

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.

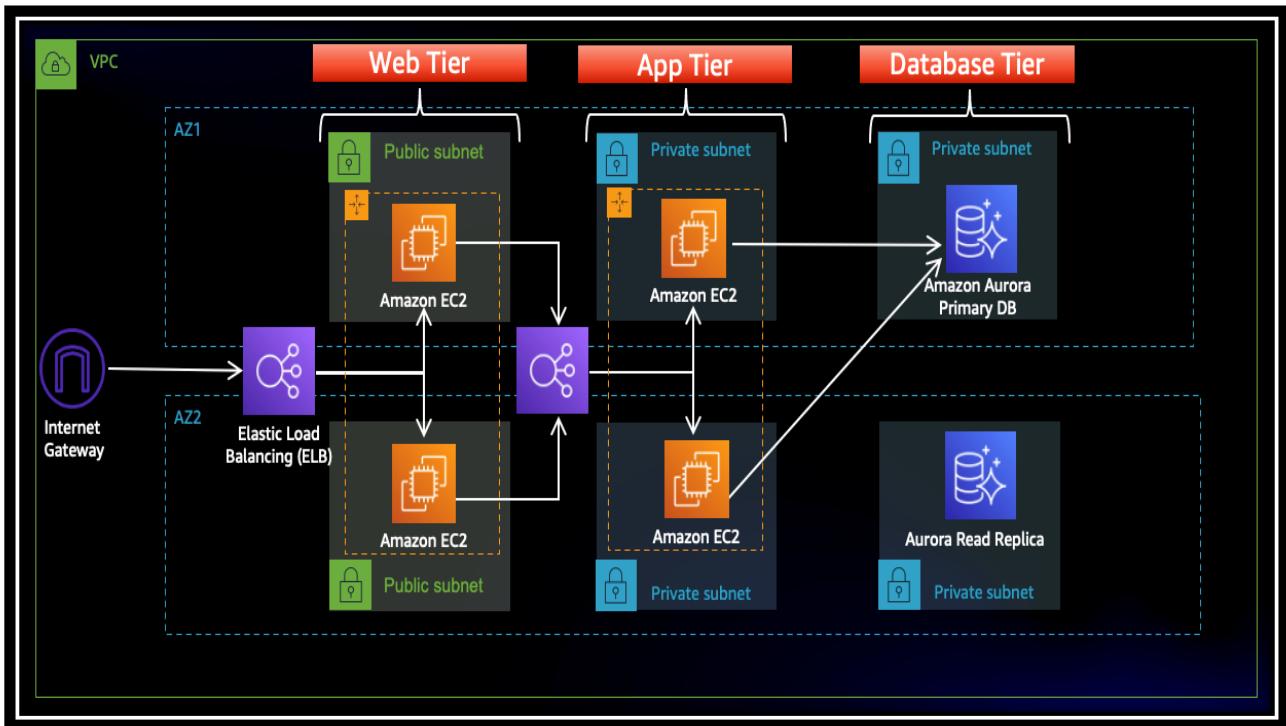
Overview

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

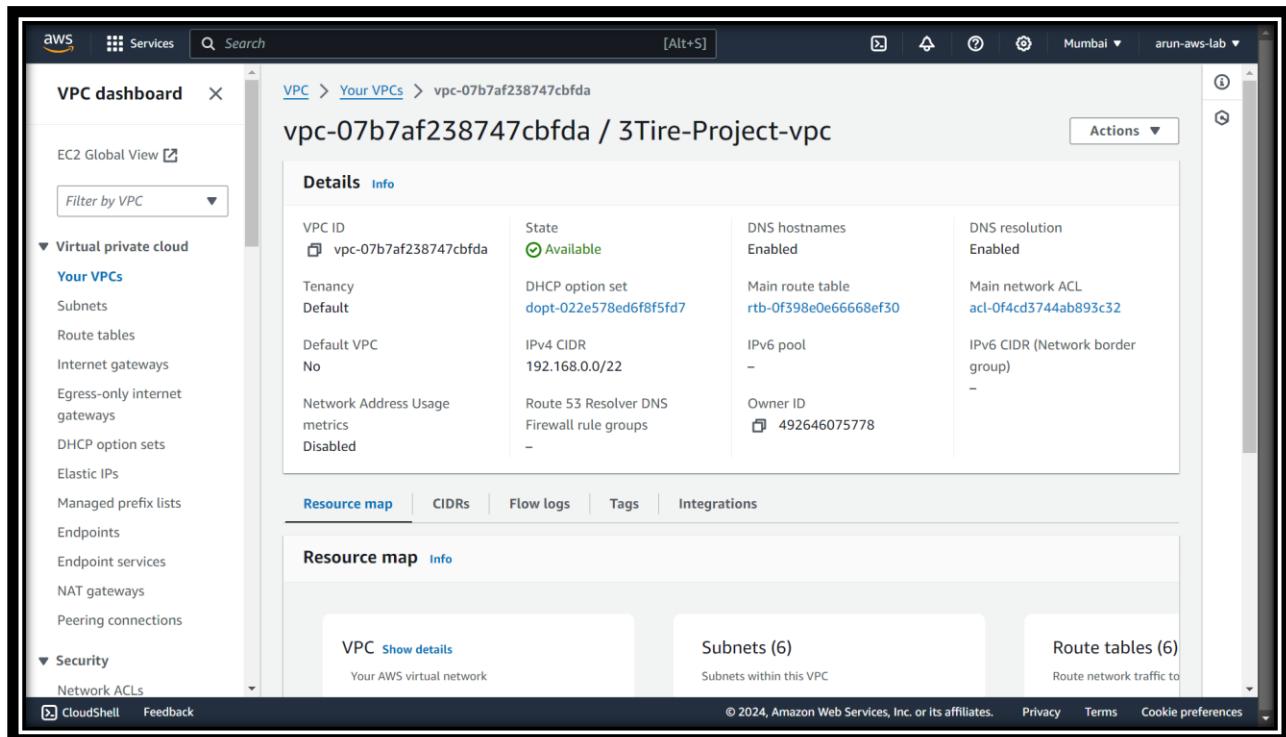
Arun M

- **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 healthy resources if needed.
-

Step-By-Step Process & Screenshot :

