Two-Tier AWS Web Application with Private Database

1. Project Overview

This project demonstrates a secure two-tier architecture on AWS where:

- **A public web server hosts a website and backend service.**
- **A private database server hosts MariaDB with no public access.**
- **The database can still access the internet only for updates and patching using a NAT Gateway.**

The project focuses on real-world AWS networking, security, troubleshooting, and cost awareness.

---

2. Architecture Summary

High-Level Design

- **VPC with a custom IPv4 CIDR**
- **Public Subnet**
  - **Web EC2 instance**
  - **Internet Gateway attached to VPC**
- **Private Subnet**
  - **DB EC2 instance (MariaDB)**
  - **No public IP**
- **NAT Gateway (Public, Regional)**
  - **Provides outbound internet access to private subnet**
- **Security Groups**
  - **Web SG: allows HTTP (80) and SSH (22 from admin IP)**
  - **DB SG: allows MySQL (3306) and SSH only from Web SG**

Traffic Flow

- **User → Web EC2 (HTTP)**
- **Web EC2 → DB EC2 (MySQL via private IP)**
- **DB EC2 → Internet (updates via NAT Gateway)**

---

3. AWS Resources Used

Resource	Purpose
VPC	Network isolation
Public Subnet	Hosts web server
Private Subnet	Hosts database server
Internet Gateway	Internet access for public subnet
NAT Gateway (Regional, Public)	Outbound internet for private subnet
EC2 (Web)	Apache + Node.js backend
EC2 (DB)	MariaDB server
Security Groups	Traffic control
Route Tables	Traffic routing

---

4. Step-by-Step Implementation

4.1 VPC and Networking Setup

- **Created a VPC with CIDR block (example: `10.0.0.0/16`).**
- **Created:**
  - `public-subnet`
  - `private-subnet`
- **Attached Internet Gateway to the VPC.**

Route Tables

- **Public Route Table:**
  - `0.0.0.0/0 → Internet Gateway`
- **Private Route Table:**
  - `10.0.0.0/16 → local`
  - `0.0.0.0/0 → NAT Gateway`

4.2 Web Server EC2 (Public)

Configuration:

- **Amazon Linux 2023**
- **Public IP enabled**
- **Security Group allows:**
  - **HTTP (80) from `0.0.0.0/0`**
  - **SSH (22) from admin IP**

Installed Services:

sudo dnf install httpd nodejs -y

sudo systemctl start httpd

sudo systemctl enable httpd

Website Deployment:

- **Copied frontend files to:**

/var/www/html/

- **Verified website access via:**

http://<Web-EC2-Public-IP>

4.3 Private Database EC2

Configuration:

- **Amazon Linux 2023**
- **No public IP**
- **Placed in private subnet**
- **Security Group allows:**
  - **MySQL (3306) only from Web SG**
  - **SSH (22) only from Web SG**

Access Method:

- **SSH hop:**

Laptop → Web EC2 → DB EC2

4.4 NAT Gateway Setup

Reason:
Private subnet cannot access the internet directly. NAT Gateway enables outbound-only access.

Configuration:

- **NAT Gateway type: Public**
- **Availability mode: Regional**
- **Elastic IP: Automatic allocation**

Troubleshooting:

- **Encountered `Blackhole` route status**
- **Resolved by re-adding the `0.0.0.0/0 → NAT Gateway` route after NAT became `Available`**

4.5 MariaDB Installation and Configuration

After NAT was active:

sudo dnf install mariadb105-server -y

sudo systemctl start mariadb

sudo systemctl enable mariadb

Database Security:

sudo mysql_secure_installation

Database and User Creation:

CREATE DATABASE lokha_db;

CREATE USER 'lokha_user'@'%' IDENTIFIED BY 'StrongPassword';

GRANT ALL PRIVILEGES ON lokha_db.* TO 'lokha_user'@'%';

FLUSH PRIVILEGES;

4.6 Web → DB Connectivity Test

From Web EC2:

mysql -h <DB-PRIVATE-IP> -u lokha_user -p

Result:

- **Successful connection**
- **Confirms:**
    - **DB has no public access**

- Only web server can connect

4.7 Node.js Backend Integration

- Installed dependencies using:

npm install

- Backend started with:

node server.js

Issue Faced:

- Frontend showed `connection lost`

Root Cause:

- Frontend JavaScript was calling:

http://localhost:3000

- `localhost` in browser refers to user's PC, not EC2

Fix:

- Updated frontend API URL to:

http://<Web-EC2-Public-IP>:3000

---

5. Common Issues & Troubleshooting

Issue	Cause	Resolution

`dnf install` timeout	No internet in private subnet	Added NAT Gateway
SSH permission denied	Missing key on web EC2	Converted `.ppk` to `.pem`
MariaDB access denied	User/host mismatch or password	Recreated DB user
`Cannot GET /`	No root route in Node.js	Expected behavior
`connection lost`	Frontend using localhost	Used EC2 public IP

