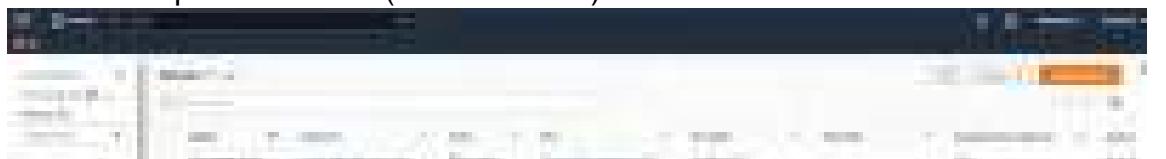
- e. Attach Internet Gateway for VPC-A



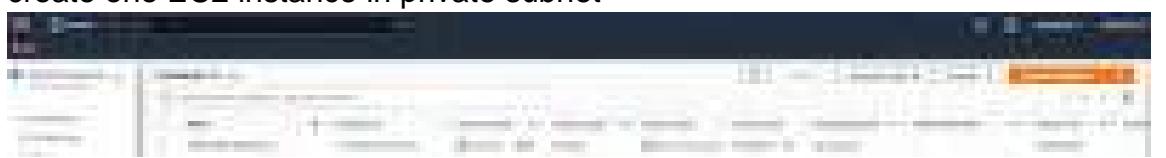
3. Create a "VPC B" with 1 subnet (10.200.0.0/16)



- a. Create one private subnet (10.200.1.0/24)



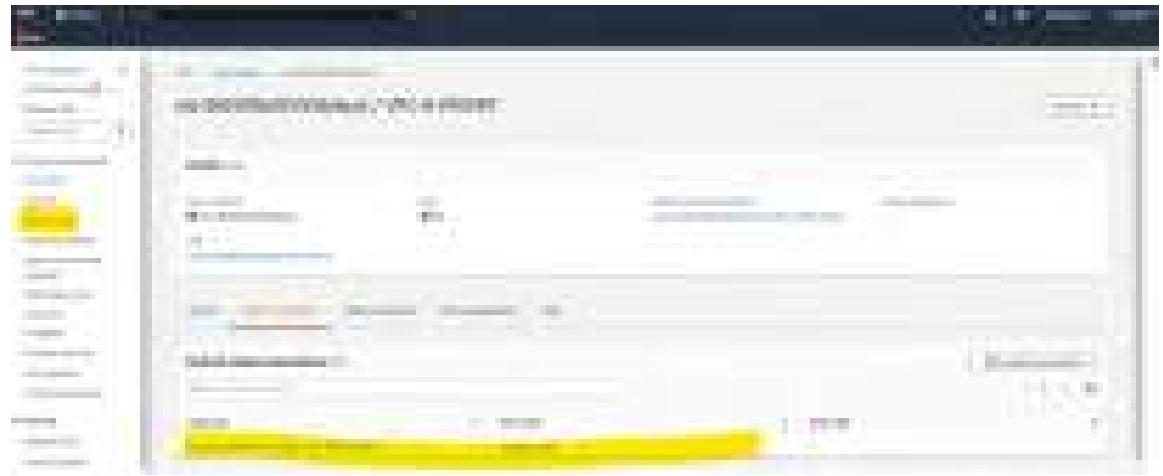
- b. create one EC2 instance in private subnet



