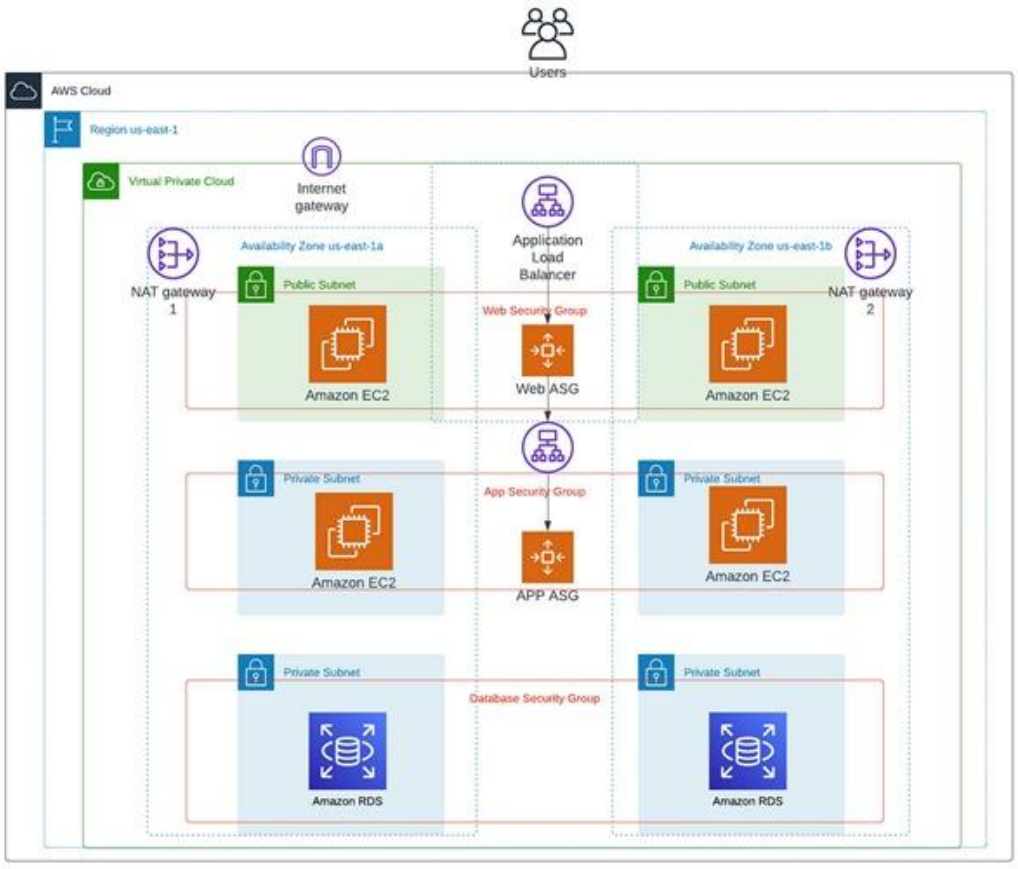
AWS 3-Tier Architecture Implementation Guide

Project: Scalable 3-Tier Web Application on AWS
Author: Bhargav Mandapati

Project Overview

This document provides a step-by-step implementation guide for building a 3-tier web application architecture on AWS, including all necessary screenshots and configurations.