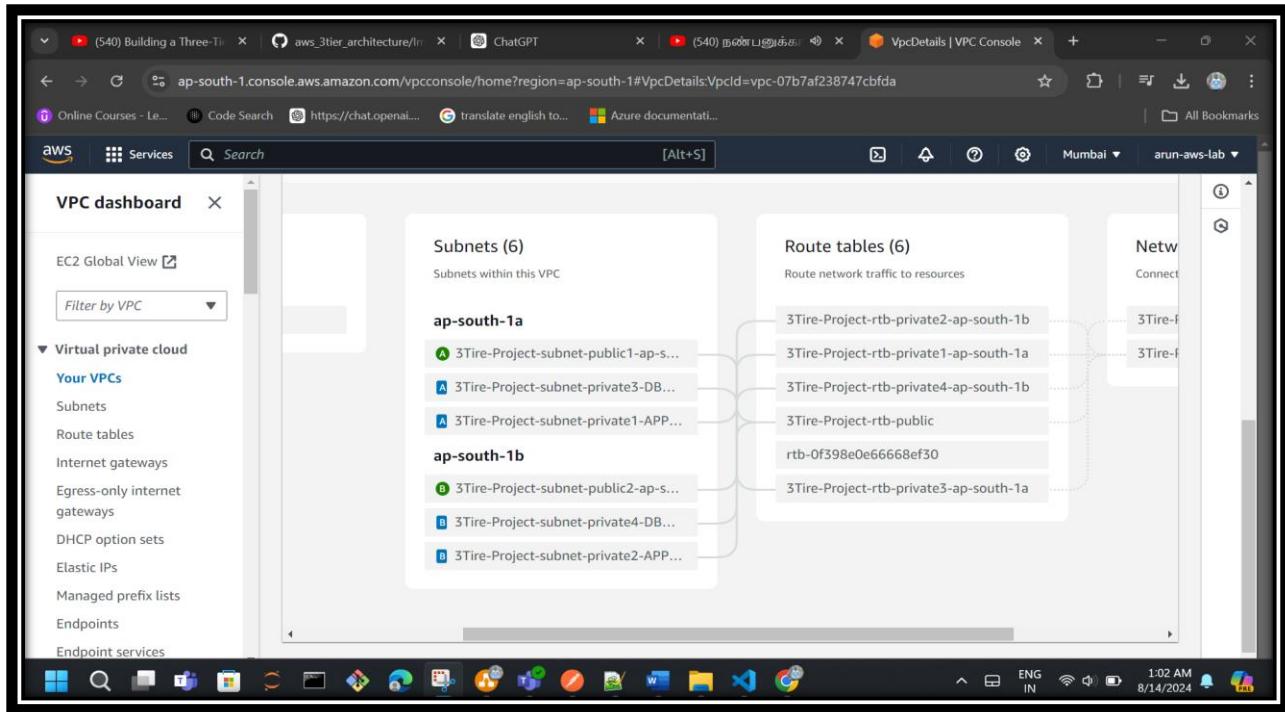
VPC : VPC named as “3Tire-Project-vpc”



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.

The screenshot shows the AWS VPC Subnets page. The sidebar is identical to the one in the previous dashboard. The main table lists 9 subnets:

Name	Subnet ID	State	VPC
3Tire-Project-subnet-public2-ap-south-1b	subnet-0941483efb0ebbb8a	Available	vpc-07b7
3Tire-Project-subnet-public1-ap-south-1a	subnet-01d7892fafc17938b	Available	vpc-07b7
3Tire-Project-subnet-private4-DB2-ap-south-1b	subnet-0014548305d14953b	Available	vpc-07b7
3Tire-Project-subnet-private3-DB1-ap-south-1a	subnet-01e008a777be64c87	Available	vpc-07b7
3Tire-Project-subnet-private2-APP2-ap-south-1b	subnet-0154f75ca4fddfbdd	Available	vpc-07b7
3Tire-Project-subnet-private1-APP1-ap-south-1a	subnet-06f5c5cc953f0020	Available	vpc-07b7
-	subnet-08e3b5dab9b7d6625	Available	vpc-0d7a
-	subnet-0ee73907c0ebb9074	Available	vpc-0d7a
-	subnet-0a955d2dc466488a1	Available	vpc-0d7a

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 :

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. The left sidebar shows navigation options like EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets, Route tables (selected), Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security, Network ACLs, CloudShell, and Feedback.

The main content area displays a table titled 'Route tables (6/7)'. The table has columns for Name, Route table ID, Explicit subnet associations, and Edge associations. The routes listed are:

- (unchecked) rtb-0d85a582d888aa0c9
- (checked) rtb-0f398e0e66668ef30
- 3Tire-Project-rtb-private1-ap-south-1a (checked) rtb-04417a65d05248b9a subnet-06f5c5cccd953f00...
- 3Tire-Project-rtb-private2-ap-south-1b (checked) rtb-0fe77e63a21a462a9 subnet-0154f75ca4fddfb...
- 3Tire-Project-rtb-private3-ap-south-1a (checked) rtb-04a640445026244e7 subnet-01e008a777be64...
- 3Tire-Project-rtb-private4-ap-south-1b (checked) rtb-01cb61f6af0f302ed subnet-0014548305d149...
- 3Tire-Project-rtb-public (checked) rtb-06c4a8038adb95017 2 subnets