- c. Create one route table VPC-B-PRIVRT

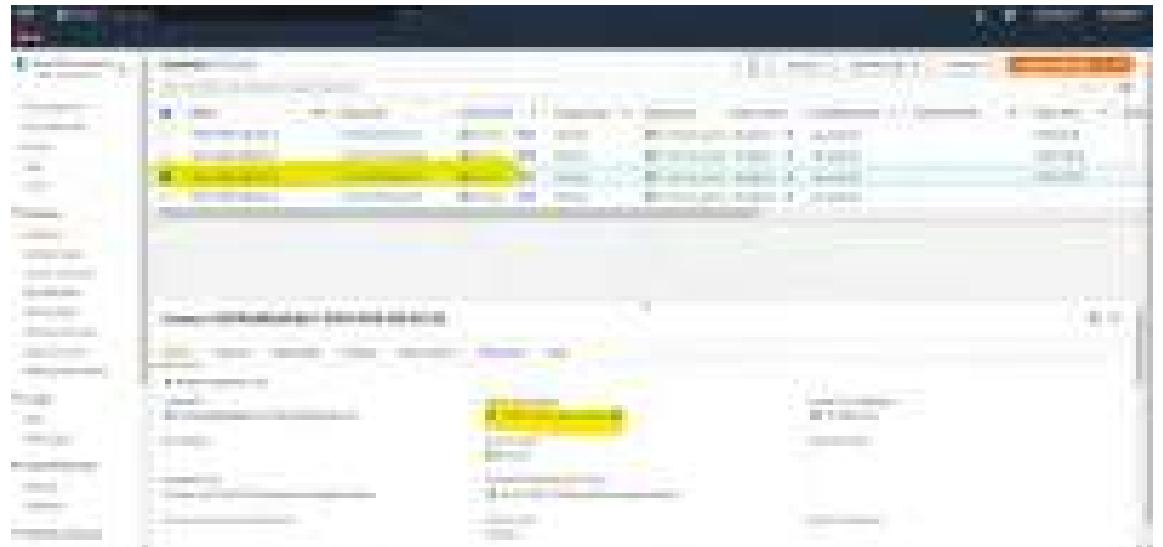


- d. Connect the EC2 machine to Route table



4. Check connectivity

- Connect to VPCA-PUB-SUB-EC2-A machine



Successfully established connection

```
ec2-user@ip-10-200-1-85:~$  
  
macion@MINGW64 ~ /OneDrive/Desktop/AWS  
$ ssh -i "AWS-Hyderabad.pem" ec2-user@18.69.157.83  
Last login: Sun Jan  8 14:03:09 2023 from 49.206.46.11  
  
[ec2-user@ip-10-100-0-122 ~]$  
[ec2-user@ip-10-100-0-122 ~]$  
[ec2-user@ip-10-100-0-122 ~]$  
[ec2-user@ip-10-100-0-122 ~]$
```

- 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.pem 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