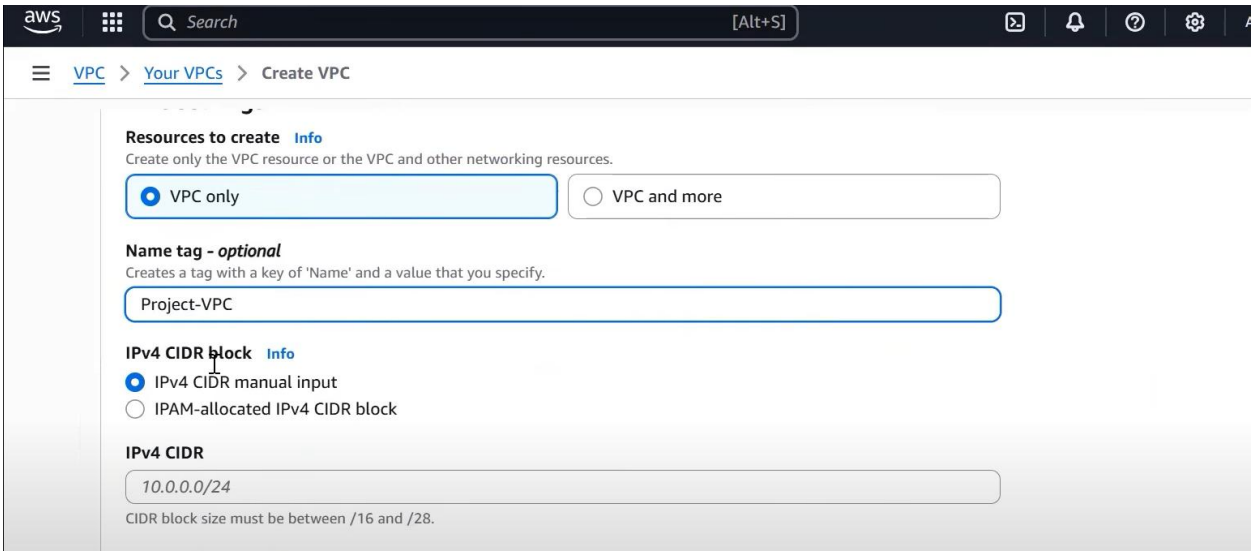
Architecture Diagram



Step 1: VPC Setup

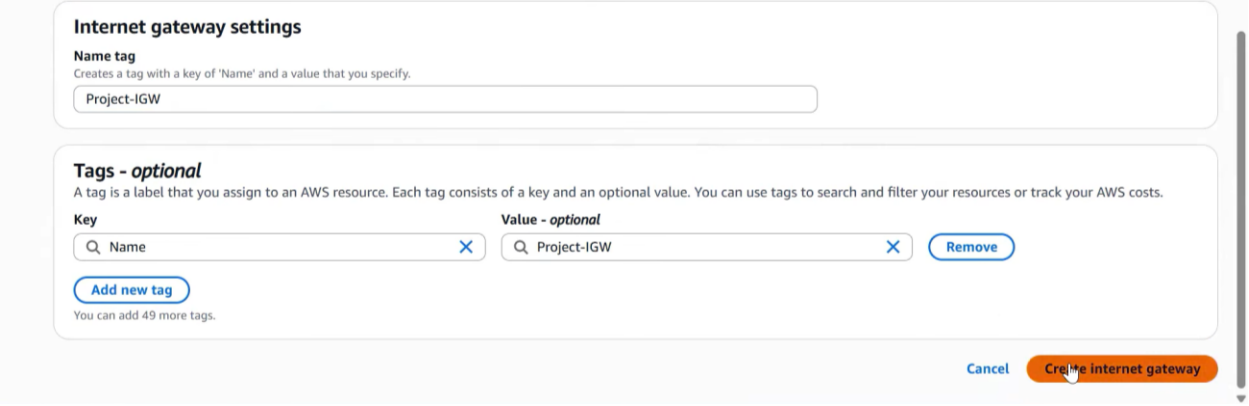
1.1 Create VPC

1. Navigate to VPC Dashboard in AWS Console
2. Click "Create VPC"
3. Configure VPC settings:
    - Name: Project-VPC
    - Tenancy: Default



1.2 Create Internet Gateway

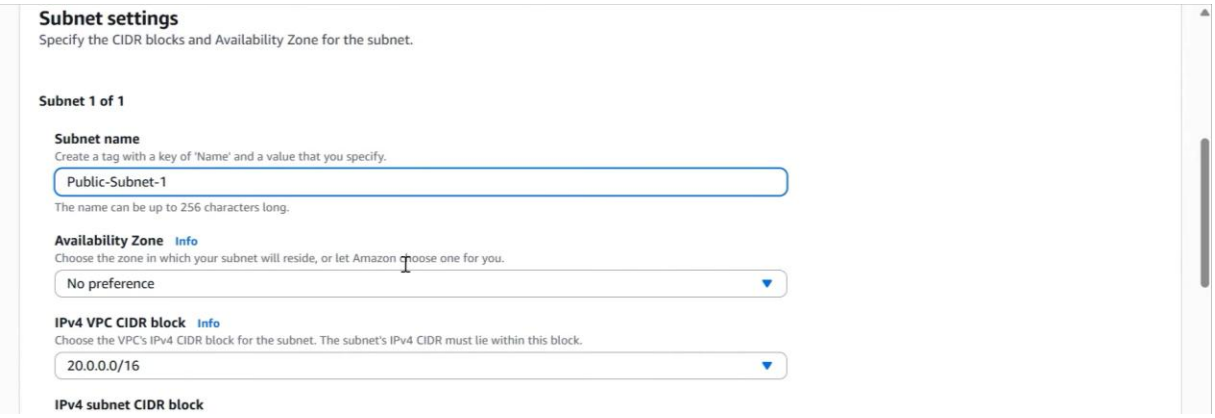
1. Go to Internet Gateways in VPC Dashboard
2. Click "Create Internet Gateway"
3. Name: Project-IGW
4. Attach to VPC: Project-VPC



1.3 Create Subnets

Public Subnet

1. **Public Subnet 1:**
    - Name: Public-Subnet-1
    - VPC: Project-VPC
    - Availability Zone: us-east-1a
2. **Public Subnet 2:**
    - Name: Public-Subnet-2
    - VPC: Project-VPC
    - Availability Zone: us-east-1b

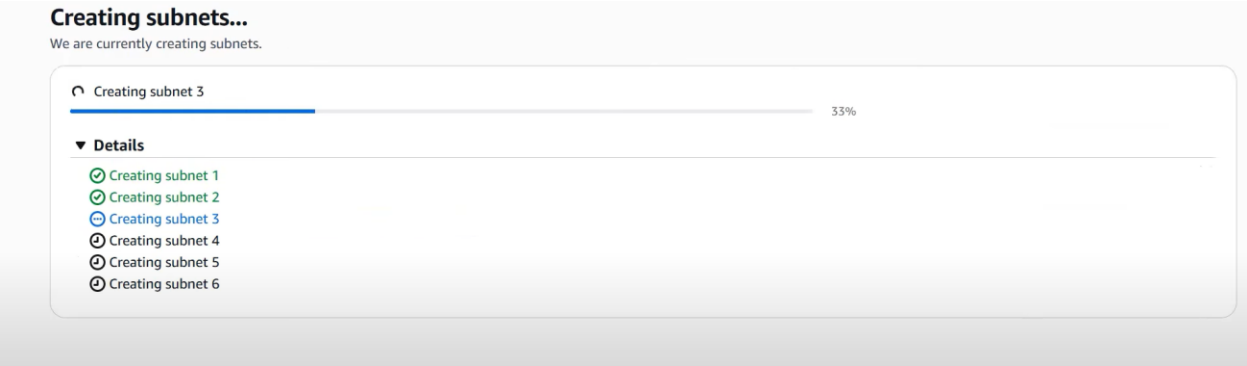


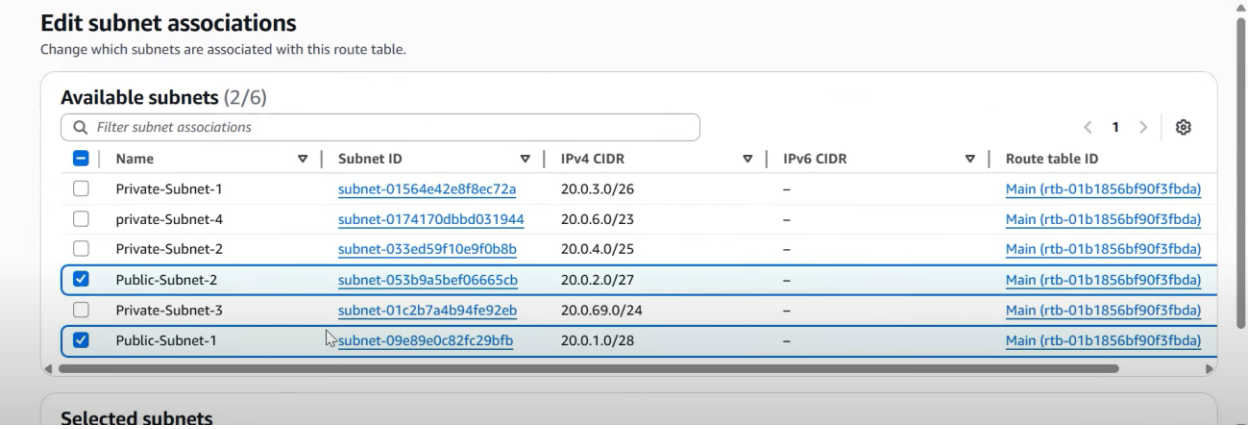
Private Subnets (App Tier)