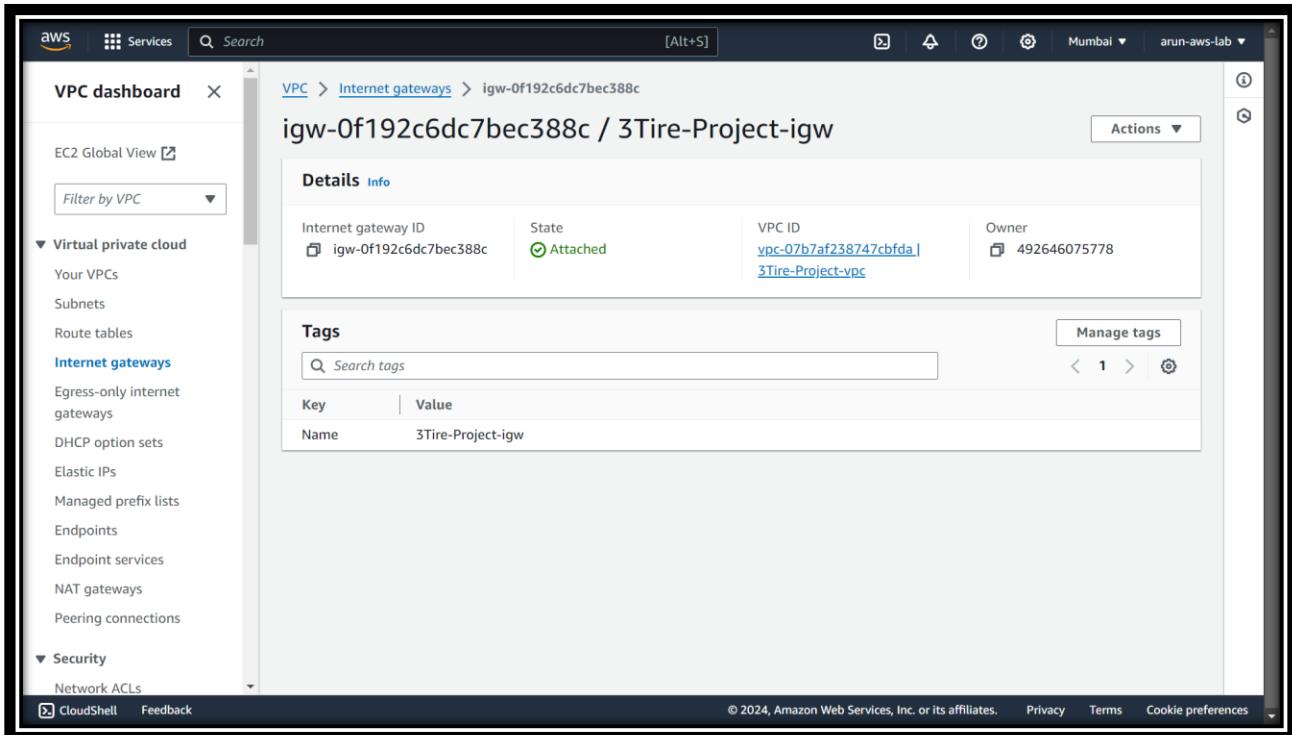
At the bottom of the table, it says 'Route tables: rtb-0fe77e63a21a462a9, rtb-04417a65d05248b9a, rtb-01cb61f6af0f302ed, rtb-06c4a8038adb95017, rtb-04a640445026244e7, rtb-0f398e0e66668ef30'

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

The screenshot shows the AWS VPC console with the path: VPC > Elastic IP addresses > 13.201.255.190. The main content area displays the following summary information:

Allocated IPv4 address	Type	Allocation ID	Reverse DNS record
13.201.255.190	Public IP	eipalloc-Oce2420f19681da27	-
Association ID	Scope	Associated instance ID	Private IP address
eipassoc-09f5909264550c751	VPC	-	192.168.0.17
Network interface ID	Network interface owner account ID	Public DNS	NAT Gateway ID
eni-0da67d8e85d6c07e7	492646075778	ec2-13-201-255-190.ap-south-1.compute.amazonaws.com	nat-083c3a24f08bc9651 (3Tire-Project-nat-public1-ap-south-1a)
Address pool	Network border group		
Amazon	ap-south-1		

Below the summary, there is a section for Tags(1) with a Manage tags button. The page footer includes standard AWS navigation links.

NAT Gateway :

The screenshot shows the AWS VPC console with the path: VPC > NAT gateways > nat-083c3a24f08bc9651. The main content area displays the following details:

NAT gateway ID	Connectivity type	State	State message
nat-083c3a24f08bc9651	Public	Available	Info
NAT gateway ARN	Primary public IPv4 address	Primary private IPv4 address	Primary network interface ID
arn:aws:ec2:ap-south-1:492646075778:natgateway/nat-083c3a24f08bc9651	13.201.255.190	192.168.0.17	eni-0da67d8e85d6c07e7
VPC	Subnet	Created	Deleted
vpc-07b7af238747cbfda / 3Tire-Project-vpc	subnet-01d7892fafc17938b / 3Tire-Project-subnet-public1-ap-south-1a	Tuesday, August 13, 2024 at 20:17:52 GMT+5:30	-

Below the details, there are tabs for Secondary IPv4 addresses, Monitoring, and Tags. The Secondary IPv4 addresses tab shows a table with columns for Private IPv4 address, Network interface ID, and Status. The page footer includes standard AWS navigation links.

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

The screenshot shows the AWS VPC Security Groups console. The left sidebar is collapsed. The main area displays the details for the security group 'sg-0be37caa143e324aa - Web-App-LB-SG'. The 'Details' section includes:

Security group name	Security group ID	VPC ID
Web-App-LB-SG	sg-0be37caa143e324aa	vpc-07b7af238747cbfda

Owner: 492646075778, Inbound rules count: 2 Permission entries, Outbound rules count: 1 Permission entry.

Below the details, there are tabs for 'Inbound rules', 'Outbound rules', and 'Tags'. The 'Inbound rules' tab is selected, showing 'Inbound rules (2)'. At the bottom right of the main area are buttons for 'Manage tags' and 'Edit inbound rules'.

“WEB-SG” : Security Group for Webserver

The screenshot shows the AWS VPC Security Groups console. The left sidebar is collapsed. The main area displays the details for the security group 'sg-017a677a297b1ee02 - WEB-SG'. The 'Details' section includes:

Security group name	Security group ID	Description	VPC ID
WEB-SG	sg-017a677a297b1ee02	SG for Web-Server	vpc-07b7af238747cbfda

Owner: 492646075778, Inbound rules count: 2 Permission entries, Outbound rules count: 1 Permission entry.

Below the details, there are tabs for 'Inbound rules', 'Outbound rules', and 'Tags'. The 'Inbound rules' tab is selected, showing 'Inbound rules (2)'. At the bottom right of the main area are buttons for 'Manage tags' and 'Edit inbound rules'.

