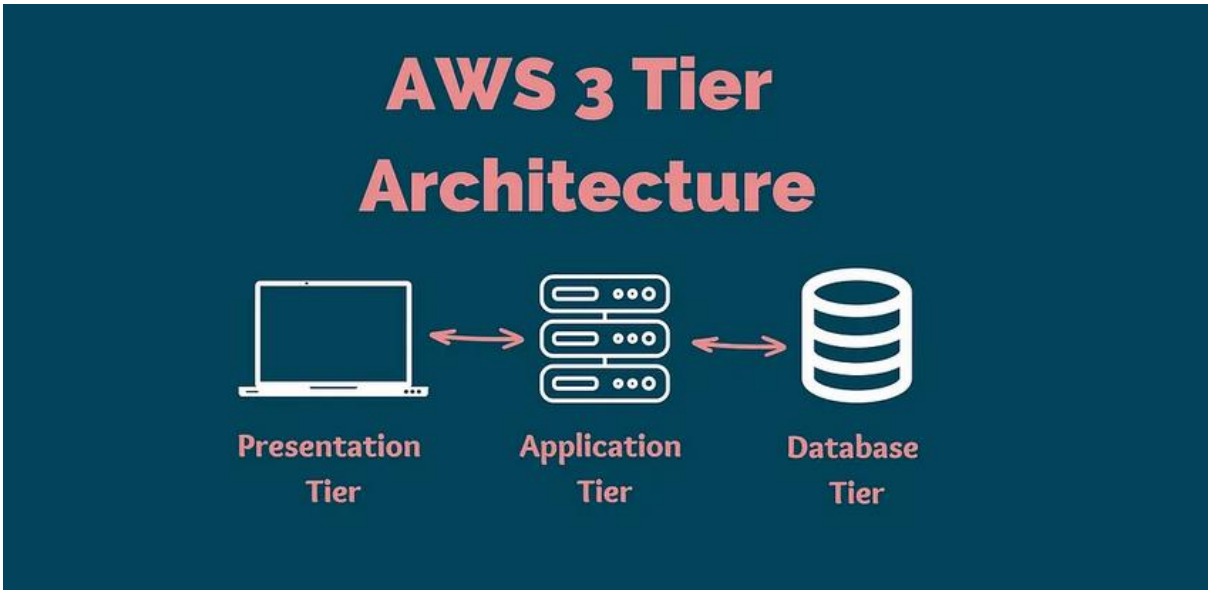
# PROJECT-01

# 3 TIER ARCHITECTURE

**What is three tier architecture?**

- Three tier architecture is a well established software application architecture that organizes applicaions into three logical and physical computing tiers. The presentation tier or web tier, the application tier, and the database tier, where application data is stored and managed.




**Presentation tier :** The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information and collect information from the user. Thus top level tier can run on web,desktop application, or a graphical user interface. Web tiers are developed by using HTML,CSS and javascript.

**Application tier :** The application tier is also known as the logic tier or middle tier, is the heart of the application. In application tier the data that is collected in the presentation tier is processed sometimes against other informstion in the data tier. The application tier can also add, delete, or modify data in the data tier.

**Database tier :** The database tier is also known as data access tier or back-end, is where the information that is processed by the application is stored and managed. This can be a RDS management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or

Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB, or MongoDB.

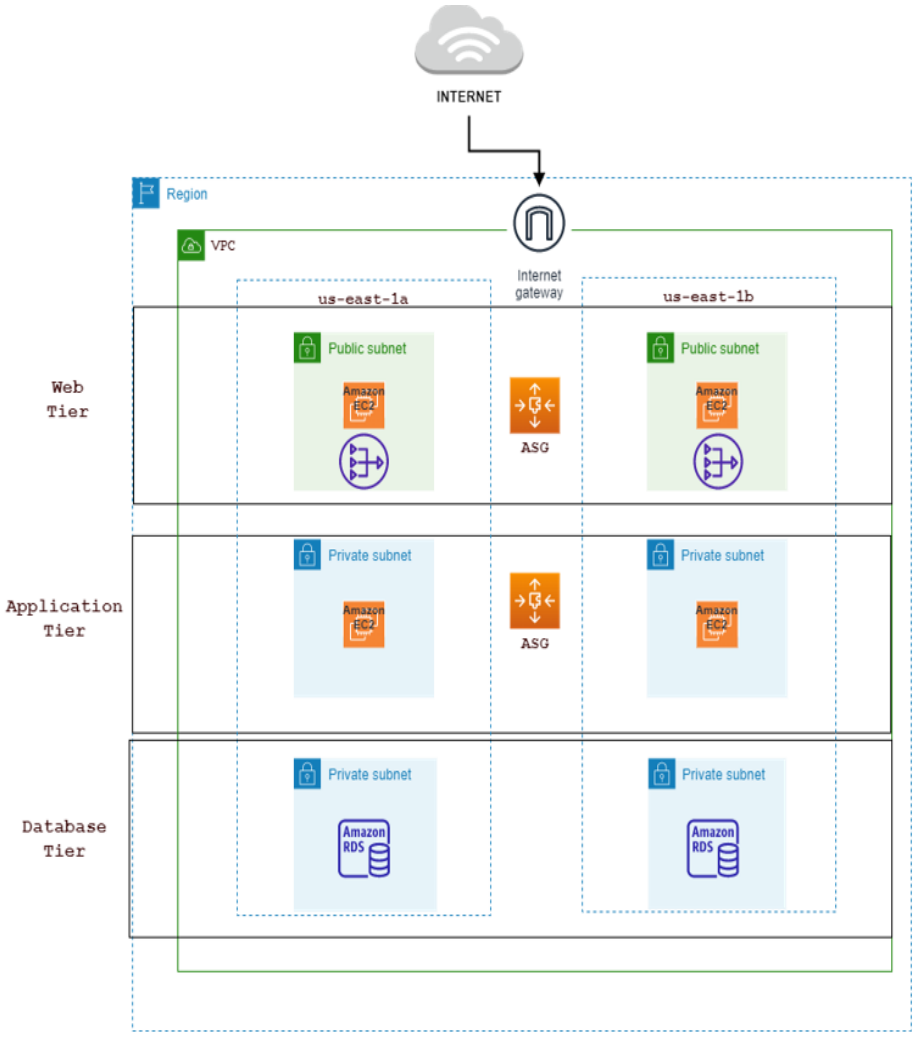
In a three tier application, all communication goes through the application tier. The web tier and the database tier cannot communicate directly with one another.

**Benefits of three tier architecture :**

It's a logical and physical separation of functionality. Each tier can run on a separate OS and server platform-for example, web server, application server, database server. And each tier runs on atleast one dedicated server hardware or virtual server, so each services of each tier can be customized and optimized without impacting the other tiers.

And also faster development, improved scalability, improved reliability, and improved security.