1. **Private Subnet 1:**
    - Name: Private-Subnet-1
    - VPC: Project-VPC

Private Subnet 2:

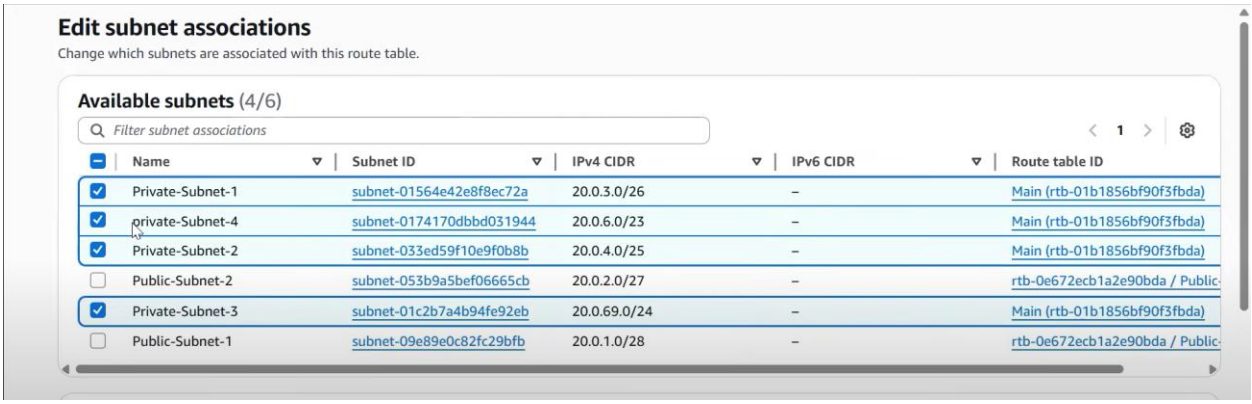
- Name: Private-Subnet-2
- VPC: Project-VPC

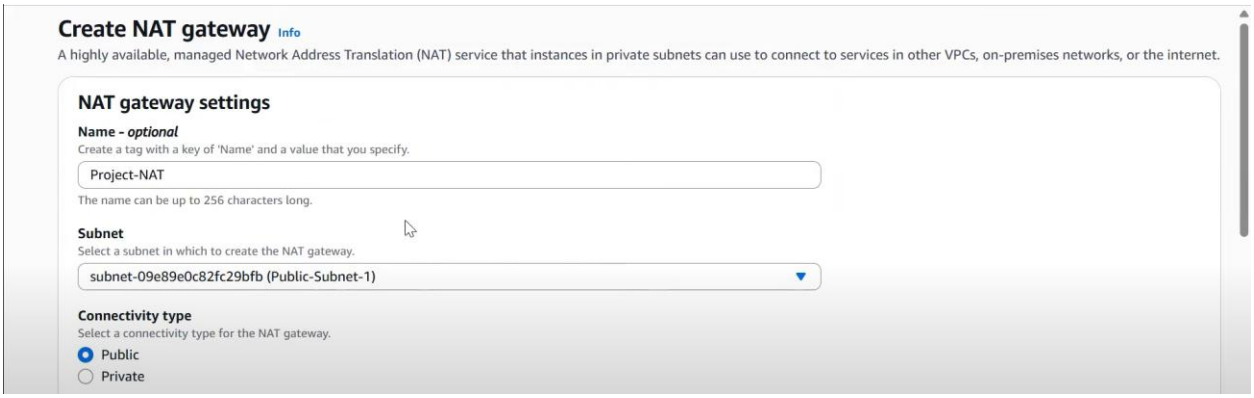




Database Subnets

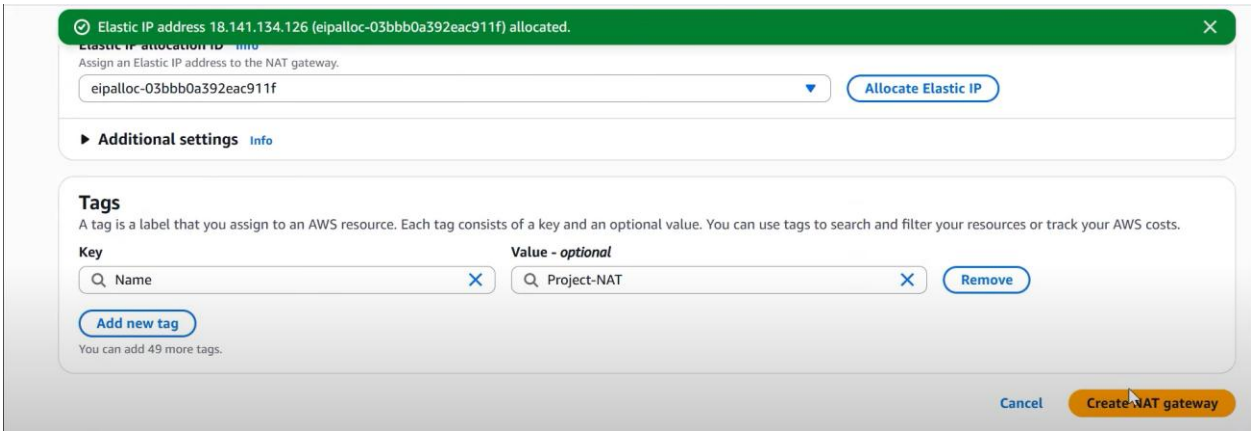
1. **Private-Subnet-3:**
   - Name: Private-Subnet-3
   - VPC: Project-VPC
2. **Private-Subnet-4:**
   - Name: Private-Subnet-4
   - VPC: Project-VPC





