AWS DEVSECOPS -139

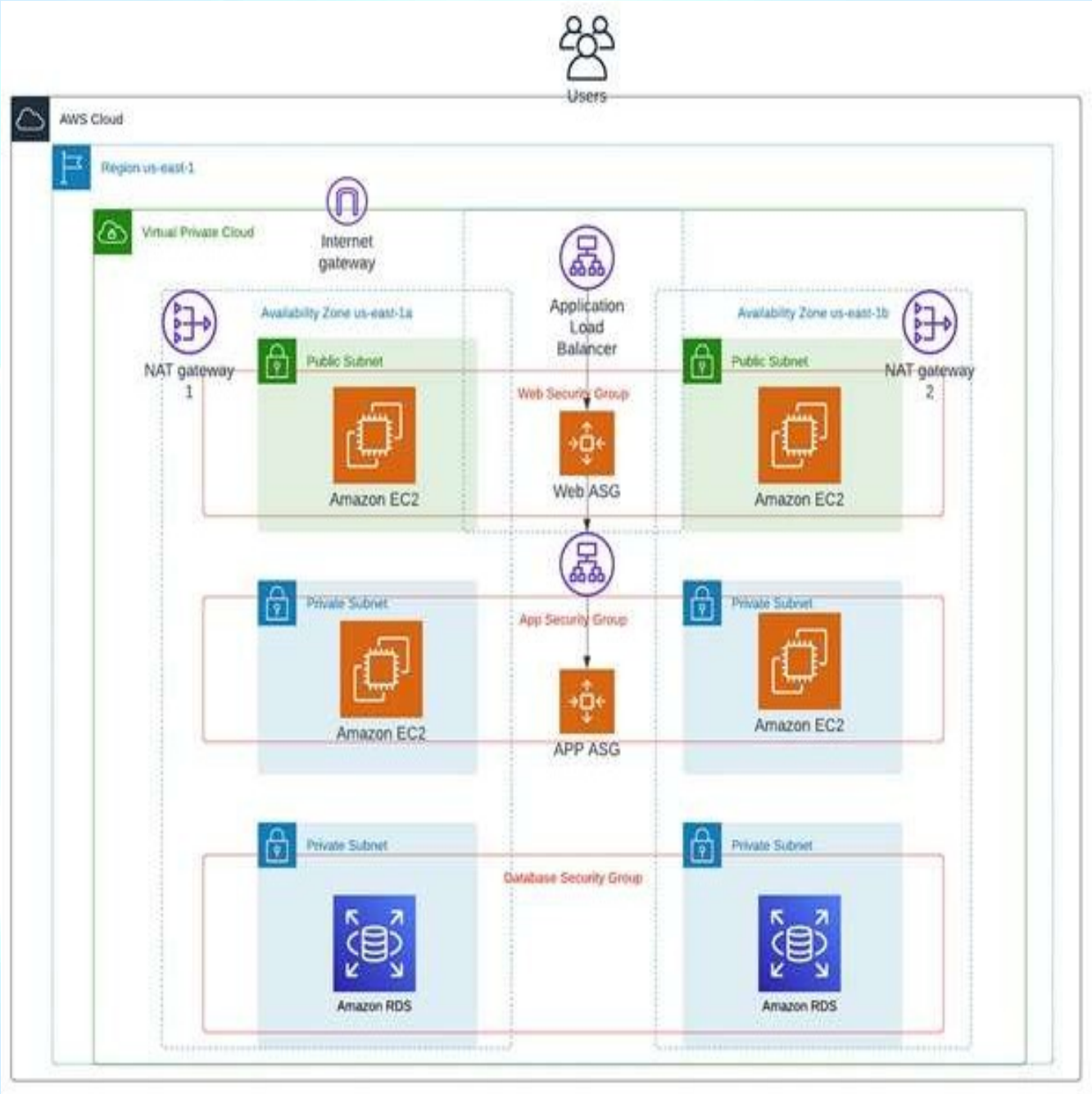
PROJECT – 01

3 – TIER ARCHITECTURE

N GOPI KRISHNA

gopikrishnanunna@gmail.com

3 – TIER ARCHITECTURE



What Are the 3 Tiers?

1) **Web Tier (Frontend)**

- Amazon EC2 in Public Subnet: Handles incoming user traffic.
- Connected to an Application Load Balancer.
- Protected by Web Security Group.

2) **App Tier (Business Logic)**

- Amazon EC2 in Private Subnet: Processes the logic (e.g., Node.js, Flask, or Spring Boot).
- Not directly accessible from the internet.
- Protected by App Security Group.
- Deploy it on a separate EC2 instance or container.
- Ensure it can communicate with both the frontend and database

3) **Database Tier**

- Amazon RDS in Private Subnet: Stores app data.
- Protected by Database Security Group.
- Install it on a separate EC2 instance
- Only accessible from App tier.

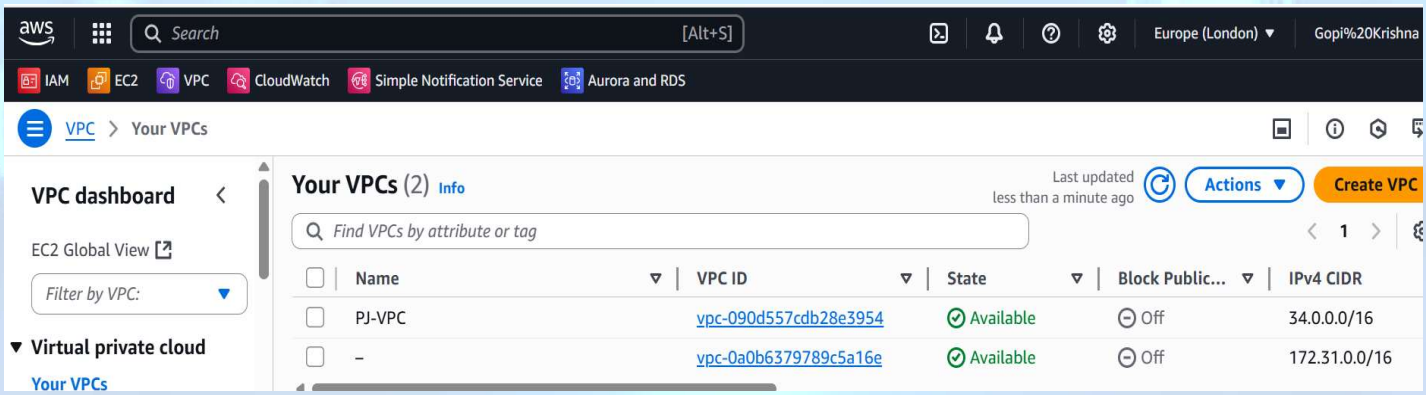
Overview of the Three Tiers:

- **Web Tier**:
  This is the **presentation layer** where users interact with the application. It includes the user interface built using technologies like HTML, CSS, JavaScript, or frameworks like React. It handles user input and displays data received from the backend.
- **App Tier**:
  Also known as the **application or logic layer**, this tier processes user requests, applies business logic, and communicates with the database. It's typically built using backend technologies like Node.js, Python (Flask/Django), Java (Spring Boot), etc.
- **Database Tier**:
  This is the **data layer** responsible for storing, retrieving, and managing data. It uses relational databases like MySQL or PostgreSQL, or NoSQL databases like MongoDB. The app tier interacts with this layer to perform CRUD operations.

PROCESS :

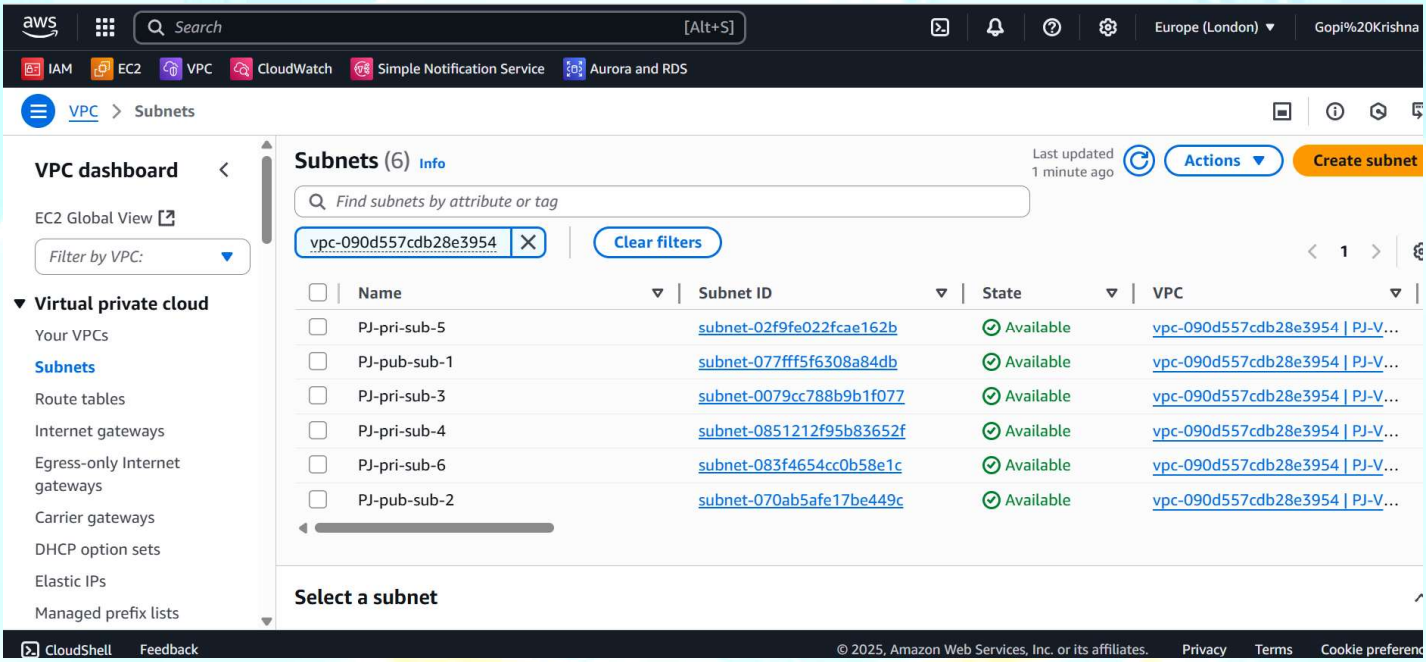
Step 1: Create a VPC (Virtual Private Cloud).

- Cerate VPC
- Name the Tag : PJ-VPC
- IPV4 address : 34.0.0.0/16
- VPC created



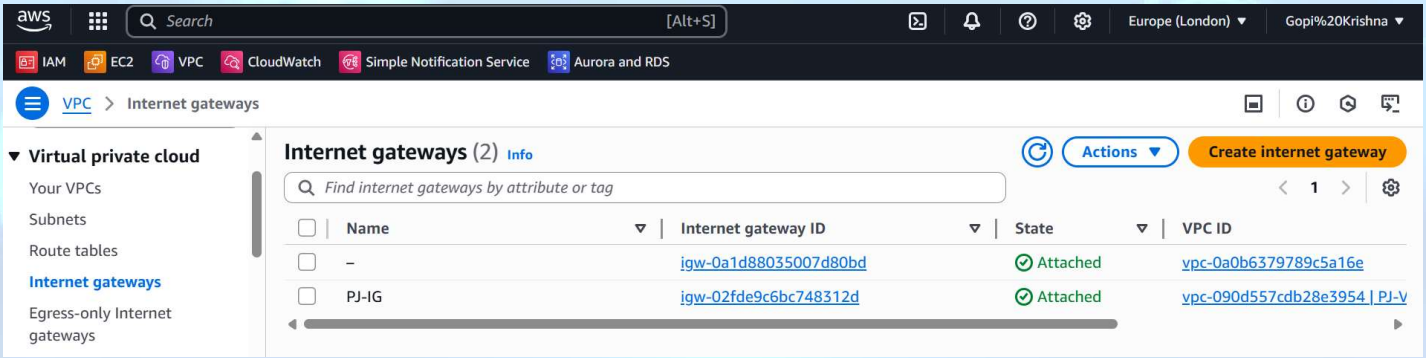
Step 2: Create a Subnets(6).