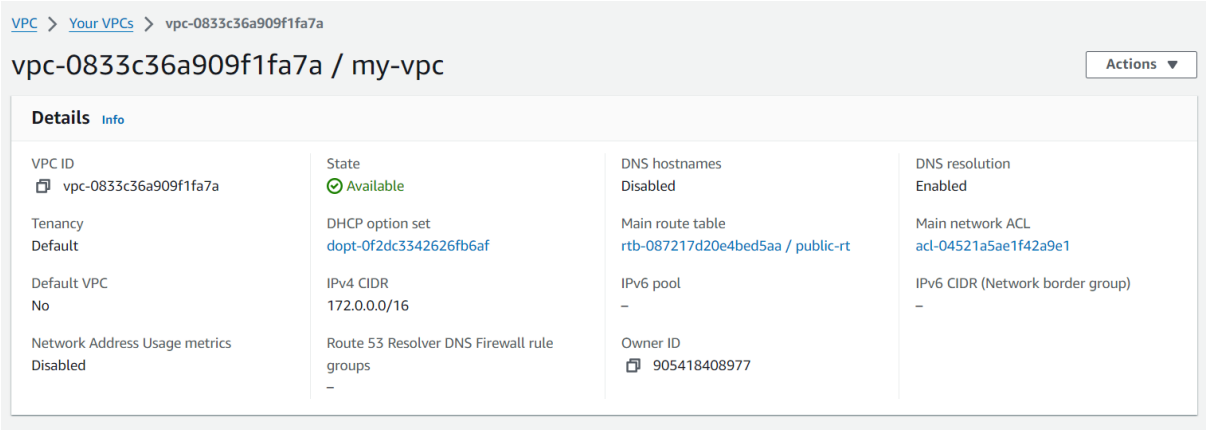
**Project:**



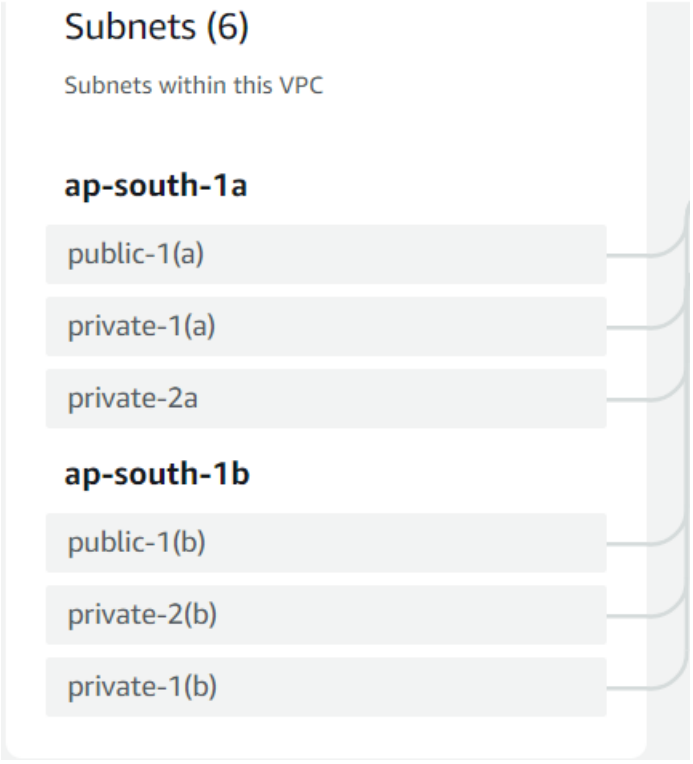

';'

**VPC :**

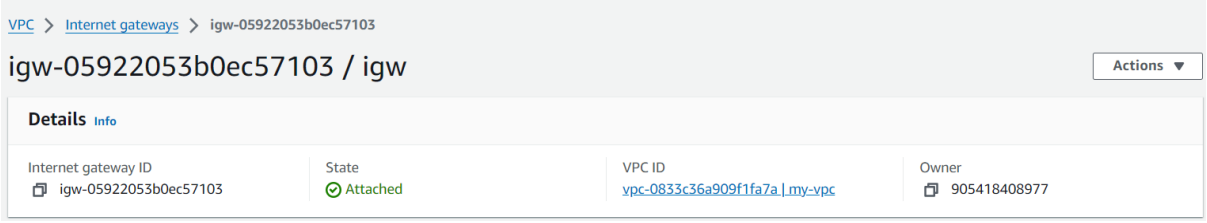
- Create a VPC with specific name [my-vpc] in a region Mumbai.



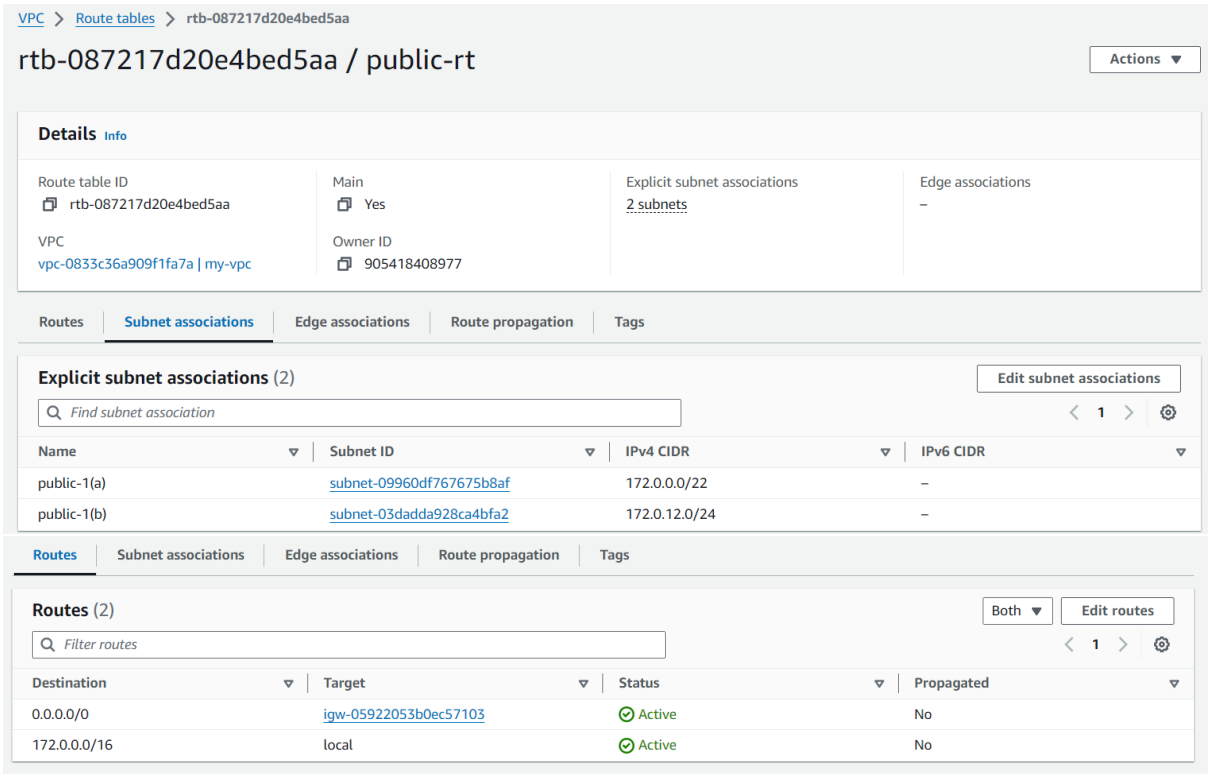

- Now create 6 subnets [1 public, 2 private subnets in one availability zone and same for the other subnets but in different availability zone] with in the same VPC.