Three-Tier Web Architecture

Arun M

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

Details			
Security group name	sg-0188dd2936f9dad0	Security group ID	sg-0188dd2936f9dad0
Owner	492646075778	Description	Application server SG
		Inbound rules count	1 Permission entry
		Outbound rules count	1 Permission entry

Inbound rules (1) Manage tags Edit inbound rules

“RDS-SG” : Security Group for Database

Details			
Security group name	sg-012e11e95a0b54485	Security group ID	sg-012e11e95a0b54485
Owner	492646075778	Description	Database Security Group
		Inbound rules count	1 Permission entry
		Outbound rules count	1 Permission entry

Inbound rules (1) Manage tags Edit inbound rules

Three-Tier Web Architecture

Arun M

“Internal-LB-SG” : Internal-LB-SG

The screenshot shows the AWS VPC Security Groups console. The left sidebar is collapsed. The main area displays the details for a security group named "Internal-LB-SG".

Details			
Security group name Internal-LB-SG	Security group ID sg-Occab6e4e03d47311	Description SG for Internal Load Balancer	VPC ID VPC-07b7af238747cbfda
Owner 492646075778	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	
Inbound rules Outbound rules Tags			
Tags			
Manage tags			

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

The screenshot shows the AWS RDS Subnet Group details page. On the left, the navigation menu includes options like Dashboard, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main content area displays the Subnet group details, including the VPC ID (vpc-07b7af238747cbfda), ARN (arn:aws:rds:ap-south-1:492646075778:subgrp:three-tier-subnet-group), Supported network types (IPv4), and Description (3Tier-Subnet-Group). Below this, a table lists two subnets: ap-south-1a (subnet-01e008a777be64c87) with CIDR block 192.168.2.128/26 and ap-south-1b (subnet-0014548305d14953b) with CIDR block 192.168.2.192/26. A section for Tags (0) is present with a Manage tags button.

DataBase

The screenshot shows the AWS RDS Database details page for 'my-rds-db'. The navigation menu is identical to the previous screenshot. The main content area shows the database summary, including the DB identifier (my-rds-db), Status (Available), Role (Instance), Engine (MySQL Community), and Region & AZ (ap-south-1a). Below the summary, tabs for Connectivity & security, Monitoring, Logs & events, Configuration, and Maintenance & backups are visible. The Connectivity & security tab is selected, displaying the Endpoint & port, Networking, and Security sections. The Endpoint is listed as my-rds-db.c18804ywoj.0.ap-south-1.rds.amazonaws. The Networking section shows the Availability Zone (ap-south-1a) and VPC. The Security section lists the VPC security groups (RDS-SG (sg-012e11e95a0b54485)).