- Create 2 public subnets (for frontend EC2s).
- Create 4 private subnets (2 for App Tier, 2 for DB Tier).
- Select the given VPC id : PJ-VPC
- Subnet name : PJ-pub-sub-1 & add the 5 subnets with proper names for private and public scenario
- Select Availability zones for all subnets
- Customize the IP address with the proper ranges for all subnets
- Create the Subnet



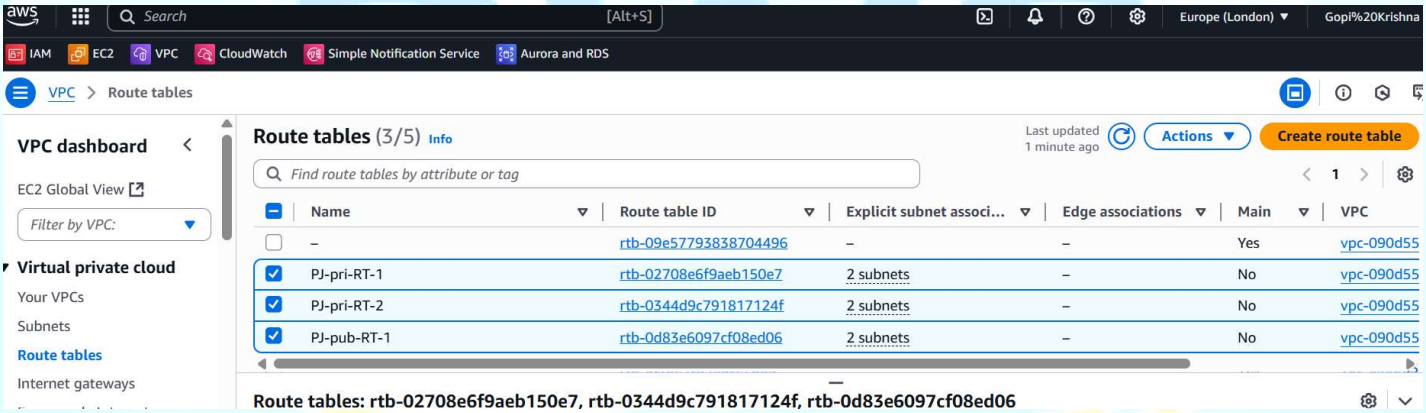
Step 3: Create an Internet Gateway (IG).

- Create Internet Gateway for public subnets access the internet connection
- Name : PJ-IG
- Create IG
- In Actions select and Attach VPC (in Notifications)



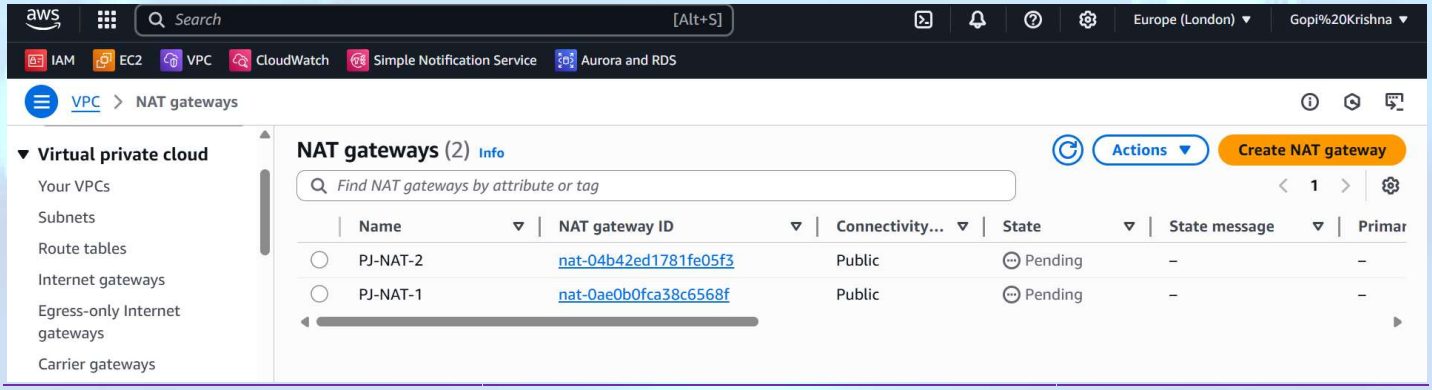
Step 4: Create Route tables (3)

- Create 1 public Route table & 2 private Route tables
- Name : PJ-pub-RT-1
- Select and choose the PJ-VPC
- Created the root table
- Actions and Edit the routes for internet connectivity and choose the all traffic and save it
- Then choose the subnet association for routing purpose
- Edit and select the public or private (Ex: public RT = public subnets and private RT = private subnets choose NAT gateway later)
- Done the RT,IG and subnet association
- For PJ-pri-RT-1 (select the subnets is pri-sub-3 and pri-sub-4) (NAT1)
- For PJ-pri-RT-2(select the subnets is pri-sub-4 and pri-sub-6) (NAT2)



Step 5: Create NAT Gateways (2) & add NAT Gateways to allow private instances to access the internet.

- Create NAT Gateway for Private subnets access the internet connection
- Name : PJ-NAT-1(private subnets 3,4) and PJ-NAT-2(private subnets 5,6)
- Subnets : select public only
- Connectivity Type : public
- Allocate the Elastic IP with specific number (Constant public IP)
- Then create NAT Gateway

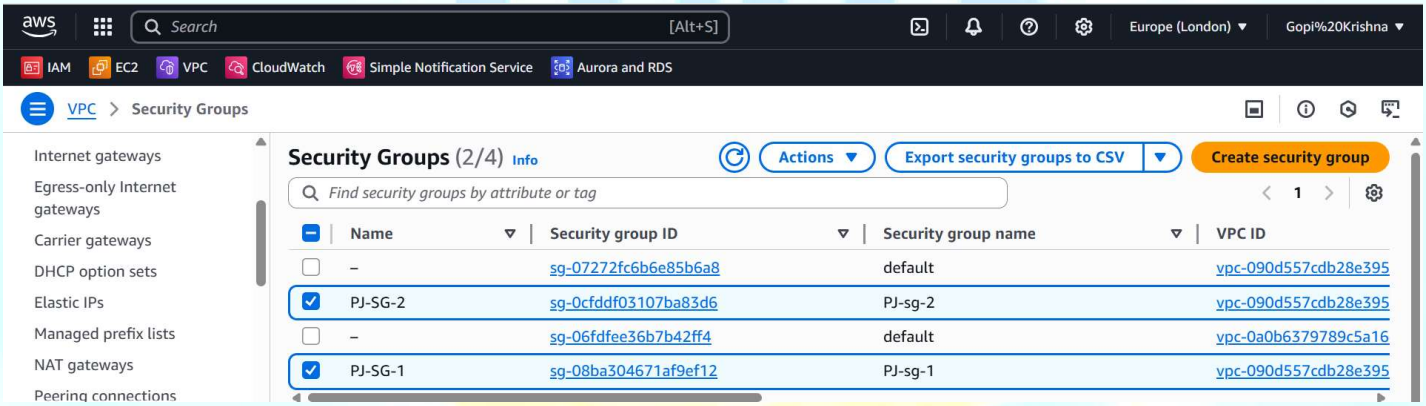


Step 6: Create Security Groups (2)