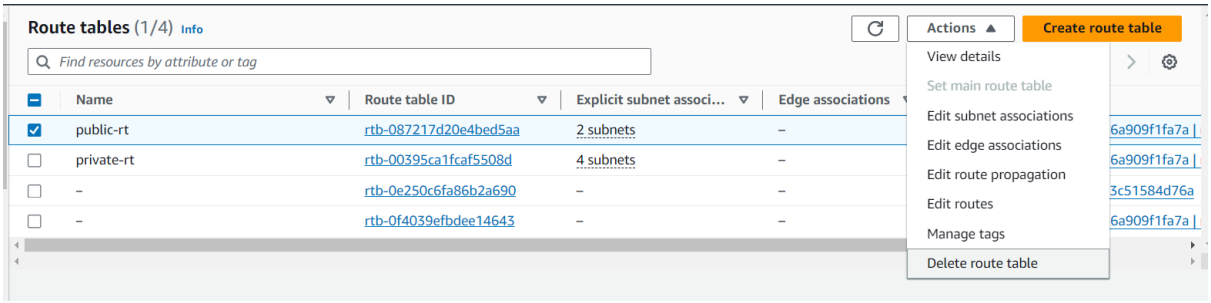

- Create a internet gateway and click on actions and attach it to the respective VPC.



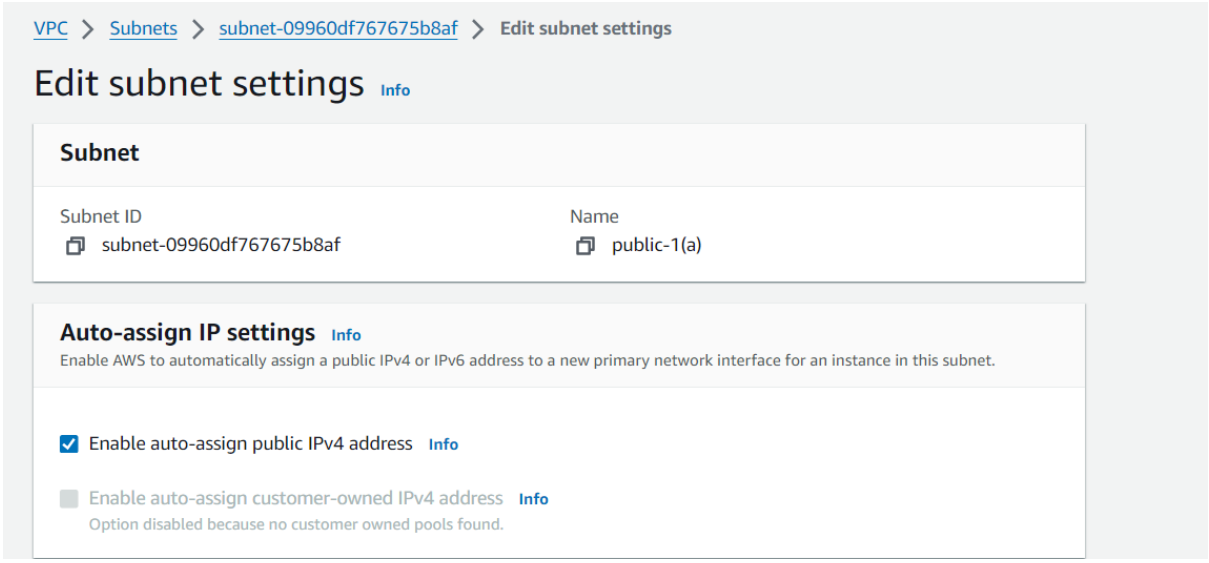

- And create route tables and associate them with subnets along with the internet connection attaches to the [1a & 2a] public subnets.



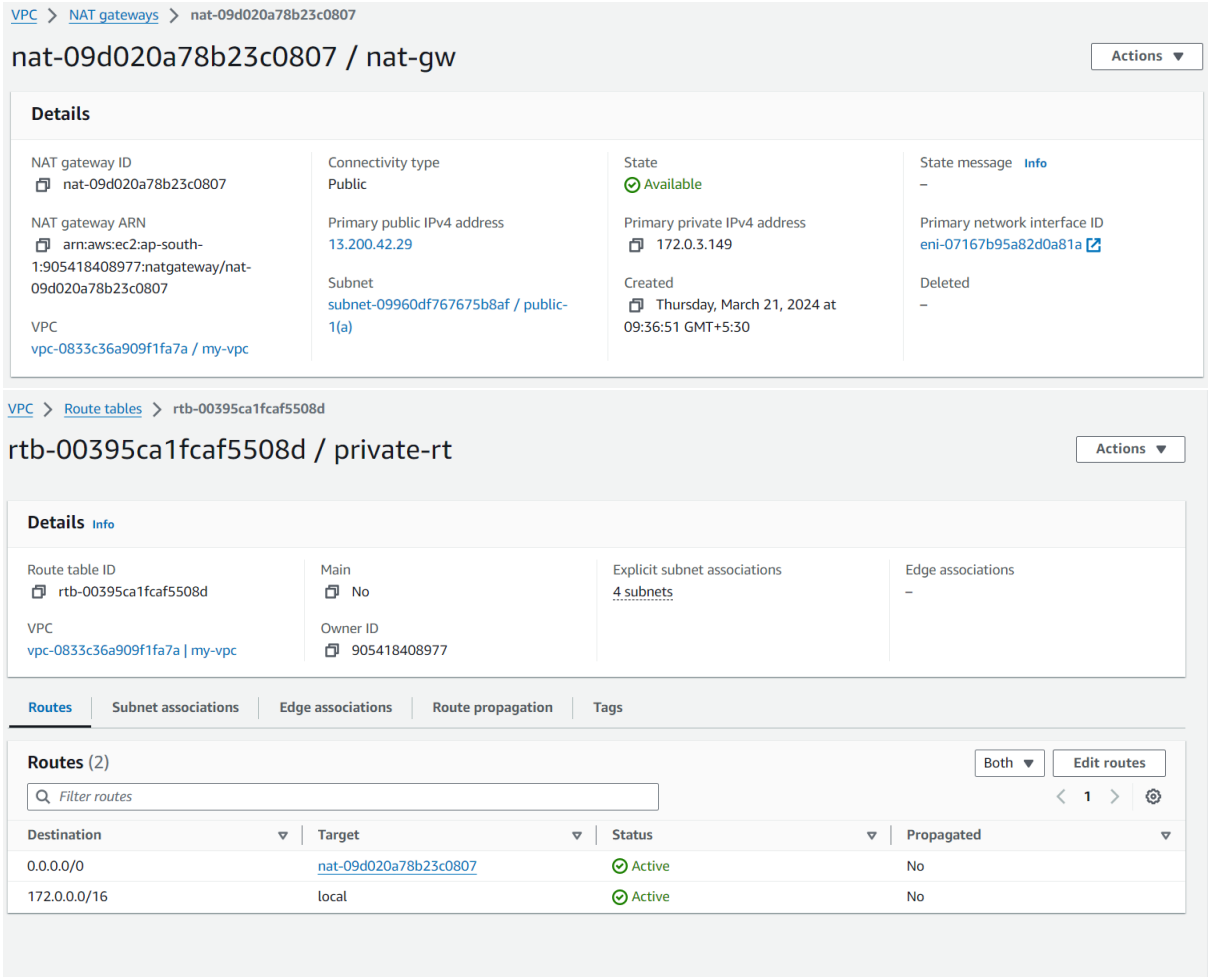

- And also set the public route tables as main route tables.



