(Name: Nipur Jain)

CI/CD Pipeline with GitHub Actions & Docker - Project Report

Introduction

This project demonstrates the implementation of a complete CI/CD (Continuous Integration/Continuous Deployment) pipeline using modern DevOps tools. The objective was to create an automated workflow that builds, tests, and deploys a Java application using containerization technology without requiring cloud infrastructure costs.

Abstract

A full-stack CI/CD pipeline was successfully developed integrating GitHub Actions for automation, Docker for containerization, and Kubernetes for local deployment. The pipeline automatically triggers on code commits, runs comprehensive tests, builds Docker images, pushes to Docker Hub registry, and deploys to a local Kubernetes cluster using Minikube. This implementation showcases industry-standard DevOps practices while maintaining cost-effectiveness through local deployment strategies.

Tools Used

Development & Version Control

- Java 11 - Application development language
- Maven - Build automation and dependency management
- Git - Version control system
- GitHub - Source code repository and CI/CD platform

CI/CD & Containerization

- GitHub Actions - Automated CI/CD pipeline orchestration
- Docker - Application containerization platform
- Docker Hub - Container image registry
- JUnit - Unit testing framework

Deployment & Orchestration

- Kubernetes - Container orchestration platform
- Minikube - Local Kubernetes cluster
- kubectl - Kubernetes command-line interface

Steps Involved in Building the Project

1. Application Development

• Created Java HTTP server application (HelloWorld.java)
• Configured Maven build system with pom.xml
• Implemented JUnit test cases for CI validation
• Added proper JAR manifest configuration for executable deployment

2. Containerization Setup

• Designed minimal Dockerfile using OpenJDK 11 base image
• Created docker-compose.yml for local development environment
• Optimized container size and security practices

3. CI/CD Pipeline Configuration

• Implemented GitHub Actions workflow (.github/workflows/ci-cd.yml)
• Configured automated triggers on push/pull requests to main branch
• Integrated Maven test execution and build processes
• Set up Docker Hub authentication using GitHub Secrets
• Automated Docker image building and registry pushing

4. Kubernetes Deployment

• Created Kubernetes deployment manifest (k8s-deployment.yml)
• Configured NodePort service for external access
• Set up Minikube local cluster environment
• Implemented rolling deployment strategies

5. Integration & Testing

• Established end-to-end pipeline validation
• Verified automated image pulls from Docker Hub
• Tested application accessibility through Kubernetes services
• Validated complete workflow from code commit to live deployment

Technical Architecture

```
Code Commit → GitHub Actions → Maven Tests → Docker Build →
Docker Hub Push → Kubernetes Deployment → Live Application
```

Key Features Implemented:

• Automated Testing: JUnit tests execute on every commit
• Container Security: Minimal base images and non-root execution
• Zero-Downtime Deployment: Kubernetes rolling updates
• Image Versioning: Tagged Docker images with latest strategy
• Local Development: Complete local testing environment

Conclusion

The project successfully demonstrates a production-ready CI/CD pipeline implementation using industry-standard tools and practices. The automated workflow reduces manual deployment errors, ensures consistent testing, and provides rapid feedback cycles for development teams.

Key achievements include:
• 100% Automated Pipeline: From code commit to live deployment
• Cost-Effective Solution: No cloud infrastructure costs using local Minikube
• Scalable Architecture: Kubernetes-ready for production scaling
• Security Best Practices: Containerized applications with proper secret management

This implementation serves as a foundation for enterprise-level DevOps practices and can be easily extended to cloud platforms (AWS EKS, Google GKE, Azure AKS) when scaling requirements demand it. The project demonstrates proficiency in modern software delivery practices essential for contemporary software development workflows.

Project Repository: https://github.com/NipurJain4/java-ci-cd-pipeline
Docker Image: nipurjain/java-cicd-pipeline:latest
Completion Date: September 2025