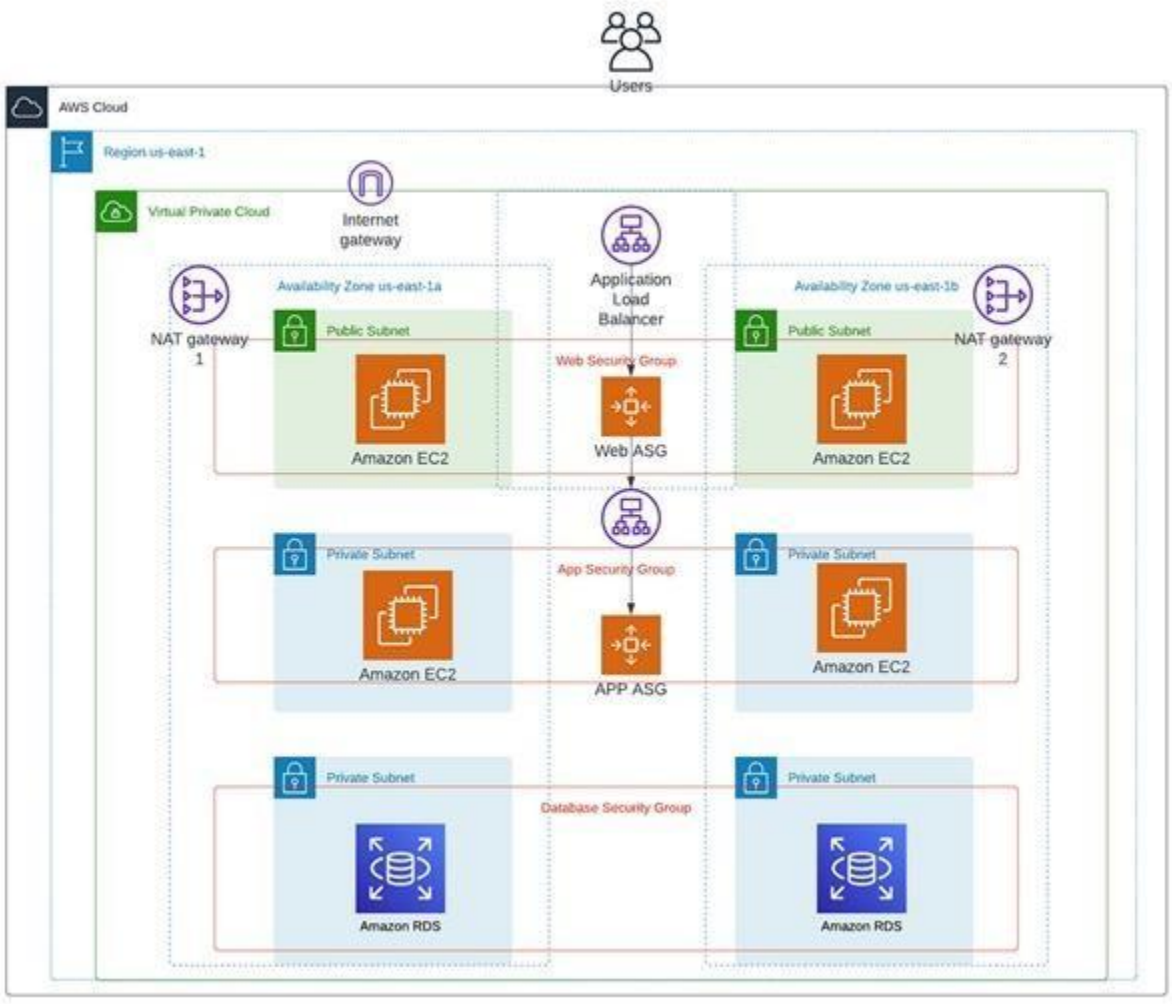
# PROJECT-1

## 3-TIRE ARCHITECTURE

**Presented by:** K. kethana
**Email ID:** kethanakothapali16@gmail.com
**Vcube Software solutions**
**Batch Number:139**

# 3- Tier Architecture




## What is 3-Tier Architecture?

A 3-Tier Architecture in AWS is a modular cloud design that separates an application into three logical and physical layers to improve scalability, security, and manageability.

## 1. Presentation Tier (Web Layer)

- Purpose: User interface
- AWS Components:

- Elastic Load Balancer (ELB) – Distributes traffic
- EC2 instances (Web servers) – Hosts front-end code (HTML, CSS, JS)
- Amazon Route 53 – DNS service
- Subnet: Public Subnet
- Access: Internet-facing

## 2. Application Tier (Logic Layer)

- Purpose: Business logic processing
- AWS Components:
  - EC2 instances – Host backend applications (e.g., Java, Node.js, Python)
  - Auto Scaling Groups – For dynamic scaling
- Subnet: Private Subnet
- Access: Only accessible by Web Tier

## 3. Database Tier (Data Layer)

- Purpose: Data storage and management
- AWS Components:
  - Amazon RDS (e.g., MySQL, PostgreSQL)
  - Amazon Aurora or DynamoDB (NoSQL)
- Subnet: Private Subnet (with no direct internet access)
- Access: Only by Application Tier

## Security Best Practices:

- Use Security Groups to allow only necessary traffic (e.g., Web → App, App → DB).

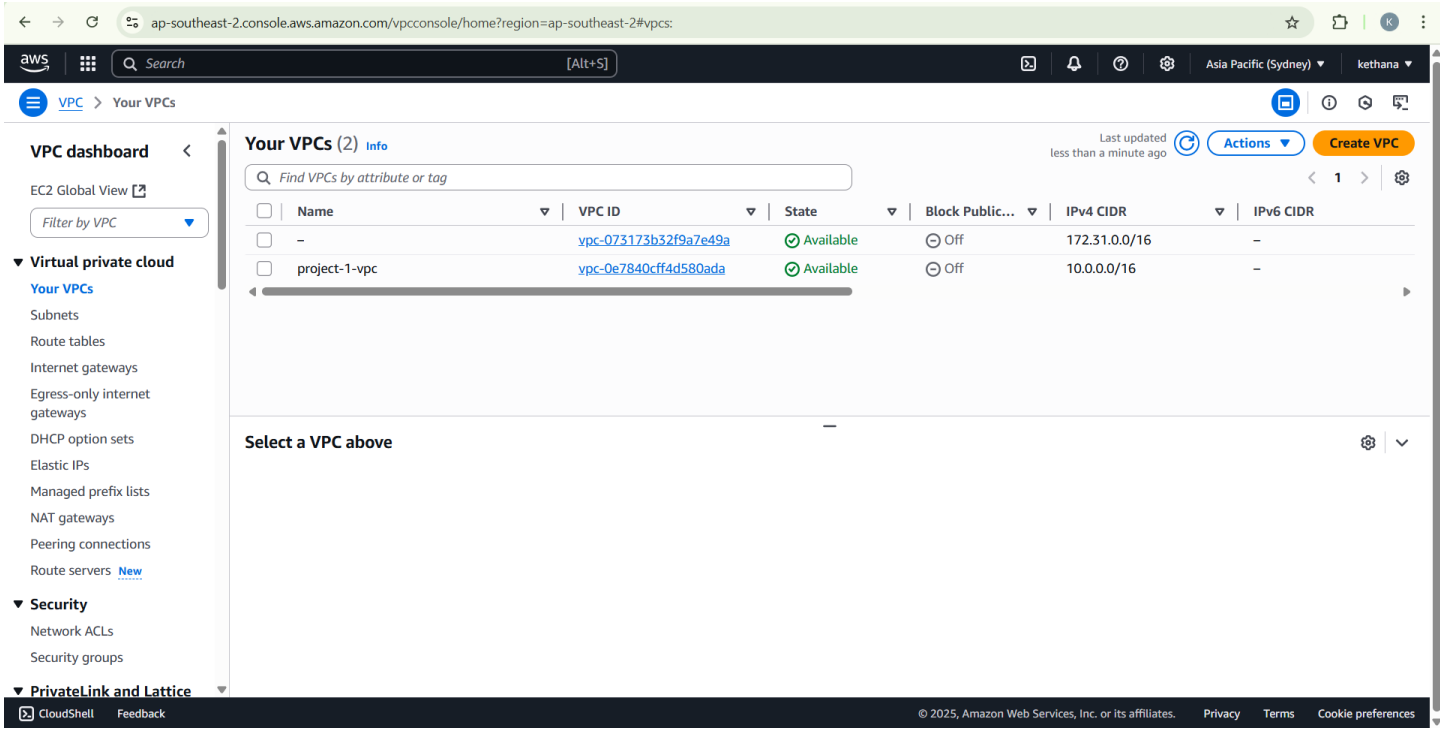
- Use Network ACLs and Subnets to segment the layers properly.
- Database should never be exposed directly to the public internet.

