Three-Tier Web Architecture

Arun M

Explanation Three-Tire Web Architecture:

- **Web Server**: This is the front part of a web application that displays the website to users. It handles how the site looks and interacts with users, like showing product pages or allowing people to sign up.
- **Application Server**: This middle part manages the business logic and processes user actions. For example, it checks the inventory to see if a product is in stock or updates a user's account details.
- **Database Server**: This is the back-end part that stores all the data for the application. It uses databases like MySQL or PostgreSQL to keep track of information.

Abstract

In a multi-tier architecture, the system is divided into three distinct layers, each responsible for a specific function.

- **Web Tier:** This layer handles client requests and serves the front-end user interface, enabling users to interact with the website.
- **Application Tier:** This intermediate layer processes API requests and manages the business logic, acting as a bridge between the web and database tiers.
- **Database Tier:** The final layer is responsible for data storage, retrieval, and management, ensuring that the application's data is securely maintained and efficiently accessed.

This separation of concerns enhances scalability, maintainability, and security within the system.

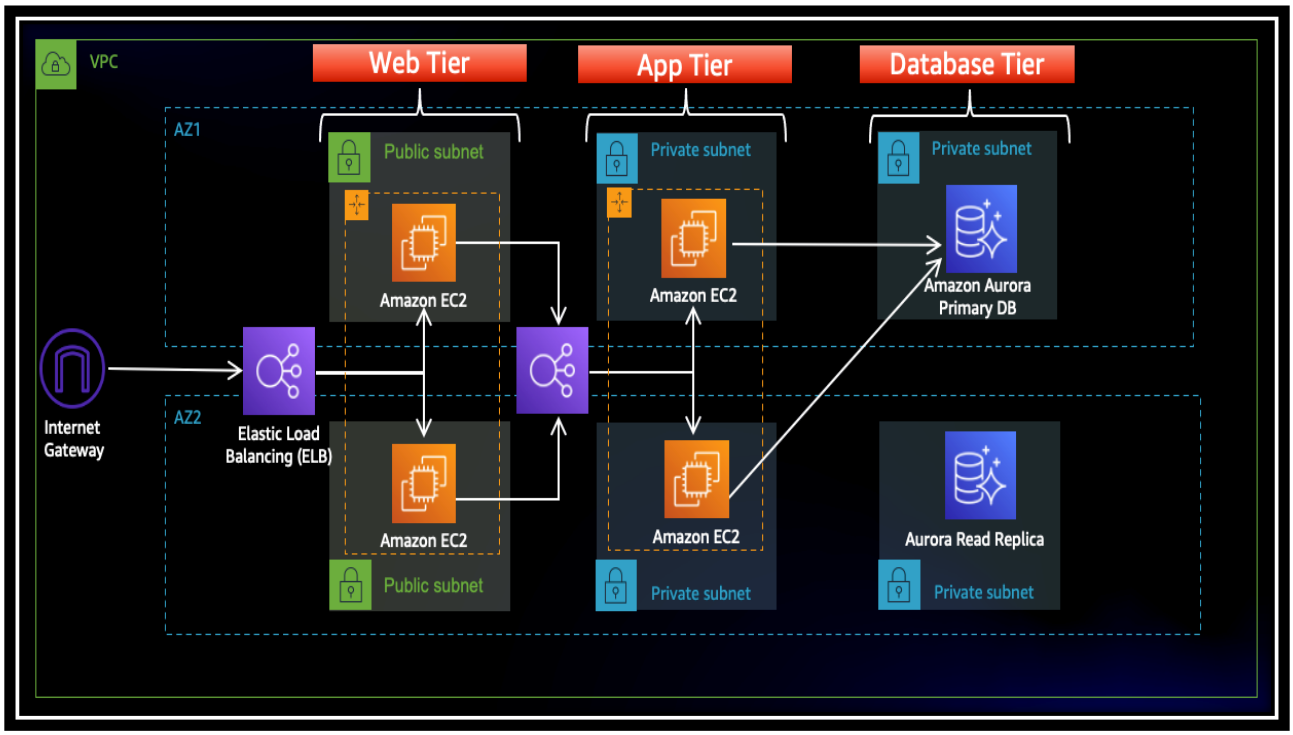
Overview

1. **Web Tier:** Handles client requests and serves the front-end website.
2. **Application Tier:** Processes API requests and handles business logic.
3. **Database Tier:** Manages data storage and retrieval.

Three-Tier Web Architecture

Arun M

Workflow – Diagram



Explanation Work-Flow Diagram :

Architecture is designed to be scalable, secure, and highly available, making it suitable for production-grade applications.

1. VPC

- The entire architecture resides within an AWS VPC, which is a logically isolated network in the AWS cloud. The VPC contains public and private subnets across two Availability Zones (AZs) for high availability.

2. Internet Gateway

- The Internet Gateway (IGW) is attached to the VPC. It allows communication between the VPC and the internet. Resources in public subnets, such as the Web Tier, can communicate with the internet via the IGW.

3. Web Tier

- **Public Subnets:** The Web Tier is deployed in public subnets across two Availability Zones (AZ1 and AZ2) to ensure redundancy and availability.

Three-Tier Web Architecture

Arun M

- **Amazon EC2 Instances:** EC2 instances in the Web Tier host the frontend web application that users interact with. These instances are accessible from the internet because they are in public subnets.
- **Elastic Load Balancing (ELB):** The ELB distributes incoming traffic from the internet to the EC2 instances in the Web Tier. It ensures that traffic is balanced across the instances in both AZ1 and AZ2, providing fault tolerance.

4. App Tier

- **Private Subnets:** The App Tier is deployed in private subnets, which are isolated from direct internet access for security reasons.
- **Amazon EC2 Instances:** EC2 instances in the App Tier run the business logic of the application. They process requests from the Web Tier and communicate with the Database Tier. Since these instances are in private subnets, they can only be accessed internally, not directly from the internet.
- **Traffic Flow:** The ELB routes traffic to the EC2 instances in the Web Tier, which then forwards requests to the EC2 instances in the App Tier. The App Tier processes the requests and interacts with the database as needed.

5. Database Tier

- **Private Subnets:** The Database Tier is also deployed in private subnets for enhanced security.
- **Amazon MYSQL Primary DB:** The primary Amazon Mysql database is hosted in one of the private subnets. It stores the application's data and handles read and write operations.
- **Mysql Read Replica:** A read replica of the Aurora database is deployed in another private subnet. This replica can handle read requests, improving the overall read performance and fault tolerance. The primary DB asynchronously replicates data to the read replica.

6. Traffic Flow Overview

- **User Requests:** Users interact with the web application through their browser, sending requests over the internet.
- **ELB:** The ELB receives these requests and distributes them to the appropriate EC2 instances in the Web Tier.
- **Web Tier to App Tier:** The EC2 instances in the Web Tier process the initial request and forward it to the EC2 instances in the App Tier for further processing.
- **App Tier to Database Tier:** If the request involves database interaction, the App Tier EC2 instances communicate with the Aurora database in the Database Tier.
- **Response:** The processed data is then sent back through the tiers, with the Web Tier returning the final response to the user via the ELB.

Three-Tier Web Architecture

Arun M

7. High Availability and Fault Tolerance

- **Multiple AZs:** The deployment across multiple Availability Zones ensures that the application remains available even if one AZ fails.
- **ELB:** The use of an Elastic Load Balancer distributes traffic evenly, preventing any single EC2 instance from becoming a bottleneck.
- **Aurora Replication:** The Aurora Read Replica ensures that database read requests can still be processed if the primary database becomes unavailable.

Components explanation

1. External Load Balancer (Public-Facing Application Load Balancer)

- **Role**: This acts as the entry point for all client traffic.
- **Functionality**:
  - Distributes incoming client requests to the web tier EC2 instances.
  - Ensures even distribution of traffic for better performance and reliability.
  - Performs health checks to ensure only healthy instances receive traffic.

2. Web Tier

- **Role**: Serves the front-end of the application and redirects API calls.
- **Components**:
  - **Nginx Webservers**: Running on EC2 instances.
  - **React.js Website**: The front-end application served by Nginx.
- **Functionality**:
  - **Serving the Website**: Nginx serves the static files for the React.js application to the clients.
  - **Redirecting API Calls**: Nginx is configured to route API requests to the internal-facing load balancer of the application tier.

3. Internal Load Balancer (Application Tier Load Balancer)

- **Role**: Manages traffic between the web tier and the application tier.
- **Functionality**:
  - Receives API requests from the web tier.

Three-Tier Web Architecture

Arun M

- Distributes these requests to the appropriate EC2 instances in the application tier.
- Ensures high availability and load balancing within the application tier.

4. Application Tier

- **Role**: Handles the application logic and processes API requests.
- **Components**:
  - **Node.js Application**: Running on EC2 instances.
- **Functionality**:
  - **Processing Requests**: The Node.js application receives API requests, performs necessary computations or data manipulations.
  - **Database Interaction**: Interacts with the Aurora MySQL database to fetch or update data.
  - **Returning Responses**: Sends the processed data back to the web tier via the internal load balancer.

5. Database Tier (Aurora MySQL Multi-AZ Data base)

- **Role**: Provides reliable and scalable data storage.
- **Functionality**:
  - **Data Storage**: Stores all the application data in a structured format.
  - **Multi-AZ Setup**: Ensures high availability and fault tolerance by replicating data across multiple availability zones.
  - **Data Retrieval and Manipulation**: Handles queries and transactions from the application tier to manage the data.

Additional Components

Load Balancing

- **Purpose**: Distributes incoming traffic evenly across multiple instances to prevent any single instance from becoming a bottleneck.
- **Implementation**:
  - **Web Tier**: The external load balancer distributes traffic to web servers.

Three-Tier Web Architecture

Arun M

- **Application Tier**: The internal load balancer distributes API requests to application servers.

Health Checks

- **Purpose**: Continuously monitor the health of instances to ensure only healthy instances receive traffic.
- **Implementation**:
  - **Web Tier**: Health checks by the external load balancer to ensure web servers are responsive.
  - **Application Tier**: Health checks by the internal load balancer to ensure application servers are operational.

Auto Scaling Groups

- **Purpose**: Automatically adjusts the number of running instances based on traffic load to maintain performance and cost efficiency.
- **Implementation**:
  - **Web Tier**: Auto-scaling based on metrics like CPU usage or request count to add or remove web server instances.
  - **Application Tier**: Auto-scaling based on similar metrics to adjust the number of application server instances.

AWS Certificate Manager (ACM)

- **Purpose**: Manages SSL/TLS certificates to secure data in transit between clients and your application, ensuring encrypted communication.
- **Implementation**:
  - **Certificate Provisioning**: ACM provides and manages SSL/TLS certificates for your domain learnaws.co.in.
  - **Certificate Deployment**: The ACM certificates are associated with the public-facing Application Load Balancer (ALB) to enable HTTPS traffic.
  - **Automatic Renewal**: ACM automatically renews certificates before they expire, ensuring uninterrupted secure connections.

Amazon Route 53

- **Purpose**: Manages DNS records and directs user traffic to the appropriate AWS resources, optimizing for performance and reliability.

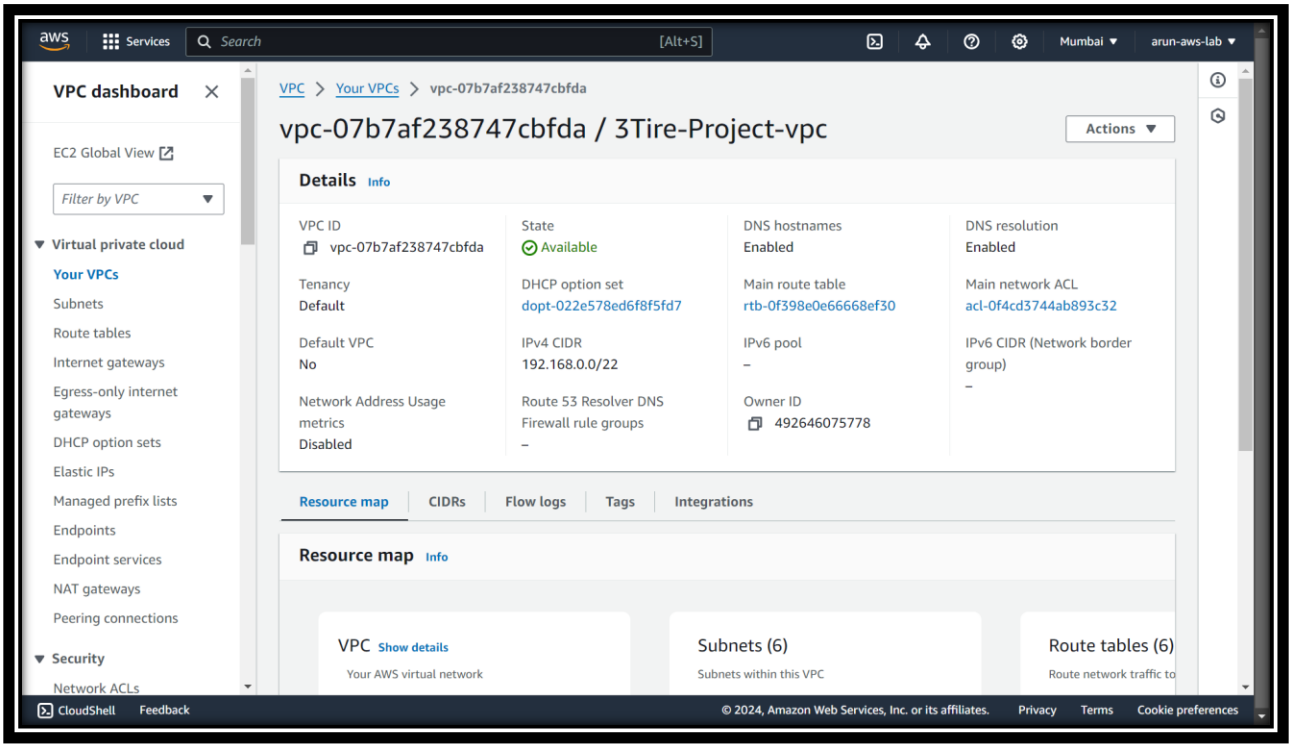
Three-Tier Web Architecture

Arun M

- **Implementation**:
    - **DNS Management**: Route 53 handles DNS queries for the domain learnaws.co.in, translating it into IP addresses for your Application Load Balancer.
    - **Traffic Routing**: Route 53 directs client requests to the public-facing Application Load Balancer based on DNS records.
    - **Health Checks and Failover**: Optionally, Route 53 performs health checks on your endpoints and can automatically reroute traffic to health resources if needed.

