AWS 3-Tier Web Application Deployment

# Description

This project demonstrates the deployment of a scalable 3-tier web application architecture using AWS services. The application consists of a React frontend, Node.js/Express backend, and a MySQL database, all hosted using AWS infrastructure.

# Project Architecture

Route 53 -> Application Load Balancer -> Auto Scaling Group (EC2: React + Node.js) -> Amazon RDS (MySQL)

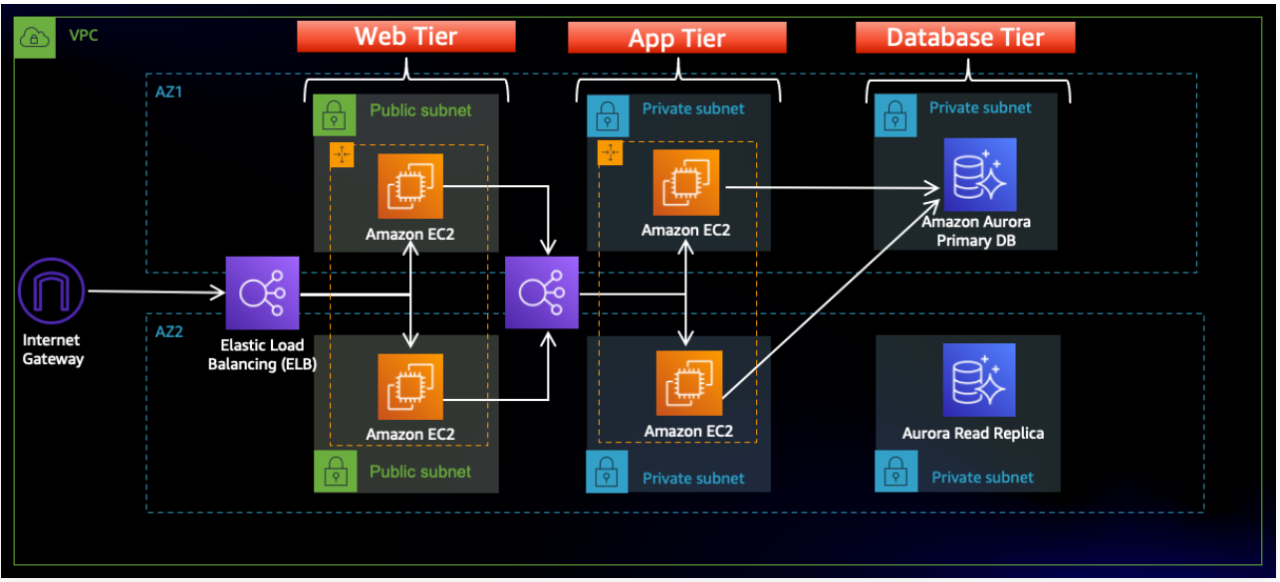
# Tech Stack & AWS Services Used

|  |  |
| --- | --- |
| Layer | Technology / AWS Service |
| Frontend | React, hosted on EC2 |
| Backend | Node.js/Express on EC2 |
| Database | Amazon RDS (MySQL) |
| Networking | VPC, Subnets, Internet Gateway |
| Load Balancer | Application Load Balancer |
| Auto Scaling | Launch Templates, Scaling Policies |
| Domain | Amazon Route 53 |
| Storage | S3 for static content/assets |
| IAM | Roles and Policies for EC2 and RDS |

# Project Structure

/aws-3tier-deployment  
├── frontend/ # React code  
├── backend/ # Node.js + Express API  
├── terraform/ # IaC for infrastructure (optional)  
├── diagrams/ # Architecture diagrams  
├── scripts/ # EC2 setup scripts  
├── .env.example # Environment template  
└── README.md

**Project Architecture Diagram**

****

# Setup Instructions

## Prerequisites

- AWS Account  
- IAM User with EC2, RDS, S3, VPC permissions  
- Git, Node.js, NPM, AWS CLI

## Infrastructure Setup

1. Create a VPC with 2 public and 2 private subnets.  
2. Set up an Internet Gateway and NAT Gateway.  
3. Launch EC2 instances for frontend and backend (Amazon Linux 2).  
4. Set up a Security Group allowing HTTP/HTTPS/DB access as needed.  
5. Set up Auto Scaling Group with Launch Template.  
6. Create an RDS MySQL instance in a private subnet.  
7. Configure an Application Load Balancer.  
8. Use Route 53 for domain pointing (optional).  
9. Store media/assets in S3 if needed.

## App Deployment

SSH into EC2 instances.  
Clone the repository:  
git clone https://github.com/yourusername/aws-3tier-deployment.git  
  
Install dependencies and run:  
cd frontend && npm install && npm run build  
cd ../backend && npm install && node index.js

# Features

- Scalable 3-tier AWS architecture  
- Load-balanced web app  
- Auto-scaling EC2 instances  
- MySQL DB in private subnet (RDS)  
- Custom domain with Route 53  
- S3 asset storage (optional)

# Future Improvements

- CI/CD pipeline with CodePipeline  
- HTTPS with ACM  
- Monitoring with CloudWatch  
- Dockerized deployment

# Author

Bobby Vijay Arora V  
LinkedIn: https://www.linkedin.com/in/bobby-vijay-arora-v-0617b7274/  
Email:bobbyvijayarora6@example.com

# License

This project is licensed under the MIT License.