

AWS Infrastructure Setup Guide

This document describes a secure, production-standard AWS infrastructure for a React frontend hosted on S3, a Python backend running on EC2, and a PostgreSQL database on RDS.

High-Level Architecture

Users access the React frontend via CloudFront (HTTPS). The frontend is served from a private S3 bucket. API requests go through an Application Load Balancer to EC2 instances running the Python backend in private subnets. The backend connects to PostgreSQL hosted on RDS in private subnets.

Networking (VPC Setup)

Create a single VPC with a /16 CIDR block and at least two Availability Zones.

- Public subnets: Application Load Balancer, NAT Gateway
- Private app subnets: EC2 backend instances
- Private DB subnets: RDS PostgreSQL

Frontend (React on S3 + CloudFront)

- S3 bucket with public access blocked
- Enable versioning and server-side encryption
- CloudFront with Origin Access Control
- HTTPS using ACM certificate

Backend (Python on EC2)

- EC2 instances in private subnets
- Access via SSM Session Manager (no SSH)
- Application served via Gunicorn/Uvicorn
- Environment variables from SSM Parameter Store

Load Balancer

- Public Application Load Balancer
- HTTPS only, ACM certificate
- Health checks on /health endpoint

Database (PostgreSQL on RDS)

- RDS PostgreSQL in private subnets
- Multi-AZ enabled
- Storage encryption enabled
- Credentials stored in AWS Secrets Manager

IAM & Security Best Practices

- Use IAM roles with least privilege
- No hard-coded AWS credentials
- Security groups restrict traffic between layers

CI/CD (Recommended)

- Frontend: GitHub Actions → S3 upload → CloudFront invalidation
- Backend: GitHub Actions → EC2 deployment via SSM or CodeDeploy

Cost Optimization Defaults

- EC2: t3.small
- RDS: db.t3.micro (scale later)
- Single NAT Gateway initially