Step-By-Step Process & Screenshot :

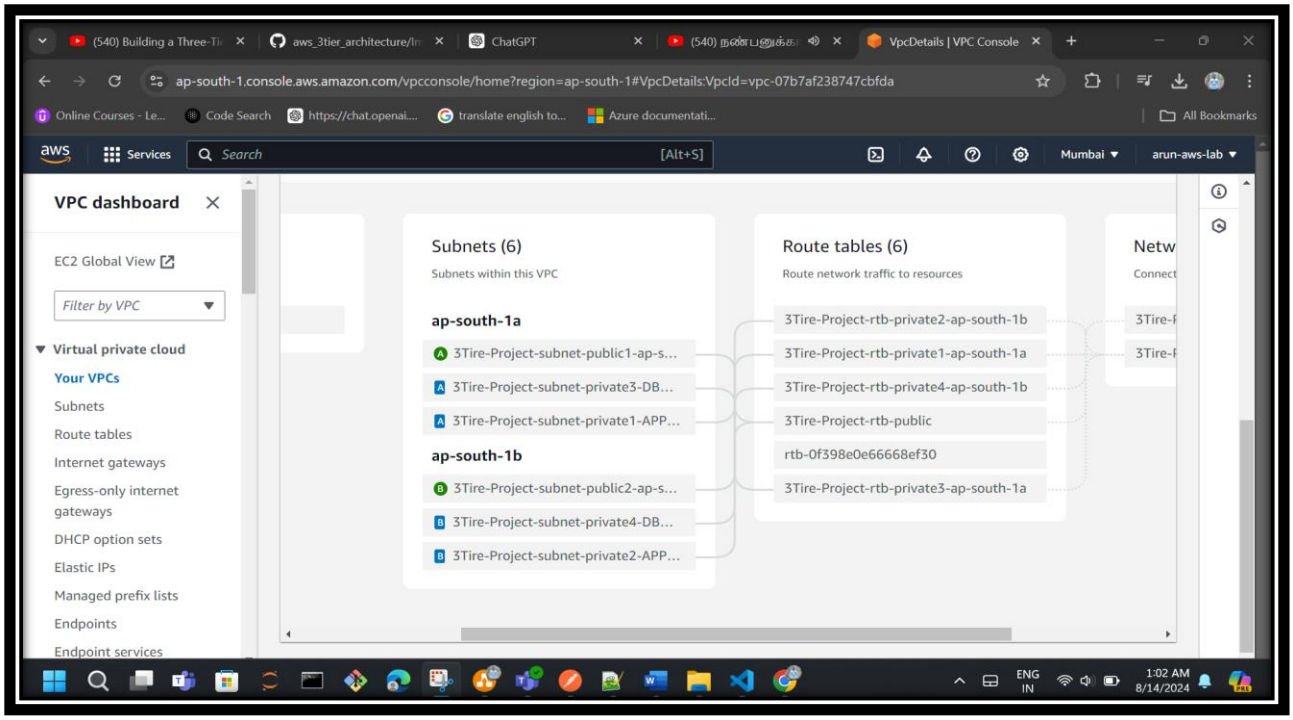
VPC : VPC named as “3Tire-Project-vpc”



Three-Tier Web Architecture

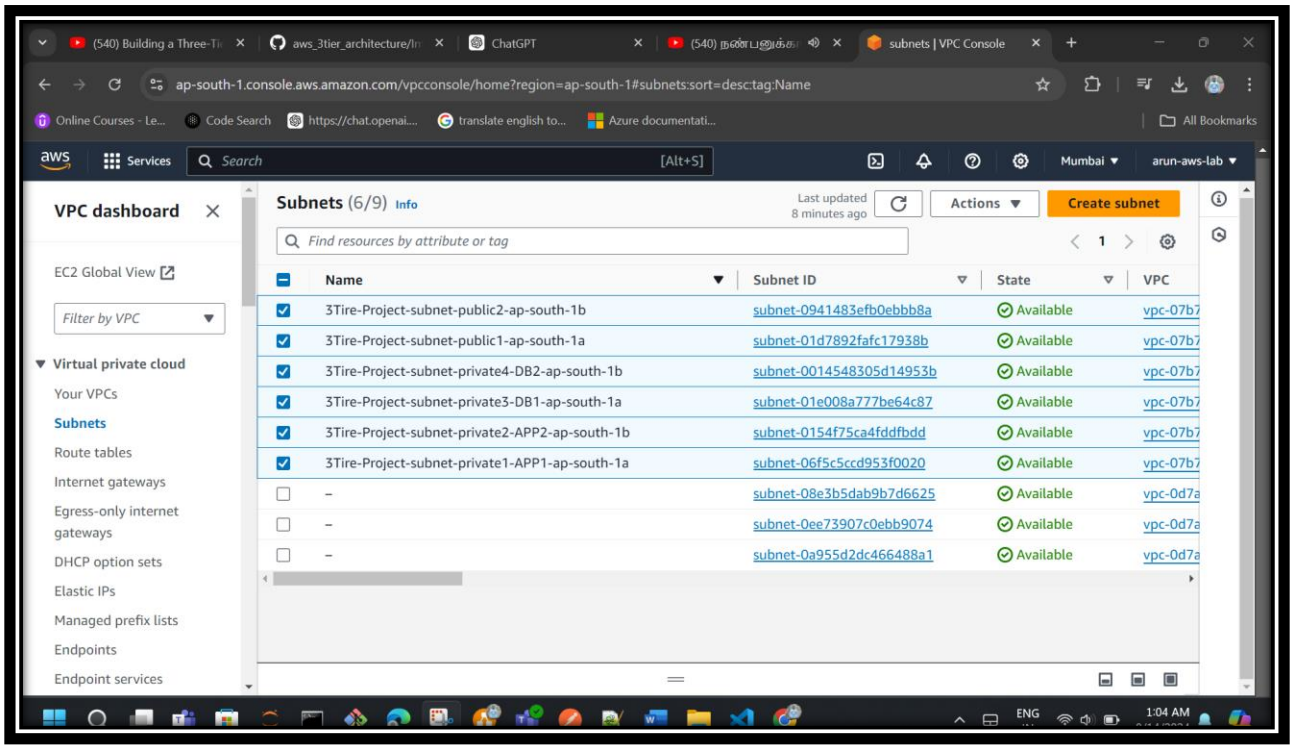
Arun M

- This is a Road Map of VPC and its Configuration.



Subnets

- They were 6 different subnet was used in two different Availability Zone.

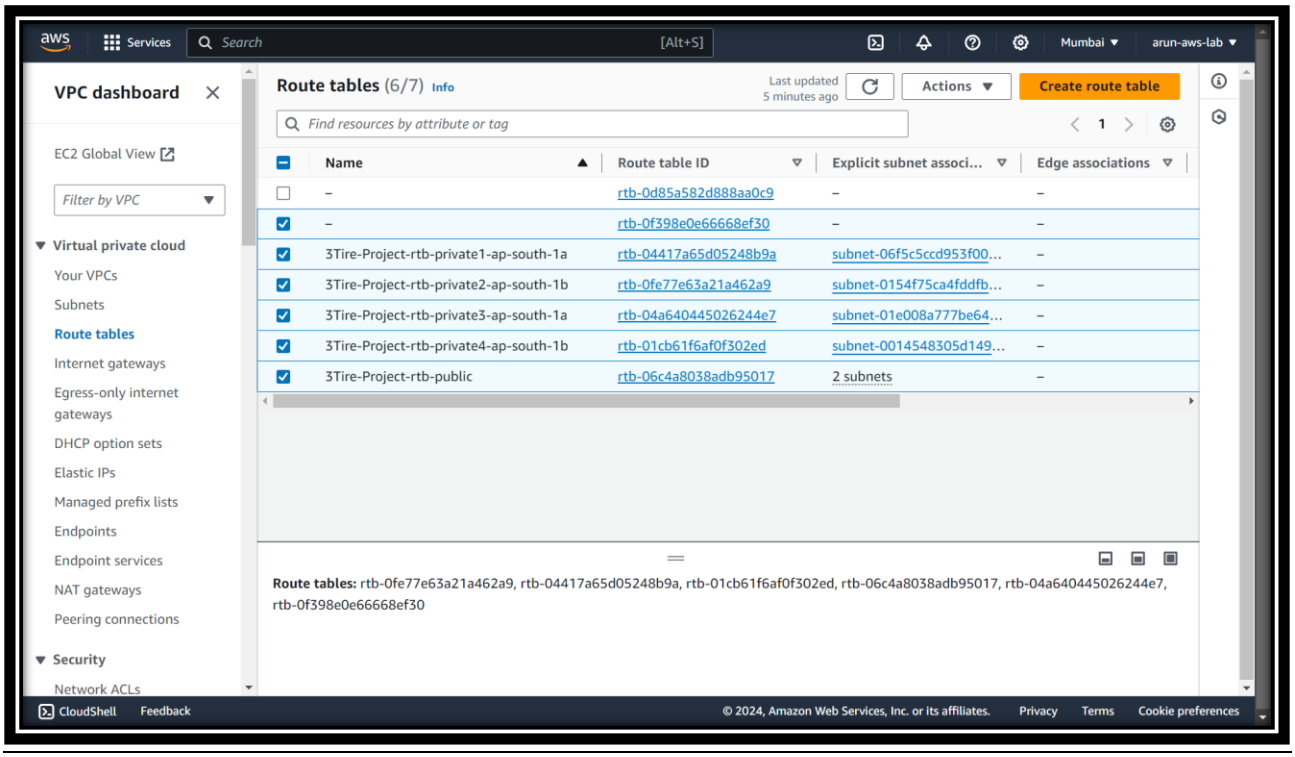


Three-Tier Web Architecture

Arun M

Subnet Name	CIDR range	Availability Zone
3Tire-Project-subnet public2-ap-south-1a	192.168.0.0/26	ap-south-1a
3Tire-Project-subnet public2-ap-south-1b	192.168.2.64/26	ap-south-1b
3Tire-Project-subnet-private3-DB1-ap-south-1a	192.168.2.128/26	ap-south-1a
3Tire-Project-subnet-private3-DB1-ap-south-1b	192.168.2.192/26	Ap-south-1b
3Tire-Project-subnet-public1-ap-south-1a	192.168.0.0/26	Ap-south-1a
3Tire-Project-subnet-public1-ap-south-1b	192.168.0.64/26	Ap-south-1b

Route Table :

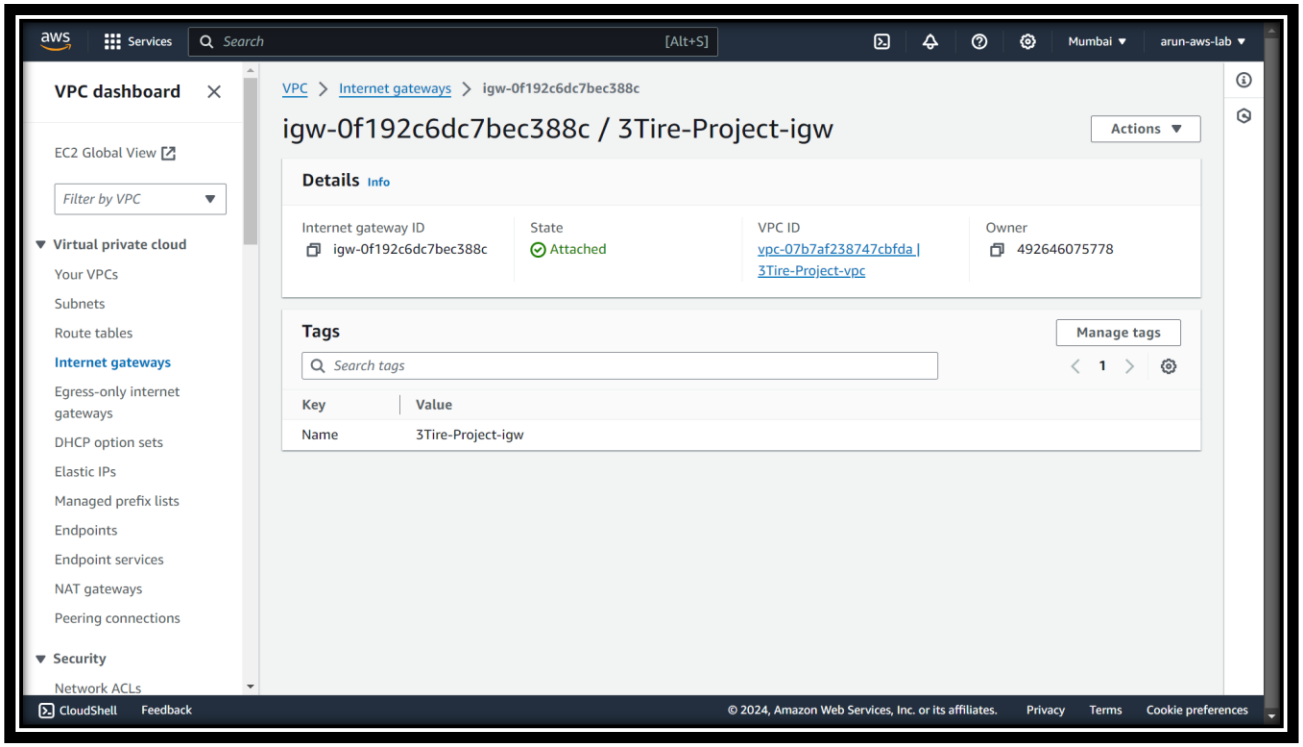


Three-Tier Web Architecture

Arun M

Internet Gateway :