## Open AWS Website and login to your Account

## PROCESS:

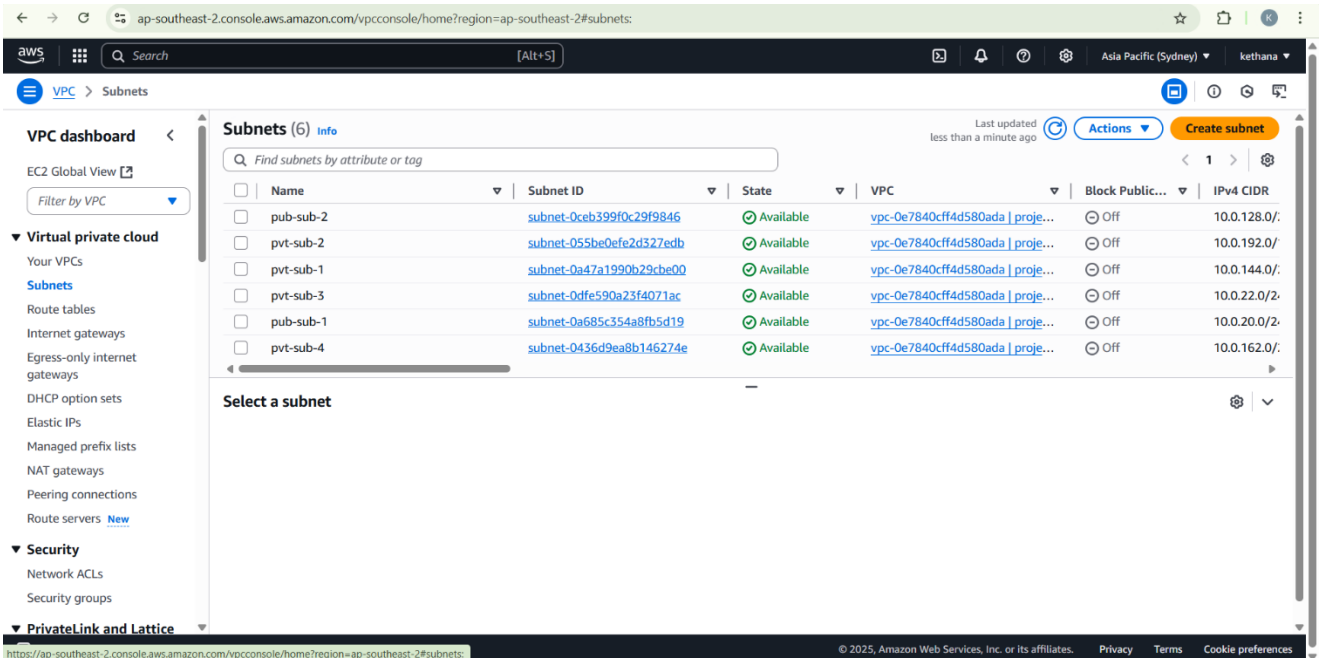
### Step-1: Create VPC

a) Open VPC and Click on "Create VPC".
b) Label the VPC (ex: Project-1-vpc).
c) Specify Ipv4 CIDR block—10.0.0.0/16.
d) Create it.



## 2. Create Subnets:

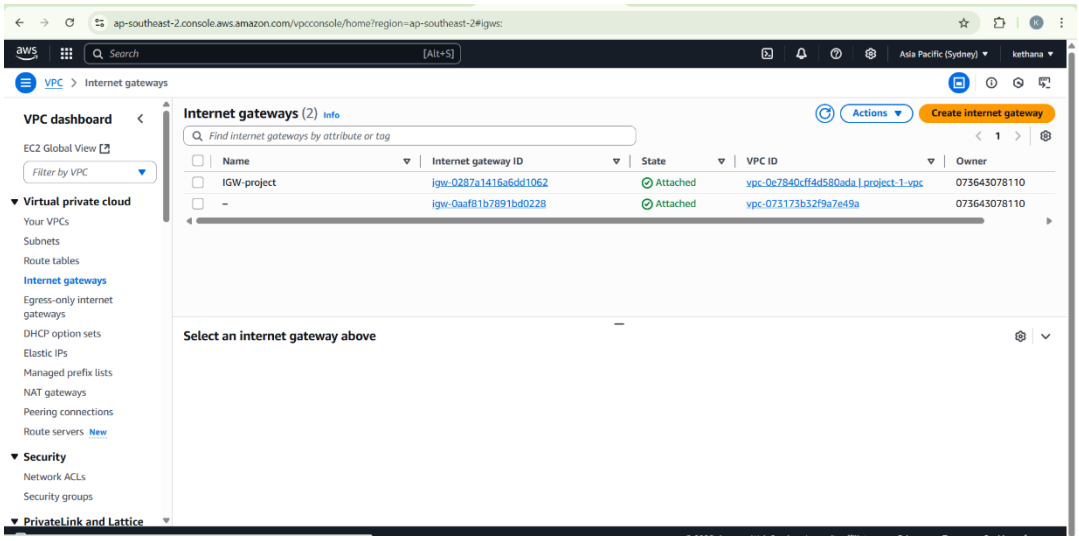
a) Click On Create Subnet.
b) Attach Vpc for subnets.
c) Create Two Public and four Private Subnets.
d) In Public Subnets Choose Different Availability Zones and Different CIDR Block.
e) In Private Subnets Choose Different AZ and CIDR Block IP Like Public.
f) Create it.

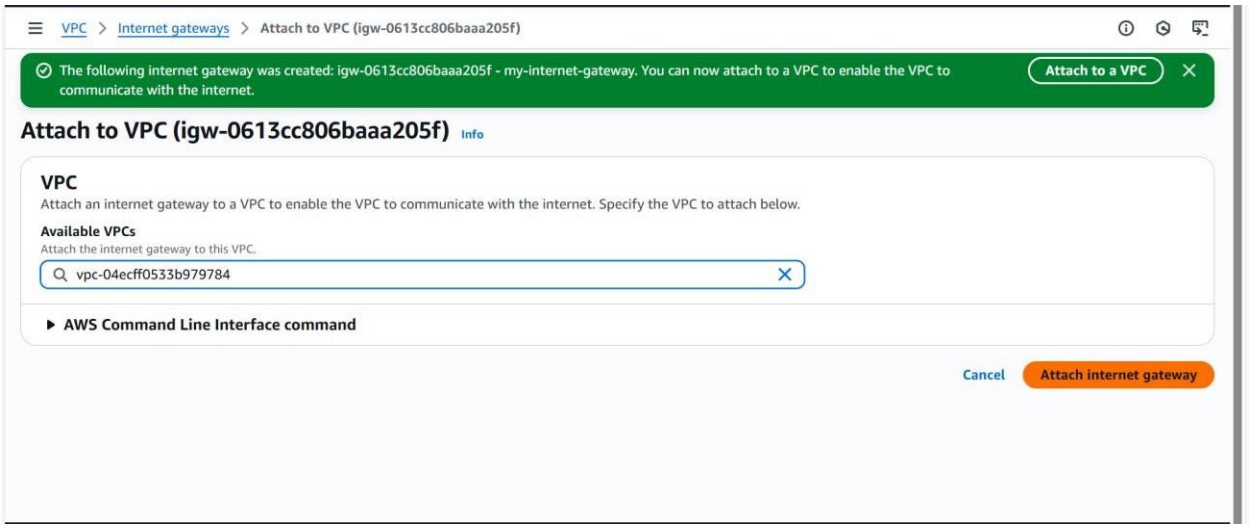



v.

## Step-3: Create Internet Gateway:

a) Select InternetGateway.
b) Attach Vpc to it.

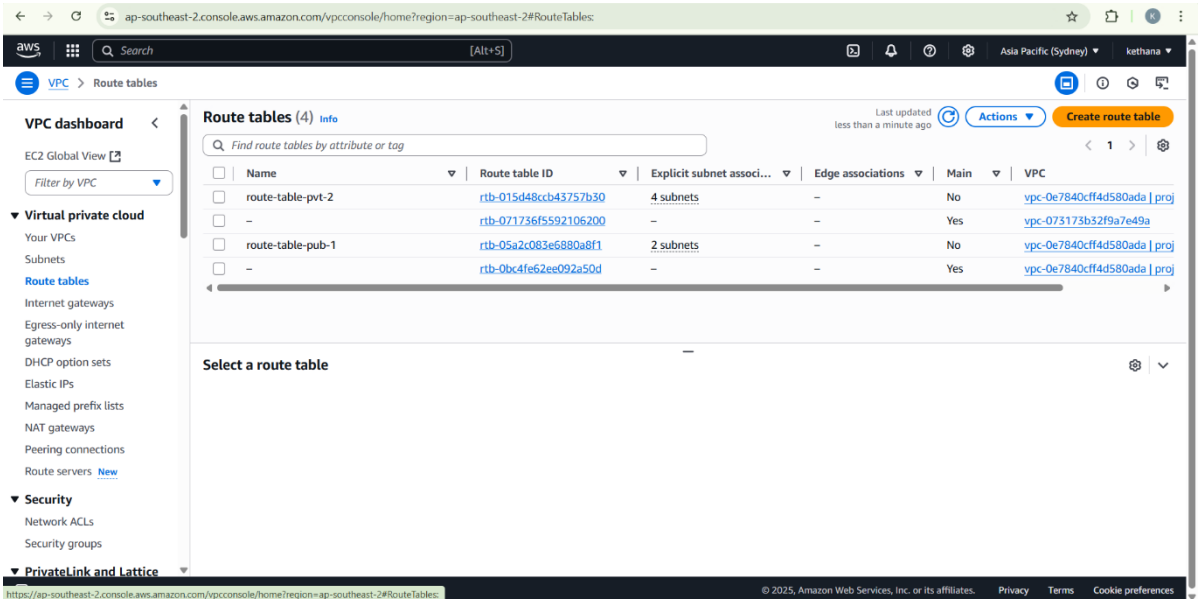



## Step-4: Create Route Tables:

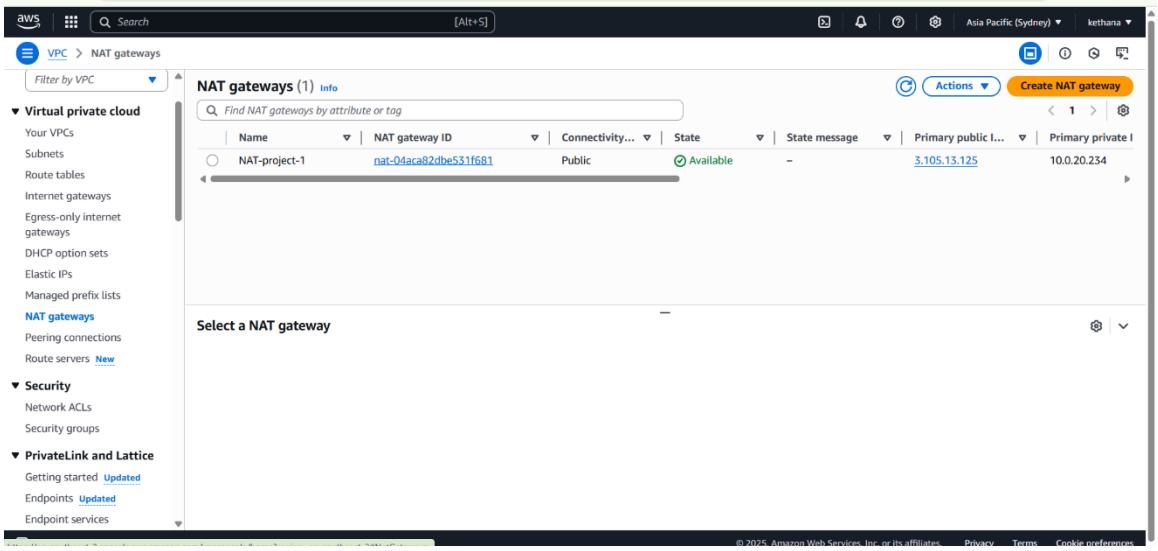
a) Open Route Tables and select main Vpc.
b) Create 2 Route Tables (public and private)

c) Attach 2 public subnets to public Route Table.
d) 4 private subnets to private Route Table.
e) Edit it and Add route (InternetGateway).