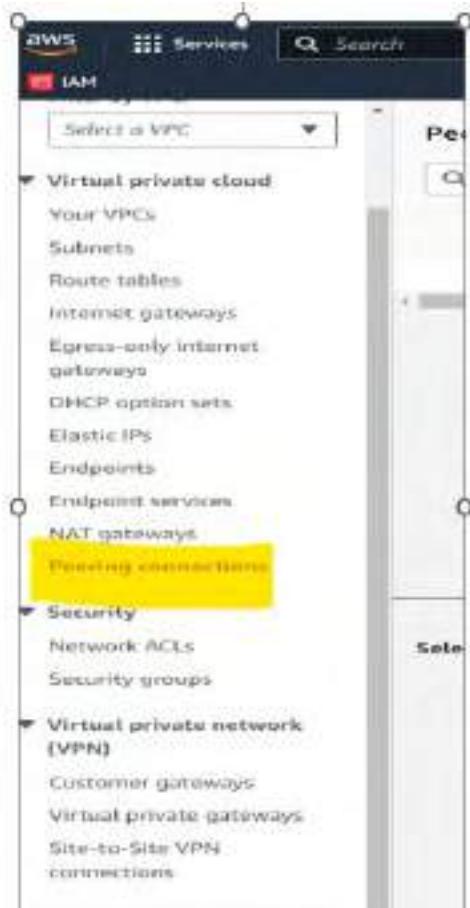
[ec2-user@ip-10-100-0-122 ~]$ Last login: Sun Jan  8 14:03:39 2023 from 10.100.0.122
[ec2-user@ip-10-100-0-122 ~]$ [ec2-user@ip-10-100-0-122 ~]$ [ec2-user@ip-10-100-0-122 ~]$
```

- 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

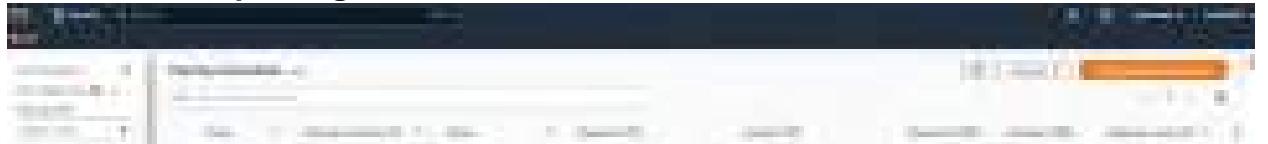


As this doesn't work because there is connectivity from VPC-A to VPC-B, so we need to create a VPC peering

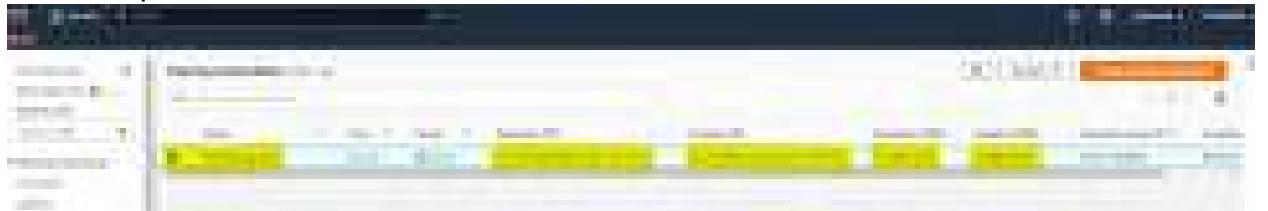
5. Setup VPC peering



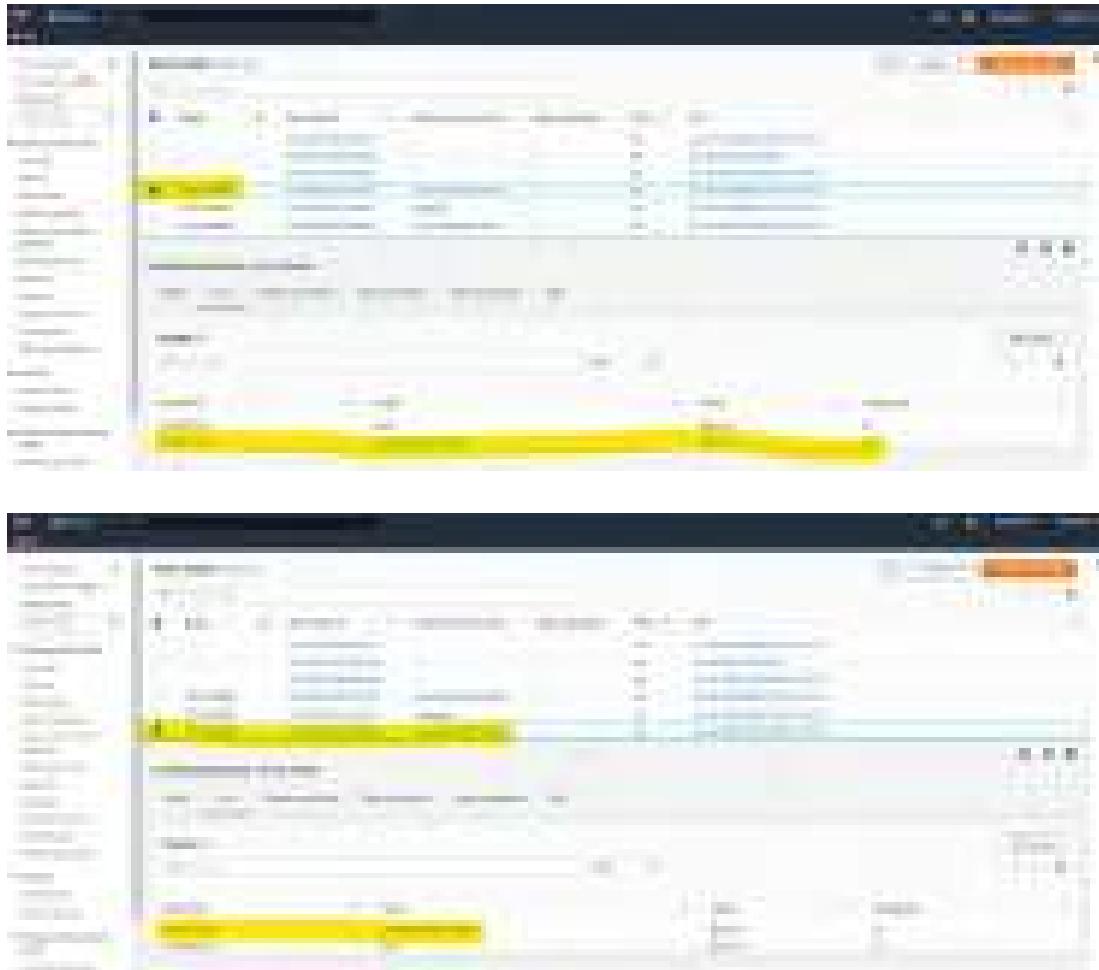
6. Choose **Create peering connection**.



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
[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
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
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
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
[ec2-user@ip-10-100-1-242 ~]$ 
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 ~]$
```