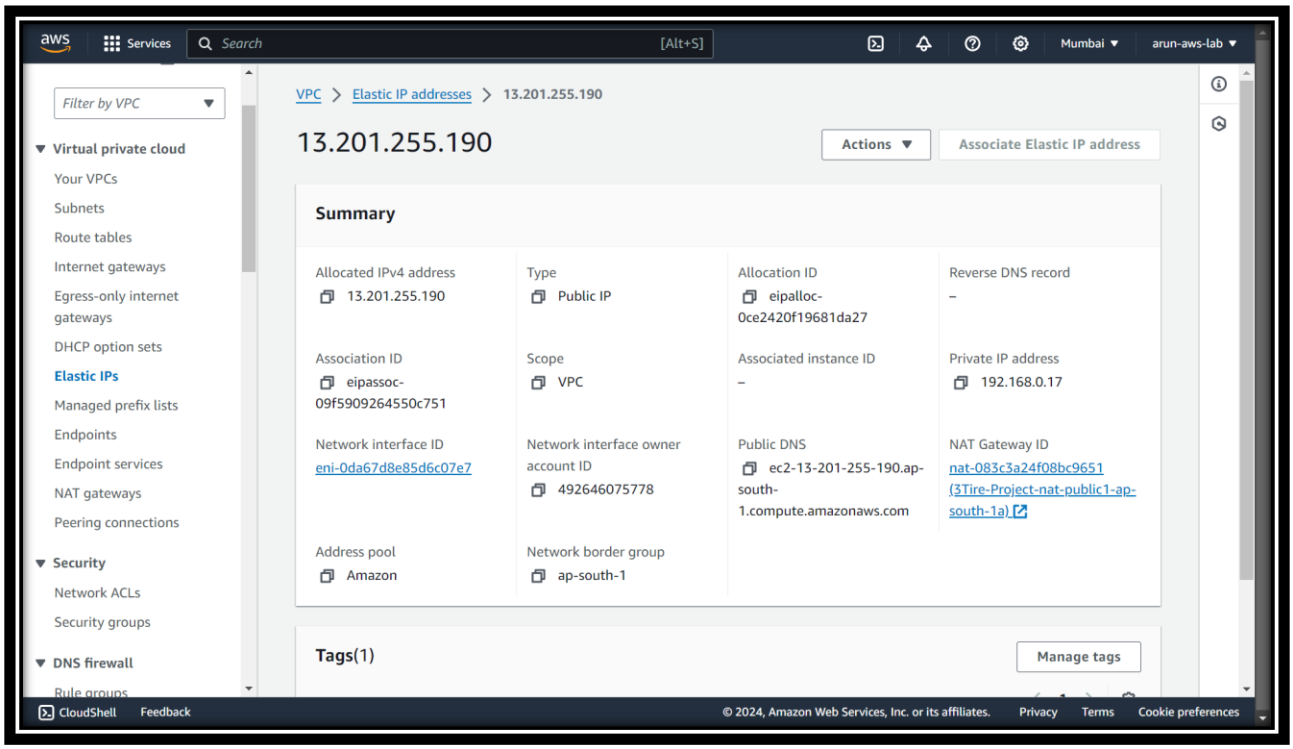
- To use an internet gateway, attach it to your VPC and specify it as a target in your subnet route table for internet-routable IPv4 or IPv6 traffic. An internet gateway performs network address translation (NAT) for instances that have been assigned public IPv4 addresses.
- Internet gateway named as "**3Tire-Project-igw**".



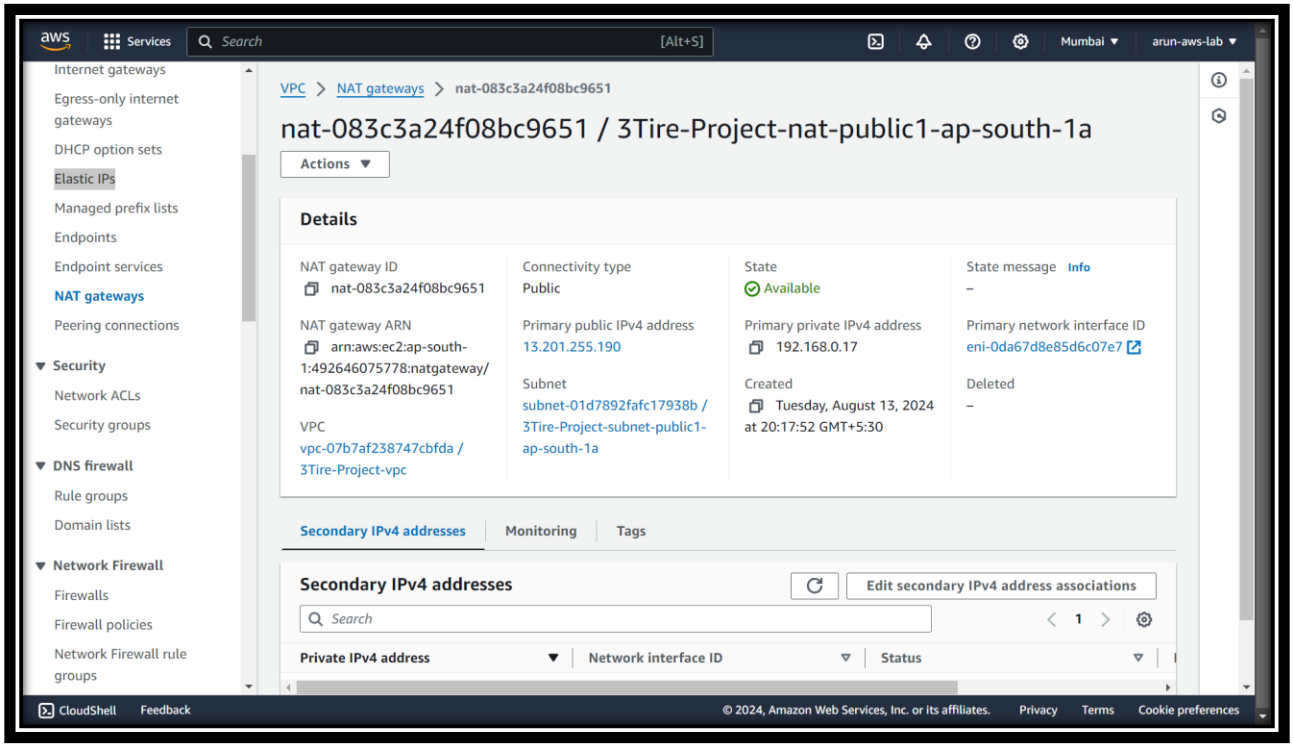
Three-Tier Web Architecture

Arun M

Elastic Ips



NAT Gateway :



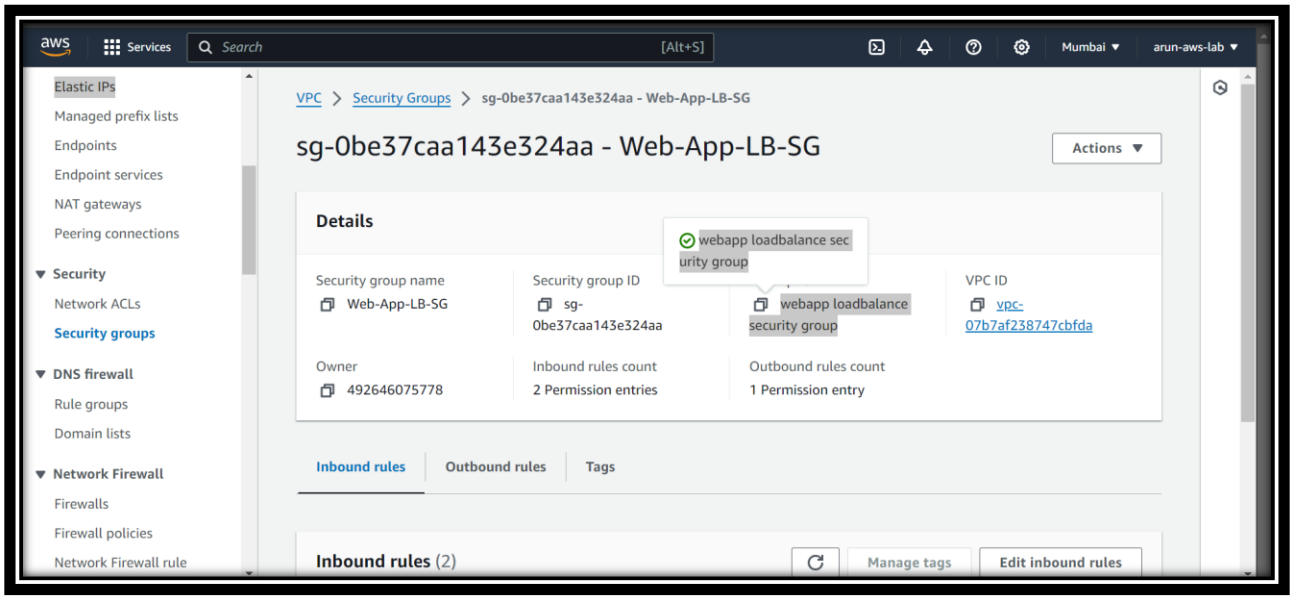
Three-Tier Web Architecture

Arun M

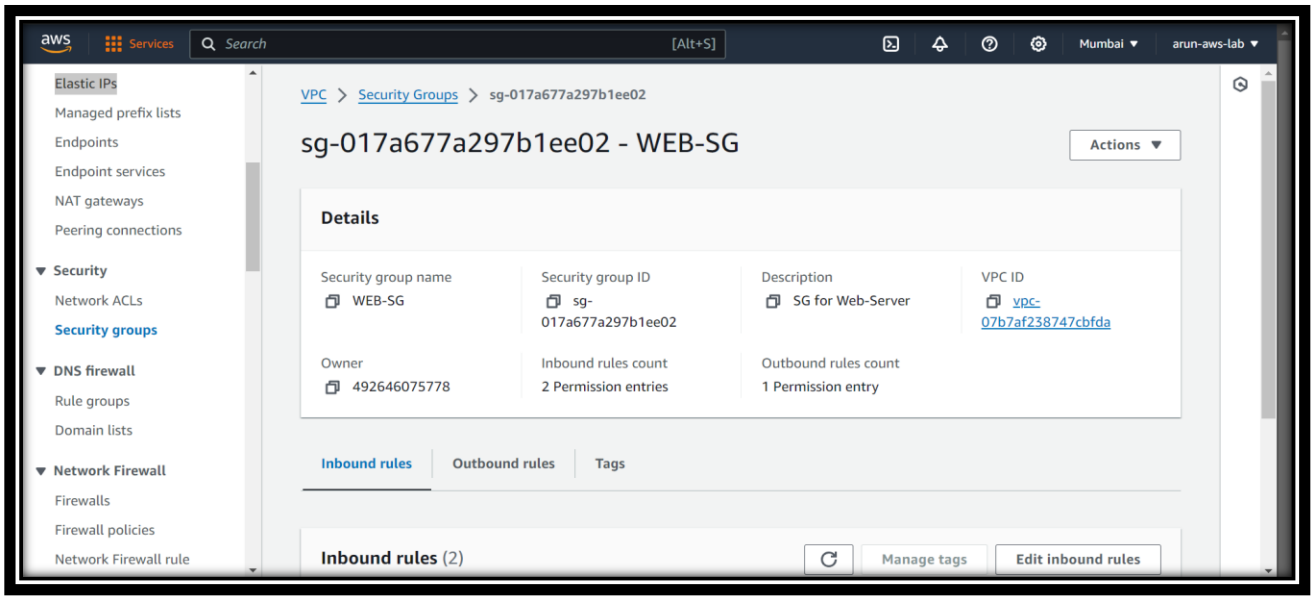
Security Group :

- There were 5 different Security Groups that were used for the process.