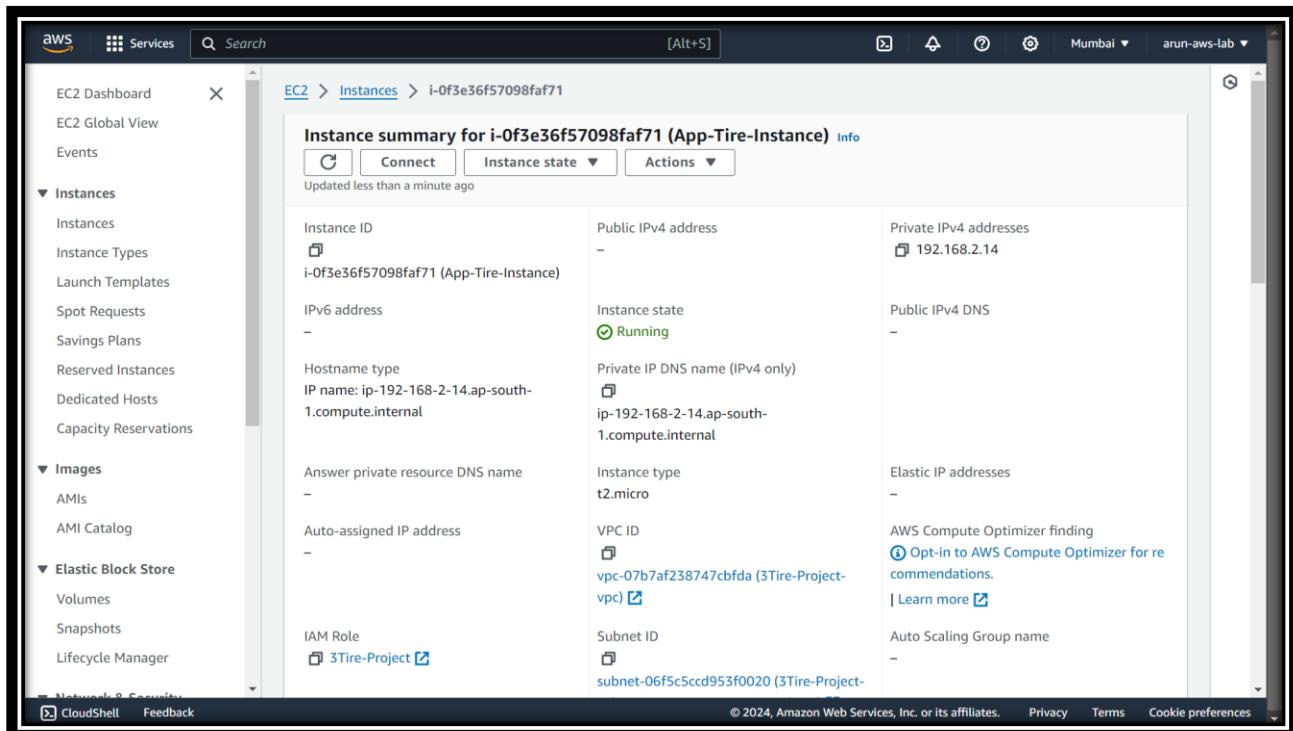
Three-Tier Web Architecture

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
```

```
Session ID: root-  
aliquvm2v6j2kup62xd44lov3u  
Instance ID: i-0f3e36f57098faf71  
Termination  
Installing:  
mariadb  
x86_64  
1:5.5.68-1.amzn2.0.1  
amzn2-core  
8.8 M  
Transaction Summary  
Install 1 Package  
Total download size: 8.8 M  
Installed size: 49 M  
Downloading packages:  
mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm  
Running transaction check  
Running transaction test  
Transaction test succeeded  
Running transaction  
Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64  
Verifying : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64  
1/1  
1/1  
Installed:  
mariadb.x86_64 1:5.5.68-1.amzn2.0.1  
Complete!  
[root@ip-192-168-2-14 ec2-user]#  
[root@ip-192-168-2-14 ec2-user]#  
[root@ip-192-168-2-14 ec2-user]#  
[root@ip-192-168-2-14 ec2-user]# mysql --version  
mysql Ver 15.1 Distrib 5.5.68-MariaDB, for Linux (x86_64) using readline 5.1  
[root@ip-192-168-2-14 ~]#
```

- 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

Session ID: root-
alqvm2v6j2kup62xd44lo3u Instance ID: i-0f3e36f57098faf71

mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64
Verifying : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64

Installed:
mariadb.x86_64 1:5.5.68-1.amzn2.0.1

Complete!
[root@ip-192-168-2-14 ec2-user]# [root@ip-192-168-2-14 ec2-user]# [root@ip-192-168-2-14 ec2-user]# [root@ip-192-168-2-14 ec2-user]# mysql --version
mysql Ver 15.1 Distrib 5.5.68-MariaDB, for Linux (x86_64) using readline 5.1
[root@ip-192-168-2-14 ec2-user]# mysql -h my-rds-db.c18804ywoj0x.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 8.0.35 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)]>

- 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

```

Session ID: root-
Instance ID: i-0f3e36f57098faf71
aliquvm2v6j2kup62xd44lovsu
Terminate

| transactions      |
+-----+
1 row in set (0.00 sec)

MySQL [webappdb]> INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');
Query OK, 1 row affected (0.01 sec)

MySQL [webappdb]> SELECT * FROM transactions;
+----+-----+
| id | amount | description |
+----+-----+
| 1  | 400.00 | groceries   |
+----+-----+
1 row in set (0.00 sec)

MySQL [webappdb]> INSERT INTO transactions (amount, description) VALUES ('400', 'ARUN');
Query OK, 1 row affected (0.00 sec)

MySQL [webappdb]> SELECT * FROM transactions;
+----+-----+
| id | amount | description |
+----+-----+
| 1  | 400.00 | groceries   |
| 2  | 400.00 | ARUN        |
+----+-----+
2 rows in set (0.00 sec)

MySQL [webappdb]>

```

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

```

Session ID: root-
Instance ID: i-0f3e36f57098faf71
aliquvm2v6j2kup62xd44lovsu
Terminate

[root@ip-192-168-2-14 ec2-user]# mysql -h my-rds-db.c18804ywoj0x.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 8.0.35 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)]> CREATE DATABASE webappdb;
Query OK, 1 row affected (0.01 sec)

MySQL [(none)]> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
| webappdb       |
+-----+
5 rows in set (0.00 sec)

MySQL [(none)]> USE webappdb;
Database changed
MySQL [webappdb]> ~

```

Three-Tier Web Architecture

Arun M

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

```

Session ID: root-
Instance ID: i-0f3e36f57098faf71
aliquvm2v6j2kup62xd44lovsu
Terminate

[root@ip-192-168-2-14 ~]# aws s3 cp s3://3tireproject01/application-code/app-tier/ app-tier --recursive
download: s3://3tireproject01/application-code/app-tier/README.md to app-tier/README.md
download: s3://3tireproject01/application-code/app-tier/DbConfig.js to app-tier/DbConfig.js
download: s3://3tireproject01/application-code/app-tier/TransactionService.js to app-tier/TransactionService.js
download: s3://3tireproject01/application-code/app-tier/package-lock.json to app-tier/package-lock.json
download: s3://3tireproject01/application-code/app-tier/package.json to app-tier/package.json
download: s3://3tireproject01/application-code/app-tier/index.js to app-tier/index.js
[root@ip-192-168-2-14 ~]# ls
app-tier
[root@ip-192-168-2-14 ~]#

```

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
```

Session ID: root-
Instance ID: i-0f3e36f57098faf71

Terminate

```
[root@ip-192-168-2-14 ec2-user]# curl -o- https://raw.githubusercontent.com/avizwayl/aws_3tier_architecture/main/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload   Total Spent   Left  Speed
100 14926  100 14926    0      0  48273      0 --:--:-- --:--:-- 48304
=> Downloading nvm as script to '/root/.nvm'

=> Appending nvm source string to /root/.bashrc
=> Appending bash_completion source string to /root/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[root@ip-192-168-2-14 ec2-user]# source ~/.bashrc
[root@ip-192-168-2-14 ec2-user]# nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (-> v16.20.2)
[root@ip-192-168-2-14 ec2-user]# nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[root@ip-192-168-2-14 ec2-user]#
```

Three-Tier Web Architecture

Arun M

Session ID: root-zq5wlkbmb3folg3epejhnrvoju Instance ID: i-0f3e36f57098faf71 Terminate

```
[root@ip-192-168-2-14 ~]# pm2 logs
[TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)
/root/.pm2/.pm2.log last 15 lines:
PM2    | 2024-08-14T02:58:52: PM2 log: PM2 version      : 5.4.2
PM2    | 2024-08-14T02:58:52: PM2 log: Node.js version   : 16.20.2
PM2    | 2024-08-14T02:58:52: PM2 log: Current arch     : x64
PM2    | 2024-08-14T02:58:52: PM2 log: PM2 home        : /root/.pm2
PM2    | 2024-08-14T02:58:52: PM2 log: PM2 PID file     : /root/.pm2/pm2.pid
PM2    | 2024-08-14T02:58:52: PM2 log: RPC socket file  : /root/.pm2/rpc.sock
PM2    | 2024-08-14T02:58:52: PM2 log: BUS socket file  : /root/.pm2/pub.sock
PM2    | 2024-08-14T02:58:52: PM2 log: Application log path: /root/.pm2/logs
PM2    | 2024-08-14T02:58:52: PM2 log: Worker Interval   : 30000
PM2    | 2024-08-14T02:58:52: PM2 log: Process dump file : /root/.pm2/dump.pm2
PM2    | 2024-08-14T02:58:52: PM2 log: Concurrent actions: 2
PM2    | 2024-08-14T02:58:52: PM2 log: SIGTERM timeout   : 1600
PM2    | 2024-08-14T02:58:52: PM2 log: -----
PM2    | 2024-08-14T02:58:53: PM2 log: App [index:0] starting in -fork mode-
PM2    | 2024-08-14T02:58:53: PM2 log: App [index:0] online

/root/.pm2/logs/index-error.log last 15 lines:
/root/.pm2/logs/index-out.log last 15 lines:
0|index  | AB3 backend app listening at http://localhost:4000
0|index  | AB3 backend app listening at http://localhost:4000
```