1.4 Project-NAT

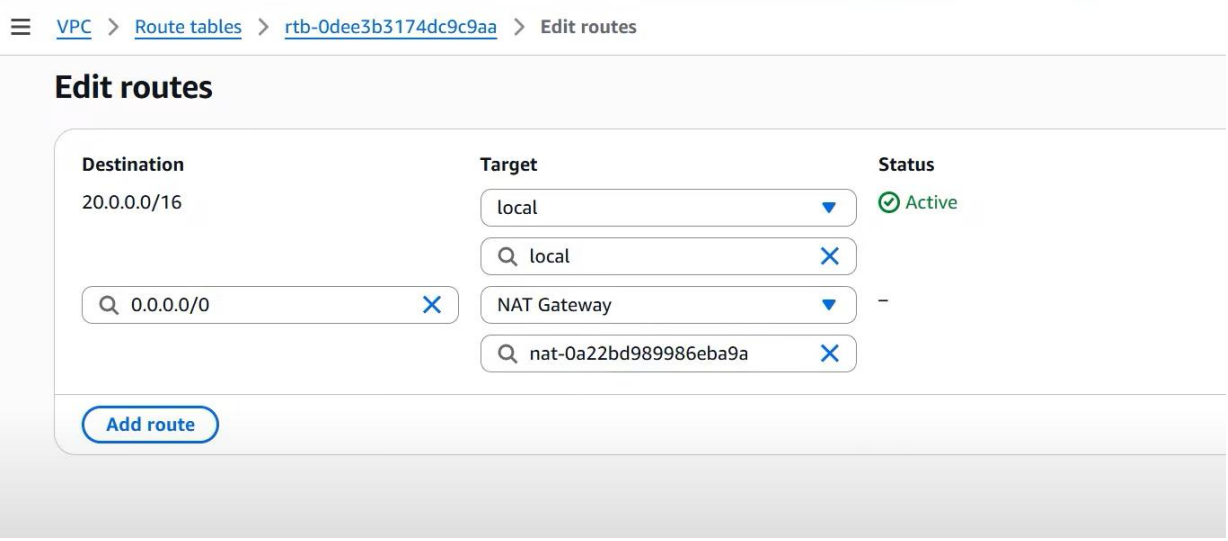
1. Go to NAT Gateways in VPC Dashboard
2. Click "Project-NAT"
3. Configure:
    - Name: 3-tier-nat-gateway
    - Subnet: Public-Subnet-1
    - Elastic IP: Allocate new



1.5 Configure Route Tables

Public Route Table

1. Create Route Table:
    - Name: Public-Route-Table
    - VPC: Project-VPC
2. Add Routes:
    - Destination: 0.0.0.0/0
    - Target: Internet Gateway
3. Associate with Public Subnet

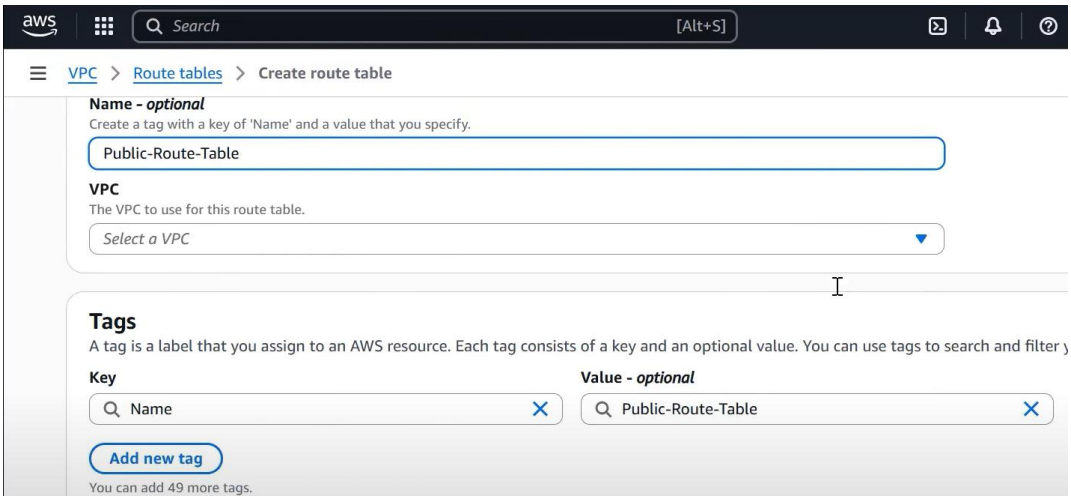


Private Route Table

1. Create Route Table:
    - Name: Private-Route-Table
    - VPC: Project-VPC
2. Add Routes:
    - Destination: 0.0.0.0/0
    - Target: NAT Gateway
3. Associate with Private Subnets

Database Route Table

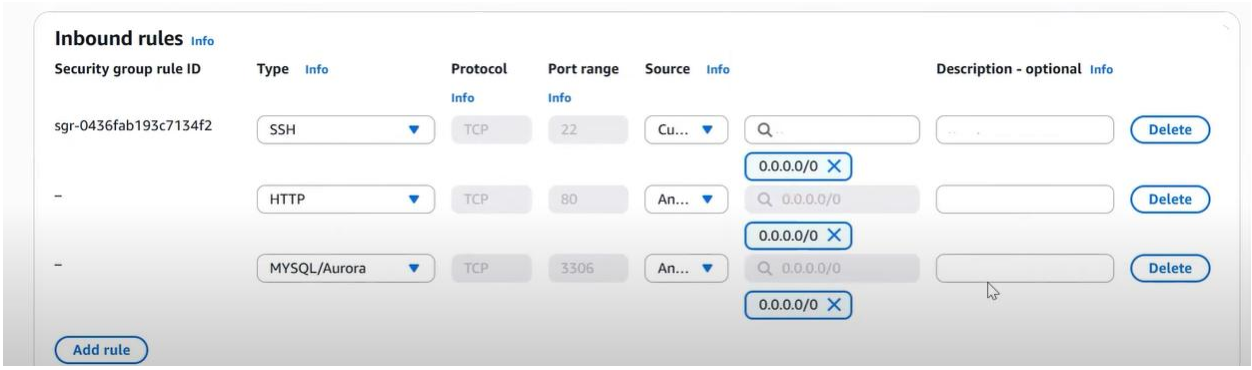
1. Create Route Table:
    - Name: Database-Route-Table
    - VPC: Project-VPC
2. Associate with Database Subnets (no internet access)



Step 2: Security Groups Configuration

2.1 Web Tier Security Group

1. Create Security Group:
    - Name: Web-Tier-SG
    - Description: Security group for web servers
    - VPC: Project-VPC
2. Inbound Rules:
    - HTTP (80): Source 0.0.0.0/0
    - HTTPS (443): Source 0.0.0.0/0
    - SSH (22): Source Your IP

