DevOps CI/CD Pipeline Implementation Project Proposal

Project Description

This project aims to implement a comprehensive DevOps CI/CD pipeline for a multi-tier microservice application based on the "microservice-app-example" repository (https://github.com/elgris/microservice-app-example). We will establish an automated build, test, and deployment workflow using industry-standard tools like Jenkins, Docker, Kubernetes, and Ansible. The project will demonstrate end-to-end automation from code commit to production deployment while implementing best practices for containerization, orchestration, and configuration management.

Group Members & Roles

- Hamid Walid: Team Leader & CI/CD Pipeline Engineer, responsible for Jenkins implementation and overall coordination
- Hafez Adel: Infrastructure Architect, handling Docker containerization and registry management
- Bassant Ali: Kubernetes Specialist, managing container orchestration and deployment strategies
- Ziad Ahmed: Configuration Management & Cloud Expert, implementing Ansible and AWS infrastructure

Team Leader

Hamid Walid

Objectives

- Create a fully automated CI/CD pipeline for the microservice application with automated builds triggered by Git commits
- Containerize all microservices (frontend, API gateway, auth-api, users-api) using Docker
- Implement container orchestration using Kubernetes for scalable deployments
- Use Ansible for configuration management and environment provisioning
- Deploy the complete application to AWS cloud infrastructure
- Establish monitoring and notification systems for the entire pipeline

Tools & Technologies

- **Version Control**: GitHub
- **CI/CD**: Jenkins
- **Containerization**: Docker, Docker Hub
- **Container Orchestration**: Kubernetes
- **Configuration Management**: Ansible
- **Cloud Platform**: AWS
- **Scripting**: Bash, Linux
- **Application Stack**: Go, Node.js, Python (from the microservice-app-example)

Milestones & Deadlines

1. Project Setup and Repository Configuration - Week 1 (Days 1-2)
    - Fork and clone the microservice application
    - Set up development environment
    - Initial documentation planning
2. Docker Containerization - Week 1 (Days 3-7)
    - Create Dockerfiles for all microservices
    - Build and test containers locally
    - Push images to Docker Hub
3. Jenkins Pipeline Implementation - Week 2 (Days 1-3)
    - Install and configure Jenkins
    - Create pipeline jobs for each microservice
    - Implement automated testing in the pipeline
4. Ansible Configuration Management - Week 2 (Days 4-7)
    - Develop Ansible playbooks for environment setup
    - Configure deployment automation
    - Test configuration reproducibility
5. AWS Infrastructure Setup - Week 3 (Days 1-3)
    - Provision AWS resources
    - Configure networking and security
    - Prepare for application deployment
6. Kubernetes Orchestration - Week 3 (Days 4-7)
    - Set up Kubernetes cluster
    - Create deployment manifests
    - Implement service discovery and scaling
7. Complete CI/CD Pipeline Integration - Week 4 (Days 1-3)

    - Connect all components into a seamless pipeline
    - Implement notification systems
    - Test full deployment workflow

8. Documentation and Presentation - Week 4 (Days 4-7)
    - Complete technical documentation
    - Prepare final presentation
    - Project delivery

KPIs (Key Performance Indicators)

Infrastructure & Automation

- Successful setup of Jenkins, Docker, and Ansible for automated CI/CD workflows
- All microservices properly containerized with optimized Docker images
- GitHub webhook integration for automated pipeline triggering

Pipeline Efficiency & Performance

- Complete pipeline execution (build through deployment) under 15 minutes
- Zero manual intervention required for deployments
- Successful implementation of multi-stage pipeline (dev, staging, production)

Code Integration & Testing

- Automated unit tests implemented for each microservice
- Integration tests to verify service communication
- All tests automated within the Jenkins pipeline

Deployment & Cloud Management

- Successful deployment to AWS using Kubernetes
- Implementation of rolling updates for zero-downtime deployments
- Proper environment separation and configuration

Monitoring & Reliability

- Implementation of basic monitoring for the application
- Pipeline success/failure notifications via email or Slack
- Documentation of error handling and recovery procedures