## Step-5: Create NAT Gateway

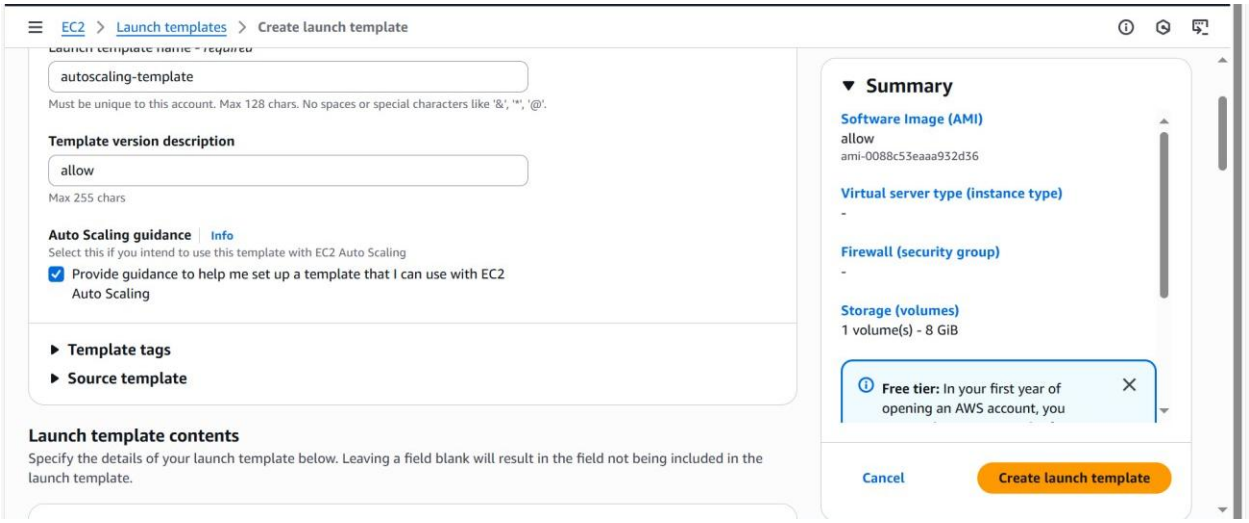
a) Now Open NAT Gateway.
b) Select Public Access.
c) Allocate ElasticIP and Select Subnets (2Public subnets)




## Step-6: Launch EC2 Instances

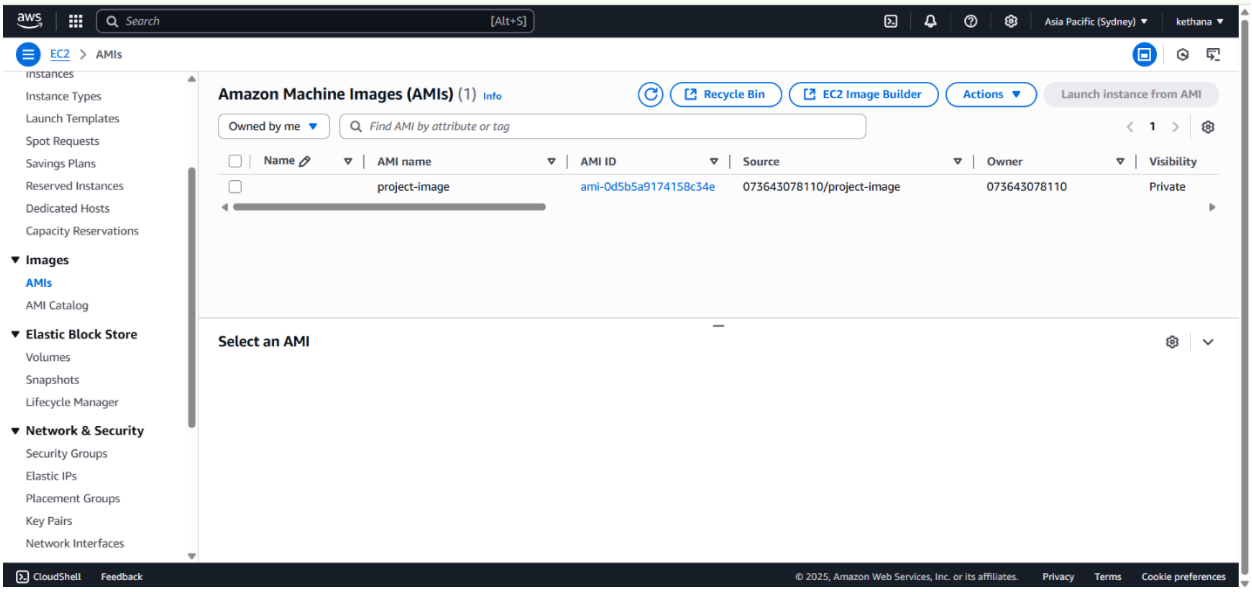
a) Create 2 Public and 2 Private EC2 instances.
b) Select UBUNTU For all Instances.
c) Create Key Pair. (Keypair is same for every Instance).
d) Select VPC – (What We Created)
e) Add Subnet and Add Security Group
f) Launch Instances.
g) Copy the Public Instance SSH link and paste it on Gitbash.
h) By Using Commands, we have to connect to the Server.

## Step 7: Create images in EC2 Instance:



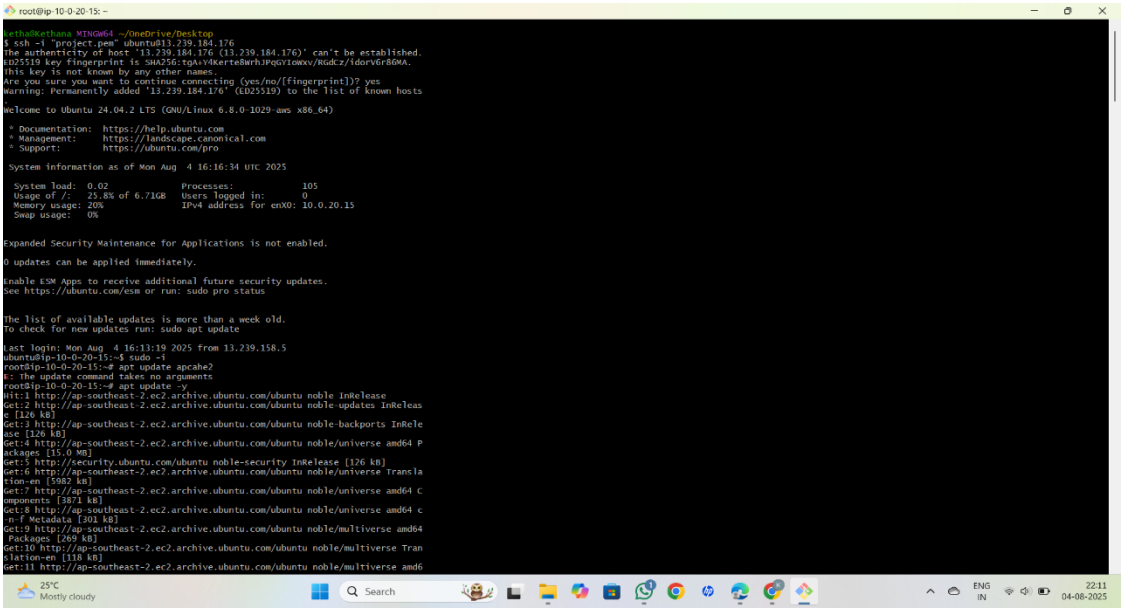

## Create image in Instance:

a) Click on Instance.
b) Go to Actions
c) Click on Image and templates.
d) Give A name to image and save it.
e) Create it.