“Web-App-Lb-SG” : webapp loadbalance security group



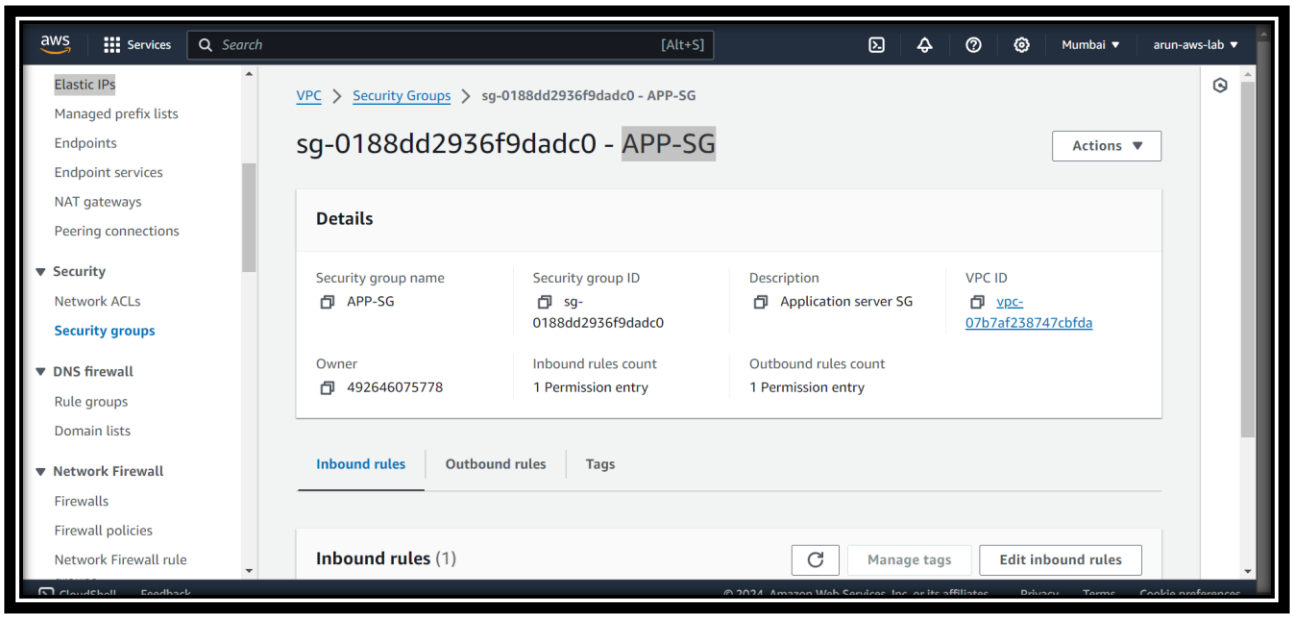
“WEB-SG” : Security Group for Webserver



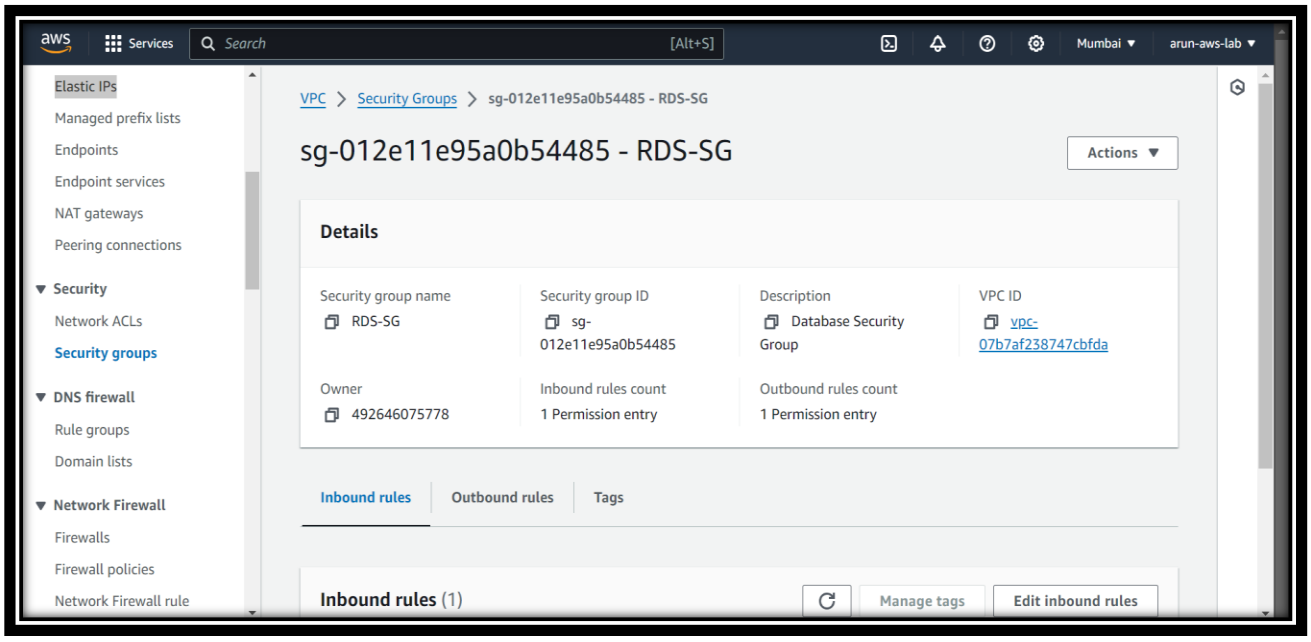
Three-Tier Web Architecture

Arun M

“APP-SG” : Security Group for Appliction server.



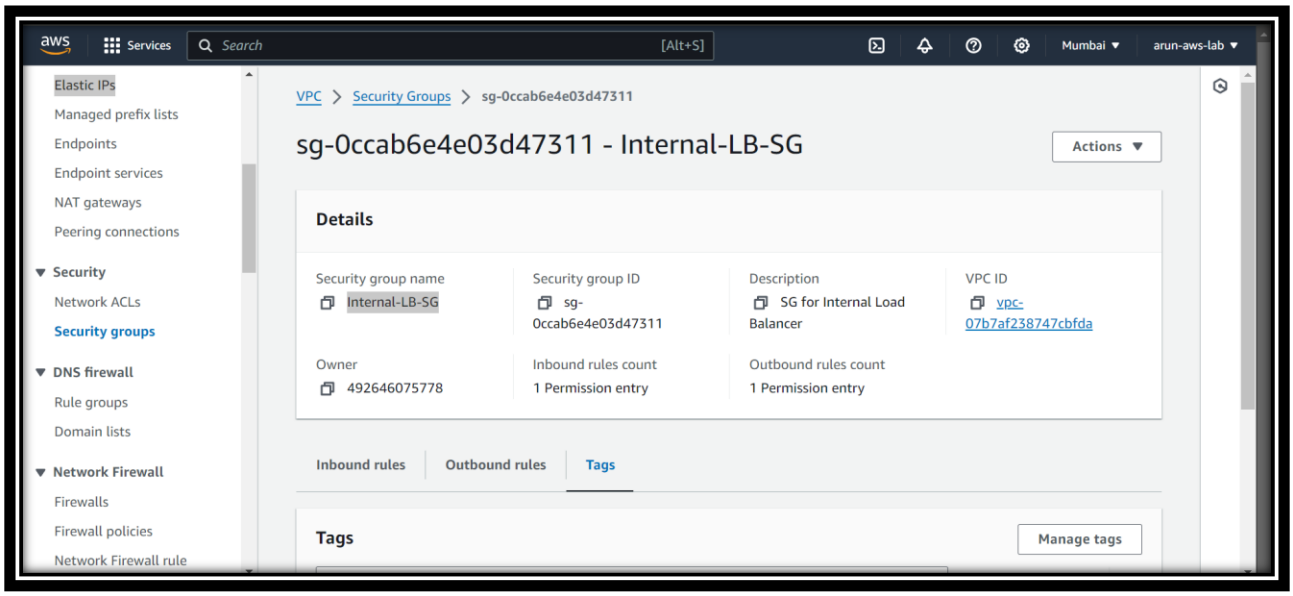
“RDS-SG” : Security Group for Database



Three-Tier Web Architecture

Arun M

“Internal-LB-SG” : Internal-LB-SG



RDS-DataBase :

- Subnet Groups
- Database

Subnet Groups :

- A subnet group is a collection of subnets (typically private) that you can designate for your self-designed clusters running in an Amazon Virtual Private Cloud (VPC) environment.
- Subnets were named as “**Three-tire-subnet-group”.**

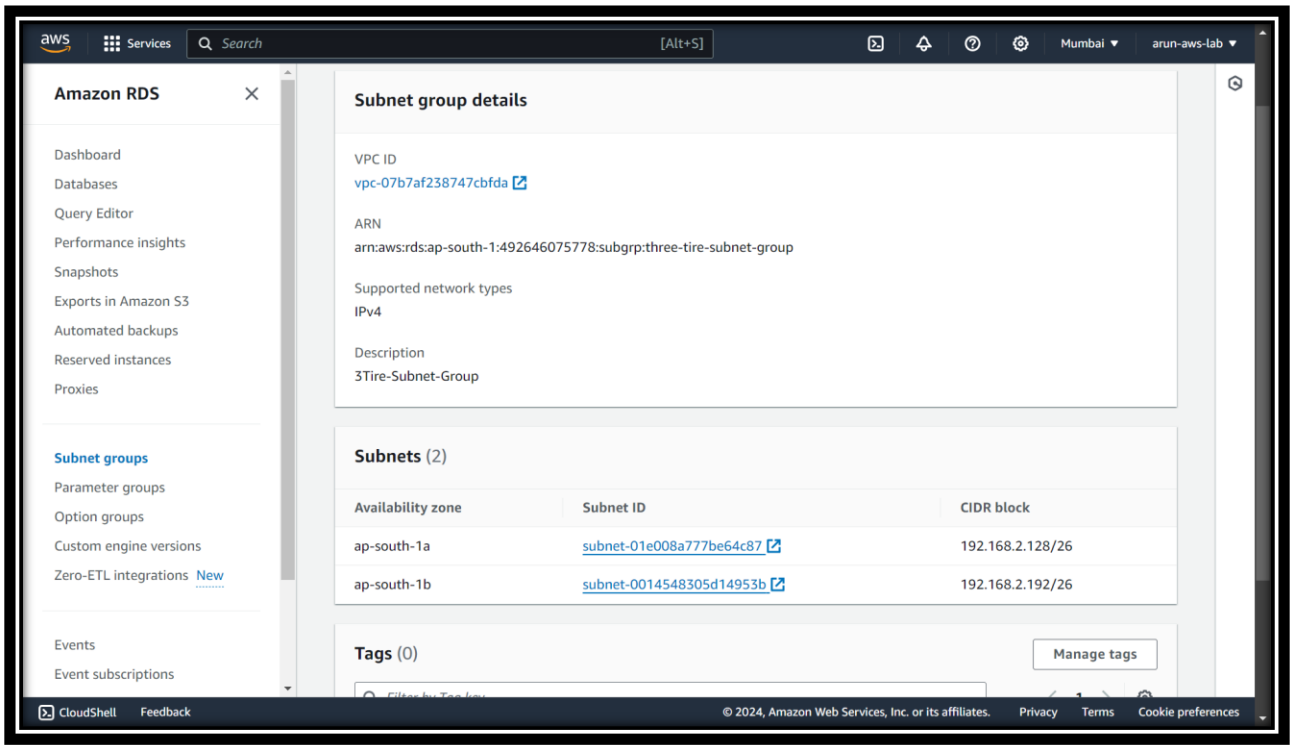
Database :

- In this three-tier architecture, I used Engine MySQL Community version : 8.0.35.
- **Amazon MYSQL Primary DB:** The primary Amazon Mysql database is hosted in one of the private subnets. It stores the application's data and handles read and write operations.