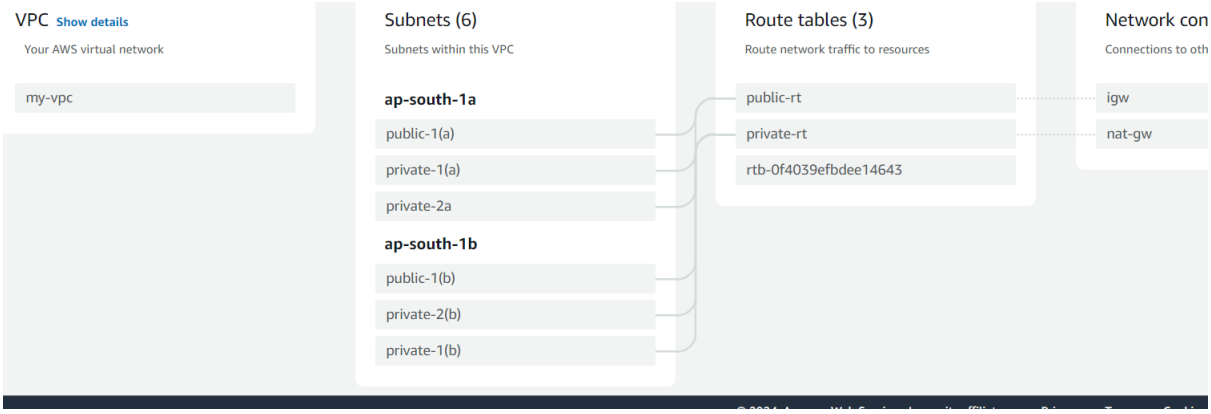

- Goto all subnet and click on edit subnets settings to auto assign IPV4.



- Create a Nat gateway for one way traffic attach to the private route table.



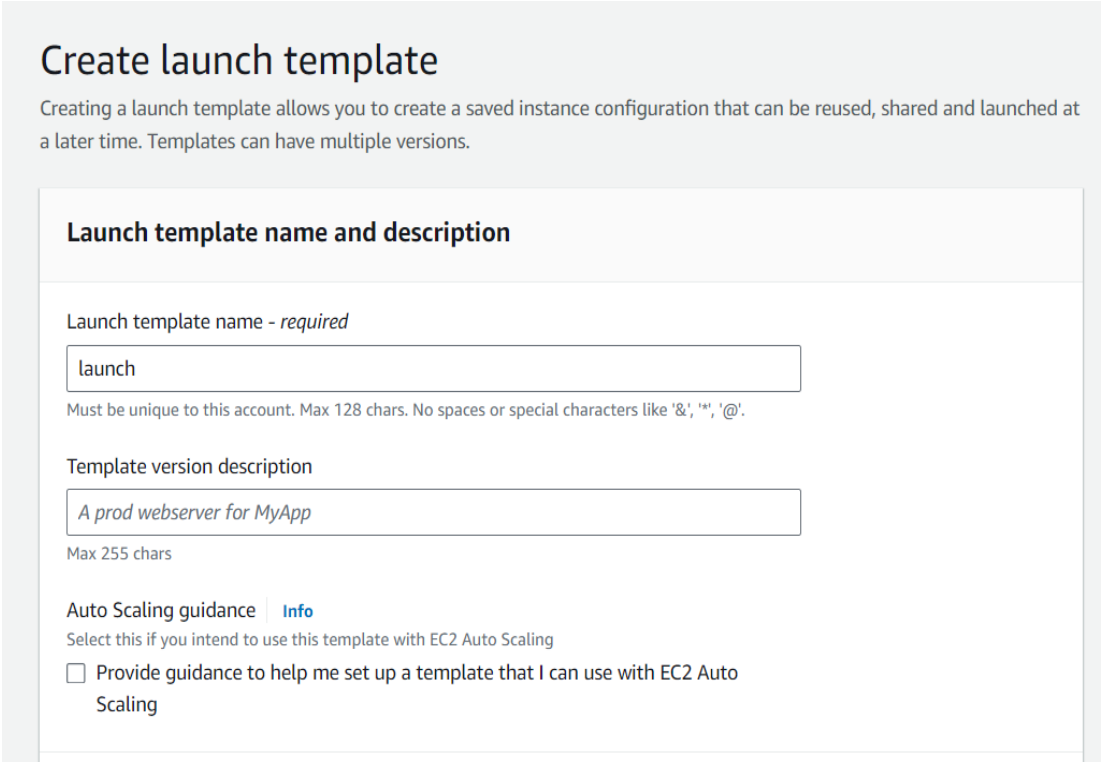

- After all connections established in VPC.