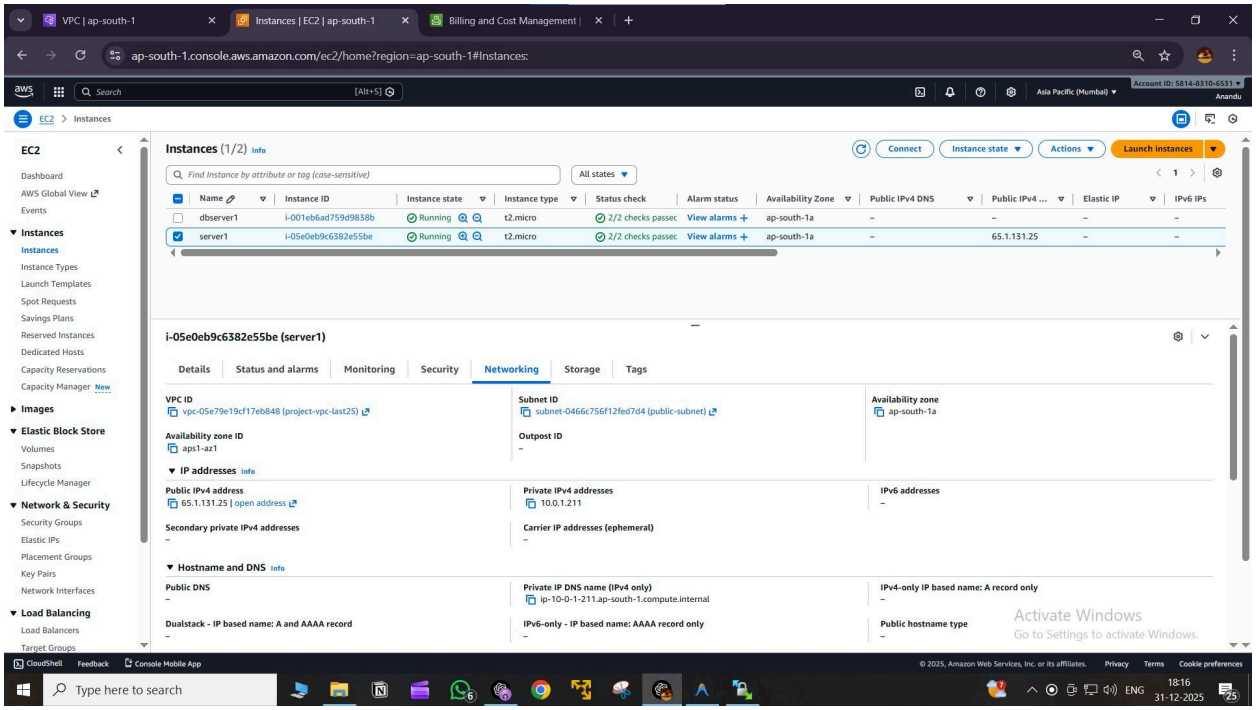
---

6. Security Best Practices Followed

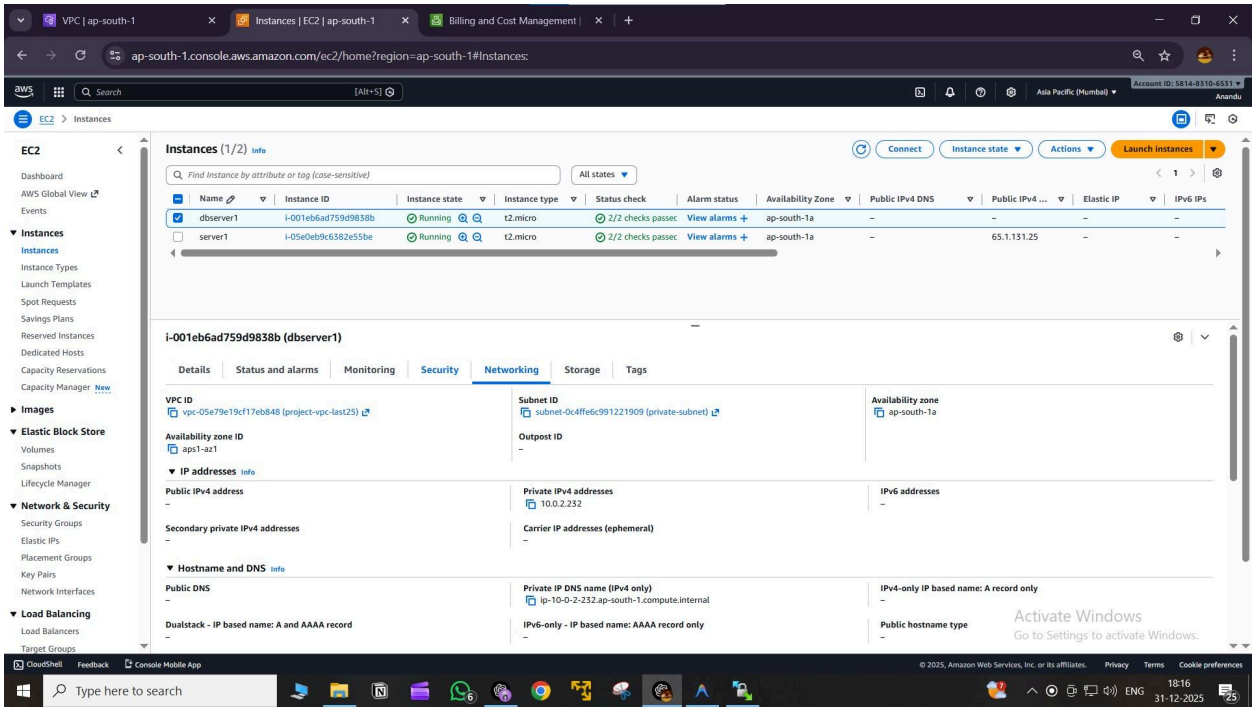
- Database has no public IP
- Security group references instead of CIDR for DB access
- NAT Gateway used instead of public DB exposure
- Root DB login disabled remotely
- Separate DB user for application