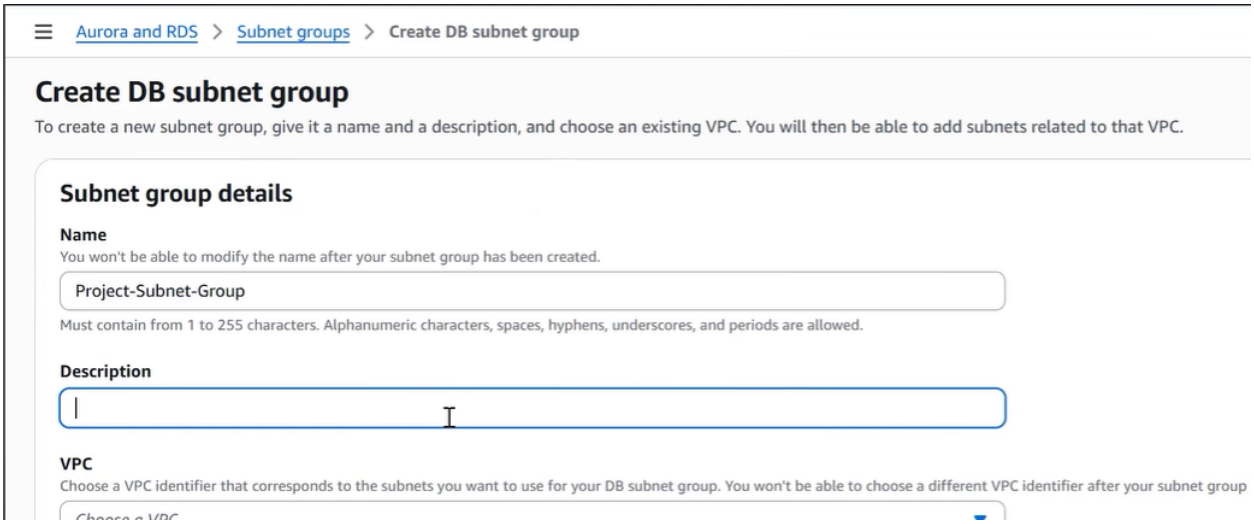


2.2 App Tier Security Group

1. Create Security Group:
    - Name: App-Tier-SG
    - Description: Security group for application servers
    - VPC: Project-VPC
2. Inbound Rules:
    - HTTP (80): Source Web-Tier-SG
    - SSH (22): Source Web-Tier-SG

2.3 Database Tier Security Group

1. Create Security Group:
    - Name: Database-Tier-SG
    - Description: Security group for database
    - VPC: Project-VPC
2. Inbound Rules:
    - MySQL (3306): Source App-Tier-SG



Step 3: Database Setup (RDS)

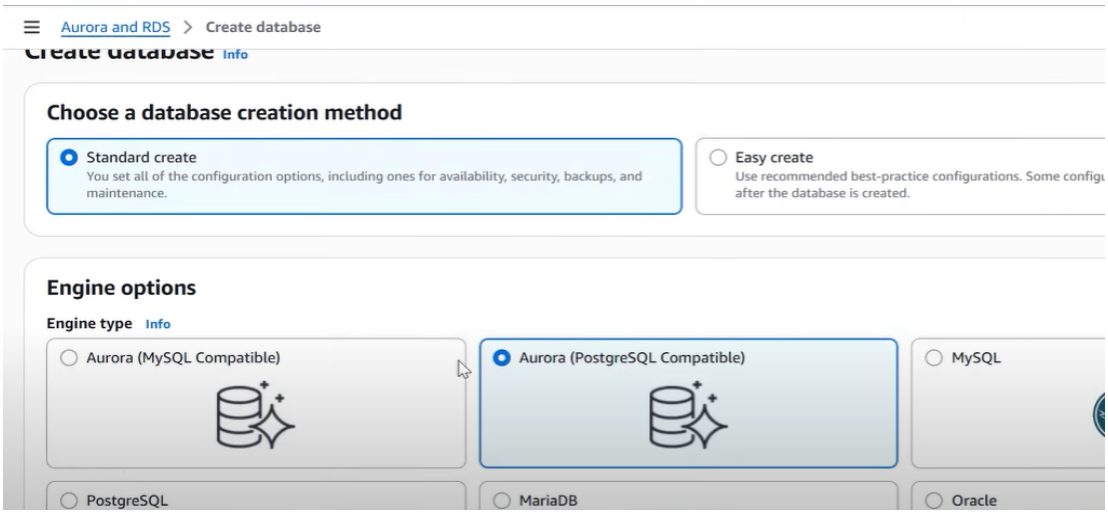
3.1 Create Database Subnet Group

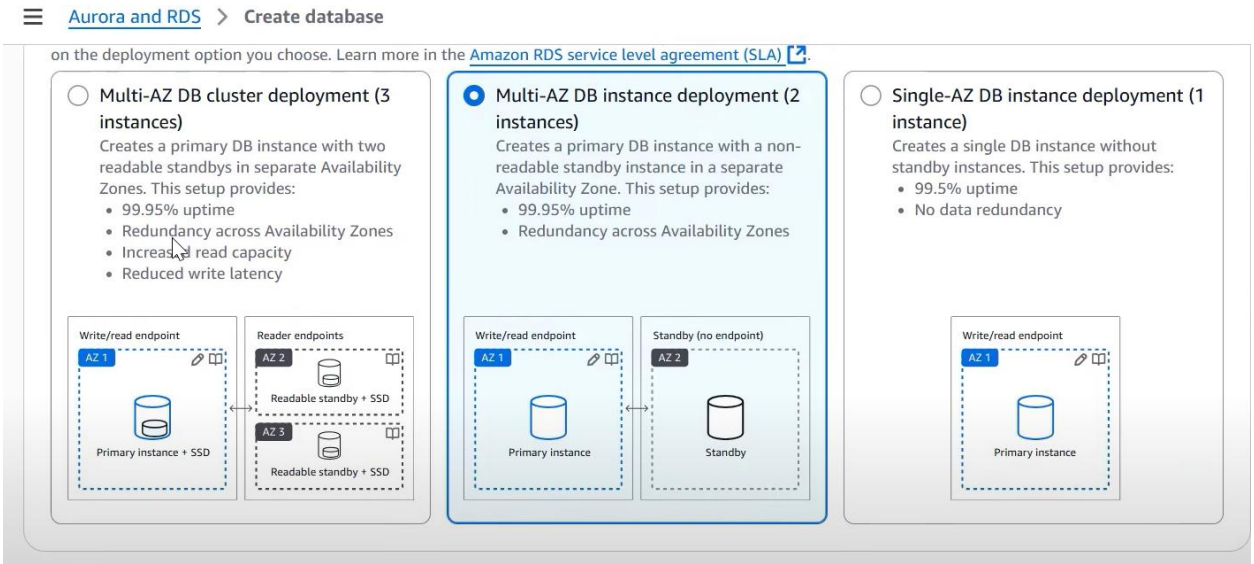
1. Navigate to RDS Dashboard
2. Go to Subnet Groups
3. Click "Create DB Subnet Group"
4. Configure:
    - Name: database-subnet-group
    - Description: Subnet group for 3-tier database
    - VPC: Project-VPC
    - Subnets: Select both database subnets