- Verify that the application is running by executing:

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

✖

Session ID: root-jrthu3nckluecnvkwh43fjogwu Instance ID: i-0f3e36f57098faf71

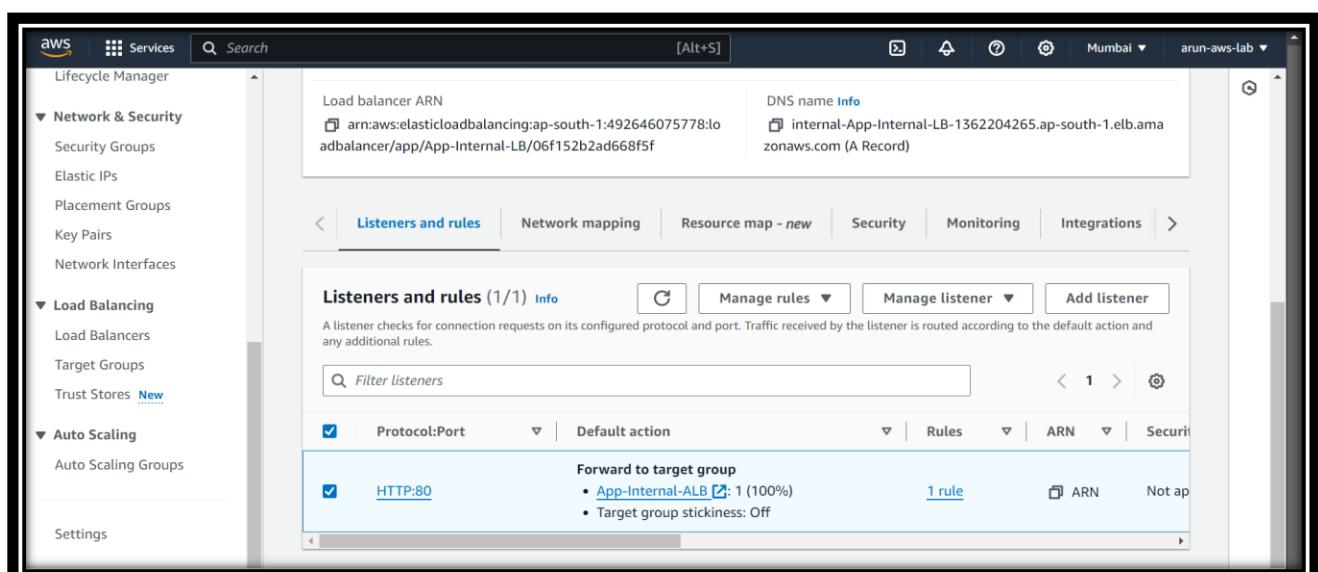
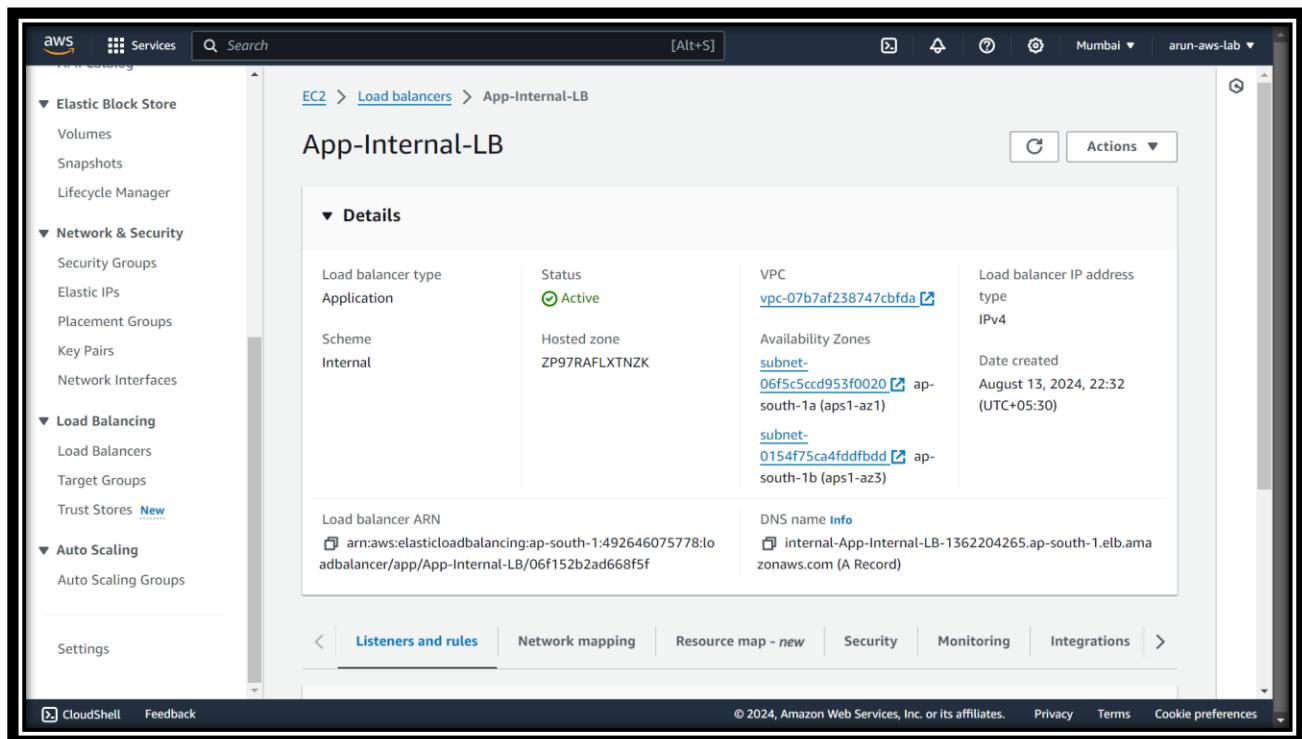
```
sh-4.2$ sudo su
[root@ip-192-168-2-14 bin]# cd
[root@ip-192-168-2-14 ~]# curl http://localhost:4000/health
"This is the health check"[root@ip-192-168-2-14 ~]#
```