---

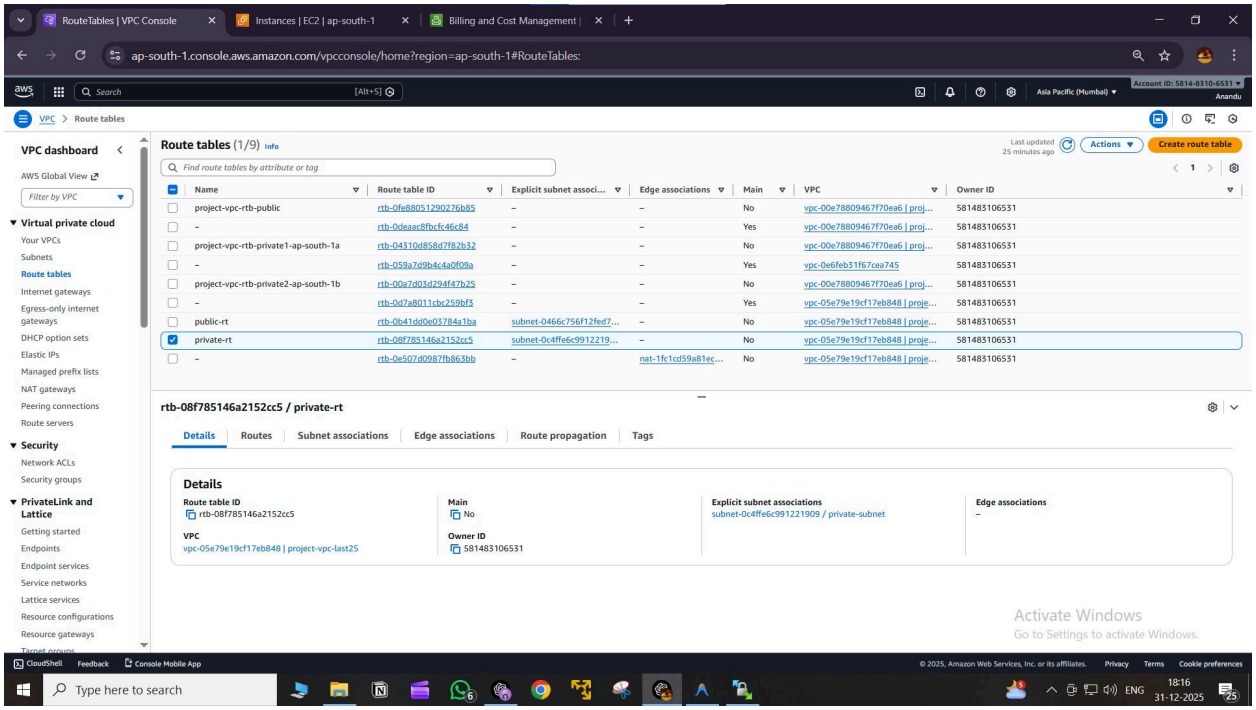
7. Screenshots



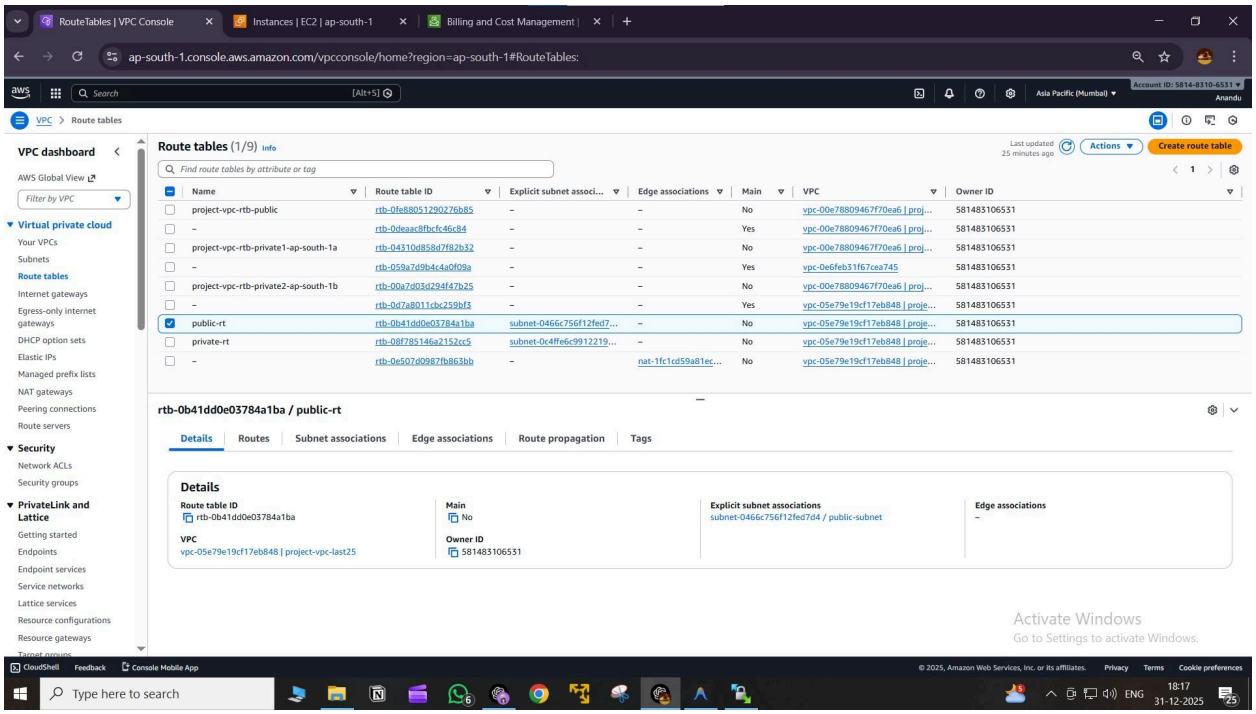
1.server1



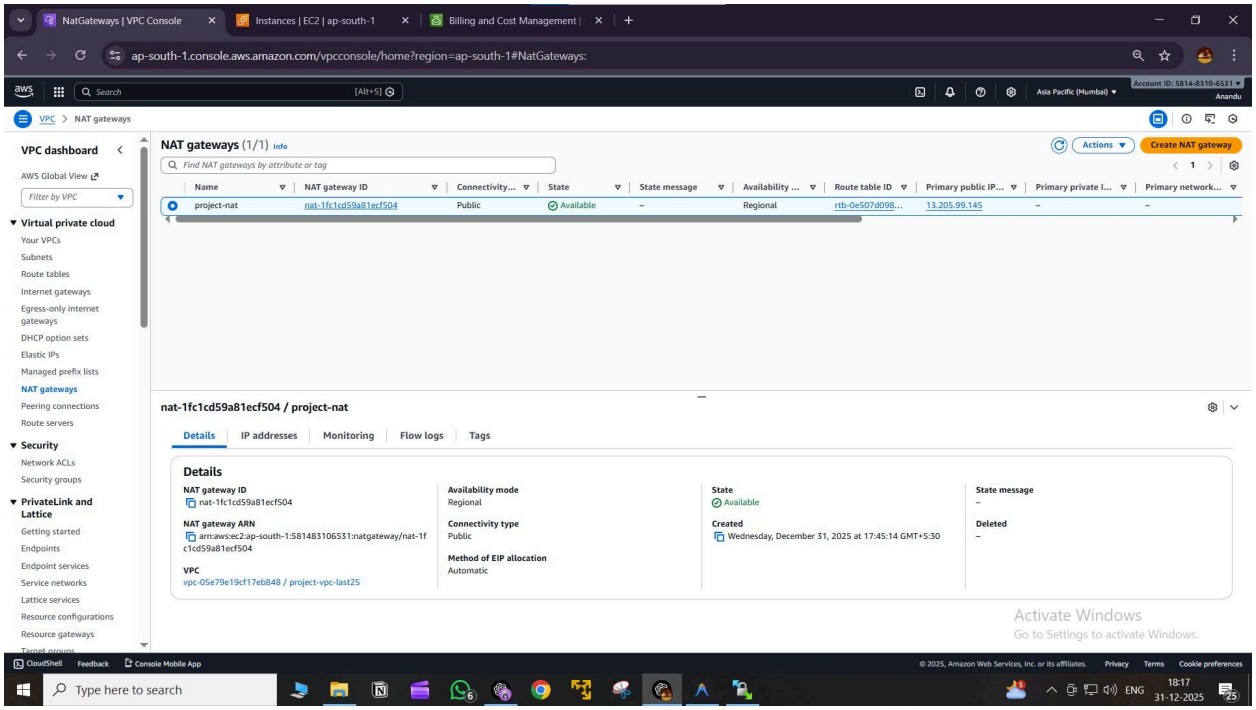
2.database server



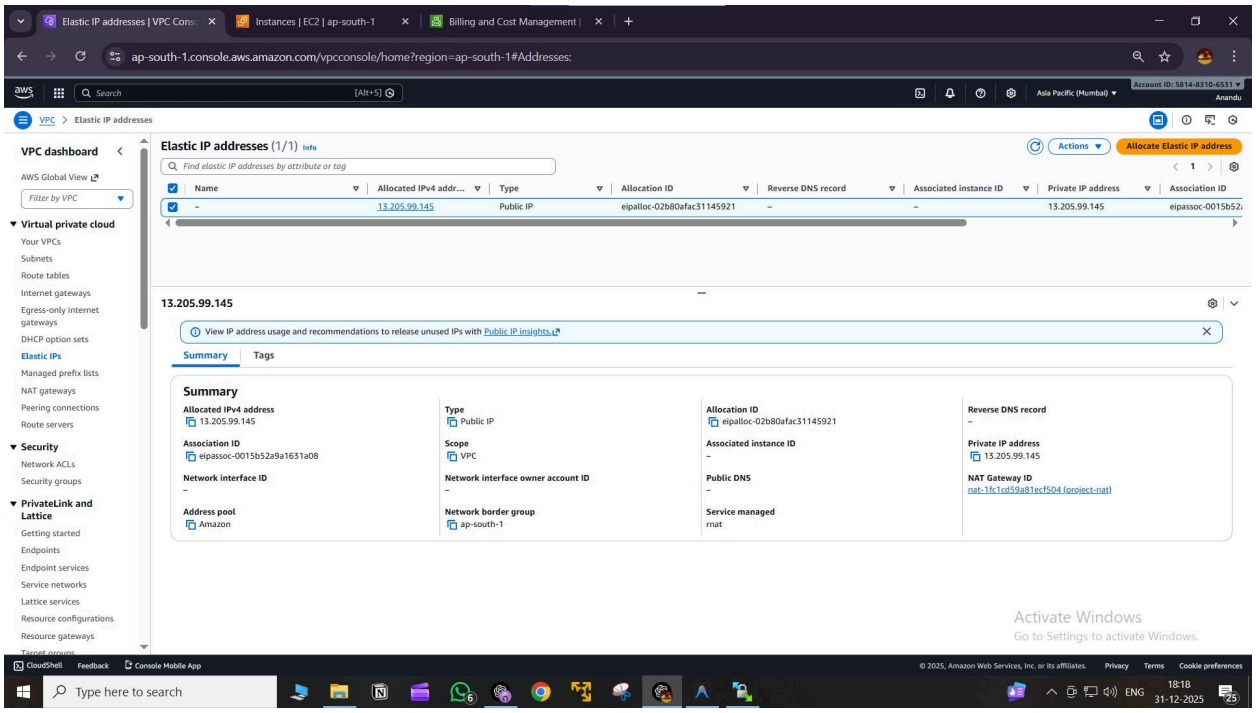
3.Routetable-private



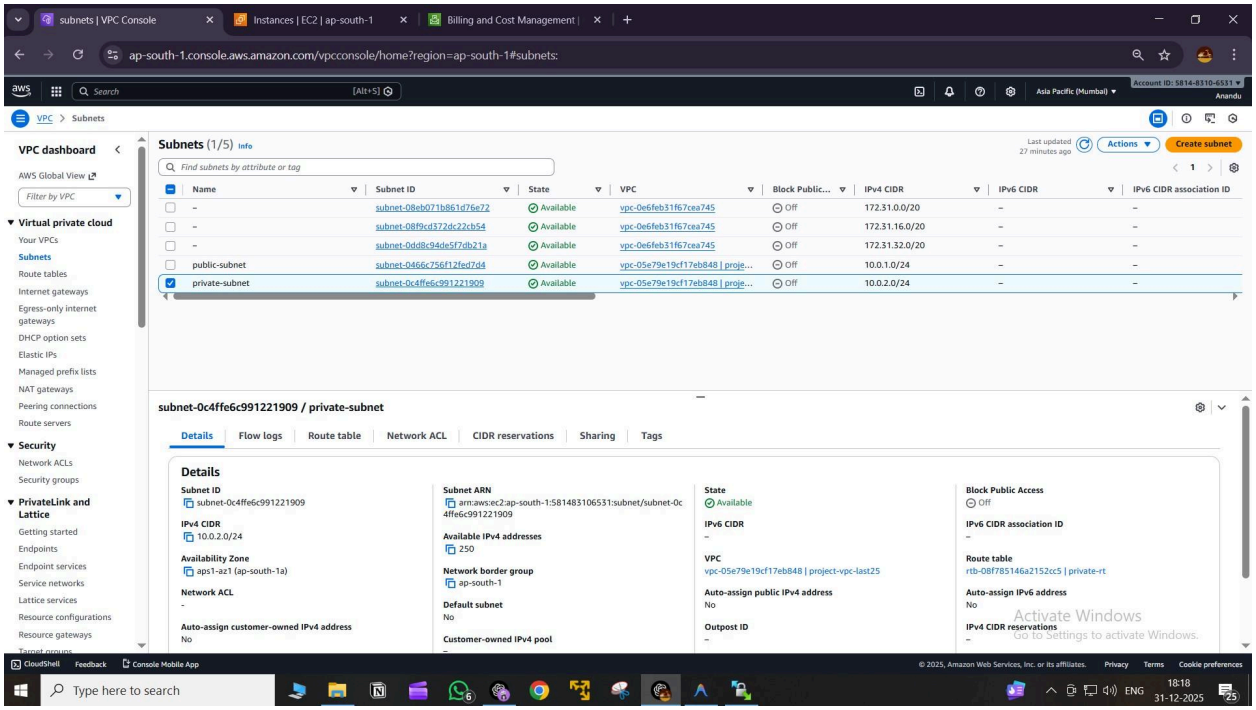
4.Routetable-public



