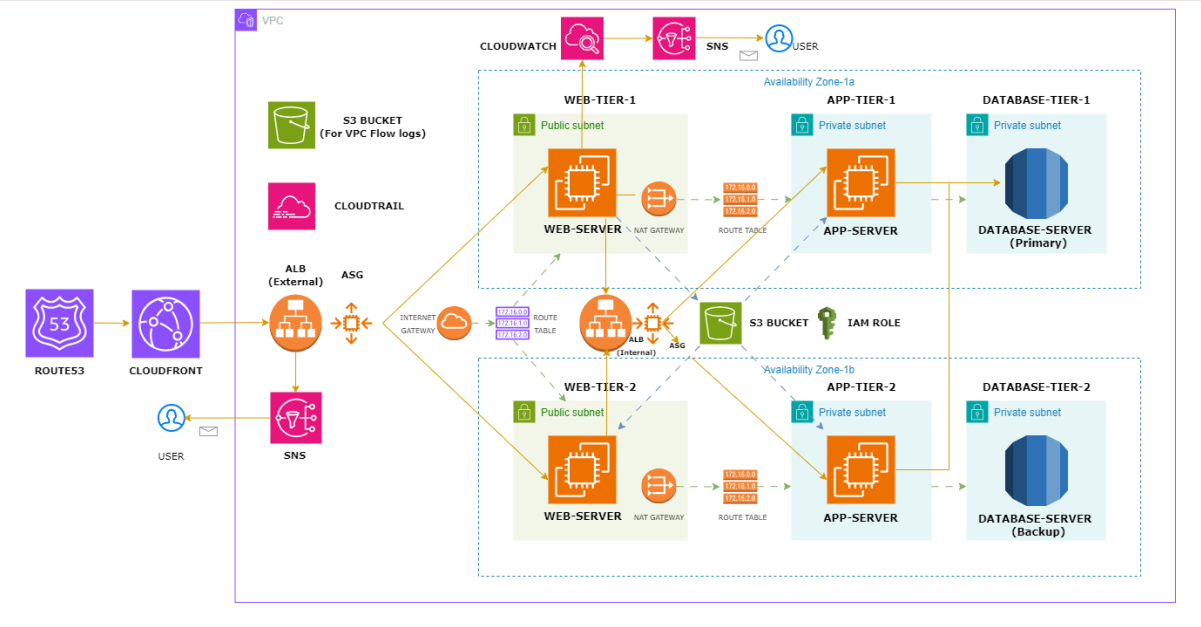
AWS Three-Tier Web Architecture

1. Introduction

This document describes a production-ready AWS Three-Tier Web Architecture designed for high availability, scalability, security, and cost optimization. The architecture separates the application into Web, Application, and Database tiers, each deployed across two Availability Zones to ensure fault tolerance.

The solution uses a React frontend served via Nginx, a Node.js (Express) backend API, and a MySQL database hosted on Amazon RDS. Traffic is distributed using Application Load Balancers (ALB) and Auto Scaling Groups (ASG), with strict security controls between tiers. SSL/TLS is enforced using AWS Certificate Manager (ACM), and global content delivery is enabled through Amazon CloudFront with Route 53 and GoDaddy domain integration.

This project is implemented without Terraform, relying instead on AWS-native services, launch templates, AMIs, scripts, and configuration templates. Monitoring, logging, and cost optimization are achieved using Amazon CloudWatch, SNS alerts, and right-sized EC2 instances with Reserved Instances, achieving up to 70% cost savings.



---

2. Architecture Overview

Tier Breakdown:

- Web Tier (Public Subnets): Nginx + React application behind an external Application Load Balancer
- Application Tier (Private Subnets): Node.js (Express) backend behind an internal Application Load Balancer
- Database Tier (Private Subnets): Amazon RDS MySQL with Multi-AZ deployment

High Availability:

- Resources deployed across two Availability Zones
- Auto Scaling Groups for web and application tiers
- RDS Multi-AZ automated failover

---

3. Technology Stack

Frontend (Web Tier)

- React.js – Frontend UI
- Nginx – Static file serving and reverse proxy
- Amazon EC2 – Web servers
- External Application Load Balancer – Internet-facing traffic distribution
- Amazon CloudFront – Global content delivery and caching

Backend (Application Tier)

- Node.js – Runtime environment
- Express.js – RESTful API framework
- bcrypt – Password hashing and security
- Internal Application Load Balancer – Private traffic routing
- Amazon EC2 – Application servers

Database Tier

- Amazon RDS MySQL – Managed relational database
- Multi-AZ Deployment – High availability and automated failover
- Automated Backups & Snapshots

Networking

- Amazon VPC – Custom VPC
- Public & Private Subnets – Across two AZs
- Internet Gateway & NAT Gateway
- Route Tables & NACLs