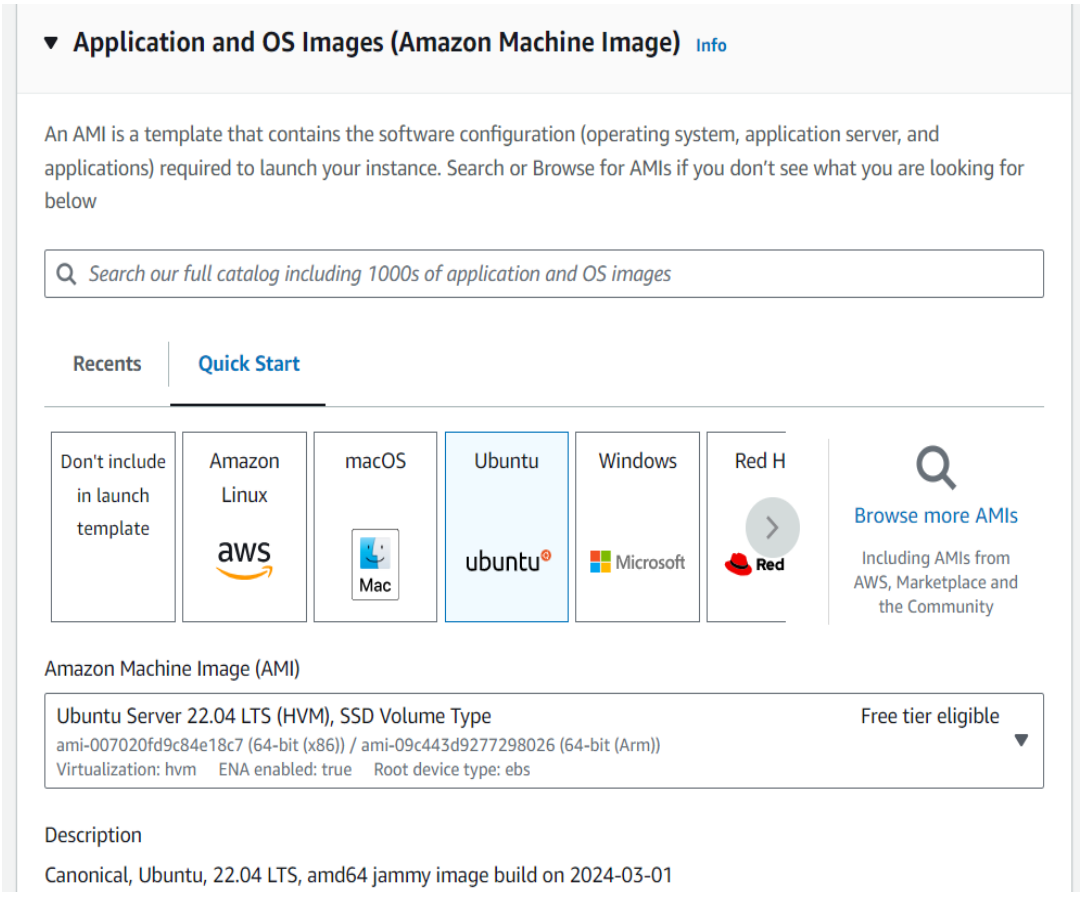

**Web tier:**

- create a web tier launch template

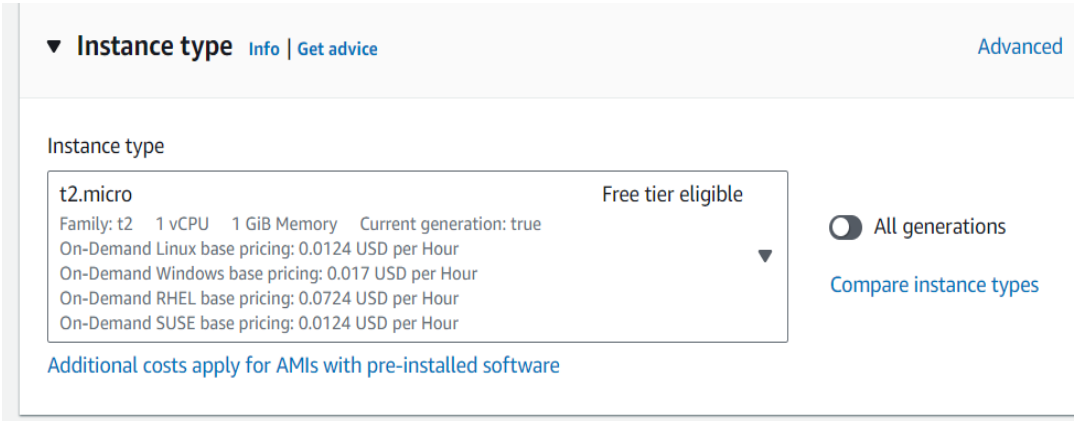
1. Give a specific name my-template1



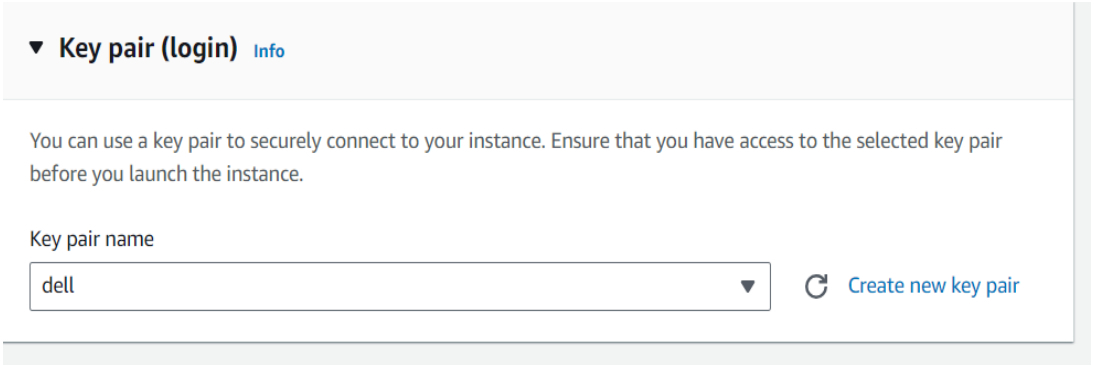

2. Click on AMI: ubuntu



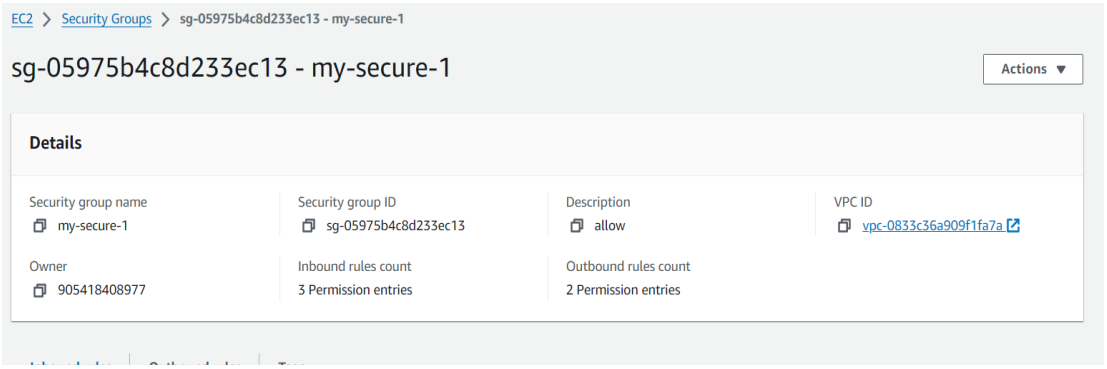
3. Instance type: t2 micro (1GB – free tier)