- It maintains inbound and outbound traffic rules
- Name : PJ-SJ-1 for public
- Web Usage and it allows traffic SSH (22) and HTTP (80) in inbound rules
- Create SG
- Name : PJ-SG-2 for private
- Database usage and it allows traffic SSH (22) and MYSQL (3306) in inbound rules.
- Create SG(it will use in RDS)

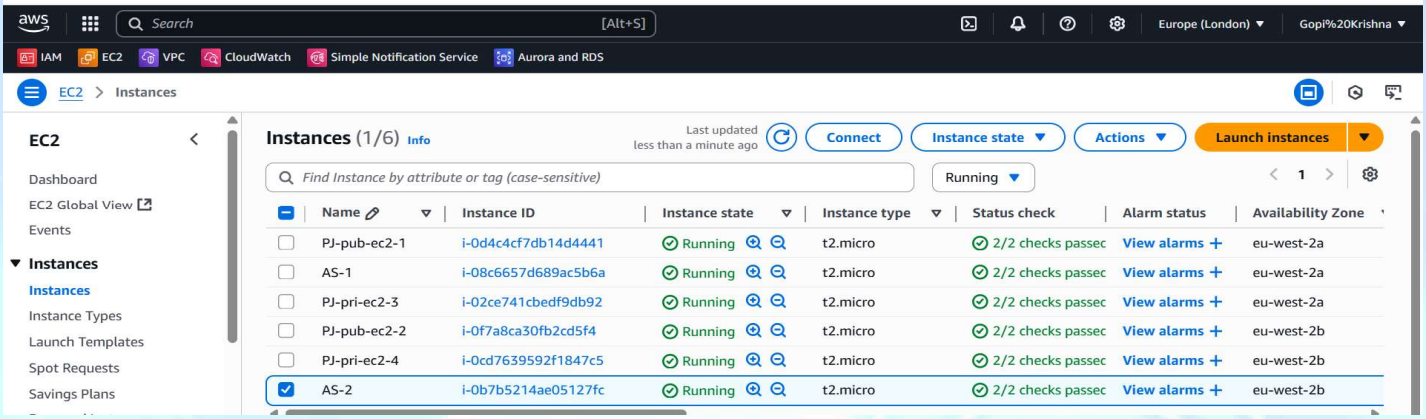
Enable DNS Hostnames :

- To enable automatic assignment of DNS hostnames to EC2 instances in your VPC, navigate to "Your VPCs," select your VPC, and edit the VPC settings to enable DNS hostnames.



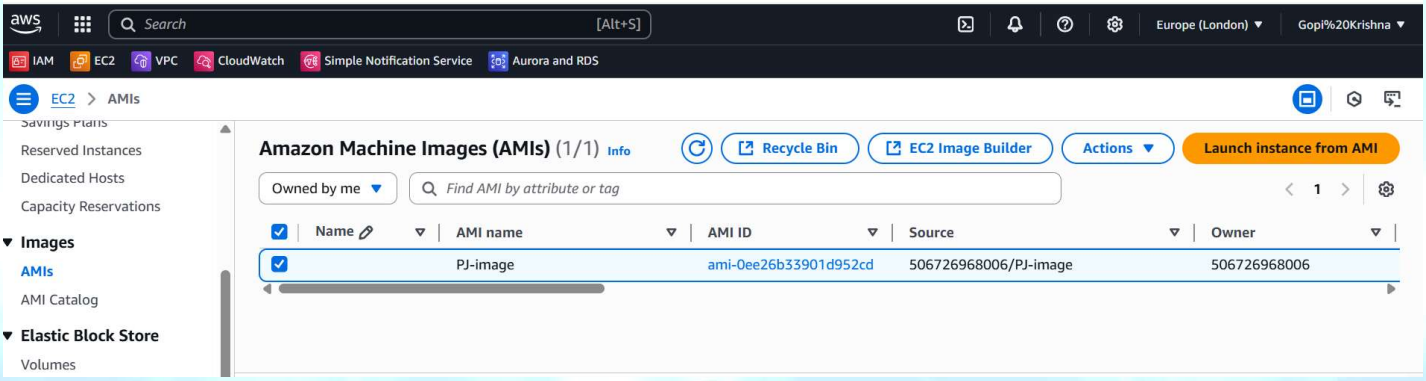
Step 7: Create and launch EC2 instances (2).

- Create two EC2 instances under our VPC (PJ-VPC) for Web layer (public subnets).
- Create two EC2 instances under our VPC (PJ-VPC) for App layer (private subnets)
- Select the Security Group (PJ-SG-1) for all instances.
- Launch instance with ubuntu OS, Instance type (t2micro)
- Select the keypair which we created (aryan.pem).
- Configure the Network settings(PJ-VPC,Subnets,SG,Public IP enable).
- Click on launch Instance.



Step 8 : Creating AMI's and Create image.

- Since we need to setup auto scaling we need AMI and launch template.
- First create a Web Instance AMI.
- To create an AMI first select the instance and in actions click on image and template option then click on image.
- Provide the details for AMI then click on create.
- Now go to instance section and click on web instance then click on actions and select image and templates then select create launch template using instance.
- Now we will be redirected to launch template page. There provide the details like name, ami here select the AMI that we created. Then check the network and security settings whether they are same as our instance. Then click on create template.