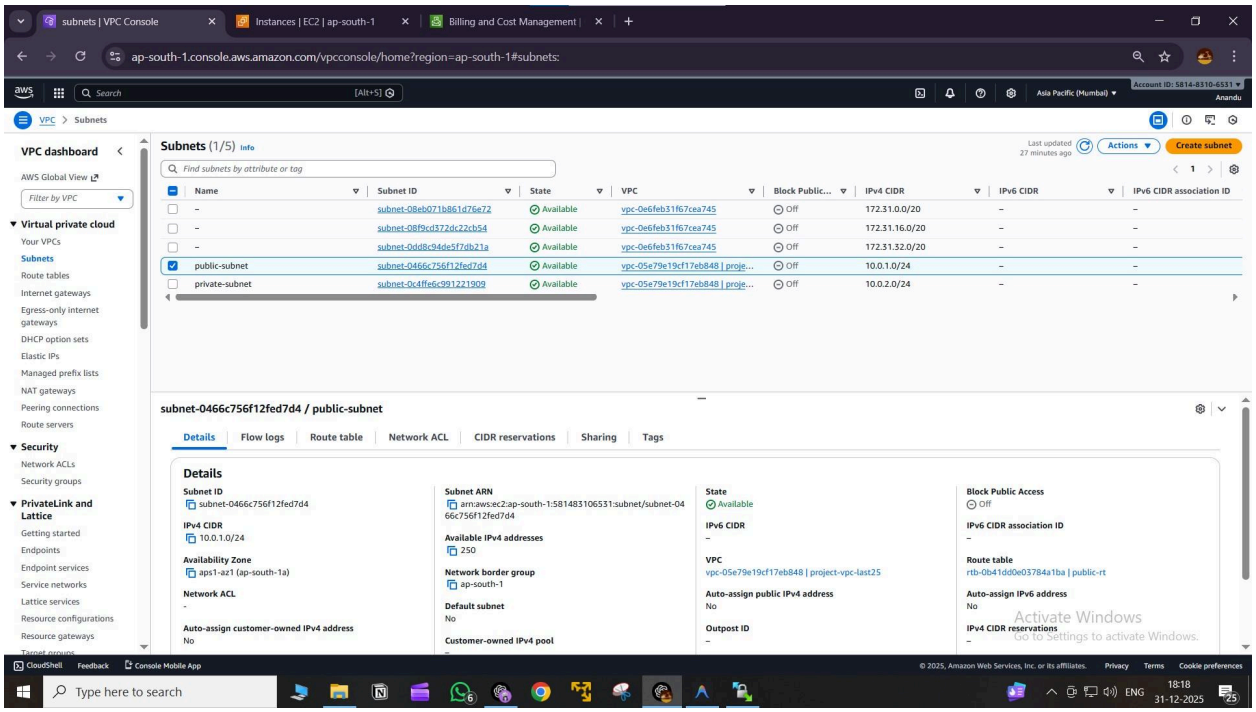
5.Nat gateways



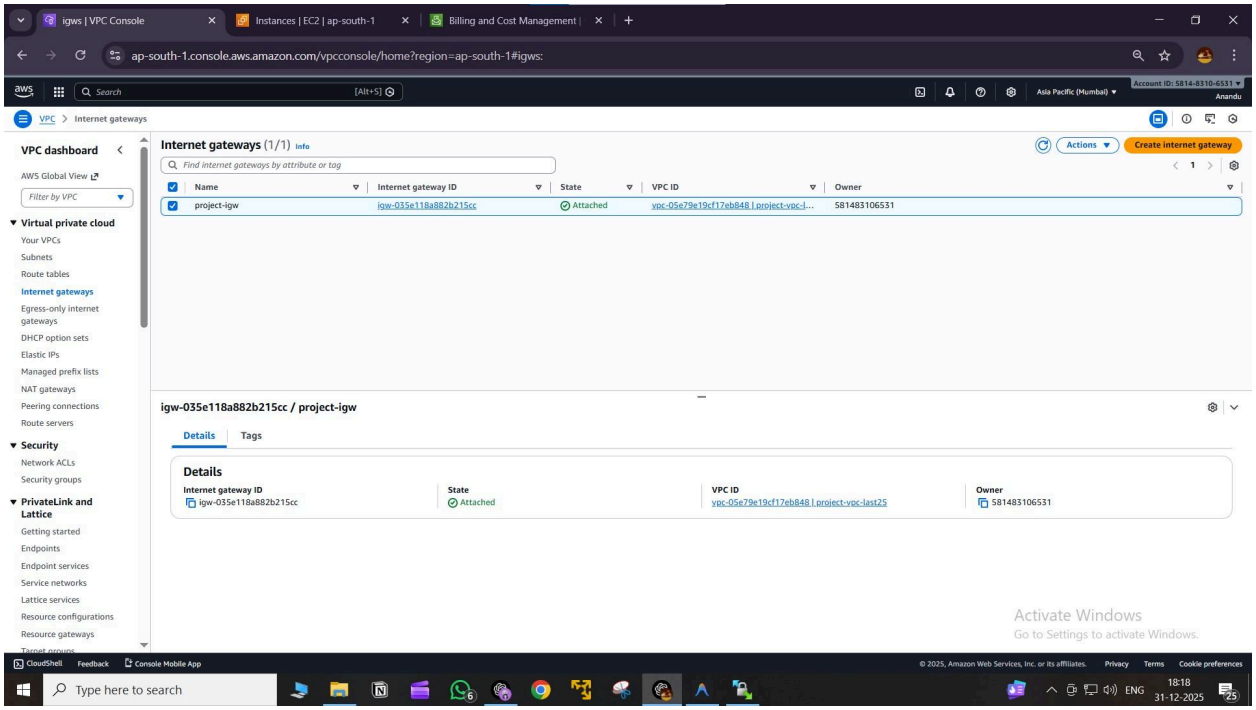
6.Elastic IP



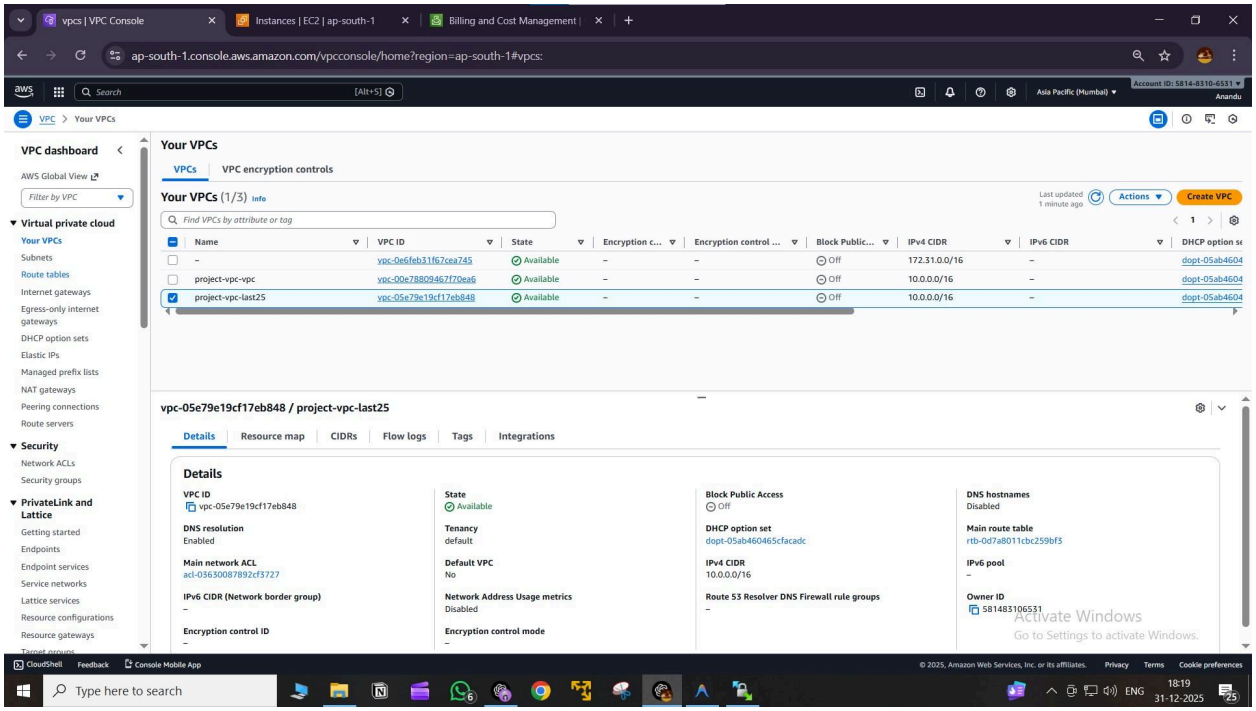
7.Private subnet



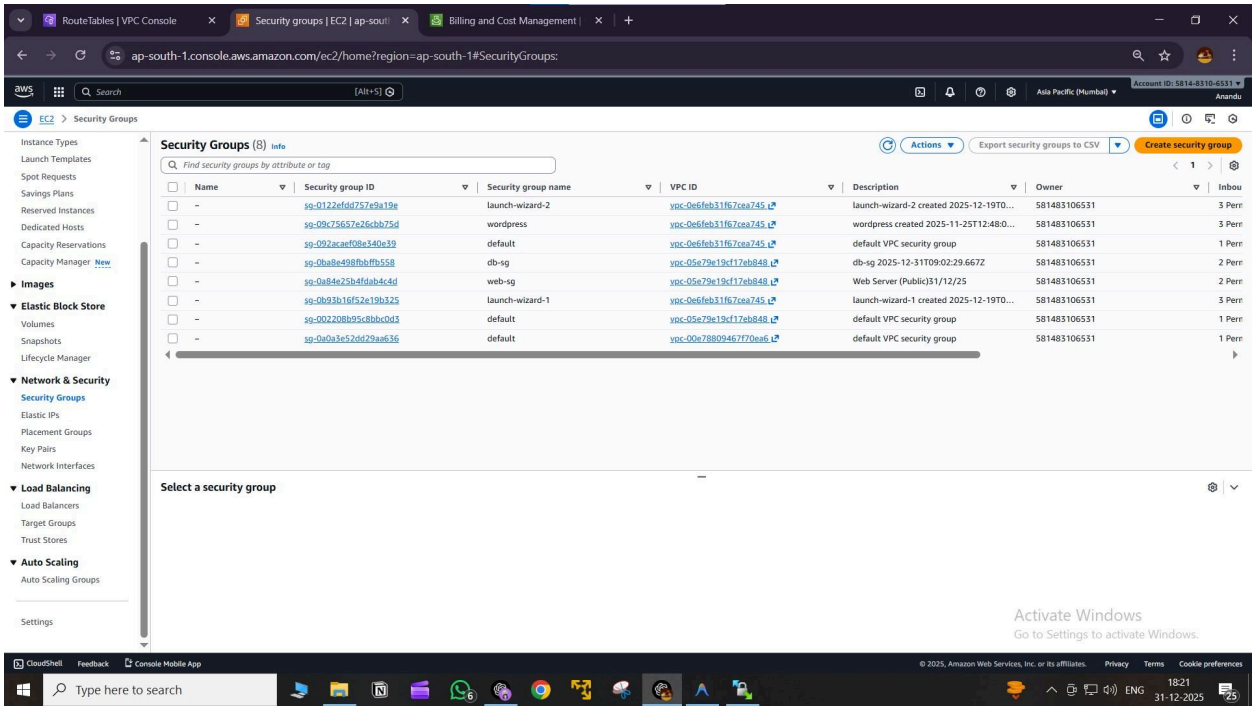
8.Public subnet



9.Internet Gateway



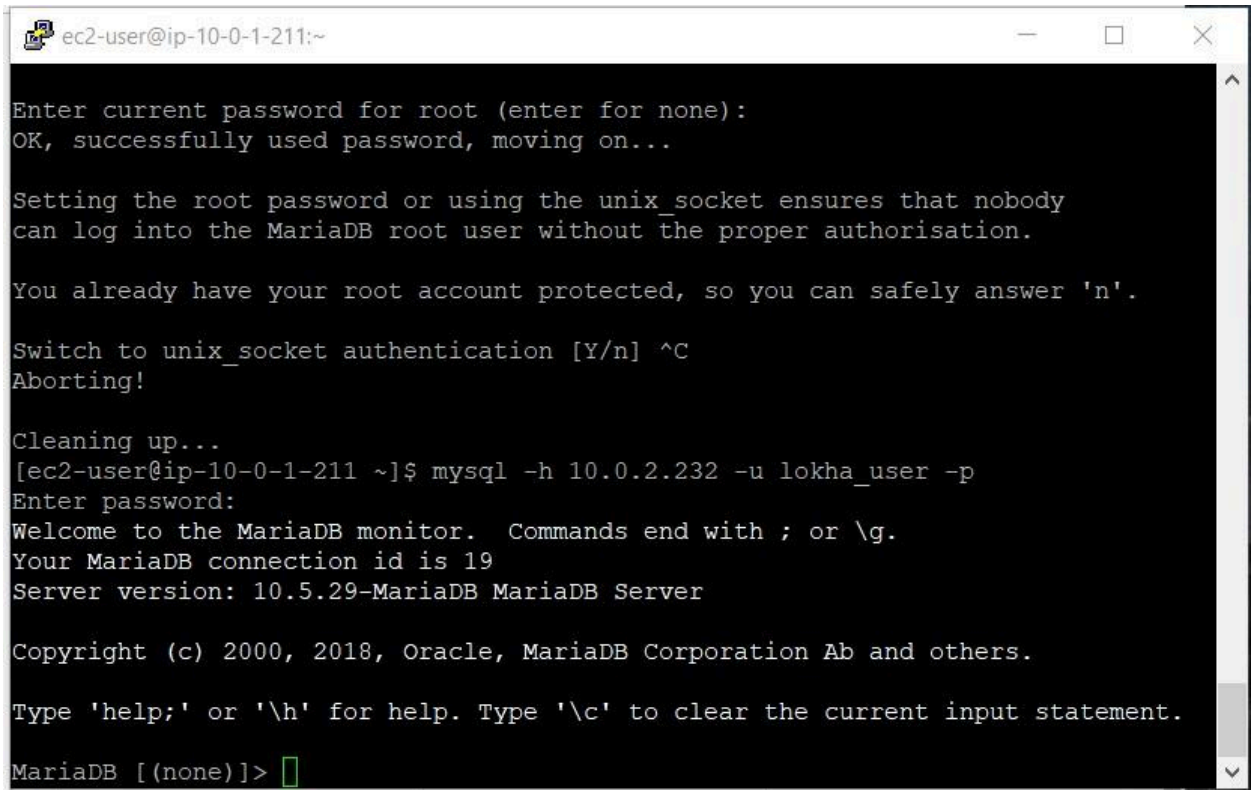
10.VPC



11.security Group

12.ssh to db server

13.checking user exits



14.connectig to db server via server1

---

7. Cost Management & Cleanup

Charged Resources

- **EC2 instances**
- **NAT Gateway**
- **Elastic IP**

Cleanup Actions

- **Terminated EC2 instances**
- **Deleted NAT Gateway**
- **Released Elastic IP**

Free Resources (no cost):

- **VPC**
- **Subnets**
- **Route Tables**
- **Security Groups**

---

8. Final Outcome

This project successfully demonstrates:

- **Secure AWS networking**
- **Real-world two-tier architecture**
- **Controlled database access**
- **Outbound-only internet from private subnet**
- **Practical troubleshooting skills**

---

9. Interview-Ready Summary

> **“I deployed a two-tier AWS architecture with a public web server and a private database server. The database has no public IP and is accessible only from the web server using security group references. A NAT Gateway provides outbound internet access for patching while maintaining isolation.”**

---

10. Conclusion

This project provides a strong foundation in AWS networking, security, and application deployment, closely matching real production environments and interview expectations.