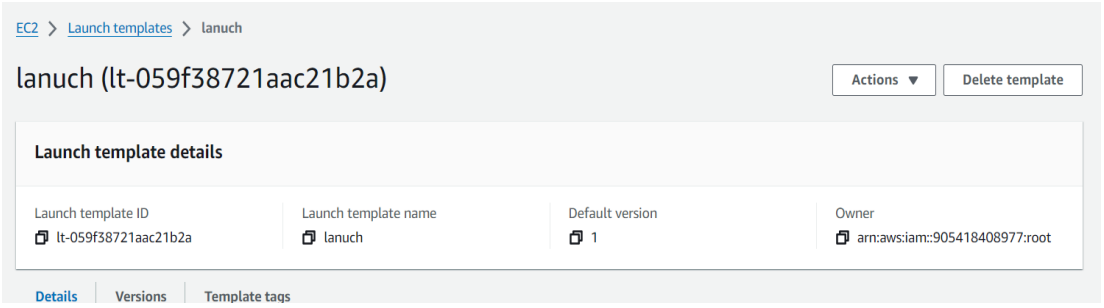

4. A new or existing key pair




5. Create a new security group and add inbound rules for SSH & HTTP

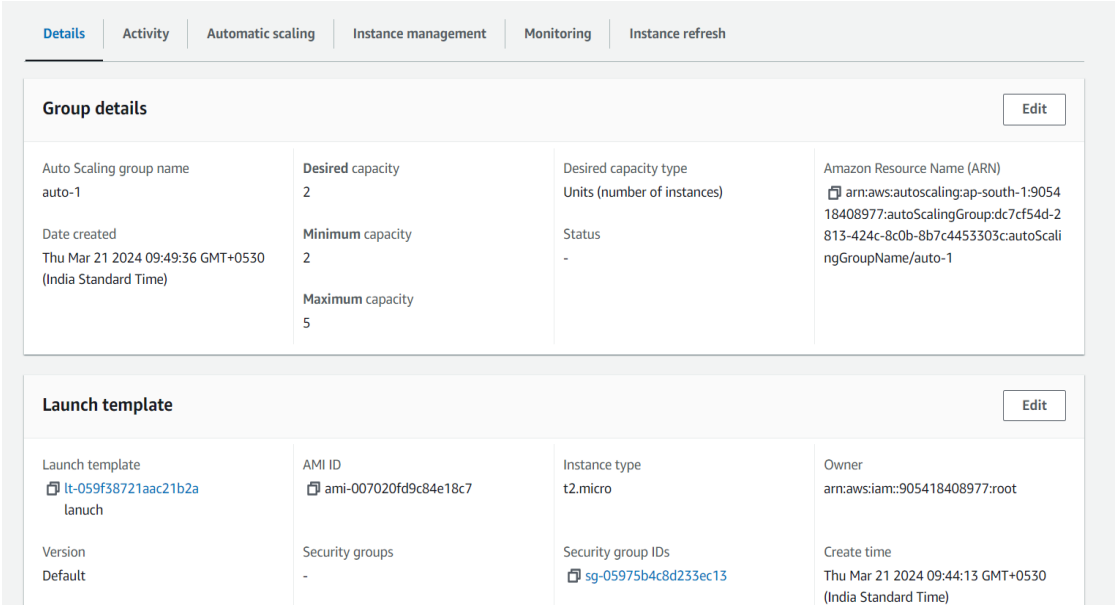



6. Now let's launch the template

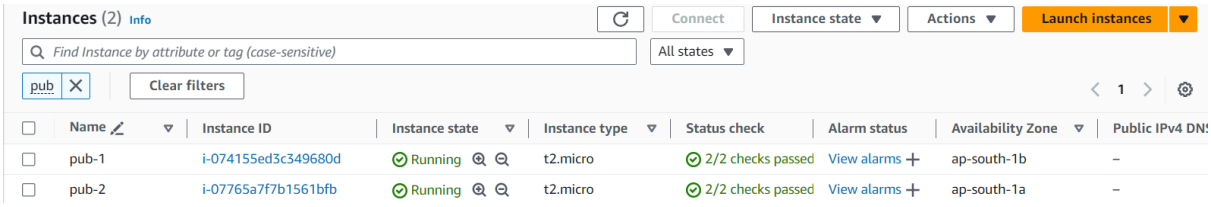



- Create auto-scaling with following settings
    1. Click on the auto scaling group and give a name
    2. Attach the created templet
    3. Select the created VPC [i.e. my-vpc] along with two public subnets
    4. Create a new application load balancer with target groups

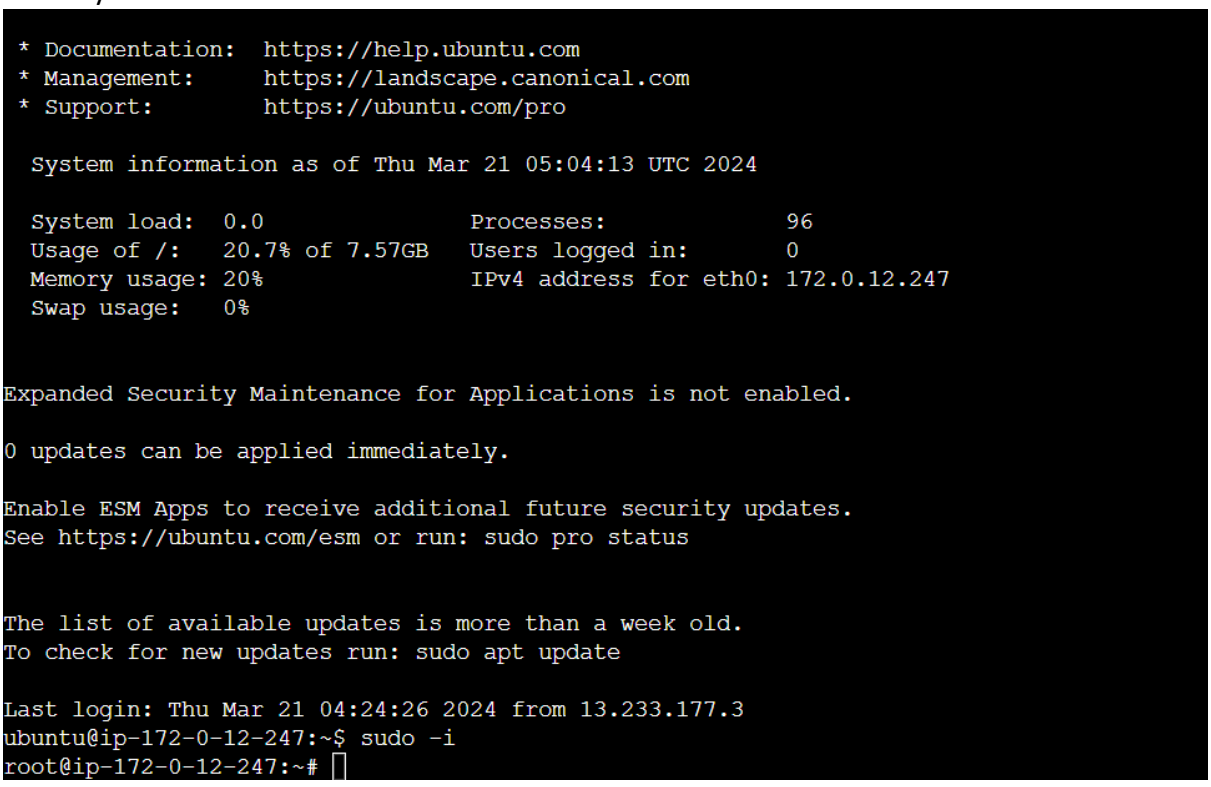
5. Configure the group size and scaling for desired capacity servers needed.




- Now we can go to EC2 to check weather the 2 instances are running successfully by using auto scaling.

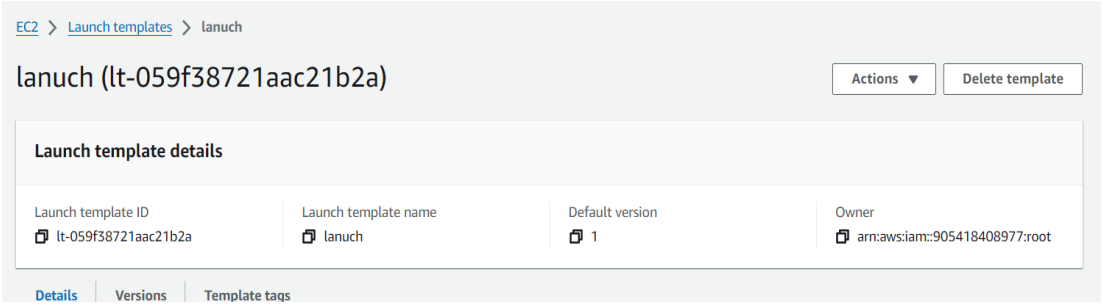



- Let's try to connect one of the instances in web.