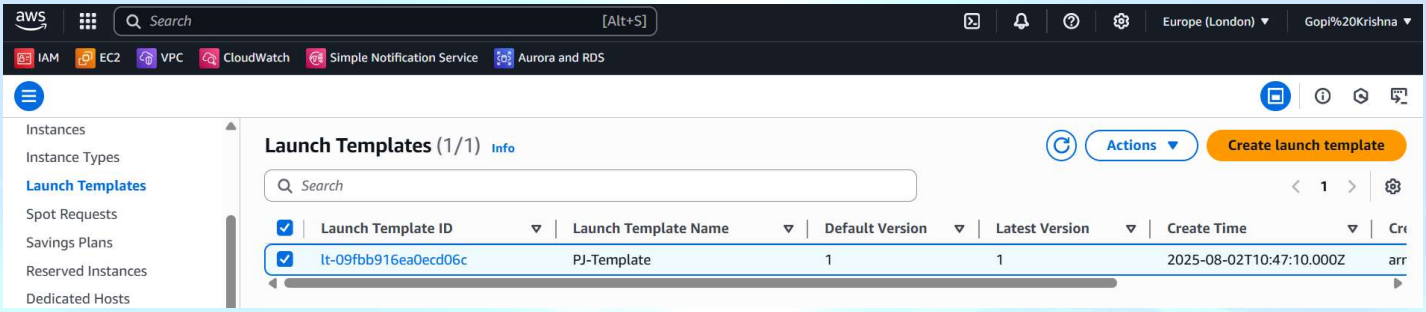


Step 8 : Creating Launch Template

Click on "Launch Templates" in the left sidebar, then "Create launch template"

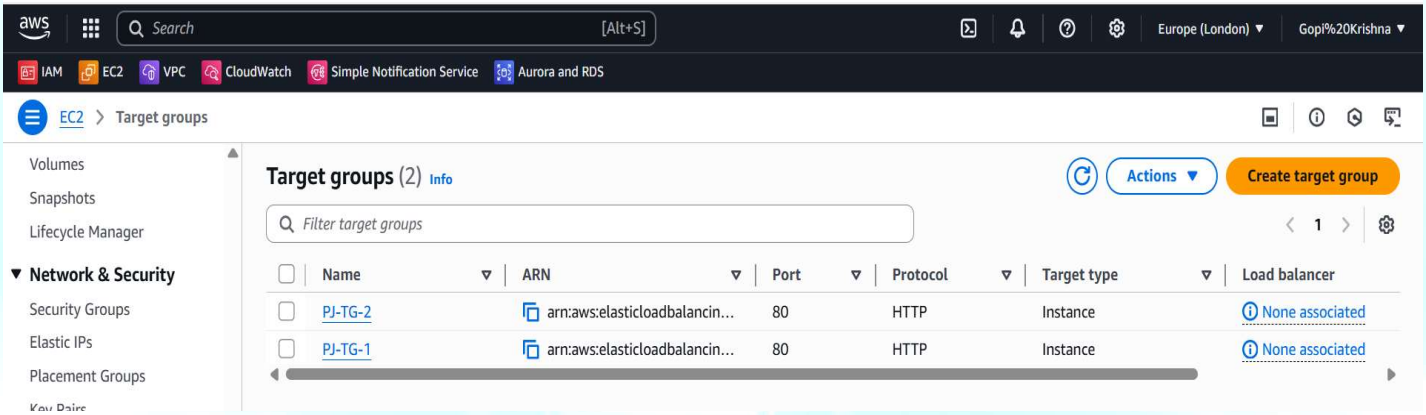
- Name and description (PJ-Template)
- AMI ID (select existing AMI)
- Instance type (t2.micro)
- Key pair (select existing key pair)
- Security groups (select existing SG)
- Storage configuration

- Launch Template.



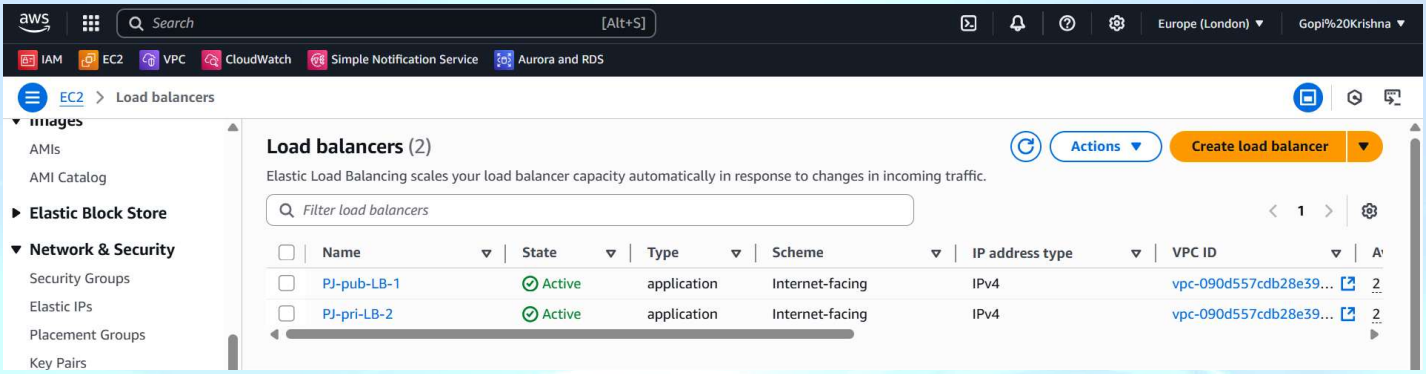
Step 8 : Creating Target Groups (2)

- Before creating a load balancer we need to create target group.
- Click on create target group. Then provide name and select VPC, add the instances and click on create target group.
- PJ-TG-1(for public)
  PJ-TG-2 (for private)



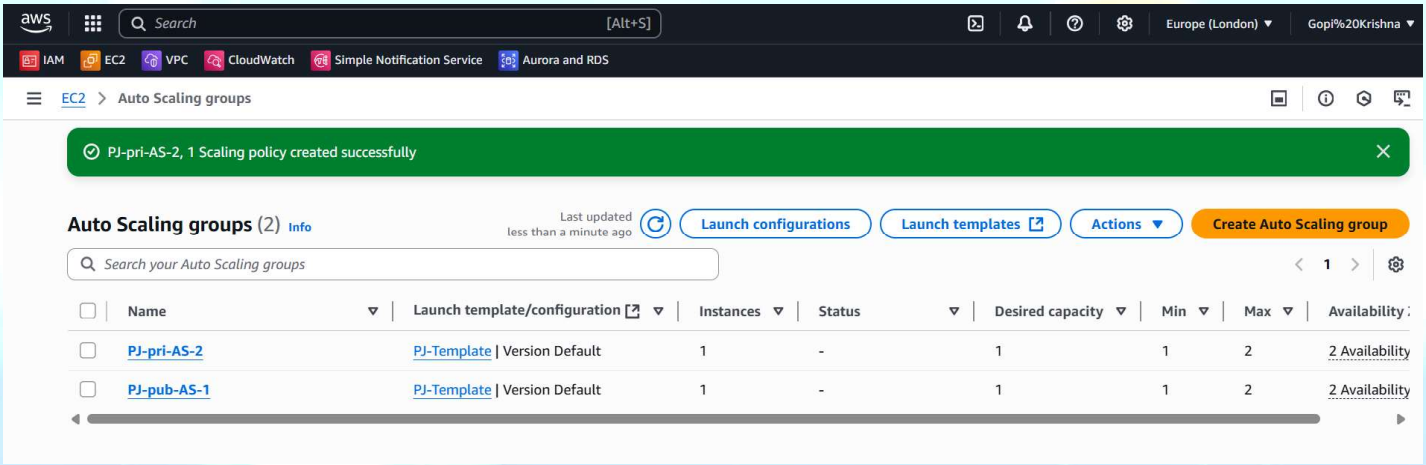
Step 8 : Creating Load Balancers LB (2)