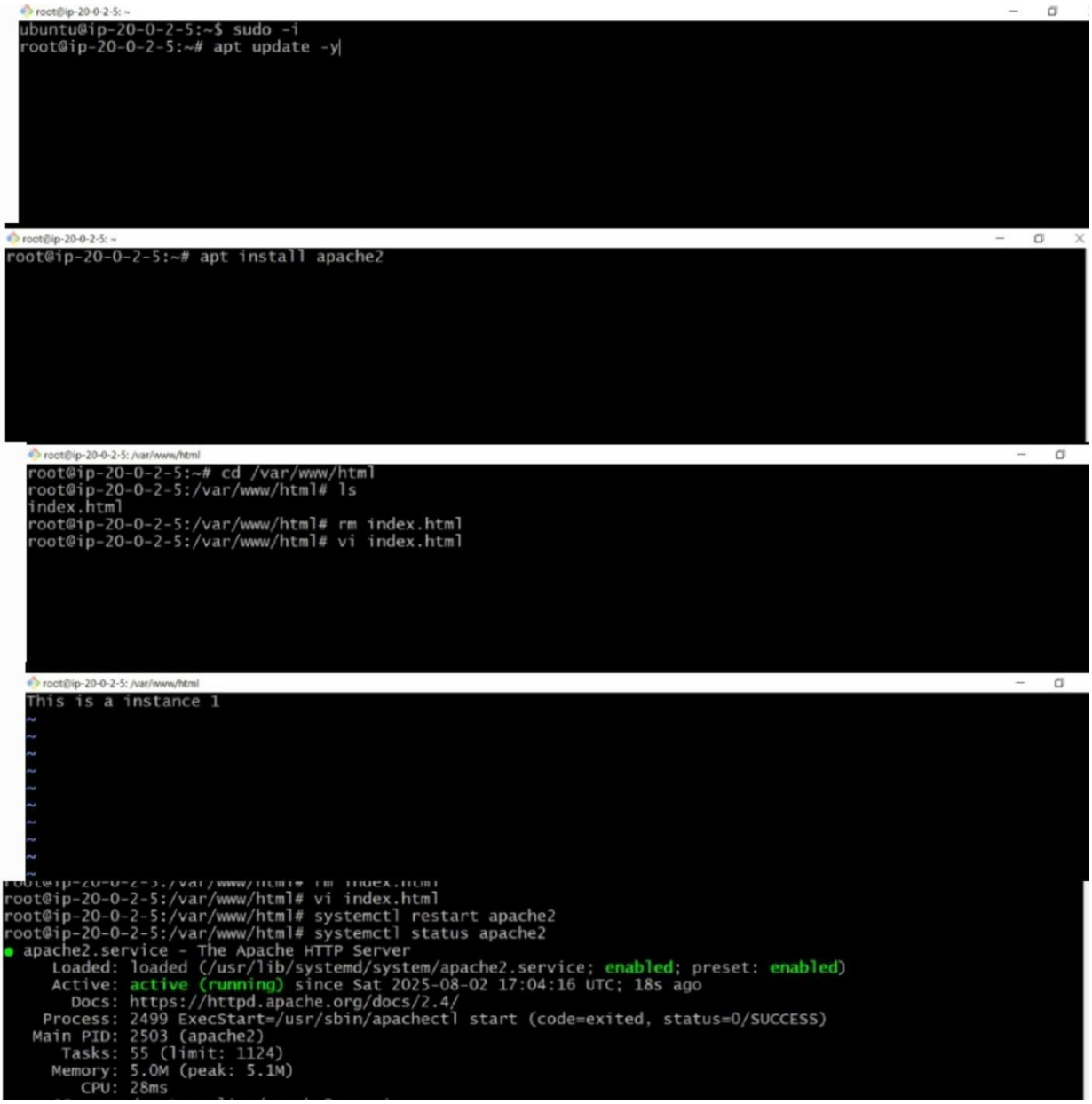


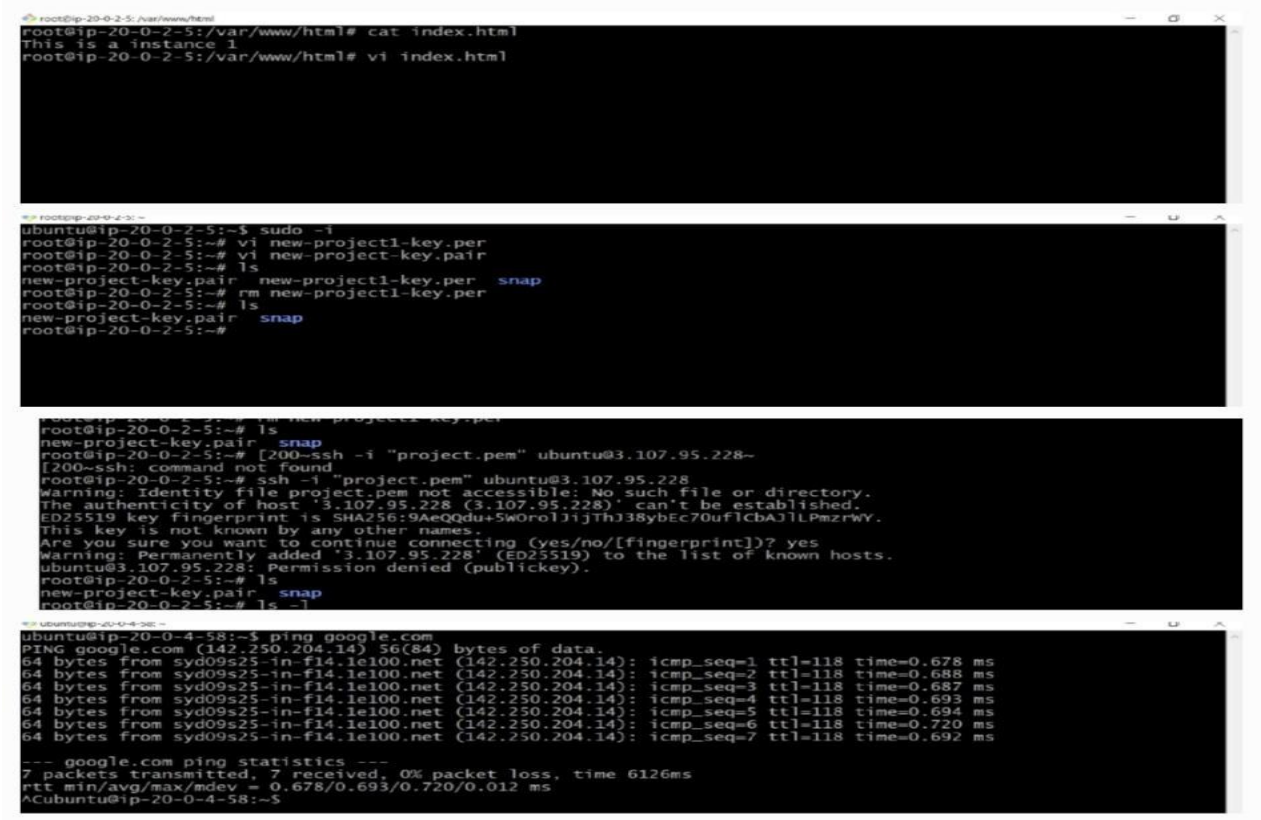

## After Connected to the Server We Have to Commands:

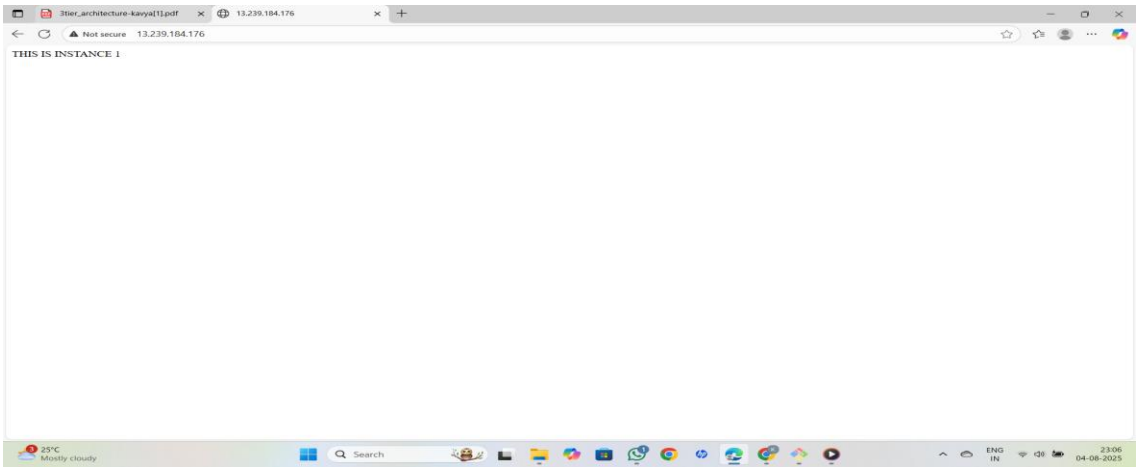
a) Sudo –i
b) apt update apcahe2
c) apt install apache2
d) cd /var/www/html
e) rm index.html
f) vi index.html (insert data)
g) systemctl restart apache2 (to restart the server)
h) systemctl status apache2 (to see the status of server)
i) cat index.html (used to insert the data in it gitbash).



x.


root@ip-20-0-2-5: ~
ubuntu@ip-20-0-2-5:~$ sudo -i
root@ip-20-0-2-5:~# apt update -y
root@ip-20-0-2-5: ~
root@ip-20-0-2-5:~# apt install apache2
root@ip-20-0-2-5: /var/www/html
root@ip-20-0-2-5:~# cd /var/www/html
root@ip-20-0-2-5:/var/www/html# ls
index.html
root@ip-20-0-2-5:/var/www/html# rm index.html
root@ip-20-0-2-5:/var/www/html# vi index.html
root@ip-20-0-2-5: /var/www/html
This is a instance 1
root@ip-20-0-2-5:/var/www/html# vi index.html
root@ip-20-0-2-5:/var/www/html# systemctl restart apache2
root@ip-20-0-2-5:/var/www/html# systemctl status apache2
● apache2.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
Active: active (running) since Sat 2025-08-02 17:04:16 UTC; 18s ago
Docs: https://httpd.apache.org/docs/2.4/
Process: 2499 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 2503 (apache2)
Tasks: 55 (limit: 1124)
Memory: 5.0M (peak: 5.1M)
CPU: 28ms