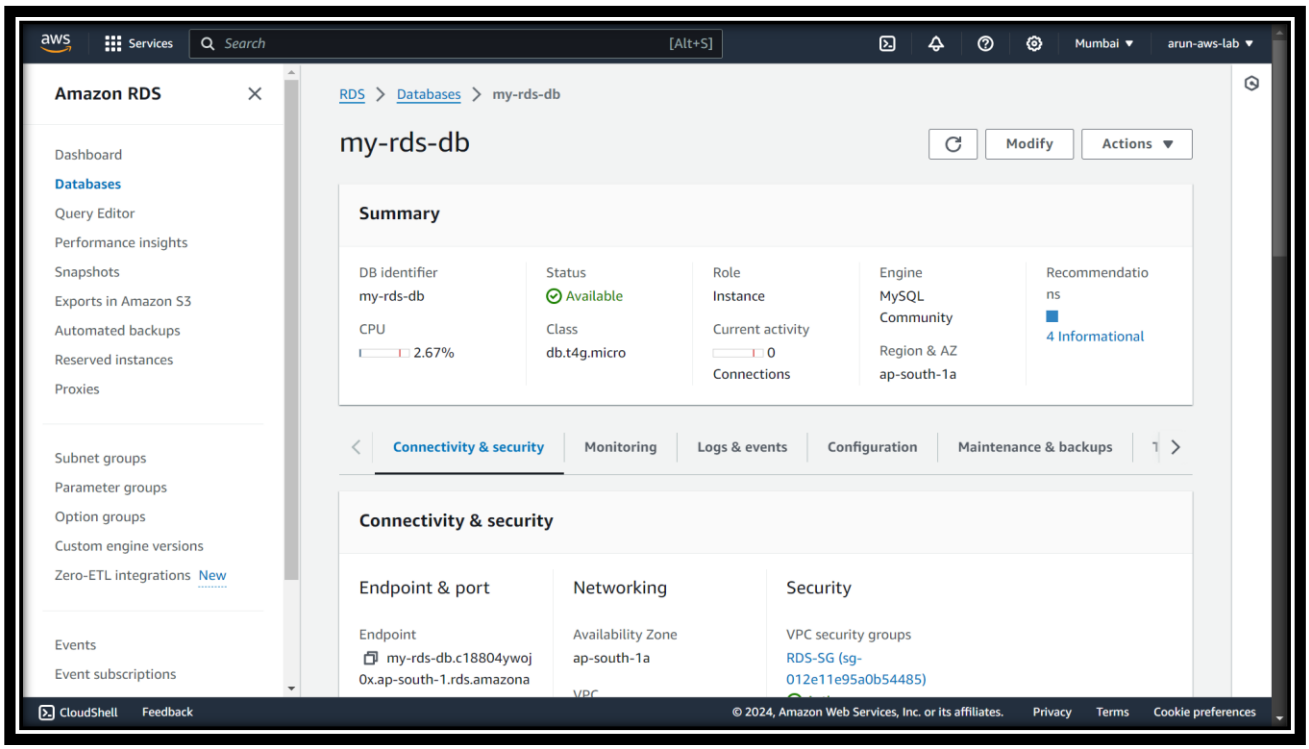
Three-Tier Web Architecture

Arun M

Subnet Group



DataBase



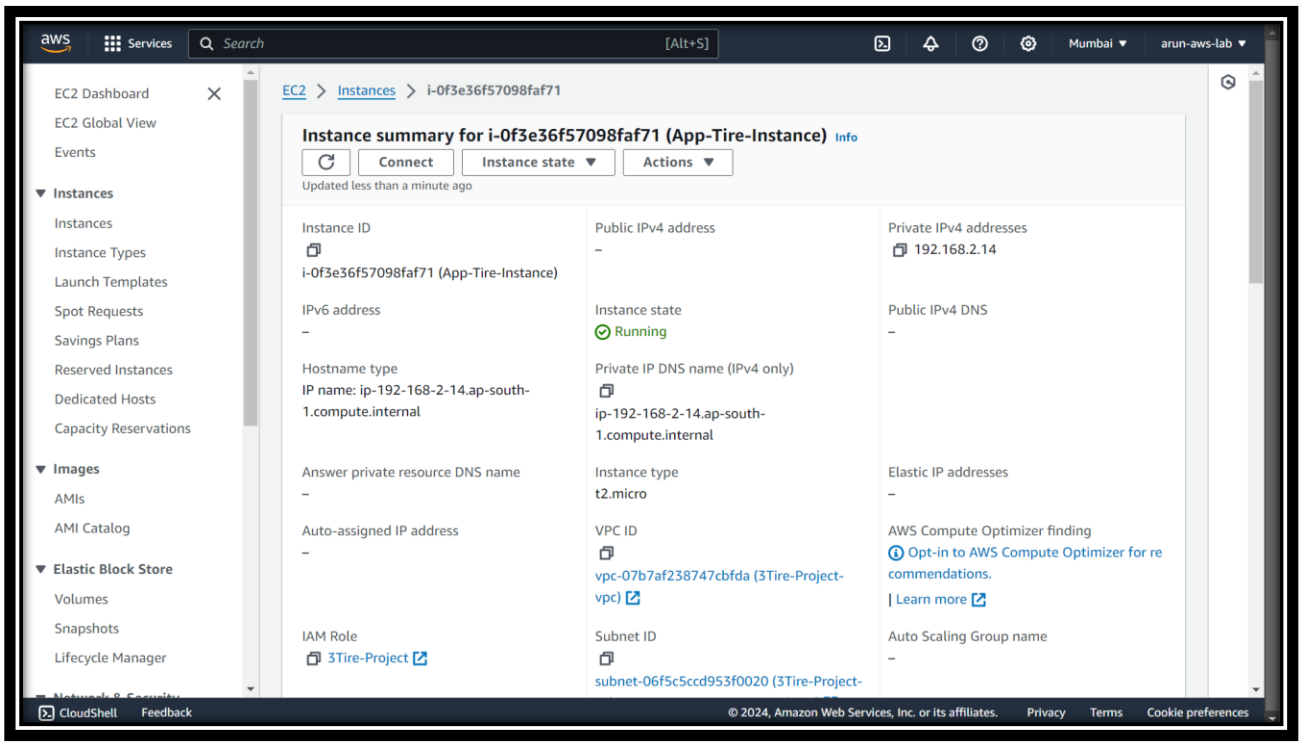
Three-Tier Web Architecture

Arun M

EC2 Instance : App Server Setup.

App Server Setup: Launch an ec2 instance in APP subnet of Custom VPC.

- App Server Setup Configuration were using EC2 Instance named as "**App-Tire-Instance**".
- The App server was "Private server" no one can access it. And it established the connection of
  DB.



Set up of Database & Mysql Packege.

- Set up and Install Mysql in the App server.
- Established the connection of DB and Appserver.

Step by Step Process:

- First, we have done with MySql Installation, open App-Server instance hit SSM Session Manager for connect the App-Server instance.

Three-Tier Web Architecture

Arun M

- Open App-Server & Install the MySql Packages.

```
sudo yum install mysql -y
```

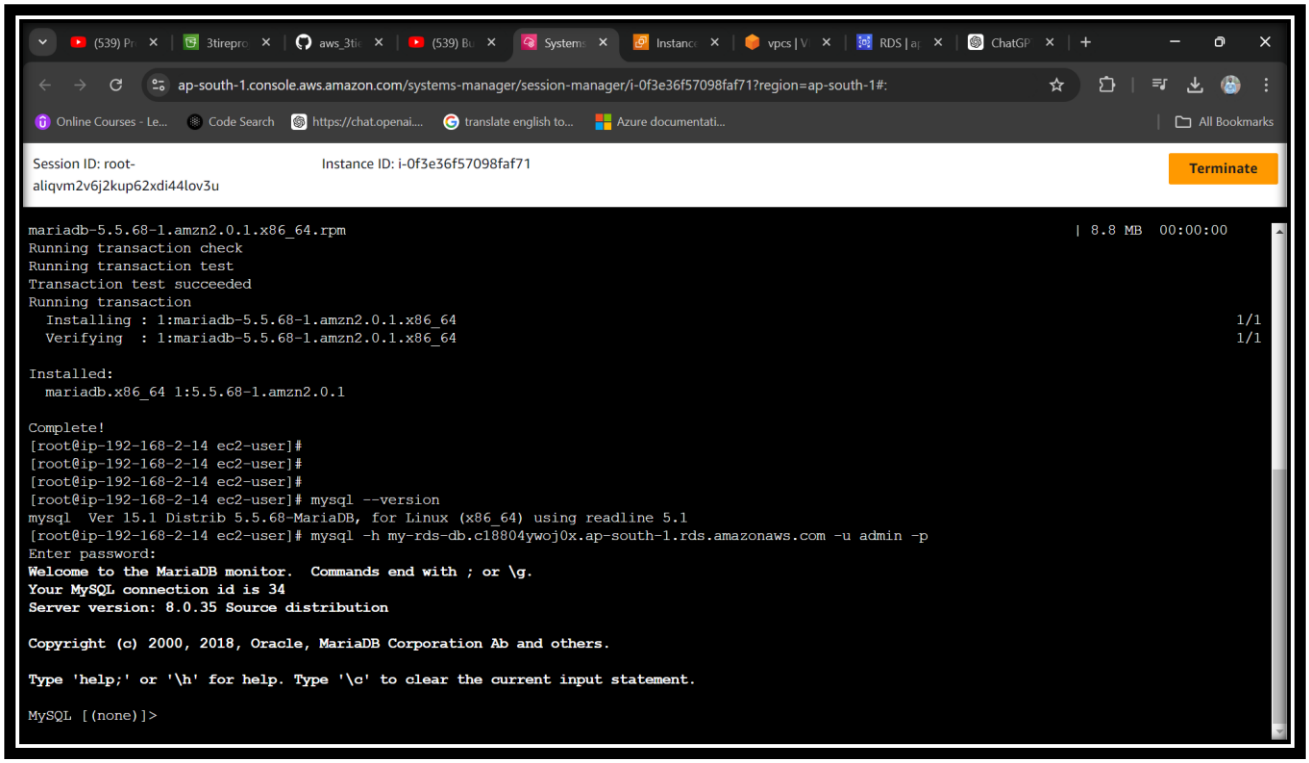
- Configure MySQL Database
- Connect to the database and perform basic configuration: Replace the info below with your DB information.

```
mysql -h mydb.cfpgnjehw330.ap-south-1.rds.amazonaws.com -u admin -p
```

- Use the above command to establish connections between app-tire with DB-tire.

Three-Tier Web Architecture

Arun M

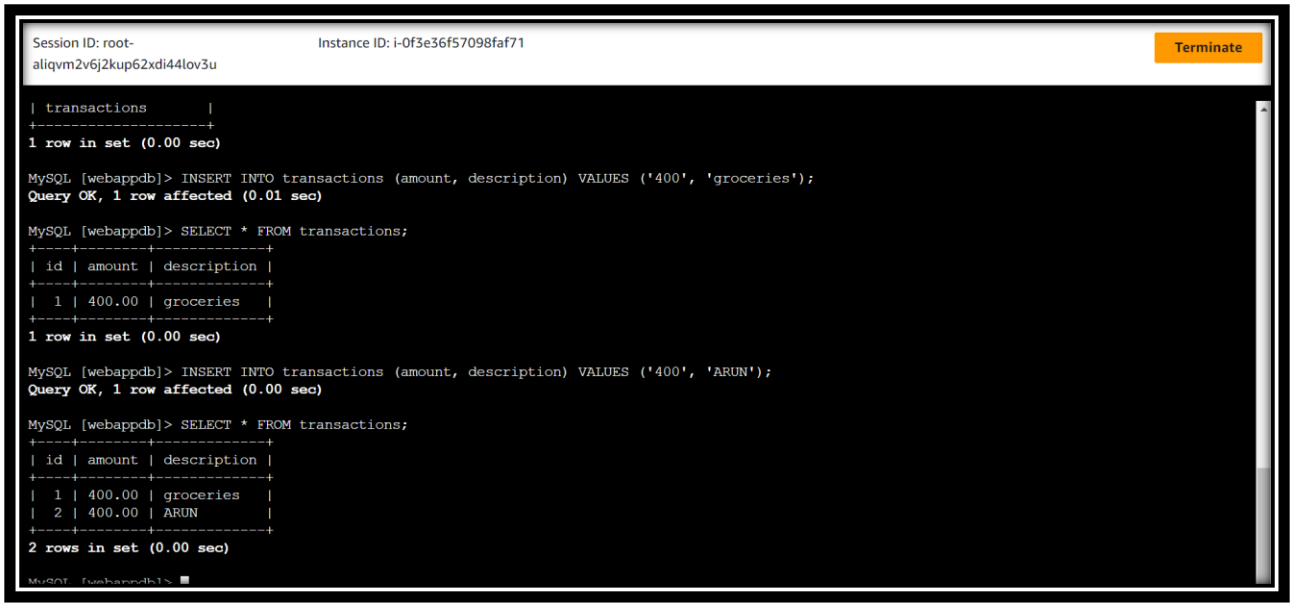


- In the MySQL shell, execute the following commands.

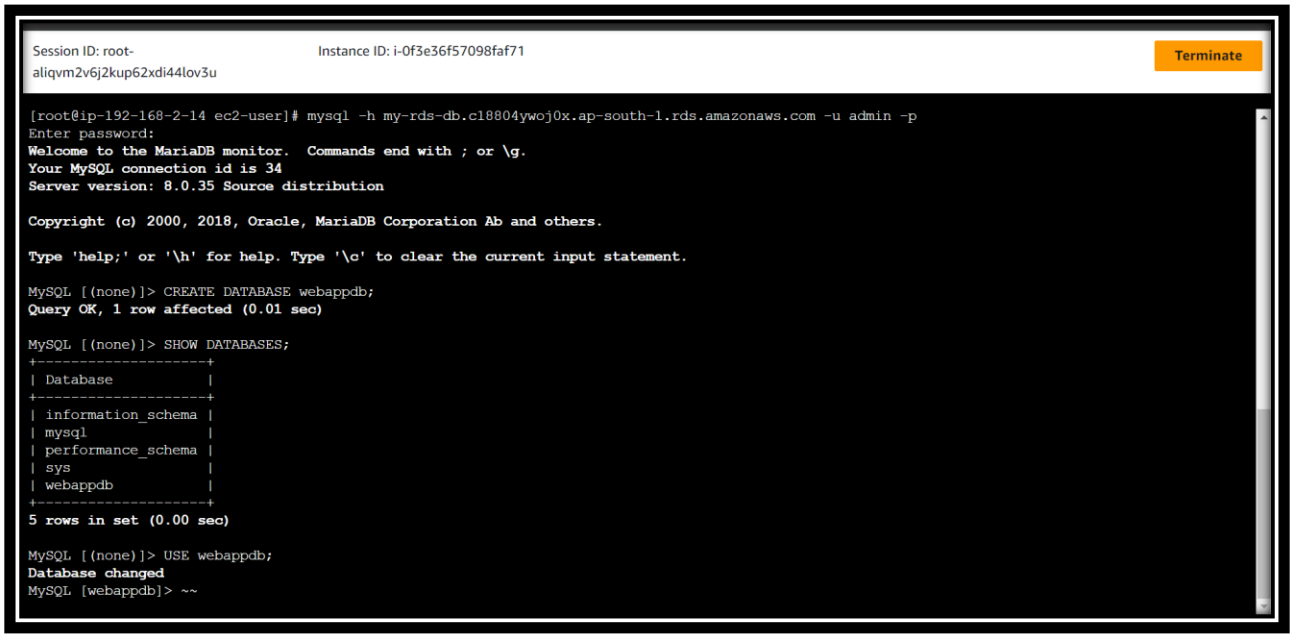
```
CREATE DATABASE webappdb;
SHOW DATABASES;
USE webappdb;

CREATE TABLE IF NOT EXISTS transactions(
  id INT NOT NULL AUTO_INCREMENT,
  amount DECIMAL(10,2),
  description VARCHAR(100),
  PRIMARY KEY(id)
);

SHOW TABLES;
INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');
SELECT * FROM transactions;
```

Three-Tier Web Architecture

Arun M



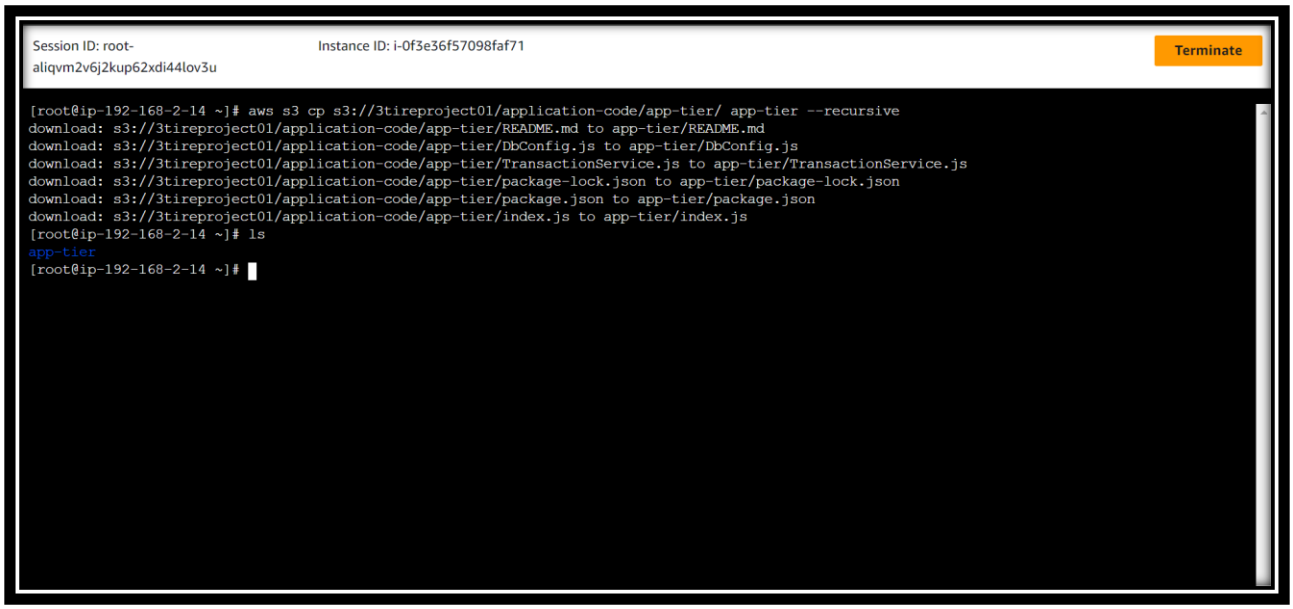
- **Update Application Configuration to with DB information**
- Update the **application-code/app-tier/DbConfig.js** file with your database credentials.