- Now Go to load balancer page and click on create. Select Application Load Balancer.
- Then provide the LB name then select VPC, Availability Zones(2) and target group.
- PJ-pub-LB-1 (for public)
  PJ-pri-LB-1 (for private)



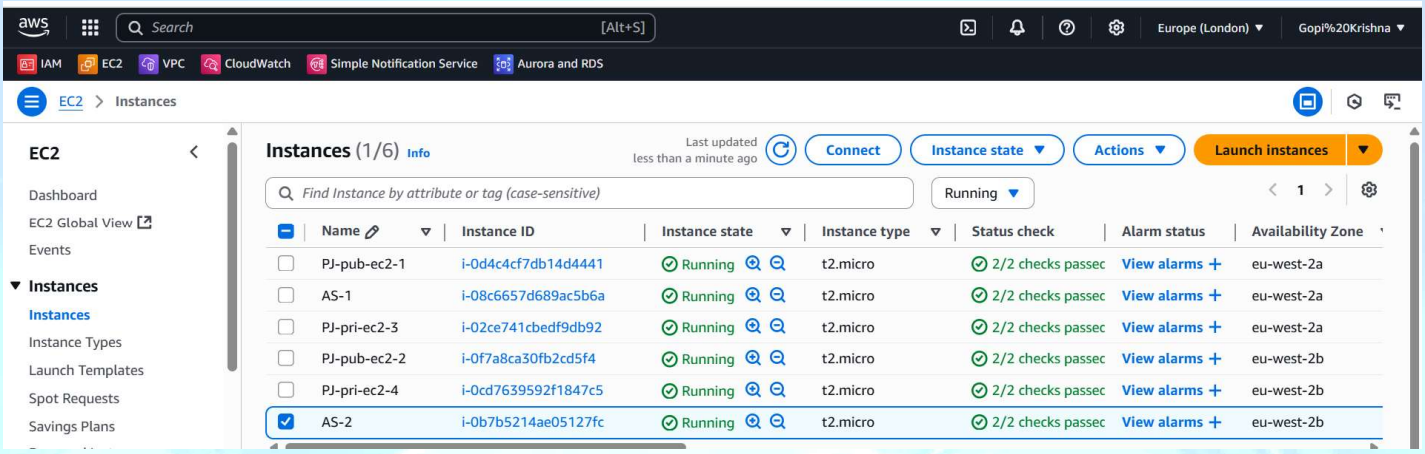
Step 9 : Creating Auto Scaling AS (2)

- As per the project requirement we need to create two auto scalling groups one for web layer and other for app layer.
- Open Auto Scaling Group page. Click on create auto scalling group option.
- Provide the name, select launch template then click on next.
- Now select VPC and availability zones and click on next.
- Now select on attach to an existing load balancer then select the target group and click on next.
- Now provide details for desired capacity in group size. And provide details for min desired and max desired capacity in scaling section. And select the target tracking policies. Select the desired policies and target value, instance warm-up and click on next for notification and tags. Then review and click on create button.



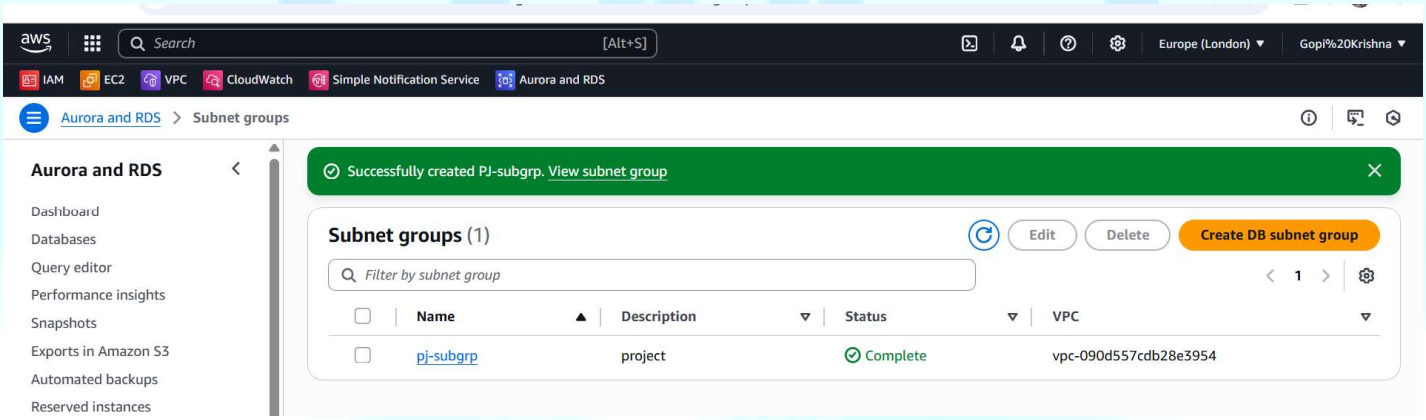
Adding Auto scaling:

- After creating auto scaling we get extra EC2 instances which we gave the desired capacity of auto scaling
- Auto scale manages the servers for increasing and decreasing the server and also performance by adding extra CPU's when needed (AS-1, AS-2) in below attachment.



Step 10 : Creating Subnet Group

- Open the RDS page in AWS console (Aurora and RDS).
- First create a database subnet group since we need Multi AZ single instance database.
- Go to subnet group in RDS
- Click on subnet group option
- Create a new subnet group.
- Fill the details Name (PJ-subgrp), Description, VPC, Subnets then click on create.