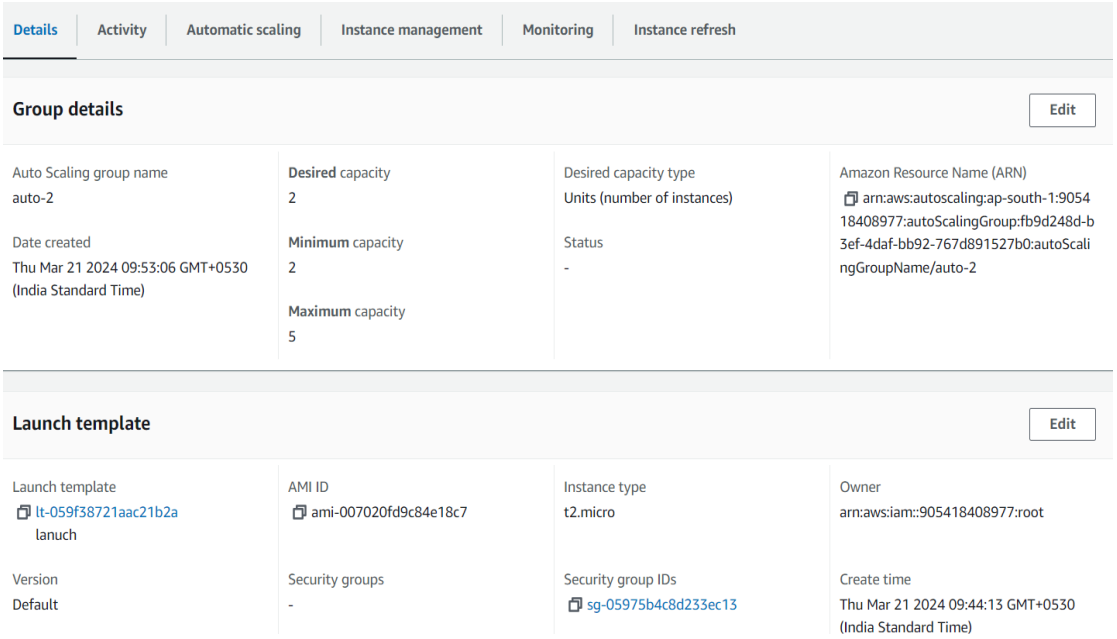


**Application tier:**

- Create a web tier launch template
    1. Give a specific name my-template2
    2. AMI: Ubuntu
    3. Instance type: t2.micro [1GB-free tier]
    4. New or existing key pair
    5. Create a new security group and add inbound rules for SSH & HTTP
    6. Now let's launch template

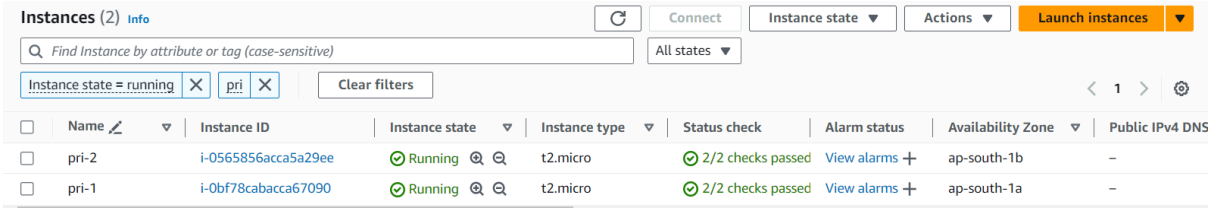



- Create auto-scaling group and give a name
    1. Click on auto-scaling group and give a name
    2. Attach the created template
    3. Select 3-tier-vpc along with two private subnets
    4. Create a new application load balancer with target groups
    5. Configure the group size and scaling for desired capacity servers needed




- Now we can see that two more instances using auto scaling are created within two private subnets that I have assigned a specific name to identify the private subnets as

private1 & private.

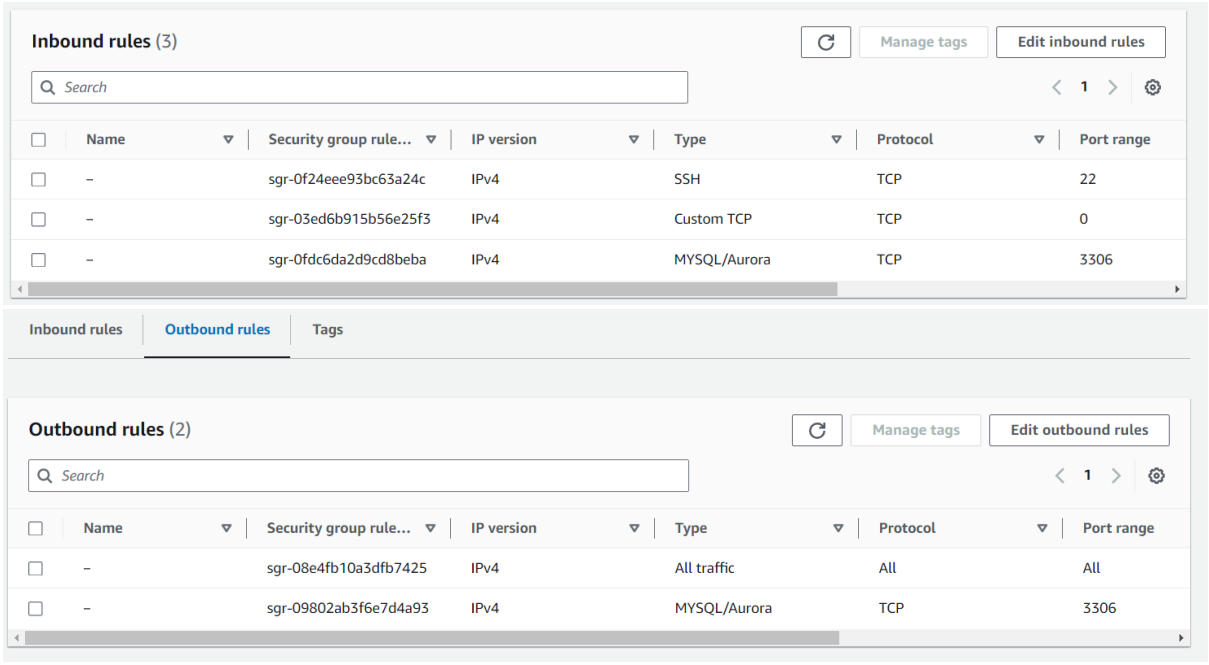



- Lets connect to the private EC2 instance from the public by using the key.pem

```
root@ip-172-0-12-247:~# ssh -i "key.pem" ubuntu@172.0.20.44
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Thu Mar 21 05:09:52 UTC 2024

  System load:  0.0               Processes:             96
  Usage of /:   20.7% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.0.20.44
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Mar 21 04:27:26 2024 from 172.0.12.247
ubuntu@ip-172-0-20-44:~$ sudo -i
root@ip-172-0-20-44:~#
```

**Data tier :**

- Create a security group add inbound and outbound rules for MySQL/Aurora.



