Deployment of Online Food Ordering and Delivery Application

A DevOps Implementation for Scalable Food Delivery Platform

Domain: Food Delivery and E-Commerce

Project Overview:
Zomato is a leading online food delivery and restaurant discovery platform. To handle increasing user demand and improve deployment efficiency, the company is transitioning from a monolithic architecture to a microservices-based architecture using DevOps practices.

The goal is to implement a CI/CD pipeline, containerized deployments, and real-time monitoring to ensure high availability, faster releases, and automated scalability. AWS will be used as the primary cloud provider for hosting and deployment.

GitHub URL: https://github.com/staragile2016/Zomato.git

Business Goals:

- Faster and reliable deployments to production.
- Reduced manual intervention in infrastructure management.
- Improved scalability and performance monitoring.
- Automated rollback in case of failures.

Current Challenges:

- Manual Deployments: Time-consuming and error-prone.
- Monitoring Gaps: No real-time visibility into application health.

CI/CD Pipeline Implementation:
The following tools will be used to automate the deployment process:

- Tools & Technologies
- Git – Version control for tracking code changes.
- Jenkins – CI/CD pipeline automation (build, test, deploy).
- Docker – Containerization of the application.
- Kubernetes (K8s) – Orchestration for scalable deployments.
- AWS (EC2, EKS, ECR) – Cloud infrastructure for hosting.
- Prometheus & Grafana – Monitoring and alerting.

Pipeline Workflow:

- GitHub Webhook Trigger
- Code push to the main/master branch triggers Jenkins automatically.
- Jenkins fetches the code
- Docker Image Creation
- The application is packaged into a Docker image and pushed to AWS ECR/DockerHub.
- Kubernetes Deployment (Staging Environment)

- The Docker image is deployed to a Kubernetes cluster (AWS EKS).
- Monitoring & Validation
- Prometheus collects metrics.
- Grafana visualizes performance dashboards.
- If tests pass, the build is promoted to production.

Production Deployment:

- The same Docker image is deployed to the production Kubernetes cluster.
- Auto-scaling ensures high availability during peak loads.

Expected Outcome:

✅ Fully Automated CI/CD Pipeline – Zero manual deployments.
✅ Scalable Microservices – Kubernetes handles traffic spikes.
✅ Real-time Monitoring – Prometheus & Grafana provide observability.
✅ Faster Release Cycles – Quick updates with rollback capability.