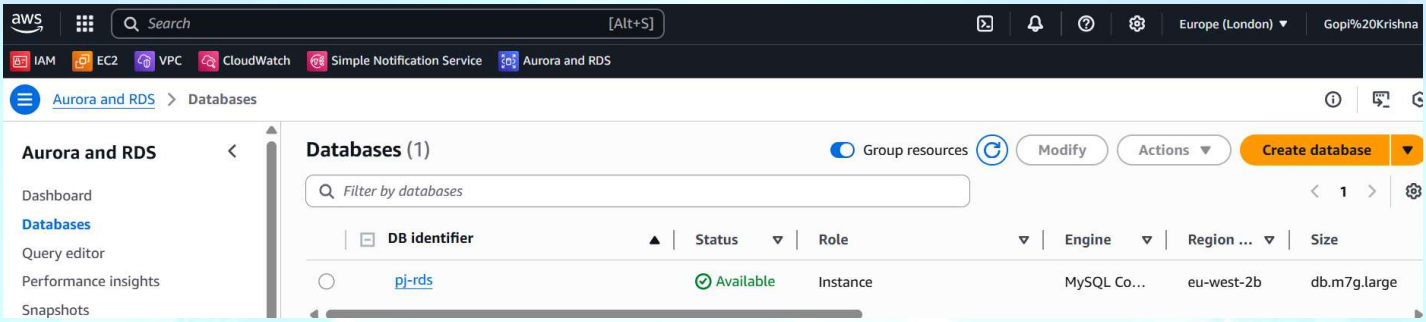


Step 11 : Creating RDS

Now create a database with the following options. Leave the rest as it is.

- Database creation method : Standard mode
- Engine Options : MySQL
- RDS Name : PJ-Rds
- Templates : Production
- Availability and durability : Multi AZ DB Instance deployment(2 ins)
- Configure connectivity : PJ-subgrp
- Initial database name : PJ-rds
- Provide DB identifier : admin and master username and master password(choose your own).
- Select the VPC, subnet group, Security Group.
- Then select yes for public access.

- Then click on create database.
- Once the database status changes to "Available," you can connect to it using a database client (e.g., MySQL Workbench) and the master user credentials.



Step 12 : Connecting Public terminal(pub-ec2-1) to Private terminal(pri-ec2-3)

- Connecting to App instances and connecting to Database from App instance.
- Since App instances are private instances we cannot connect to them directly using SSH.
- First connect to Web instance. Then create a .pem file inside the instance. Now use the following command "**chmod 400 KeyPairFilename.pem**".
- Now copy the ssh client string of private instance and paste it inside the web instance.
- Now we are connected to app instance for below stesps.
- For connecting to RDS we need to install mysql-server in app instance with following commands.

Commands List we used for this project :

- Update the server : **sudo apt update –y**
- Install the apache2 software in server **:yum install apache2**
- Change directory and check the index file : **cd /var/www/html**
- List with Index file : **Ls Index.html**
- Remove the existing index file : **rm index.html**
- Then create index.html : **vi index.html**
- Server Restart : **systemctl restart apache2**
- Server status check : **systemctl status apache2**
- Then install the mysql-server package: **sudo apt install mysql-server**
- Ensure that the server is running: **sudo systemctl start mysql.server**
- End point link : **MySQL -h pj-rds.cjkyiegucdkd.eu-west-2.rds.amazonaws.**com **-u admin -p**

Step 13: We successfully connected to public server for below commands

- Update the server : **sudo apt update –y**
- Install the apache2 software in server **:yum install apache2**
- Change directory and check the index file : **cd /var/www/html**
- List with Index file : **Ls Index.html**
- Remove the existing index file : **rm index.html**
- Then create index.html : **vi index.html**
- Server Restart : **systemctl restart apache2**
- Server status check : **systemctl status apache2**

```
ubuntu@ip-34-0-7-206:~$ sudo -i
root@ip-34-0-7-206:~# ls -l
total 8
-r-------- 1 root root 1676 Aug  2 11:39 aryan.pem
drwx------ 3 root root 4096 Aug  2 10:00 snap
root@ip-34-0-7-206:~# ssh -i "aryan.pem" ubuntu@ec2-34-0-20-67.eu-west-2.compute.amazonaws.com
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

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

 System information as of Sat Aug  2 11:50:32 UTC 2025

  System load:  0.0               Processes:             105
  Usage of /:   25.8% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 34.0.20.67
  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: Sat Aug  2 11:48:22 2025 from 34.0.7.206
ubuntu@ip-34-0-20-67:~$
```

Step 14: Here installing the mysql server with help of below command

- Then install the mysql-server package: sudo apt install mysql-server

```
root@ip-34-0-20-67:~# apt install mysql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package mysql
root@ip-34-0-20-67:~# apt install mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0 mysql-common
The following NEW packages will be installed:
  mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
0 upgraded, 4 newly installed, 0 to remove and 98 not upgraded.
Need to get 2767 kB of archives.
After this operation, 61.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Step 15: Here Restarting the SQL server for better result by using below command.

- Ensure that the server is running: **sudo systemctl restart mysql.server**

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-34-0-7-206:~# sudo systemctl start mysql.service
root@ip-34-0-7-206:~#
```

‘

Step 16: Create the database and table with Authors data for below steps

1) Creating database : **Krishna** database created for below query

   Create database Krishna

   Then use this Database for table creation: **Use Krishna**

2) Creating table: **Books** Table created for below query

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
     PublishedYear INT,
     Genre VARCHAR(50)
 );
```

3) Inserting Data into Table: By using below query

```
INSERT INTO Books (BookID, Title, Author, PublishedYear, Genre)
   VALUES
  (1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction'),
   (2, '1984', 'George Orwell', 1949, 'Dystopian'),
   (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Classic'),
   (4, 'Sapiens', 'Yuval Noah Harari', 2011, 'Non-fiction');
```

4) Check the data in the table : By using below query

   Select * from Books

O/P : It shows the below screenshot for all Authors and other details