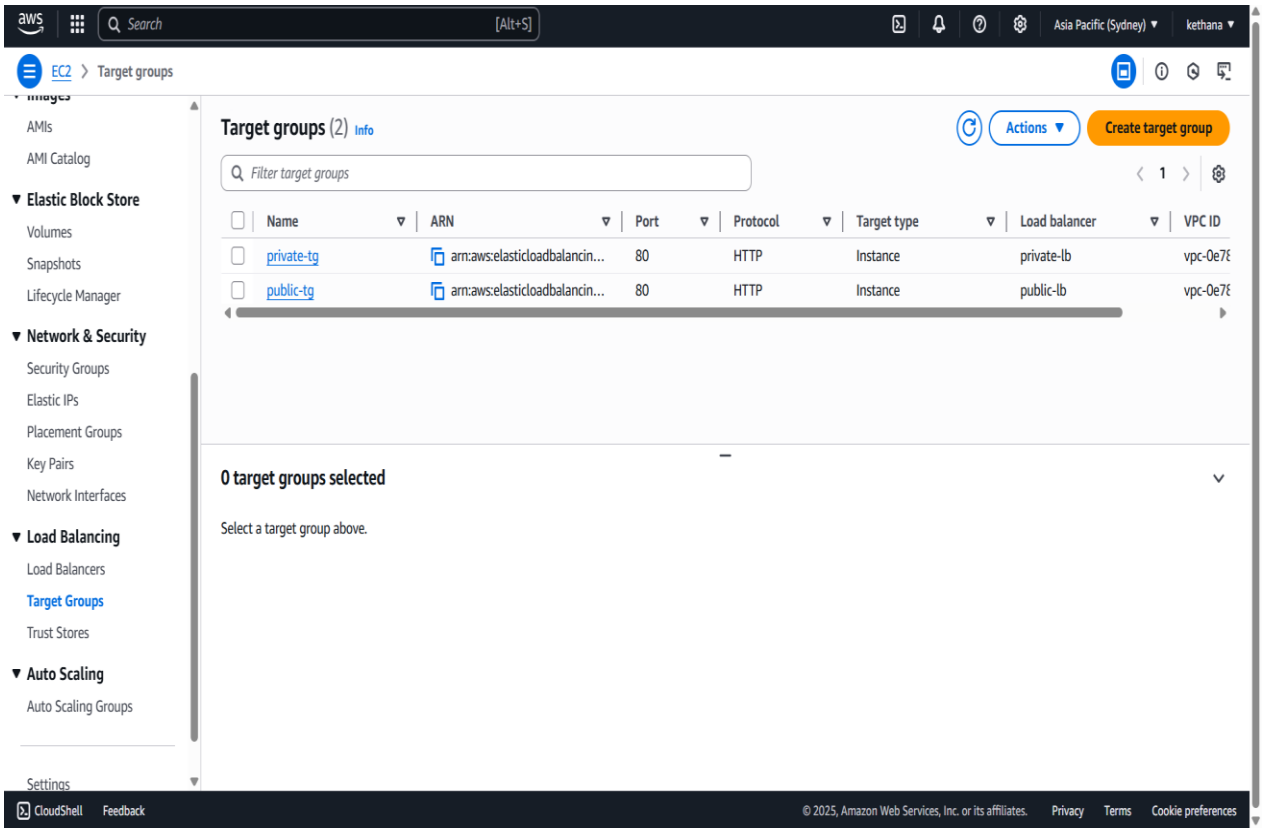



# Step-7: Create TargetGroup:

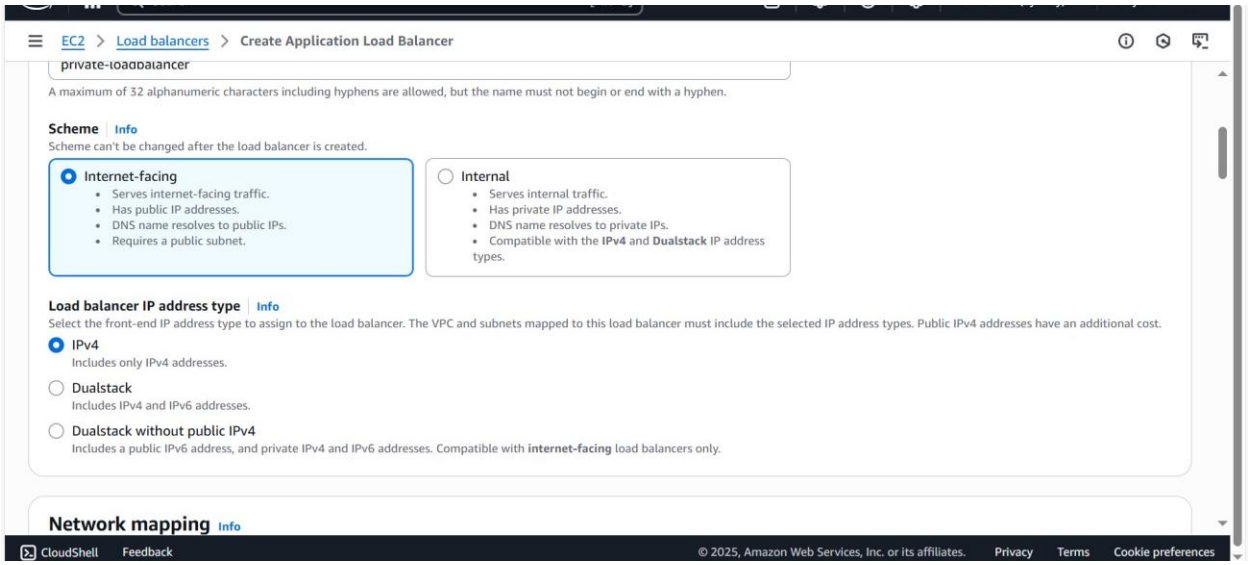
a) Create One Public TargetGroup.
b) Give name as (public)
c) Add Public Instances to Public TG.
d) Create One Private TargetGroup.

e) Give name as (Private)
f) Add Private Instances to Private TG.

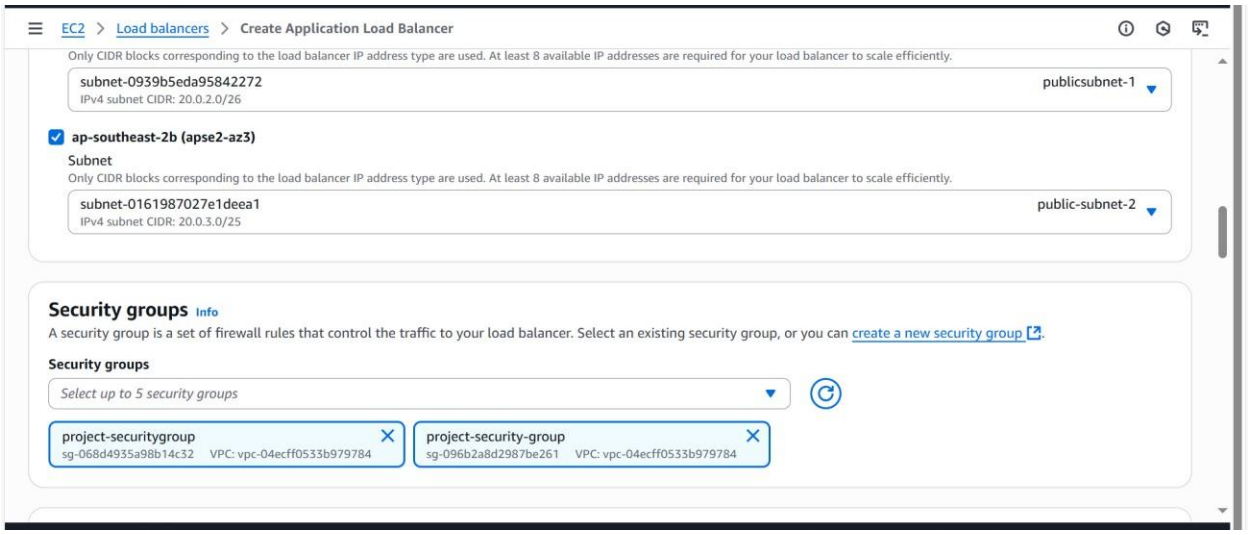



## Step-8: Create LoadBalancer:

a) Open LoadBalancer.
b) We have to create Public and private LoadBalancer.
c) Create Public LoadBalancer.
d) Select Internet Facing.
e) Select Subnets.
f) Select our SecurityGroup. (what we created).
g) Add Target Group (we Created one).
h) Create Private Load Balancer same as Public LB
i) Click On Create LoadBalancer.



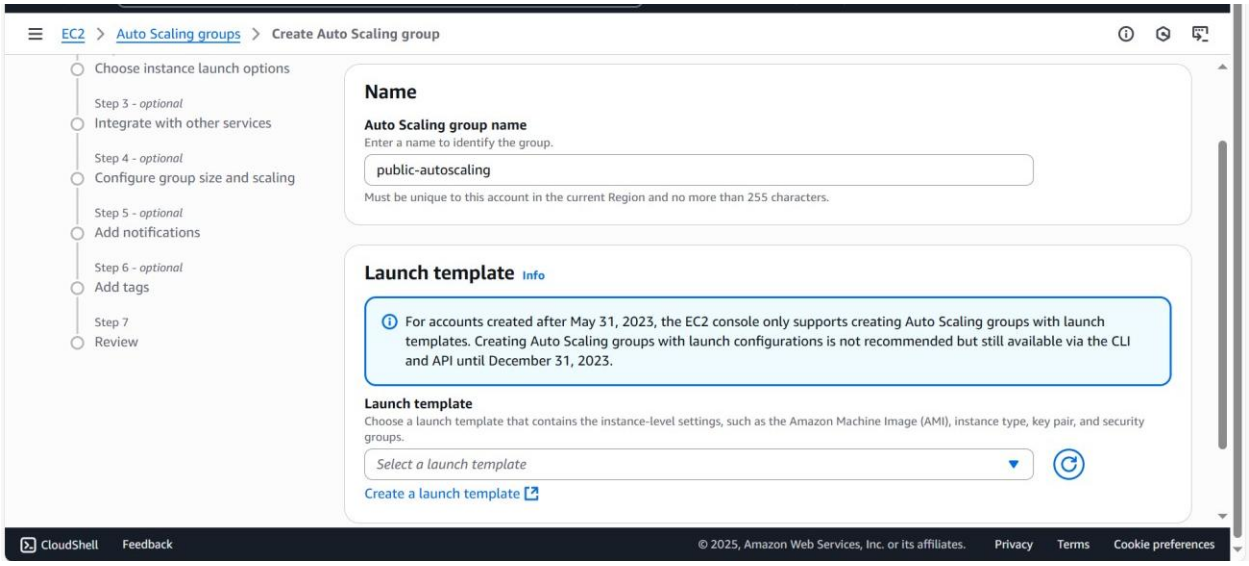

.