Three-Tier Web Architecture

Arun M

- Copy necessary File from the S3 Bucket. aws s3 cp <<path of the S3 bucket>>.



Install and Configure Node.js and PM2

Install Node.js and PM2:

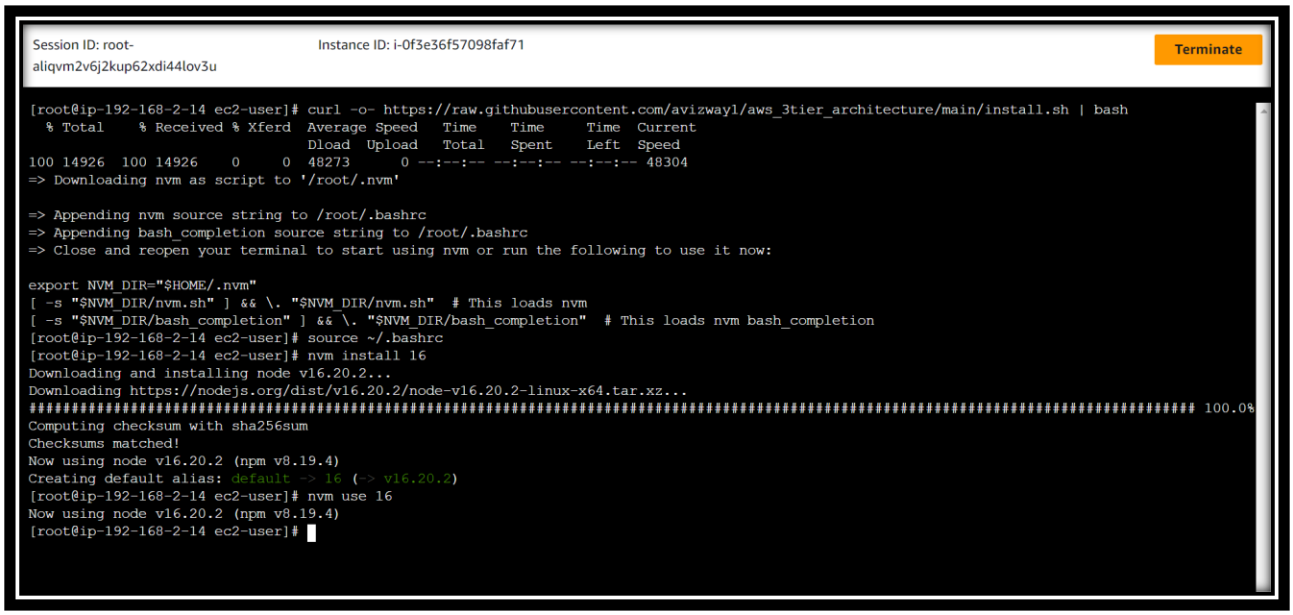
```
curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
source ~/.bashrc

nvm install 16
nvm use 16

npm install -g pm2
```

- Use the above command to configure the Node.js and PM2 from the NVM Node Version Module.

Three-Tier Web Architecture

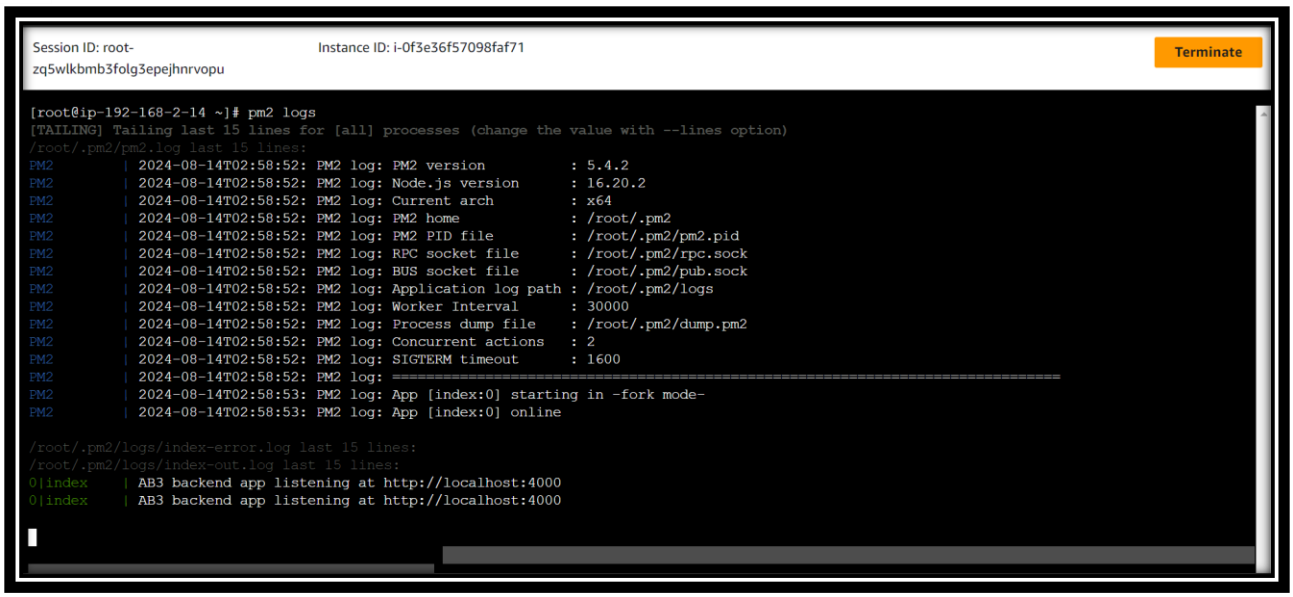
Arun M

```
cd ~/
aws s3 cp s3://3tierproject-avinash/application-code/app-tier/ app-tier --recursive

cd ~/app-tier
npm install
pm2 start index.js

pm2 list
pm2 logs
pm2 startup
pm2 save
```



Three-Tier Web Architecture

Arun M



- Verify that the application is running by executing:

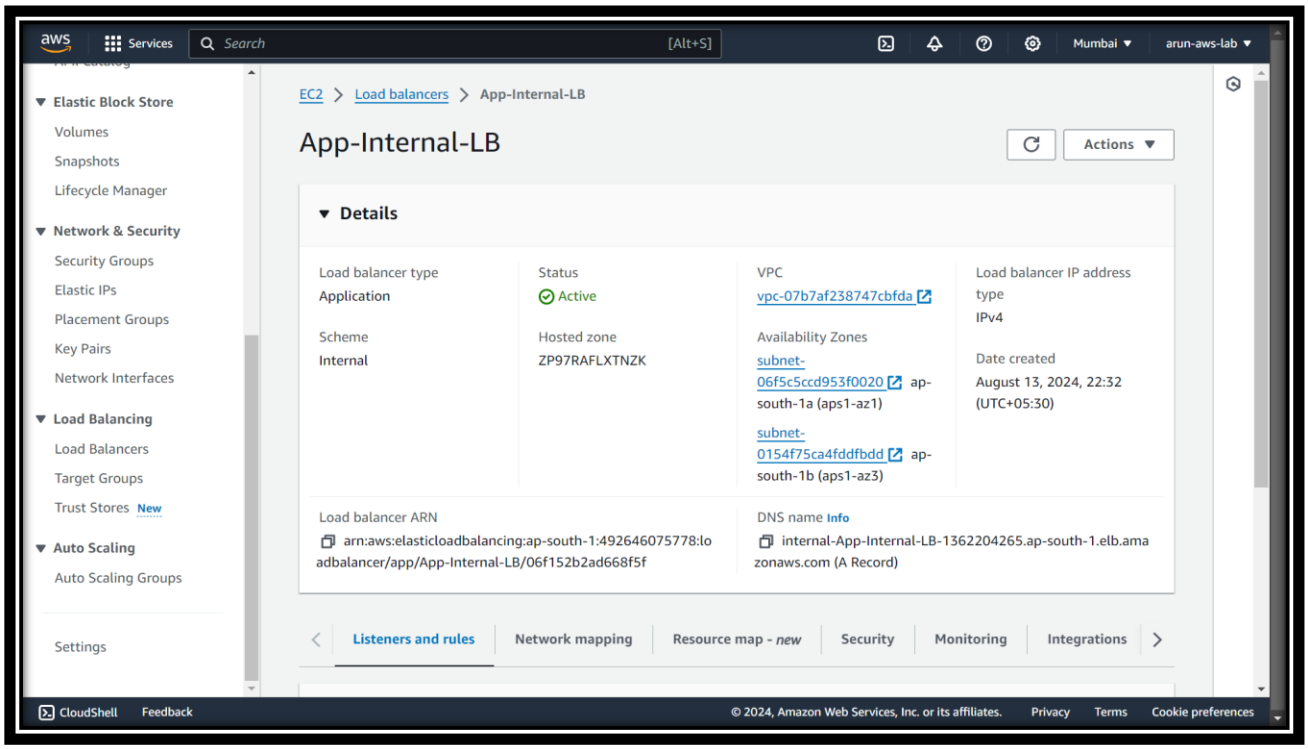
```
curl http://localhost:4000/health
```

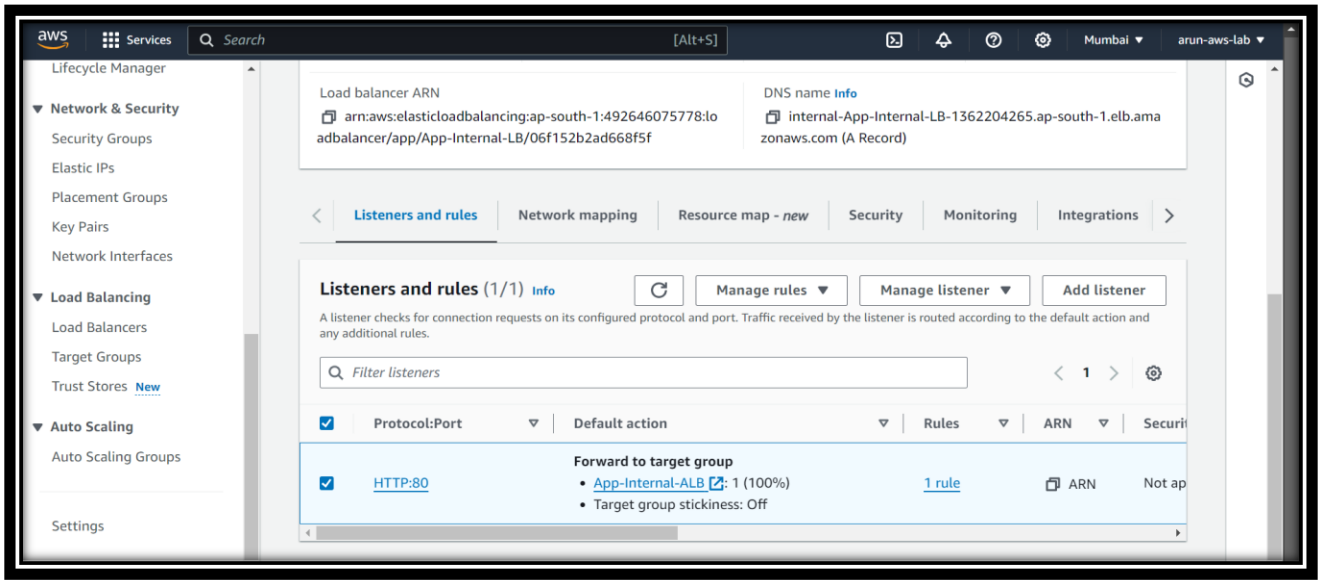
Three-Tier Web Architecture

Arun M

Set up of Internal Load Balancer :

- Create an internal load balancer and update the Nginx configuration with the internal load balancer IP.