```
Database changed
mysql> CREATE TABLE TestTable AS
    -> SELECT customername, contactname
    -> FROM customers;
ERROR 1146 (42S02): Table 'krishna.customers' doesn't exist
mysql> CREATE TABLE Books (
    ->     BookID INT PRIMARY KEY,
    ->     Title VARCHAR(100),
    ->     Author VARCHAR(100),
    ->     PublishedYear INT,
    ->     Genre VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO Books (BookID, Title, Author, PublishedYear, Genre)
    -> VALUES
    -> (1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction'),
    -> (2, '1984', 'George Orwell', 1949, 'Dystopian'),
    -> (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Classic'),
    -> (4, 'Sapiens', 'Yuval Noah Harari', 2011, 'Non-fiction');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> show tables
    -> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser
ine 2
mysql> select *from
    -> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser
ine 2
mysql> select * from books;
ERROR 1146 (42S02): Table 'krishna.books' doesn't exist
mysql> select * from Books;
+--------+-----------------------+--------------------+---------------+-------------+
| BookID | Title                 | Author             | PublishedYear | Genre       |
+--------+-----------------------+--------------------+---------------+-------------+
|      1 | To Kill a Mockingbird | Harper Lee         |          1960 | Fiction     |
|      2 | 1984                  | George Orwell      |          1949 | Dystopian   |
|      3 | The Great Gatsby      | F. Scott Fitzgerald|          1925 | Classic     |
|      4 | Sapiens               | Yuval Noah Harari  |          2011 | Non-fiction |
+--------+-----------------------+--------------------+---------------+-------------+
4 rows in set (0.00 sec)
```

Step 17: The same way connected to other private sever with same commands and install the SQL server below.

```
Get:51 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:52 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 36.0 MB in 7s (5339 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
98 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-34-0-7-206:~# sudo apt install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libclone-perl libencode-locale-perl libevent-pthreads-2.1-7t64 libfcgi-bin libfcgi-perl l
  libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-mediatypes-perl l
  liburi-perl mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.
Suggested packages:
```

Step 18: By using the End point link & port we can see the data (i.e check the data with private server SSH string) with below command and added some commands.

- End point link : **MySQL -h pj-rds.cjkyiegucdkd.eu-west-2.rds.amazonaws.com -u admin -p**

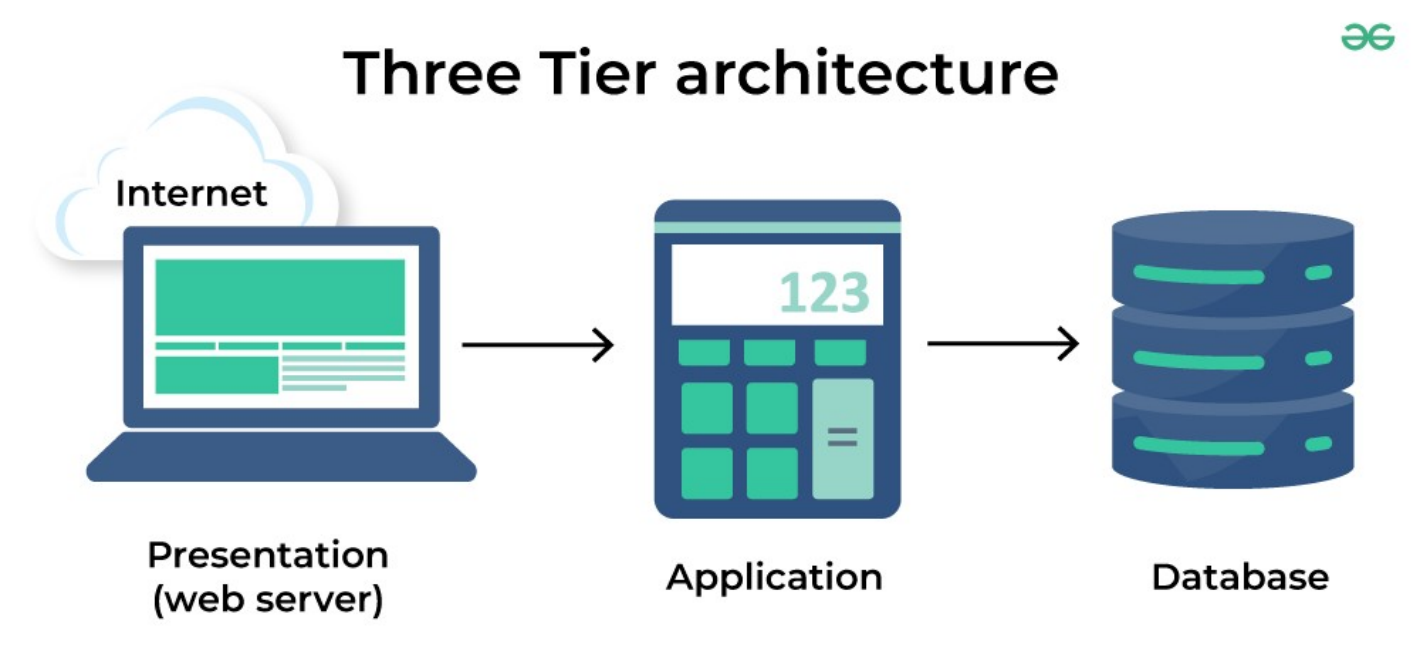
```
mysql> ^DBye
root@ip-34-0-7-206:~# mysql -h pj-rds.cjkyiegucdkd.eu-west-2.rds.ama
```

Step 19: Even we have verified with other private server with those details then get the same output below.

```
mysql> select *from
    -> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser
ine 2
mysql> select * from books;
ERROR 1146 (42S02): Table 'krishna.books' doesn't exist
mysql> select * from Books;
+--------+----------------------+--------------------+---------------+-------------+
| BookID | Title                | Author             | PublishedYear | Genre       |
+--------+----------------------+--------------------+---------------+-------------+
|      1 | To Kill a Mockingbird | Harper Lee         |          1960 | Fiction     |
|      2 | 1984                 | George Orwell      |          1949 | Dystopian   |
|      3 | The Great Gatsby     | F. Scott Fitzgerald |          1925 | Classic     |
|      4 | Sapiens              | Yuval Noah Harari  |          2011 | Non-fiction |
+--------+----------------------+--------------------+---------------+-------------+
4 rows in set (0.00 sec)
```

AWS 3-Tier Architecture Project - Conclusion

This architecture is a robust example of a well-architected 3-tier application deployment on AWS, aligning with best practices for high availability, fault tolerance, scalability, and security. It is ready for production-grade deployment, capable of handling varying loads, minimizing downtime, and safeguarding sensitive data at each layer.A secure, fault-tolerant, and scalable AWS deployment enabling resilient web application hosting across multiple availability zones.



THANK YOU