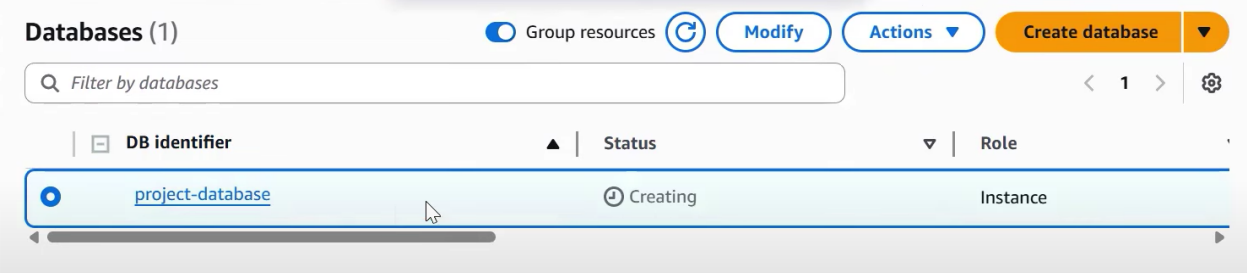
3.2 Create RDS Instance

1. Click "Create Database"
2. Engine: MySQL
3. Template: Free tier
4. Settings:
    - DB Instance Identifier: project-database

-
    - Master Username: admin
    - Master Password: [Your Password]
5. Instance Configuration:
    - DB Instance Class: db.t3.micro
6. Connectivity:
    - VPC: Project-VPC
    - Subnet Group: database-subnet-group
    - Security Group: Database-Tier-SG
    - No public access








```
root@ip-20-0-1-9:~# sudo systemctl start mysql.service
root@ip-20-0-1-9:~# mysql -h project-database.crkk88wuq912.ap-southeast-1.rds.amazonaws.com -u admi
n -p
Enter password:
```

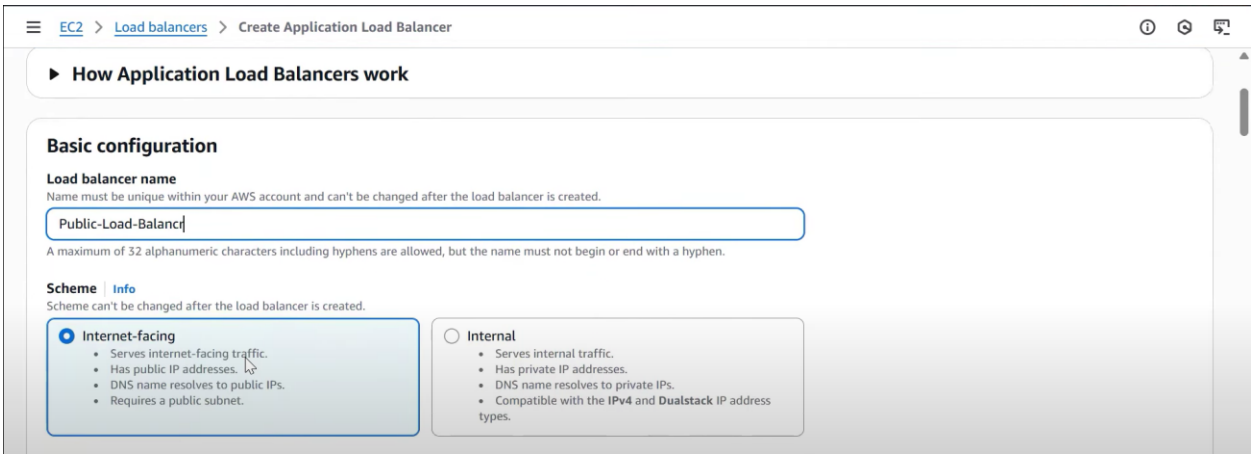
```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> create database devops;
Query OK, 1 row affected (0.01 sec)

mysql>
```

```
mysql> use devops
Database changed
mysql> clear
mysql> CREATE TABLE Details(
    ->     ID int NOT NULL,
    ->     LastName varchar(255) NOT NULL,
    ->     FirstName varchar(255),
    ->     Age int,
    ->     PRIMARY KEY(ID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+------------------+
| Tables_in_devops |
+------------------+
| Details          |
+------------------+
1 row in set (0.00 sec)

mysql> INSERT INTO Details(ID, LastName, FirstName, Age)
    -> VALUES (405, 'Teja', 'Ravi', 22);
Query OK, 1 row affected (0.01 sec)

mysql>
```