**Inbound rules** (3)

Manage tags | Edit inbound rules

Search

| Name | Security group rule... | IP version | Type | Protocol | Port range |
|---|---|---|---|---|---|
| - | sgr-0f24eee93bc63a24c | IPv4 | SSH | TCP | 22 |
| - | sgr-03ed6b915b56e25f3 | IPv4 | Custom TCP | TCP | 0 |
| - | sgr-0fdc6da2d9cd8beba | IPv4 | MYSQL/Aurora | TCP | 3306 |

Inbound rules | Outbound rules | Tags

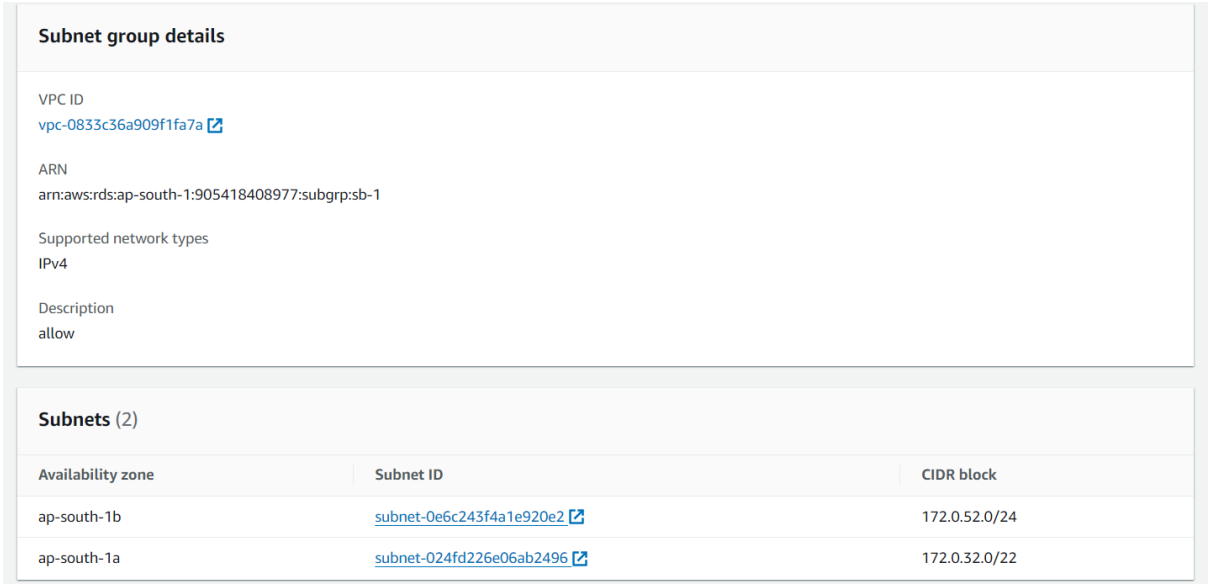
**Outbound rules** (2)

Manage tags | Edit outbound rules

Search

| Name | Security group rule... | IP version | Type | Protocol | Port range |
|---|---|---|---|---|---|
| - | sgr-08e4fb10a3dfb7425 | IPv4 | All traffic | All | All |
| - | sgr-09802ab3f6e7d4a93 | IPv4 | MYSQL/Aurora | TCP | 3306 |

- Create a database subnet groups by attaching 2 private subnets which are in different availability zones in RDS and attach my-vpc.



**Subnet group details**

VPC ID
vpc-0833c36a909f1fa7a

ARN
arn:aws:rds:ap-south-1:905418408977:subgrp:sb-1

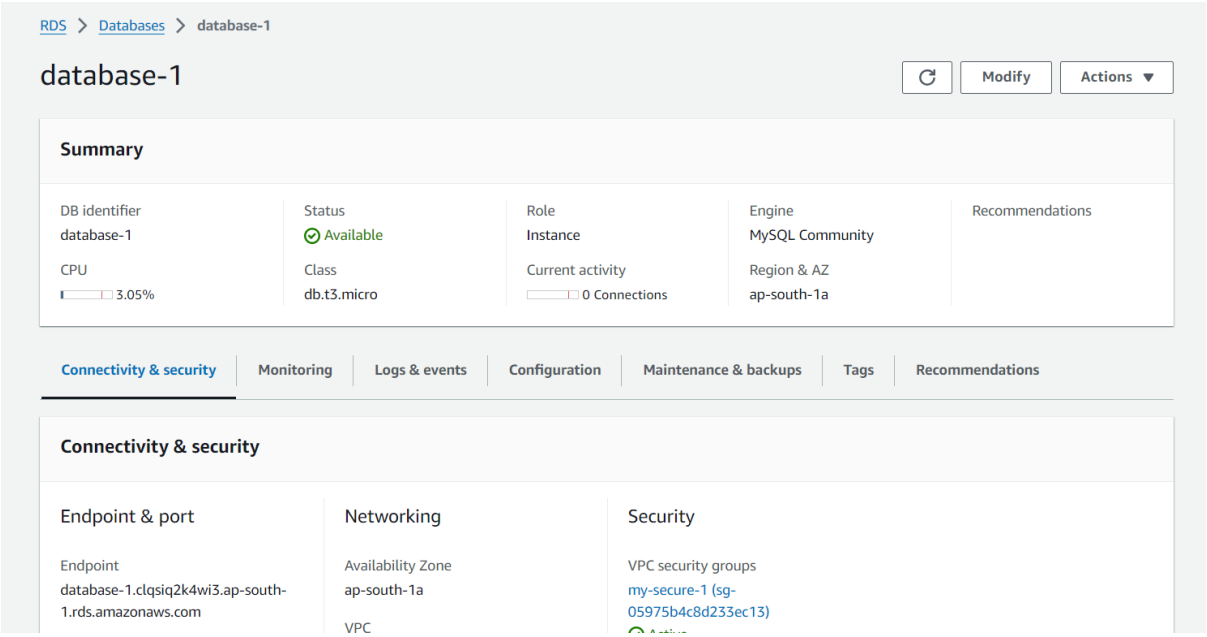
Supported network types
IPv4

Description
allow

**Subnets** (2)

| Availability zone | Subnet ID | CIDR block |
|---|---|---|
| ap-south-1b | subnet-0e6c243f4a1e920e2 | 172.0.52.0/24 |
| ap-south-1a | subnet-024fd226e06ab2496 | 172.0.32.0/22 |

- Now create [relational database] database.




- Connect to the database.
    1. Connect to the server
    2. Install MySQL
    3. And enter command `MySQL -h YOUR_DB_ENDPOINT -P 3306 -u YOUR_DB_USERNAME -p` in this command instead of your_dB_endpoint we should enter our database endpoint and also edit username as well after that enter password that you have given while creating a relational database.

```
[ec2-user@ip-10-0-156-224 ~]$ mysql -h brainiac-webapp-db.cgwzuzkgdaxy.us-east-1
.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 115
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```