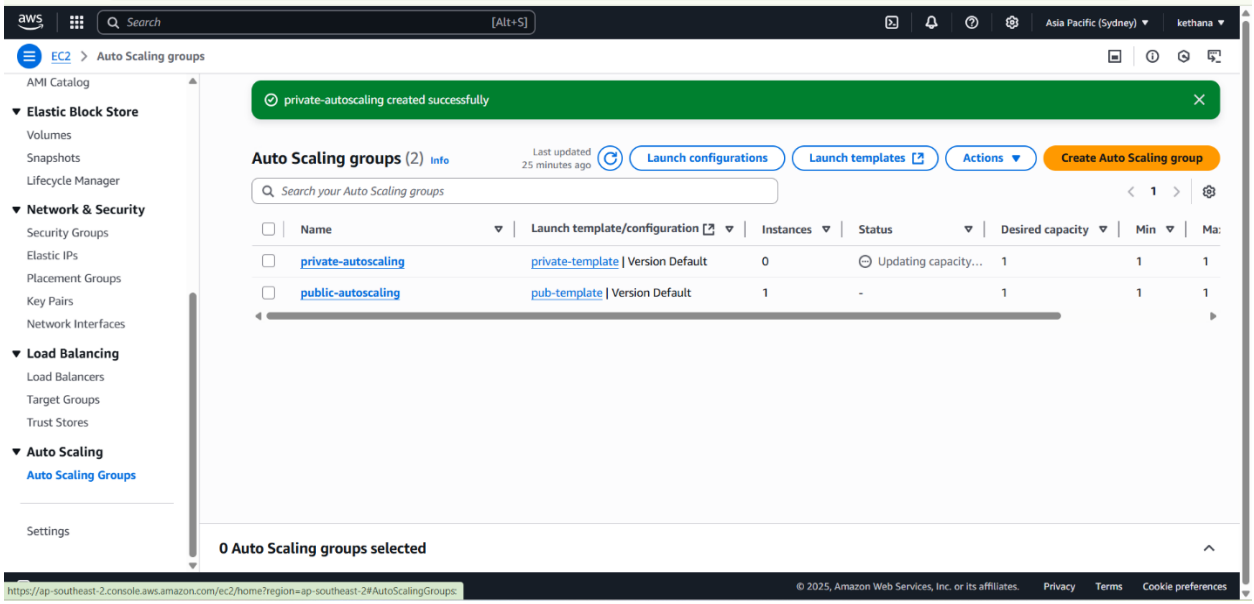



## Step-9: AWS Auto Scaling:

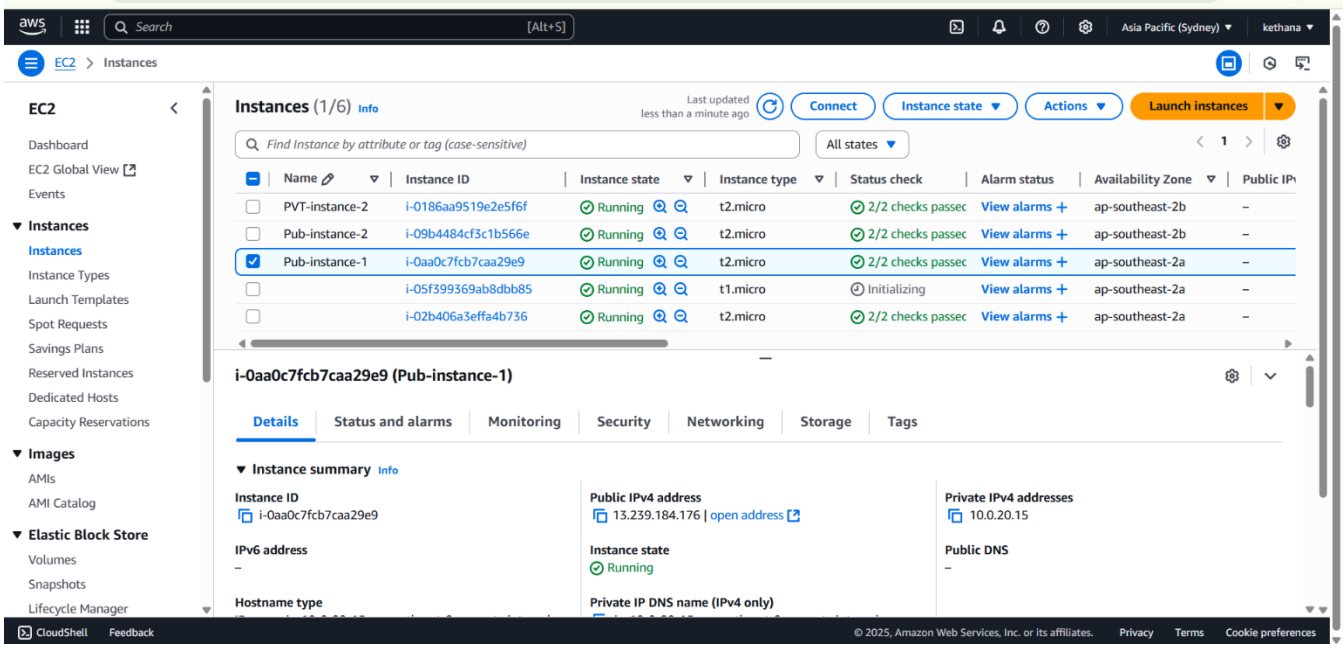
a) Click on Create Auto Scaling.
b) Auto Scaling group.
c) Give Scaling Policies.

d) Give the size (max or min).
e) Attach the load Balancer. (what we created).
f) Create Public and Private AutoScaling.
vii. Add Public Subnet to Public AutoScaling.
g) Add Private Subnet to Private AutoScaling.



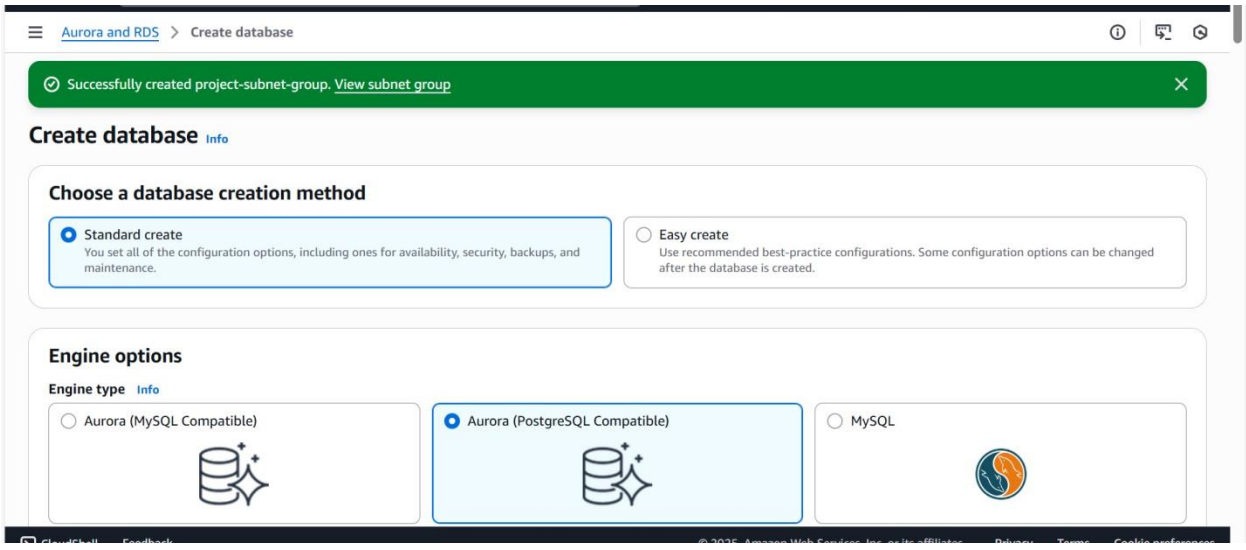


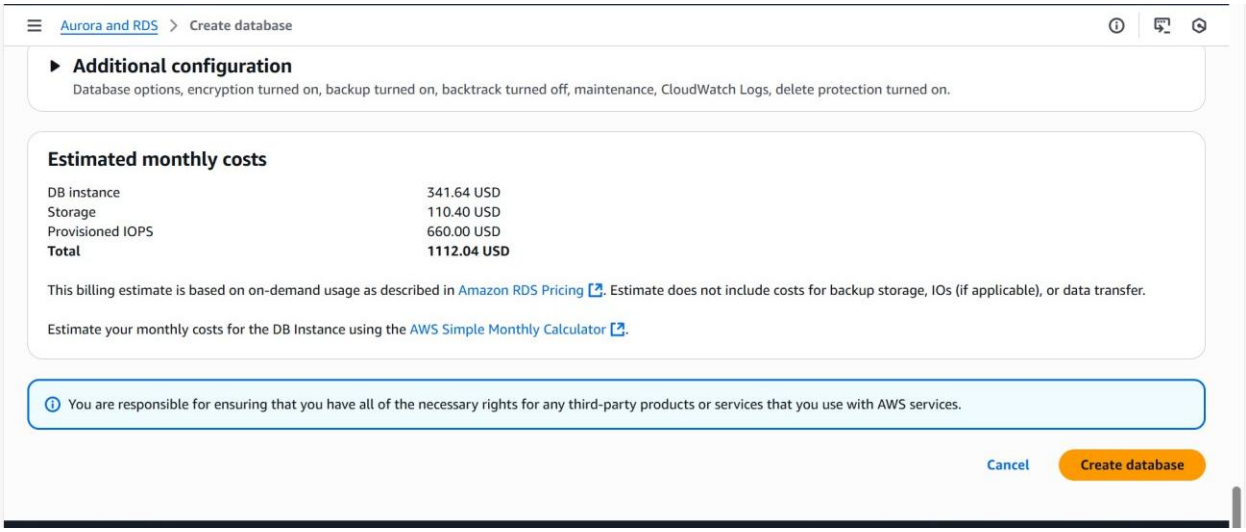

- In EC2 Instance we can see the New Instances Created.
- If we add subnets in auto Scaling. It can create EC2 Instances in EC2.