Step 4: Application Load Balancer Setup

4.1 Create Application Load Balancer

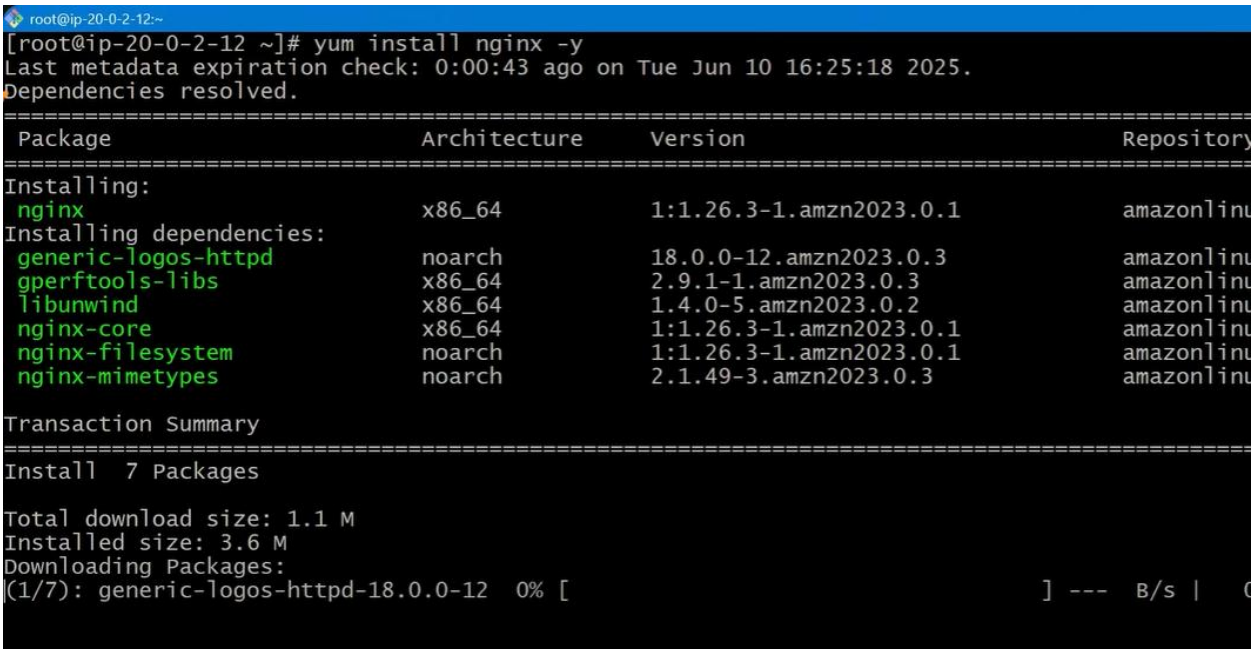
1. Navigate to EC2 Dashboard → Load Balancers
2. Click "Create Load Balancer"
3. Select "Application Load Balancer"
4. Configure:
    - Name: Public-Load-Balancer
    - Scheme: Internet-facing
    - IP Address Type: IPv4
    - VPC: Project-VPC
    - Subnets: Select both public subnets
    - Security Group: Web-Tier-SG

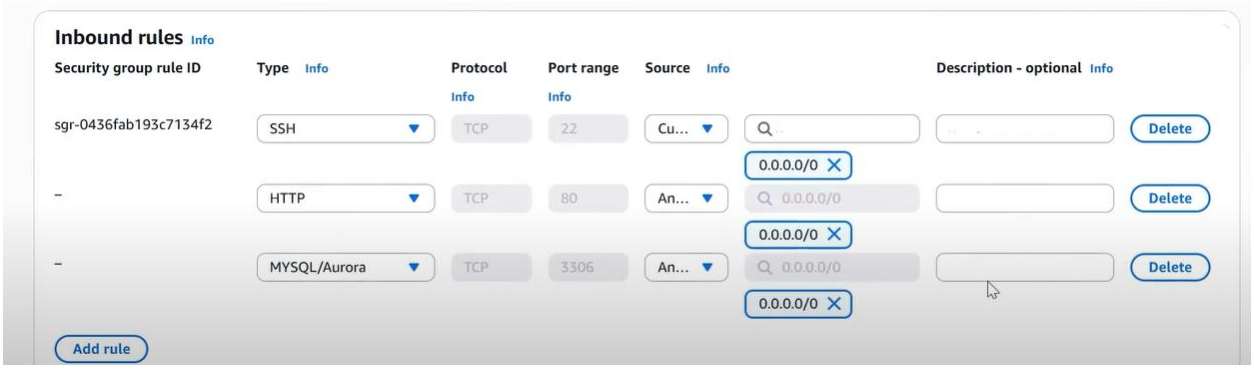


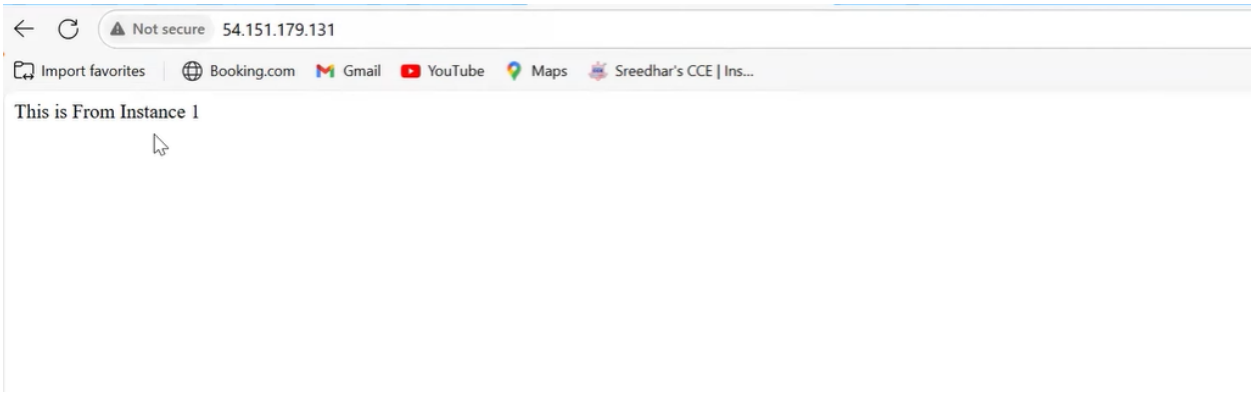
4.2 Create Target Group

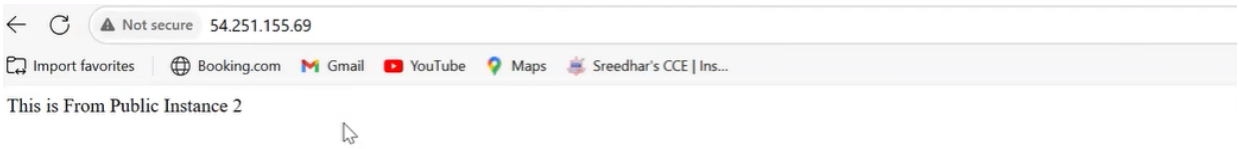
1. Create Target Group:
    - Name: Web-Tier-TG
    - Protocol: HTTP
    - Port: 80
    - VPC: Project-VPC

Step 5: Launch The Instances


root@ip-20-0-2-12:~
[root@ip-20-0-2-12 ~]# yum install nginx -y
Last metadata expiration check: 0:00:43 ago on Tue Jun 10 16:25:18 2025.
Dependencies resolved.
Package Architecture Version Repository
Installing:
nginx x86_64 1:1.26.3-1.amzn2023.0.1 amazonlinu
Installing dependencies:
generic-logos-httpd noarch 18.0.0-12.amzn2023.0.3 amazonlinu
gperftools-libs x86_64 2.9.1-1.amzn2023.0.3 amazonlinu
libunwind x86_64 1.4.0-5.amzn2023.0.2 amazonlinu
nginx-core x86_64 1:1.26.3-1.amzn2023.0.1 amazonlinu
nginx-filesystem noarch 1:1.26.3-1.amzn2023.0.1 amazonlinu
nginx-mimetypes noarch 2.1.49-3.amzn2023.0.3 amazonlinu
Transaction Summary
Install 7 Packages
Total download size: 1.1 M
Installed size: 3.6 M
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12 0% [ ] --- B/s |

