End-to-End DevOps Project
Production-Grade Cloud-Native E-Commerce Platform

Author: Sivaraj Shanmugam – Senior DevOps Engineer

1. Project Overview

This document describes a complete, production-grade DevOps implementation of a cloud-native e-commerce platform designed to handle 50,000+ concurrent users. The project demonstrates real-world DevOps practices including Infrastructure as Code, Kubernetes orchestration, CI/CD automation, scalability, observability, and disaster recovery.

2. High-Level Architecture

The platform is deployed on AWS using EKS for container orchestration. Traffic enters via an Application Load Balancer (ALB) and is routed to frontend services running in Kubernetes. Backend APIs communicate with Redis for caching and MySQL (RDS Multi-AZ) for persistence. Prometheus and Grafana provide monitoring, while Jenkins automates CI/CD pipelines.

3. Step 1 – Infrastructure Provisioning (Terraform)

Terraform is used to provision AWS infrastructure including VPC, public and private subnets, Internet Gateway, NAT Gateway, security groups, and IAM roles. This ensures repeatable, version-controlled infrastructure.

4. Step 2 – Amazon EKS Cluster Setup

An Amazon EKS cluster is provisioned using Terraform with managed node groups deployed in private subnets. IAM roles are separated for the control plane and worker nodes to follow least-privilege principles.

5. Step 3 – Application Containerization (Docker)

Frontend and backend services are containerized using Docker. Official PHP images are used, ensuring minimal attack surface and compatibility with Kubernetes. Images are built following best practices.

6. Step 4 – Kubernetes Deployment

Applications are deployed to EKS using Kubernetes Deployments and Services. Rolling update strategies ensure zero downtime. AWS ALB Ingress Controller is used for external access.

7. Step 5 – Redis Caching and Autoscaling

Redis is deployed inside Kubernetes to cache frequently accessed data and reduce database load. Horizontal Pod Autoscaler (HPA) automatically scales frontend and backend services based on CPU utilization.

8. Step 6 – CI/CD with Jenkins

A Jenkins pipeline automates code checkout, Docker image builds, pushes images to Amazon ECR, and deploys applications to EKS. Manual approval gates and automated rollback mechanisms are implemented.

9. Step 7 – Monitoring and Alerting

Prometheus and Grafana are deployed using Helm to monitor Kubernetes clusters, nodes, and pods. Alert rules are configured to detect high CPU usage and pod failures proactively.

10. Step 8 – Disaster Recovery and MySQL PITR

Disaster recovery is implemented using Amazon RDS MySQL Point-In-Time Recovery (PITR) and Velero for Kubernetes backups. The solution meets defined RPO and RTO objectives, enabling recovery from logical and infrastructure failures.

11. Conclusion

This project demonstrates senior-level DevOps expertise by combining automation, scalability, monitoring, and disaster recovery into a single production-ready system. It is suitable for showcasing in GitHub, LinkedIn, and technical interviews.