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.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-03d03dc46a8823608 (Web-tier-Instance)	52.66.155.175 open address	192.168.0.57
IPv6 address	-	Public IPv4 DNS ec2-52-66-155-175.ap-south-1.compute.amazonaws.com open address
Hostname type IP name: ip-192-168-0-57.ap-south-1.compute.internal	Instance state Running	Private IP DNS name (IPv4 only) ip-192-168-0-57.ap-south-1.compute.internal
Answer private resource DNS name -	Instance type t2.nano	Elastic IP addresses -
Auto-assigned IP address 52.66.155.175 [Public IP]	VPC ID vpc-07b7af238747cbfda (3Tier-Project-vpc)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more

```
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

```
Session ID: root-qp66xorxc5atip5k75phgqklmi           Instance ID: i-03d03dc46a8823608
Terminate

[ec2-user@ip-192-168-0-57 bin]$ cd /home/ec2-user/
[ec2-user@ip-192-168-0-57 ~]$ pwd
/home/ec2-user
[ec2-user@ip-192-168-0-57 ~]$ curl -o https://raw.githubusercontent.com/avizwayl/aws_3tier_architecture/main/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload Total Spent   Left Speed
100 14926  100 14926     0      0  47045  0 --:--:-- --:--:-- 47085
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-192-168-0-57 ~]$ source ~/.bashrc
[ec2-user@ip-192-168-0-57 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloaded https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (-> v16.20.2)
[ec2-user@ip-192-168-0-57 ~]$ nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-192-168-0-57 ~]$
```

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

```
Session ID: root-qp66xorxc5atip5k75phgqklmi           Instance ID: i-03d03dc46a8823608
Terminate

[ec2-user@ip-192-168-0-57 ~]$ ls
web-tier
[ec2-user@ip-192-168-0-57 ~]$
```

Three-Tier Web Architecture

Arun M

```
Session ID: root- Instance ID: i-03d03dc46a8823608
qp66x0rxc5atip5k75phgqklmi
Terminate

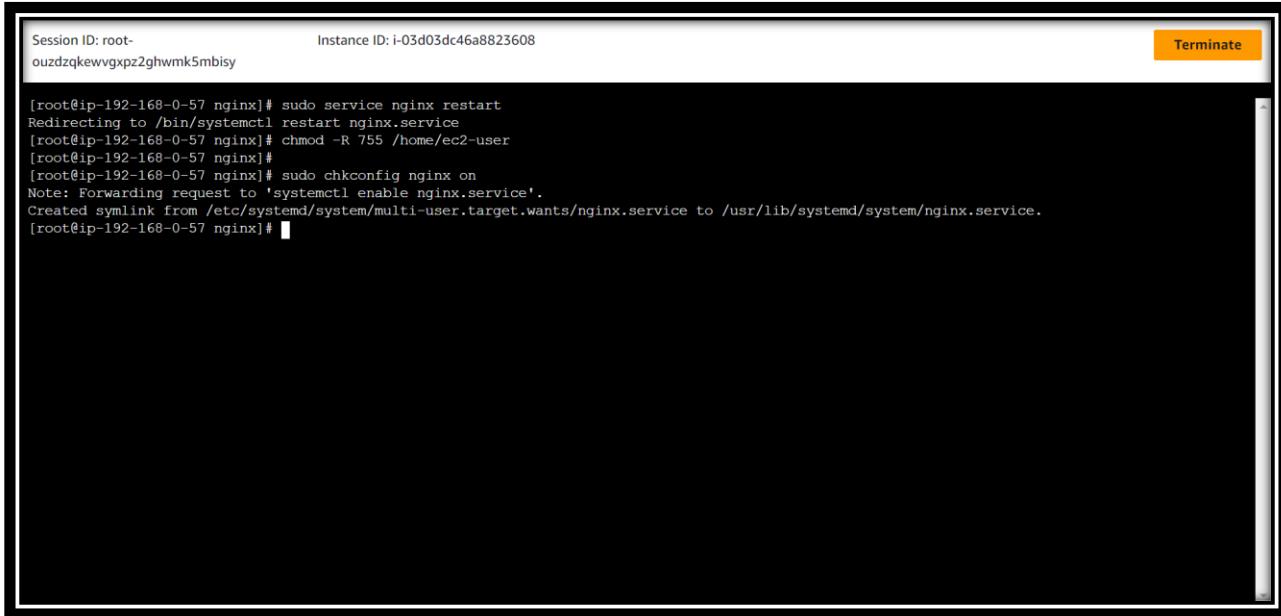
[ec2-user@ip-192-168-0-57 ~]$ sudo amazon-linux-extras install nginx -y
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10 amzn2extra-nginx
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-kernel-5.10
amzn2extra-nginx
(1/9): amzn2-core/2/x86_64/group.gz | 3.6 kB 00:00:00
(2/9): amzn2-core/2/x86_64/updateinfo | 2.9 kB 00:00:00
(3/9): amzn2extra-docker/2/x86_64/updateinfo | 3.0 kB 00:00:00
(4/9): amzn2extra-docker/2/x86_64/primary_db | 2.9 kB 00:00:00
(5/9): amzn2extra-nginx/2/x86_64/updateinfo | 2.7 kB 00:00:00
(6/9): amzn2extra-nginx/2/x86_64/primary_db | 965 kB 00:00:00
(7/9): amzn2extra-kernel-5.10/2/x86_64/updateinfo | 17 kB 00:00:00
(8/9): amzn2extra-kernel-5.10/2/x86_64/primary_db | 105 kB 00:00:00
(9/9): amzn2-core/2/x86_64/primary_db | 3.0 kB 00:00:00
Resolving Dependencies
--> Running transaction check
--> Package nginx.x86_64 1:1.22.1-1.amzn2.0.3 will be installed
--> Processing Dependency: nginx-core = 1:1.22.1-1.amzn2.0.3 for package: 1:nginx-1.22.1-1.amzn2.0.3.x86_64
--> Processing Dependency: nginx-filesystem = 1:1.22.1-1.amzn2.0.3 for package: 1:nginx-1.22.1-1.amzn2.0.3.x86_64
--> Dimension + dependencies can change
```

- 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.



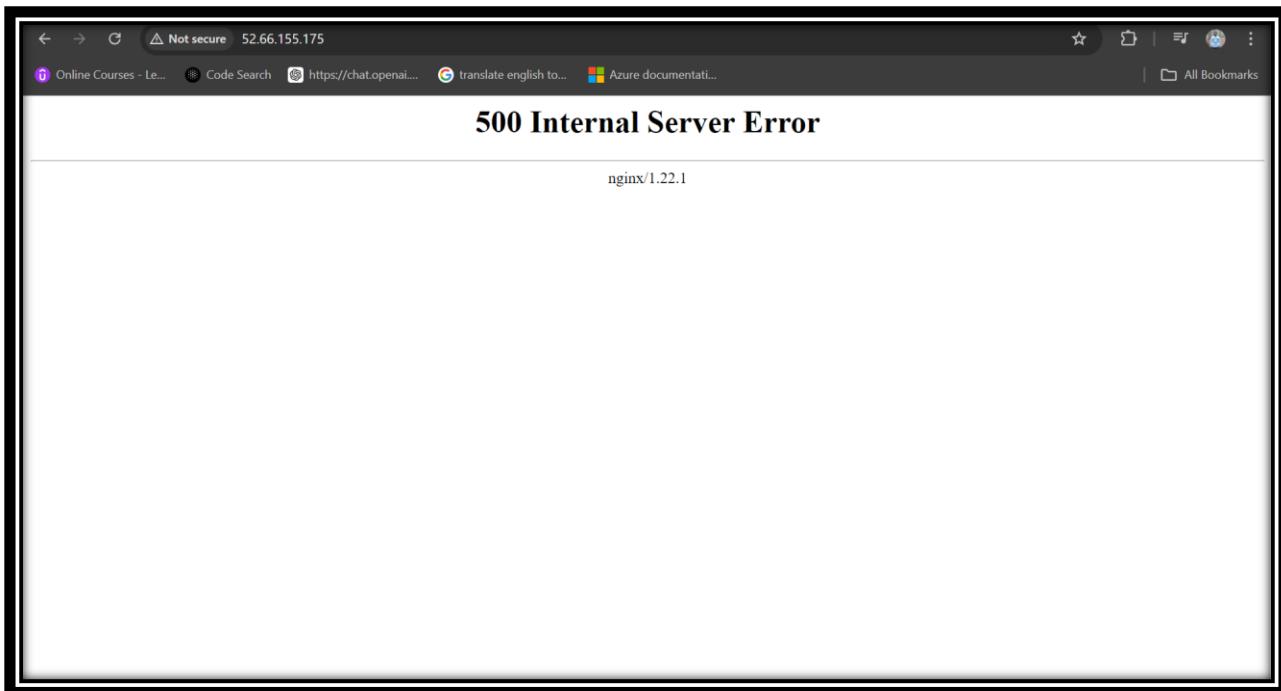
```