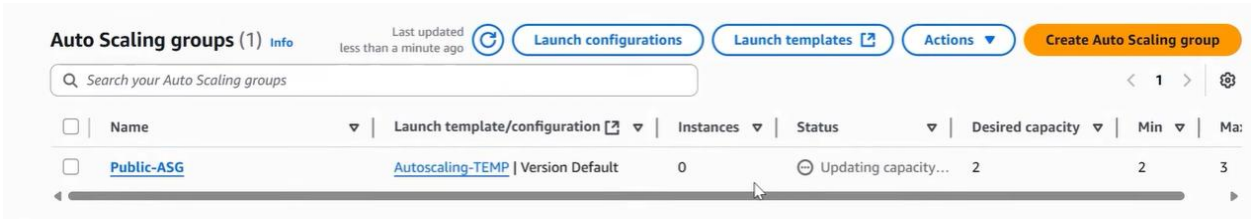

Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0436fab193c7134f2
SSH
TCP
22
Cu...
0.0.0.0/0
Delete
-
HTTP
TCP
80
An...
0.0.0.0/0
Delete
-
MYSQL/Aurora
TCP
3306
An...
0.0.0.0/0
Delete
Add rule

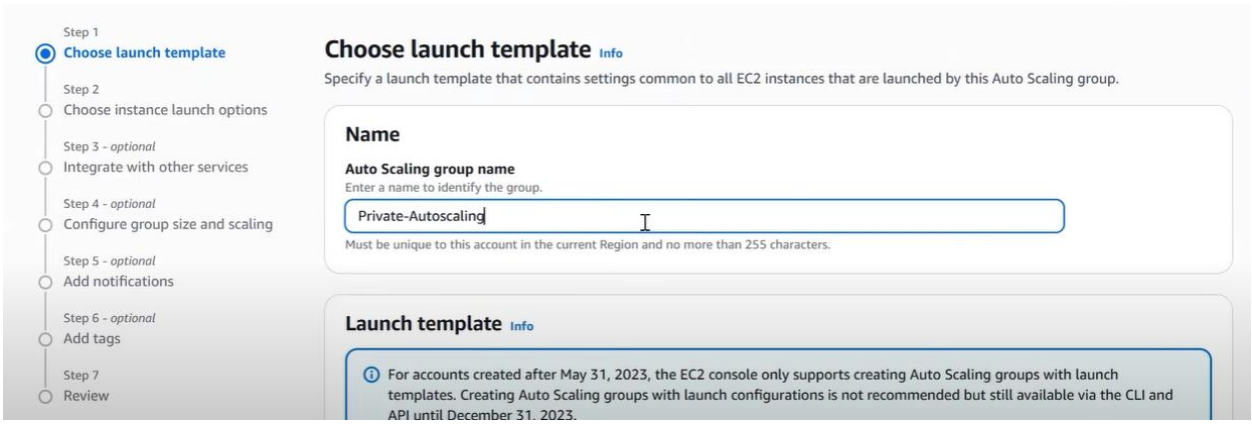

Not secure
54.151.179.131
Import favorites
Booking.com
Gmail
YouTube
Maps
Sreedhar's CCE | Ins...
This is From Instance 1



5.1 Create Launch Template

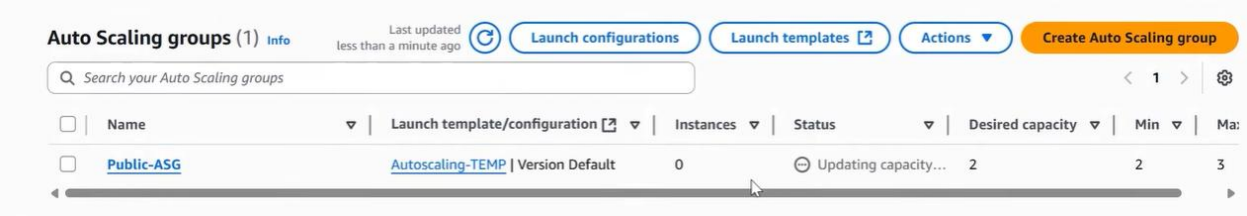
1. Navigate to EC2 Dashboard → Launch Templates
2. Click "Create Launch Template"
3. Configure:
    - Name: Web-Tier-Launch-Template
    - AMI: Amazon Linux 2
    - Instance Type: t2.micro
    - Key Pair: Select your key pair
    - Security Group: Public-ASG





5.2 Create Auto Scaling Group

1. Navigate to Auto Scaling Groups
2. Click "Create Auto Scaling Group"
3. Configure:
    - Name: Public-ASG
    - Launch Template: Web-Tier-Launch-Template
    - VPC: Project-VPC
    - Subnets: Select both public subnets
    - Load Balancer: Public-Load-Balancer
    - Target Group: Web-Tier-TG
    - Desired Capacity: 2
    - Min Capacity: 2
    - Max Capacity: 4



Step 6: Application Tier Setup

6.1 Create App Tier Launch Template

1. Create Launch Template:
    - Name: App-Tier-Launch-Template
    - AMI: Amazon Linux 2
    - Instance Type: t2.micro
    - Key Pair: Select your key pair
    - Security Group: App-Tier-SG
    - Subnet: Private subnets
    - User Data Script:

6.2 Create App Tier Auto Scaling Group

1. Configure Auto Scaling Group:
    - Name: App-Tier-ASG
    - Launch Template: App-Tier-Launch-Template
    - Subnets: Select both private subnets
    - Desired Capacity: 2

- Min Capacity: 2
- Max Capacity: 4

Project Summary

What We Built:

- VPC with public, private, and database subnets
- Internet Gateway and NAT Gateway
- Security Groups for each tier
- Application Load Balancer
- Auto Scaling Groups for web and app tiers
- RDS MySQL database
- CloudWatch monitoring

Key Achievements:

- High availability across multiple AZs
- Auto scaling based on demand
- Secure network architecture
- Cost-effective resource utilization
- Comprehensive monitoring