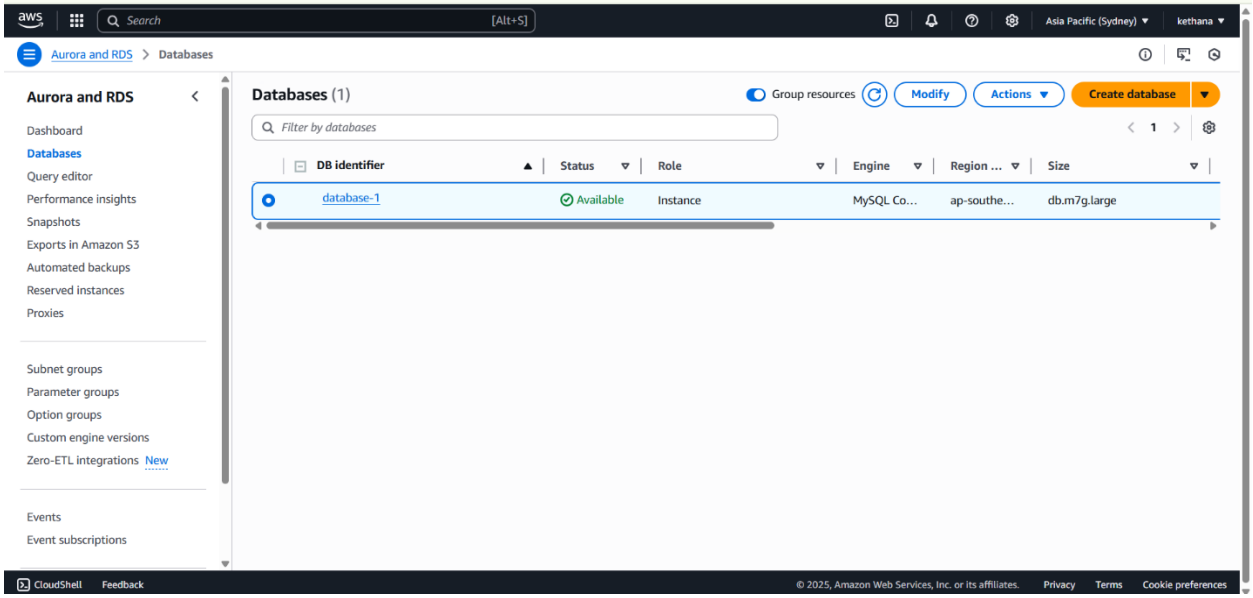


## Step-10: Create Database In RDS

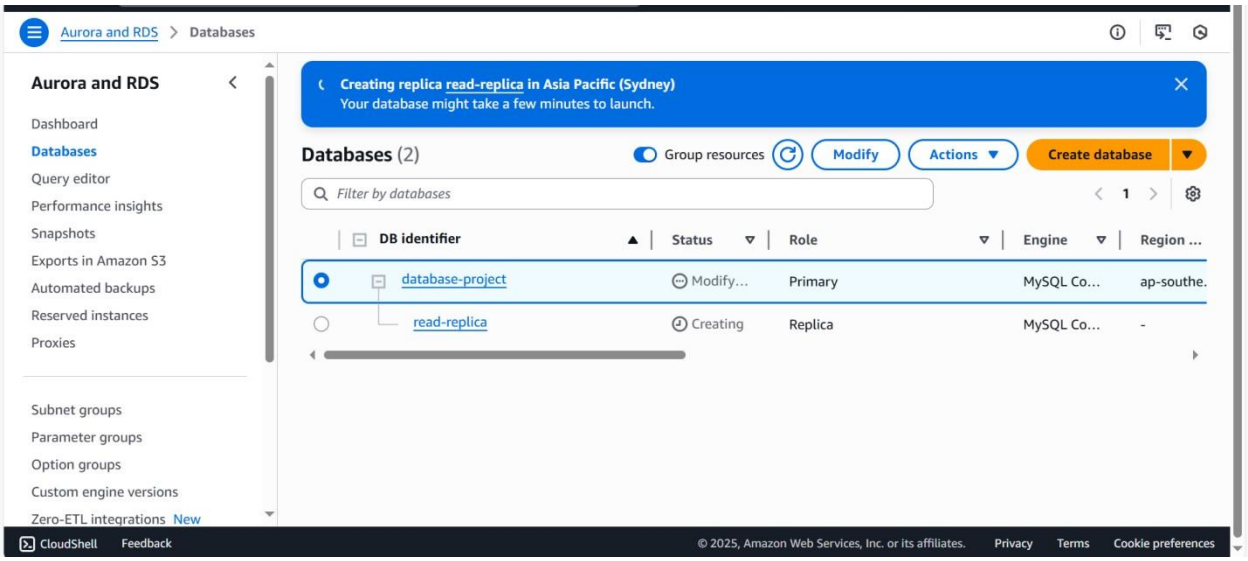
a) Open Arora and RDS.
b) Click On Databases.
c) Choose Engine as: MySQL.
d) In Credentials Management choose myself.
e) Give new password to it.
f) It shows monthly Estimated.