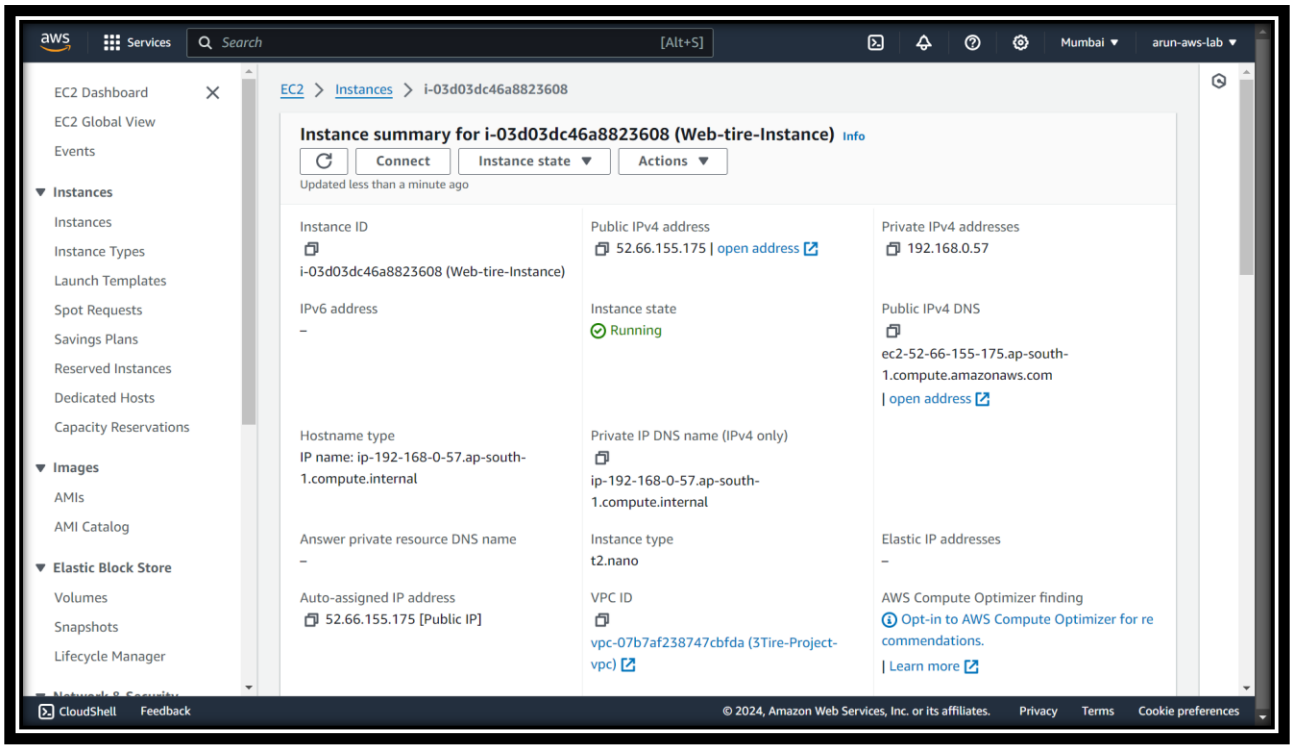


Three-Tier Web Architecture

Arun M

EC2 Instance : Webserver Set up

Web Tier Setup: Launch EC2 Instance in Web Subnets we have created in Custom VPC.

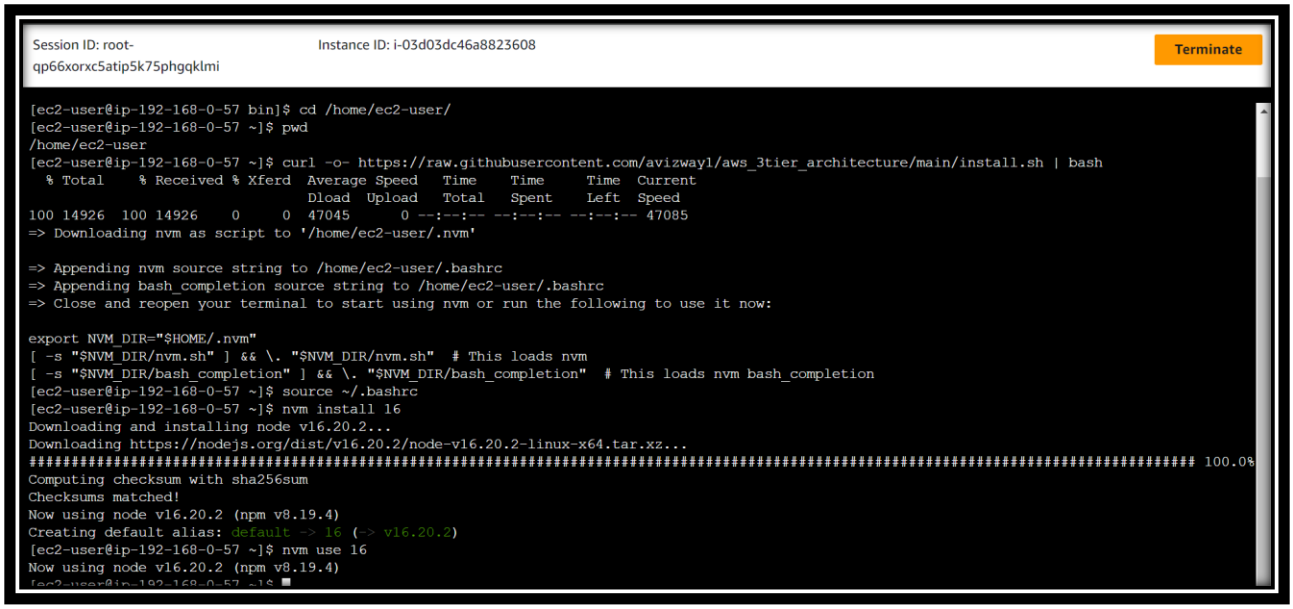



```
curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
source ~/.bashrc
nvm install 16
nvm use 16

cd ~/
aws s3 cp s3://3tierproject-avinash/application-code/web-tier/ web-tier --recursive

cd ~/web-tier
npm install
npm run build

sudo amazon-linux-extras install nginx1 -y
```

- above command used for grep the NVM Package from the S3 bucket to set web Tire configuration.

Three-Tier Web Architecture

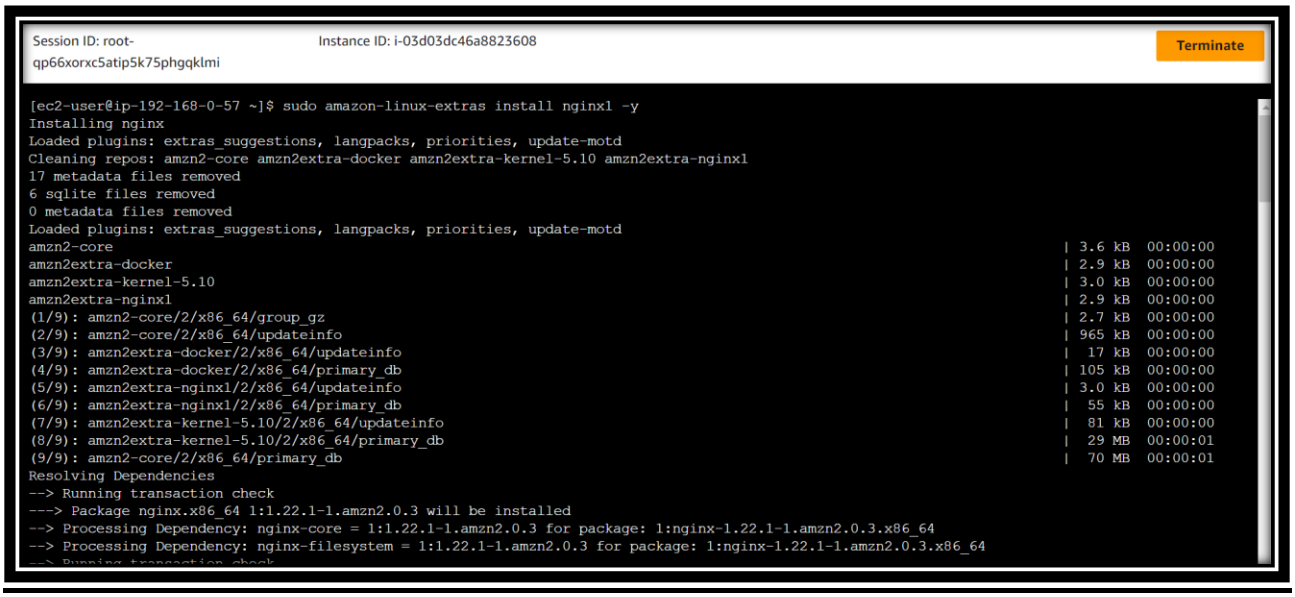
Arun M



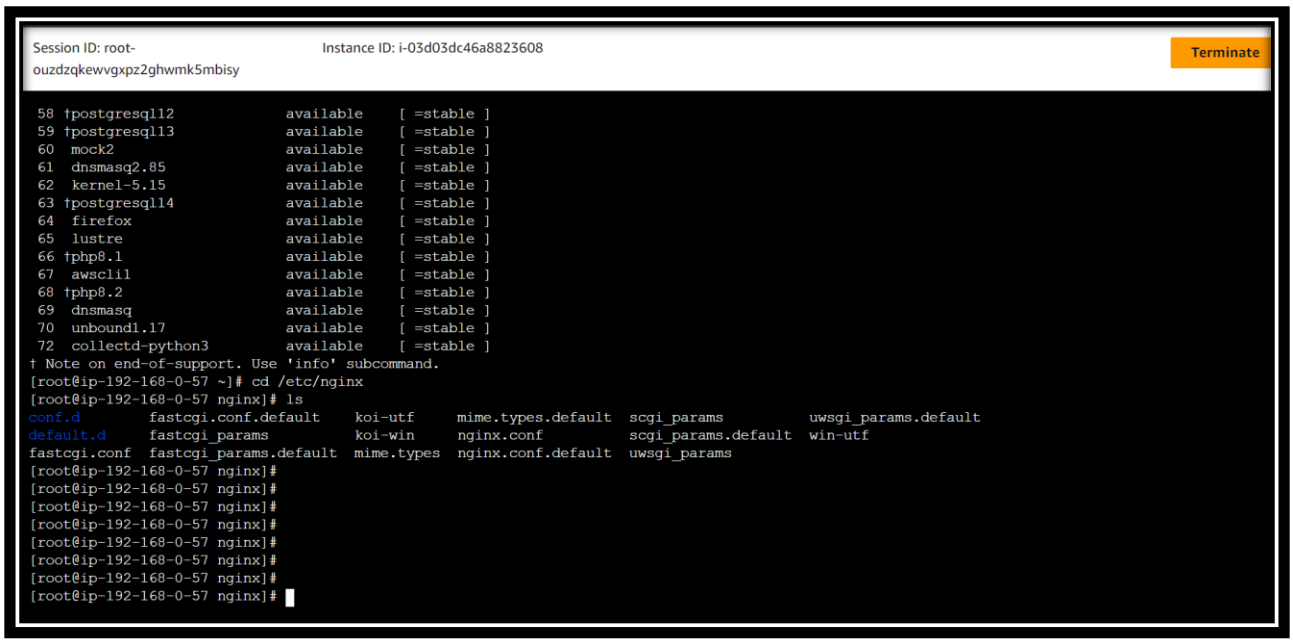
- Copy necessary file form the S3 bucket “aws s3 cp s3://3tierproject/application-code/web-tier/ web-tier –recursive”.

Three-Tier Web Architecture

Arun M



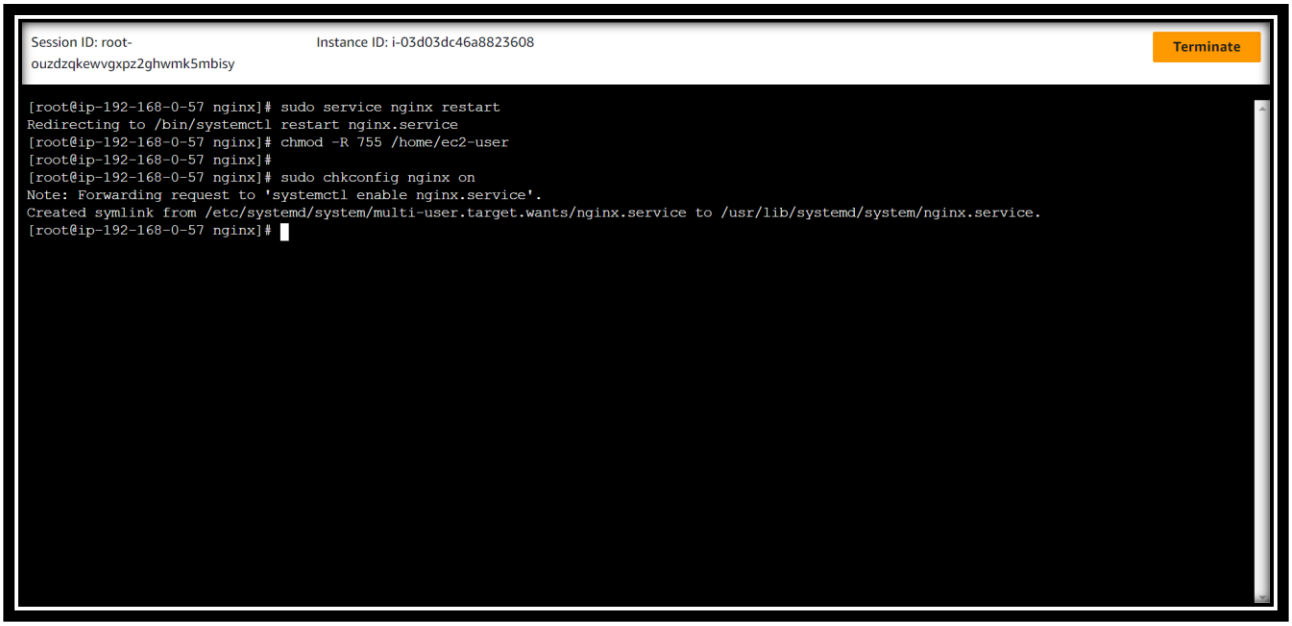
- Ngnix was successfully installed and running in the Web-tire instance.