Security & Access

- AWS Certificate Manager (ACM) – SSL/TLS certificates
- IAM Roles & Policies – Least privilege access
- AWS Systems Manager (SSM) – Secure EC2 access (no SSH)

- Security Groups – Tier-based traffic control

Storage & Deployment

- Amazon S3 – Storage for full-stack application artifacts
- EC2 AMI – Pre-baked images for faster scaling
- Launch Templates – Consistent EC2 provisioning

Monitoring & Notifications

- Amazon CloudWatch – Logs, metrics, and alarms
- Amazon SNS – Notifications and alerts

DNS & Domain

- Amazon Route 53 – DNS management
- GoDaddy – Domain registration and integration

---

4. Implementation Scope (No Terraform)

Infrastructure Setup

- Custom VPC with CIDR block
- Two public subnets (Web Tier)
- Two private subnets (Application Tier)
- Two private subnets (Database Tier)
- Internet Gateway for public access
- NAT Gateway for outbound private access

Compute & Load Balancing

- EC2 Launch Templates with custom AMIs
- Auto Scaling Groups for Web and App tiers
- External ALB (Internet-facing) for Web Tier
- Internal ALB (Private) for Application Tier
- Target Groups with health checks

Application Deployment

- React app built and stored in S3
- Nginx configured to serve React build
- Node.js Express API deployed on private EC2 instances
- Secure API communication between tiers

Database Setup

- Amazon RDS MySQL (Multi-AZ)
- Private subnet placement
- Automated backups and maintenance windows

Security & Access Control

- IAM role for EC2 to access S3, CloudWatch, and SSM
- SSM Session Manager used instead of SSH
- ACM certificates attached to ALB

CDN & DNS

- CloudFront distribution in front of external ALB
- Route 53 hosted zone
- Domain mapped from GoDaddy to Route 53

Monitoring & Scaling

- CloudWatch alarms based on CPU and request count
- Auto Scaling policies triggered by CloudWatch metrics
- SNS notifications for alarms

---

5. Security Group Design

This architecture uses five distinct security groups, each aligned to a specific tier or load balancer. This enforces least privilege access and strong network isolation.

5.1 External Load Balancer Security Group (SG-ALB-External)

Attached To: Internet-facing Application Load Balancer

Inbound Rules:

- HTTP (80) – From Internet (0.0.0.0/0)
- HTTPS (443) – From Internet (0.0.0.0/0)

Outbound Rules:

- HTTP (80) – To Web Tier Security Group

Purpose:
Accepts public traffic from the internet and forwards requests only to the web tier.

5.2 Web Tier Security Group (SG-Web)

Attached To: Nginx + React EC2 instances (Public Subnets)

Inbound Rules:

- HTTP (80) – From SG-ALB-External only

Outbound Rules:

- HTTP (80) – To Internal ALB Security Group
- HTTPS (443) – For AWS services (CloudWatch, S3, SSM)

Purpose:
Allows traffic only from the external ALB and restricts direct public access to EC2 instances.

5.3 Internal Load Balancer Security Group (SG-ALB-Internal)

Attached To: Internal Application Load Balancer (Private Subnets)

Inbound Rules:

- HTTP (80) or App Port – From SG-Web only

Outbound Rules:

- Application Port (e.g., 3000) – To App Tier Security Group

Purpose:
Routes traffic securely from the web tier to the application tier inside the VPC.

5.4 Application Tier Security Group (SG-App)

Attached To: Node.js (Express) EC2 instances (Private Subnets)

Inbound Rules:

- Application Port (e.g., 3000) – From SG-ALB-Internal only

Outbound Rules:

- MySQL (3306) – To Database Security Group
- HTTPS (443) – For AWS services and external APIs

Purpose:
Ensures only the internal ALB can access backend services while allowing controlled outbound access.

5.5 Database Tier Security Group (SG-DB)

Attached To: Amazon RDS MySQL (Private Subnets)

Inbound Rules:

- MySQL (3306) – From SG-App only

Outbound Rules:

- None (default restricted)

Purpose:
Strictly limits database access to application servers only, providing maximum data security.

---

6. Cost Optimization Strategy

- Right-sized EC2 instances based on workload
- Reserved Instances reducing compute cost by up to 70%
- Auto Scaling to handle peak and off-peak traffic
- CloudWatch monitoring for resource utilization
- CloudFront caching to reduce ALB and EC2 load

Part 1: Networking and Security

In this section we will be building out the VPC networking components as well as security groups that will add a layer of protection around our EC2 instances, Aurora databases, and Elastic Load Balancers.

Create an isolated network with the following components:

- **VPC**
- **Subnets**
- **Route Tables**
- **Internet Gateway**
- **NAT gateway**
- **Security Groups**