DevOps CI/CD Pipeline Report – ACEest Fitness & Gym

This report presents the implementation of an end-to-end DevOps CI/CD pipeline for the ACEest Fitness & Gym web application. The project demonstrates automation, version control, testing, containerization, and continuous deployment across different environments.

1. Project Overview

The ACEest Fitness & Gym project simulates a real-world digital transformation scenario for a fitness startup. A Flask-based microservice was developed to manage gym member data, and a fully automated CI/CD pipeline was built using modern DevOps tools.

Tools used:
- Version Control: Git & GitHub
- Build & CI: Jenkins
- Testing Framework: Pytest
- Code Quality: SonarQube
- Containerization: Docker
- Deployment: Kubernetes (Minikube or Cloud Cluster)
- Registry: Docker Hub

2. CI/CD Architecture

The pipeline follows a modular CI/CD architecture implemented via Jenkins Declarative Pipeline:
1. Source Stage: Jenkins polls the GitHub repository for changes and triggers a new build on commit.
2. Build & Test Stage: Python dependencies are installed, and Pytest runs automated unit tests.
3. Code Quality Stage: SonarQube performs static code analysis.
4. Build Docker Image Stage: The Dockerfile packages the application.
5. Push to Registry Stage: Docker images are tagged and pushed to Docker Hub.
6. Deploy to Kubernetes Stage: Jenkins deploys using kubectl with rolling update.

This ensures each commit is validated, tested, and deployed automatically.

3. Testing and Quality Assurance

Pytest was used for API endpoint testing, and SonarQube ensured code quality compliance. These automated validations strengthened reliability and maintainability.

4. Deployment Strategies

Multiple Kubernetes deployment strategies were configured:
- Rolling Update: gradual rollout of new pods.
- Blue-Green Deployment: two environments with traffic switch.
- Canary Deployment: partial rollout for limited users.
These minimize downtime and ensure safe updates.

5. Automation and Integration

The Jenkins pipeline automates all stages from GitHub pull to Kubernetes deployment. Docker Hub integration ensures traceable image versions, and SonarQube enforces quality gates.

6. Challenges and Mitigation

1. Integration between Jenkins, Docker, and Kubernetes – resolved with credentials and kubeconfig bindings.
2. Maintaining environment consistency – ensured via Docker containerization.
3. Testing failures – resolved with Pytest automation and SonarQube gates.
4. Rollback and recovery – managed via Blue-Green deployment.

7. Key Outcomes

- Fully automated CI/CD pipeline.
- Continuous testing and deployment achieved.
- Kubernetes-managed deployment with rollback.
- Improved speed, quality, and reliability.

8. Conclusion

This project demonstrates the power of DevOps practices in achieving faster, more reliable software delivery. Automation, testing, and container orchestration made the ACEest Fitness deployment efficient and resilient.

Prepared by: morlitarans-lab (GitHub: https://github.com/morlitarans-lab)
Date: November 2025