Three-Tier Web Architecture

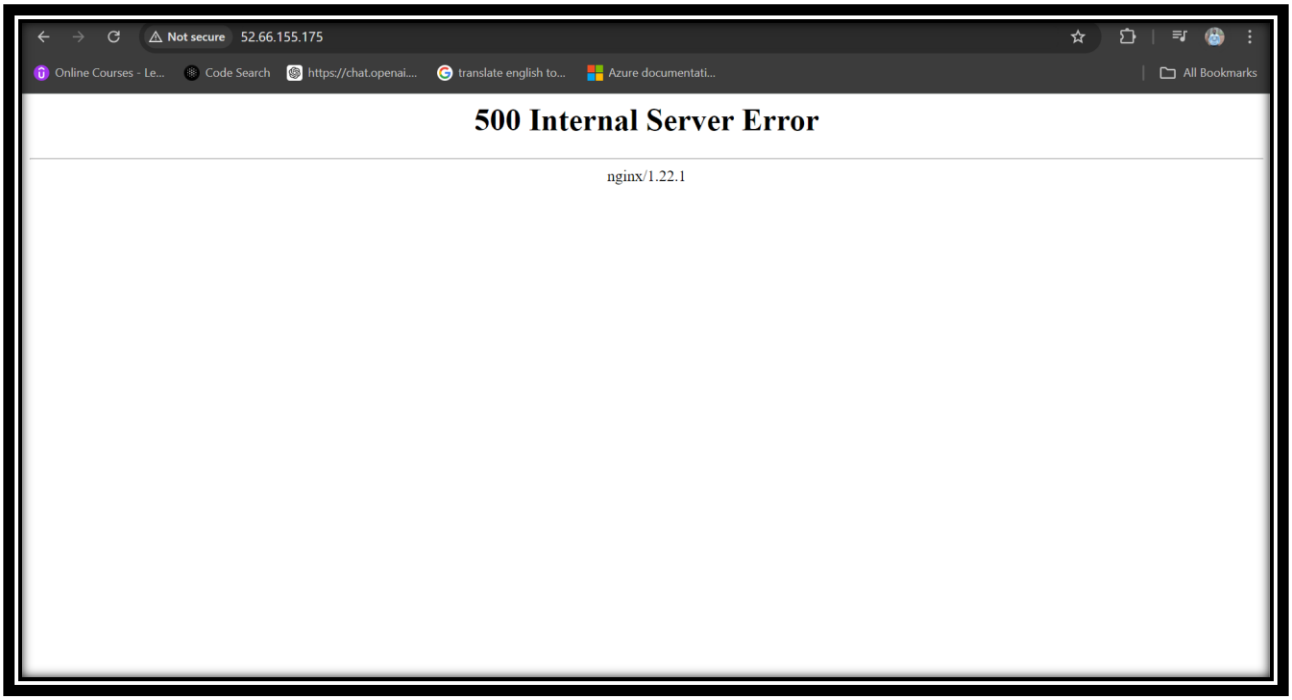
Arun M

- Nginx Was successfully launched and startup was done and read to use as the server.



Veification of Web-Tire-Server :

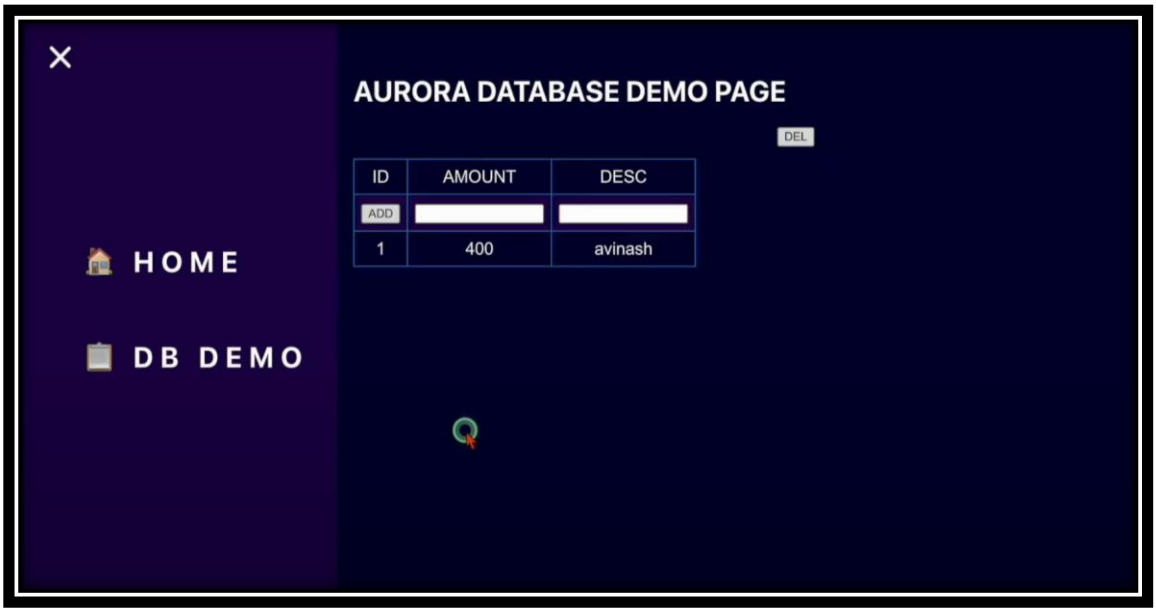
- Our Web-Tire worked perfectly as nginx version, it so 500 internal error because of our app server were downed in state.



Three-Tier Web Architecture

Arun M

- Excepected outputs were generated.

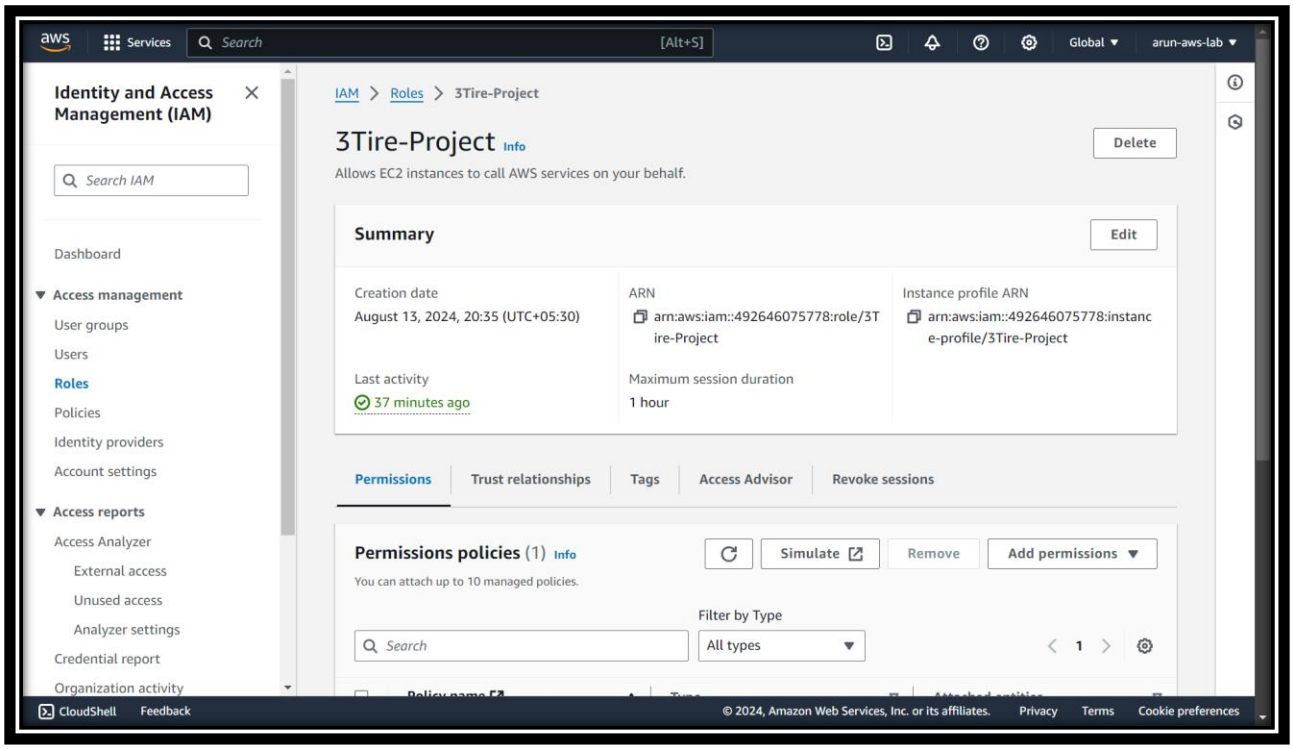


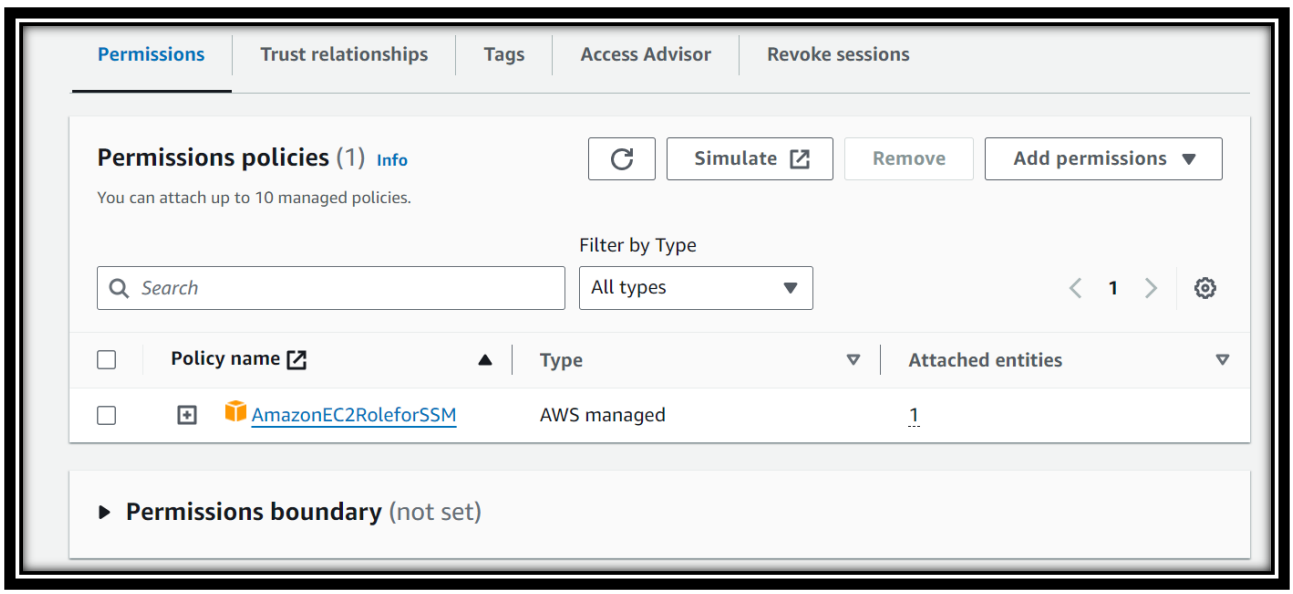
Three-Tier Web Architecture

Arun M

Additional Module SSM role management & S3 bucket.

- First, we create a role management for SSM named as "**3Tire-Project**"
- Attached the permission policy of it "AmazonEC2RoleforSSM"



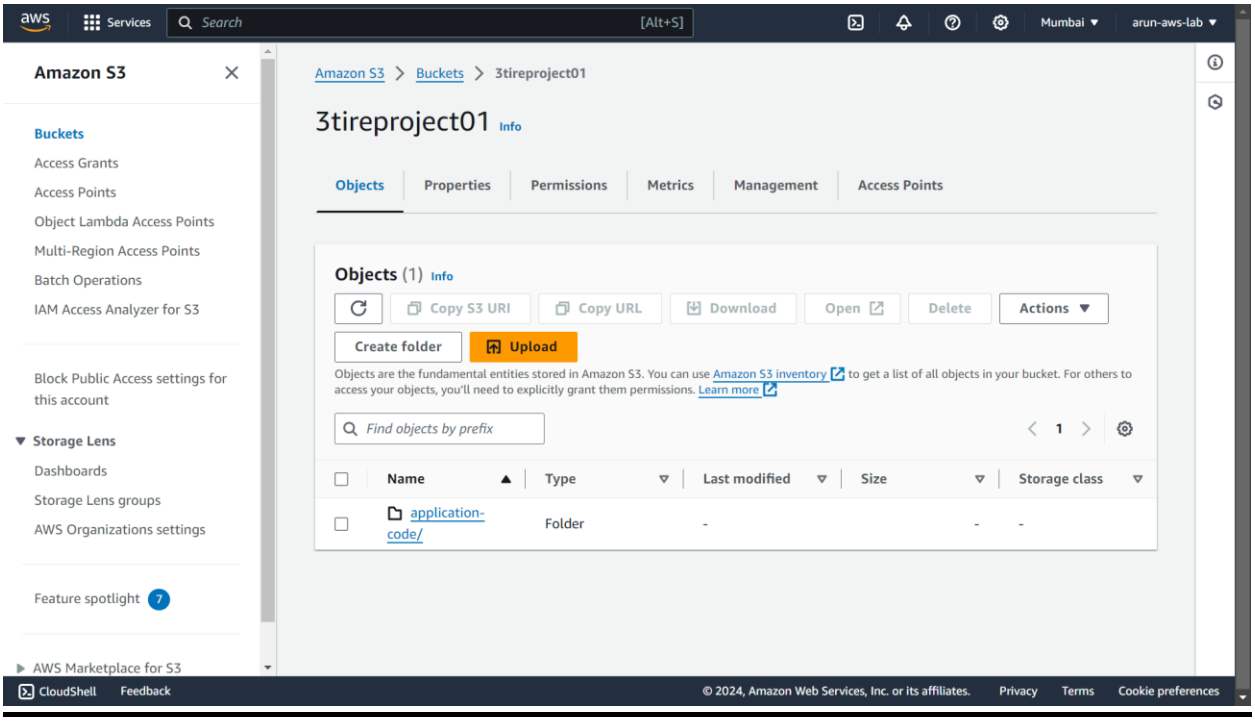


Three-Tier Web Architecture

Arun M

S3 Bucket :

- Main Purpose of S3 bucket is holds the code and necessary file of the web-tire , app-tire & DB-tire.
- S3 Bucket named as “3tireproject01”. I uploaded all the necessary file in the application code and deployed in the s3 bucket.



NOTE : I successfully Implemented the 3-tier architecture in the AWS with low fault tolerance and minimum cost , optimization purposes minimum usage of resource & services of AWS.

Three-Tier Web Architecture

Arun M