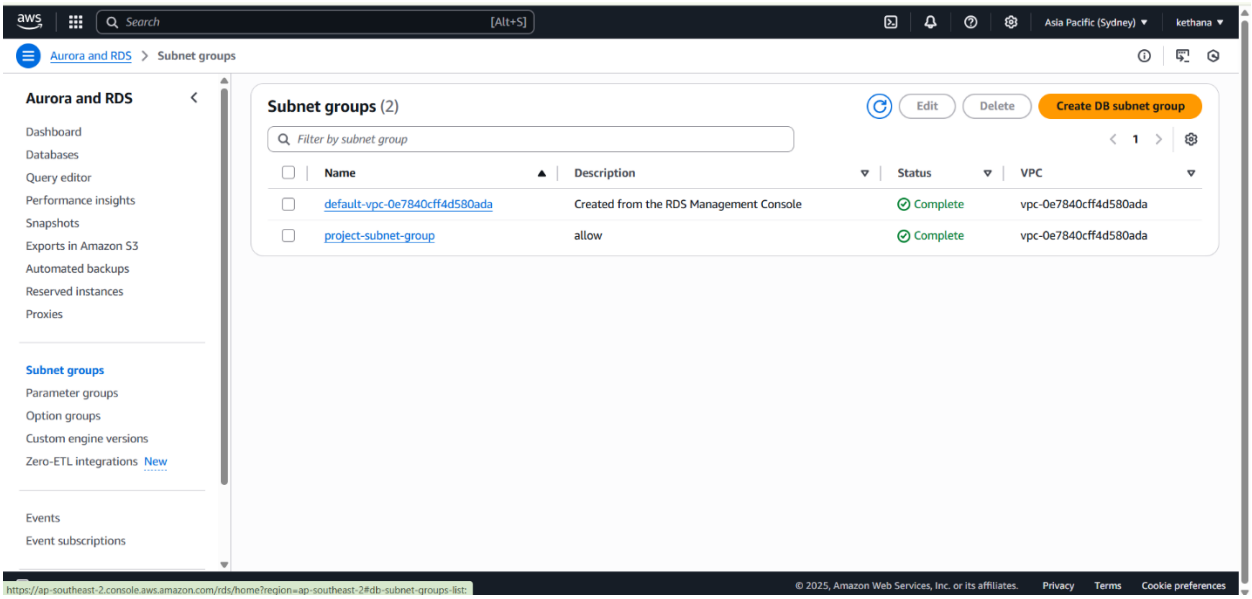




# Add Read Replica to Database:




## Step-11: Create subnet in RDS:

a) Open rds and Create a Subnets Group

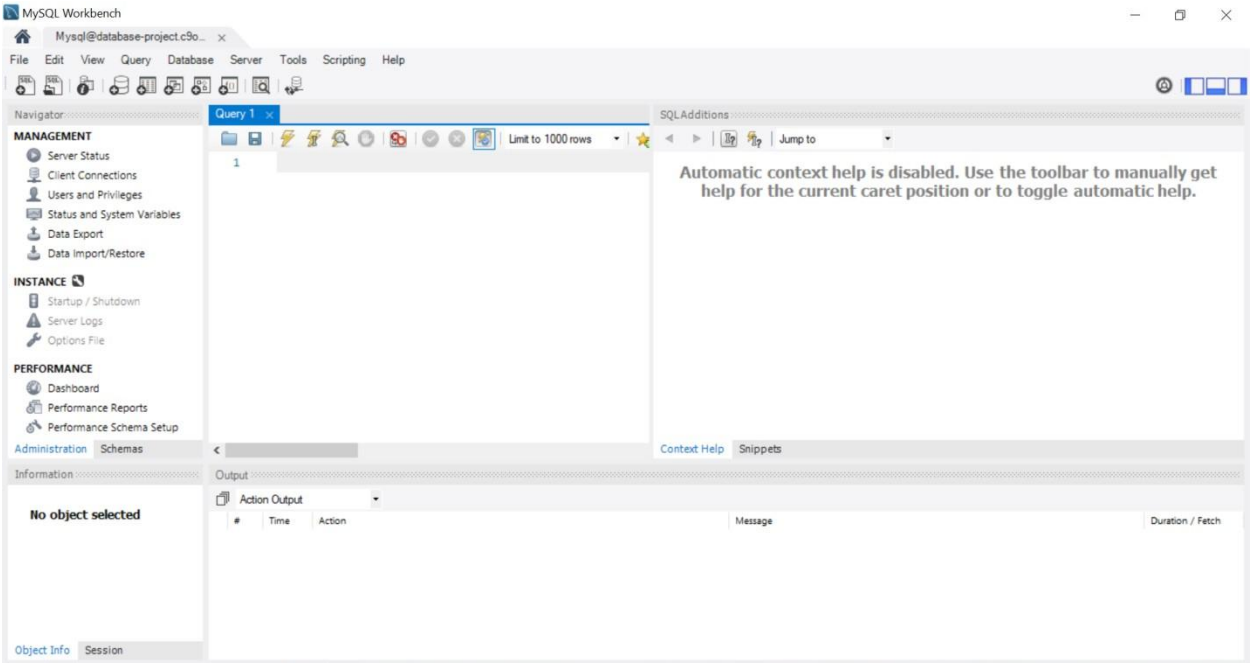


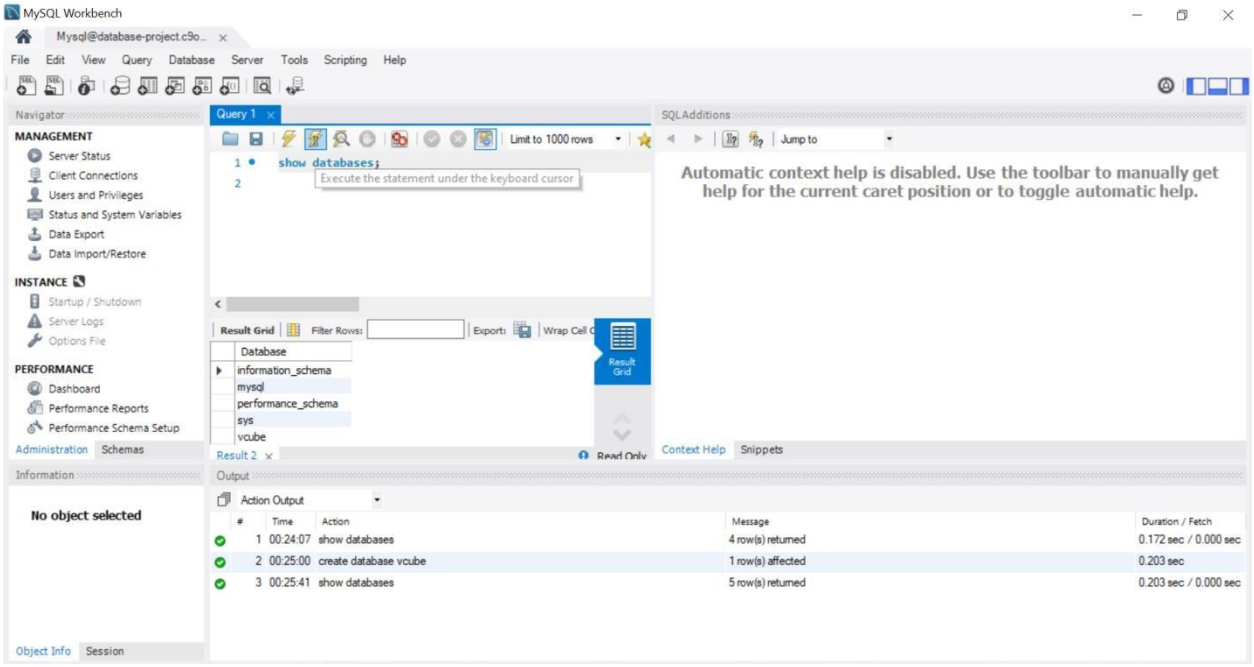

## Step-12: Create MySQL:

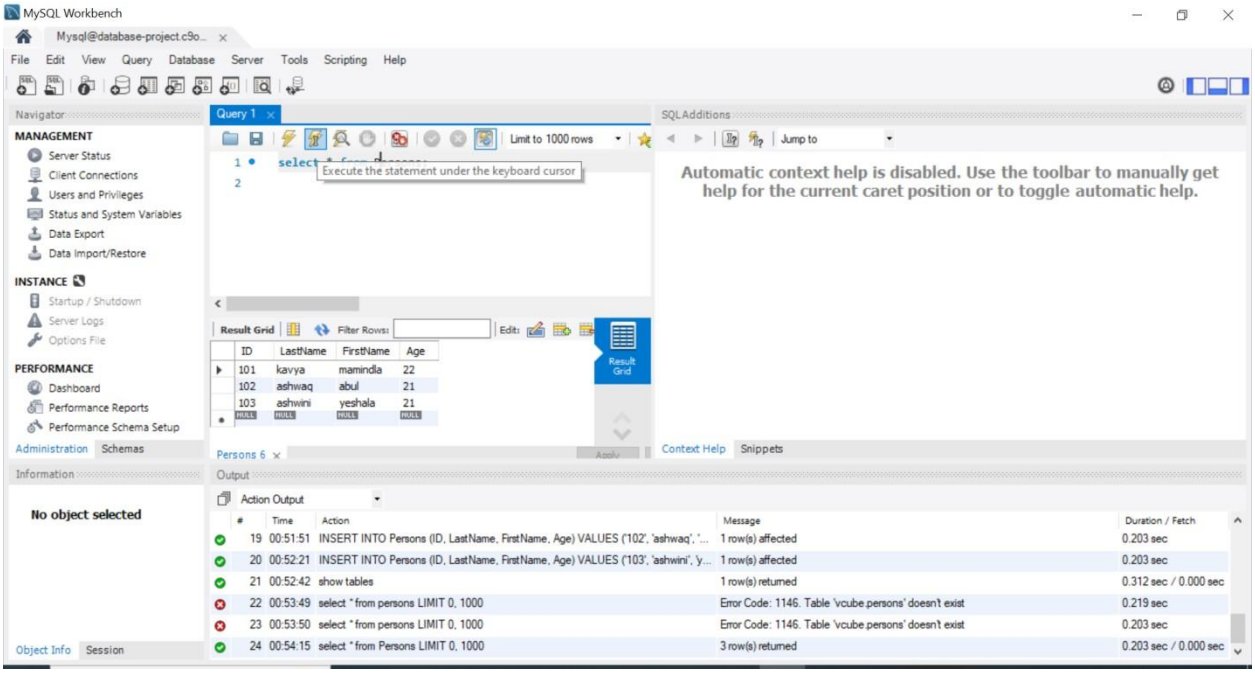
a) Open MySQL Workbench
b) In MySQL open Database.
c) Give host name and Password to it.
d) TO Create Table and Insert Data.

BY Using Commands, we have to insert Data.

b) show databases;
c) 2.create database name;
d) 3.CREATE TABLE Persons.
e) LastName varchar (255) NOT NULL;
f) FirstName varchar (255);
g) Age int;
h) PRIMARY KEY (ID)
i) 4. show tables;
j) 5.INSERT INTO Persons (ID, LastName, FirstName, Age)
k) 6. select * from Persons




MySQL Workbench
Mysql@database-project.c9o...
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 show databases;
2
Execute the statement under the keyboard cursor
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid Filter Rows: Export: Wrap Cell
Database
information_schema
mysql
performance_schema
sys
vcube
Result Grid
Result 2
Read Only
Context Help Snippets
Output
Action Output
# Time Action Message Duration / Fetch
1 00:24:07 show databases 4 row(s) returned 0.172 sec / 0.000 sec
2 00:25:00 create database vcube 1 row(s) affected 0.203 sec
3 00:25:41 show databases 5 row(s) returned 0.203 sec / 0.000 sec


MySQL Workbench
Mysql@database-project.c9o...
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 select * from persons;
2
Execute the statement under the keyboard cursor
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid Filter Rows: Edit:
ID LastName FirstName Age
101 kavya mamindla 22
102 ashwaq abul 21
103 ashwini yeshala 21
Result Grid
Persons 6
Context Help Snippets
Output
Action Output
# Time Action Message Duration / Fetch
19 00:51:51 INSERT INTO Persons (ID, LastName, FirstName, Age) VALUES ('102', 'ashwaq', '... 1 row(s) affected 0.203 sec
20 00:52:21 INSERT INTO Persons (ID, LastName, FirstName, Age) VALUES ('103', 'ashwini', 'y... 1 row(s) affected 0.203 sec
21 00:52:42 show tables 1 row(s) returned 0.312 sec / 0.000 sec
22 00:53:49 select * from persons LIMIT 0, 1000 Error Code: 1146. Table 'vcube.persons' doesn't exist 0.219 sec
23 00:53:50 select * from persons LIMIT 0, 1000 Error Code: 1146. Table 'vcube.persons' doesn't exist 0.203 sec
24 00:54:15 select * from Persons LIMIT 0, 1000 3 row(s) returned 0.203 sec / 0.000 sec

# CONCLUSION

In this project, we successfully designed and deployed a **3-tier architecture** on **Amazon Web Services (AWS)**, comprising the **Presentation Layer (Web Tier)**, **Application Layer (App Tier)**, and **Database Layer (DB Tier)**. This architecture enhances **security**, **scalability**, and **manageability** of applications.

- Implemented **VPC**, **public and private subnets**, ensuring network isolation.
- Configured **EC2 instances** for the web and application tiers.
- Deployed a **MySQL RDS instance** in the private subnet for secure data storage.
- Utilized **Security Groups** to manage fine-grained access control between layers.
- Ensured high availability and scalability using **Auto Scaling Groups** and **Elastic Load Balancer (ELB)**.

This architecture follows industry best practices for building cloud-native applications and can be scaled further to meet growing business needs. It demonstrates how AWS services can be combined effectively to create a secure, modular, and reliable infrastructure.