Session ID: root-ouzdqkewvgxpz2ghwmk5mbisy           Instance ID: i-03d03dc46a8823608
[root@ip-192-168-0-57 nginx]# sudo service nginx restart
Redirecting to /bin/systemctl restart nginx.service
[root@ip-192-168-0-57 nginx]# chmod -R 755 /home/ec2-user
[root@ip-192-168-0-57 nginx]#
[root@ip-192-168-0-57 nginx]# sudo chkconfig nginx on
Note: Forwarding request to 'systemctl enable nginx.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.
[root@ip-192-168-0-57 nginx]#

```

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

- Excepeted outputs were generated.

The screenshot shows a web browser window with a dark blue header bar. The title bar reads "AURORA DATABASE DEMO PAGE". On the left side of the header, there is a close button ("X") and a menu bar with two items: "HOME" and "DB DEMO". The main content area displays a table with three columns: "ID", "AMOUNT", and "DESC". A single row of data is present in the table, with values 1, 400, and avinash respectively. There are "ADD" and "DEL" buttons at the top of the table. A cursor is visible near the bottom center of the page.

ID	AMOUNT	DESC
1	400	avinash

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](#)”

The screenshot shows the AWS Identity and Access Management (IAM) service interface. On the left, the navigation pane is visible with options like Dashboard, Access management, Roles, Policies, and Access reports. The 'Roles' section is currently selected. In the main content area, the '3Tire-Project' role is displayed under the 'Roles' section. The 'Summary' tab is active, showing details such as Creation date (August 13, 2024, 20:35 (UTC+05:30)), ARN (arn:aws:iam::492646075778:role/3Tire-Project), Instance profile ARN (arn:aws:iam::492646075778:instance-profile/3Tire-Project), Last activity (37 minutes ago), and Maximum session duration (1 hour). Below the summary, there are tabs for Permissions, Trust relationships, Tags, Access Advisor, and Revoke sessions. The 'Permissions' tab is selected, showing the 'Permissions policies' section which lists one policy: 'AmazonEC2RoleforSSM'. There are buttons for Edit, Delete, Simulate, Remove, and Add permissions.

This screenshot shows the 'Permissions' tab for the '3Tire-Project' role. It displays the 'Permissions policies' section, which contains one policy named 'AmazonEC2RoleforSSM'. This policy is listed as 'AWS managed' and has one attached entity. Below the policy list, there is a section for 'Permissions boundary (not set)'.

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.

The screenshot shows the AWS S3 console interface. The left sidebar has a 'Buckets' section with links like Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, and Block Public Access settings. Below that is a 'Storage Lens' section with links for Dashboards, Storage Lens groups, and AWS Organizations settings. At the bottom of the sidebar are links for AWS Marketplace for S3, CloudShell, and Feedback. The main content area shows the '3tireproject01' bucket. The 'Objects' tab is selected, showing one object: 'application-code/'. The object details show it is a folder. There are buttons for Create folder, Upload, Copy S3 URI, Copy URL, Download, Open, Delete, and Actions. A search bar at the top says 'Find objects by prefix'. The footer of the page includes links for © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

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