PROJECT REPORT

MADE BY - PRIYANSHU SANWAL

Project Report

Project Title: AUTOMATED CI/CD Pipeline

Introduction:

This project implements a fully automated CI/CD pipeline for a Flask application using modern DevOps practices. It integrates GitHub Actions for continuous integration, Docker for containerization, Kubernetes for orchestration, and ArgoCD for GitOps-style continuous deployment. Whenever code is pushed to the main branch, the pipeline triggers automated linting, image builds,

manifest updates, and deployment to a Kubernetes cluster.

Abstract

The pipeline enhances developer productivity and ensures consistent deployment workflows. It performs static code analysis (using Flake8), builds and pushes Docker images with timestamp-based tags, updates Kubernetes deployment manifests, and triggers ArgoCD to deploy the new version to the cluster. This approach minimizes manual intervention, reduces errors, and supports robust CI/CD practices.

Tools Used

- Flask: for building the web application
- GitHub Actions: automating CI/CD workflow
- Flake8: linting and code quality
- checks
- Docker: containerizing the Flask application
- Kubernetes / Minikube: container orchestration
- ArgoCD: GitOps-based continuous deployment
- GitHub repository & secrets: storing code and credentials

Steps Involved in Building the Project

1. Local Testing – Verified Flask app locally and containerized with Docker.

2. CI Setup – Configured GitHub Actions for linting, build, and push workflows.

3. Container Build & Push – Built time stamped Docker images and pushed to registry.

4. Manifest Update & Commit – Automatically updated Kubernetes manifests with new image tags.

5. GitOps Deployment – Used ArgoCD to continuously sync and deploy updates to Kubernetes. Conclusion This project successfully demonstrates a complete CI/CD pipeline for a Flask application. By combining GitHub Actions, Docker, Kubernetes, and ArgoCD, it achieves end-to-end automation from code to deployment. The setup improves efficiency, reduces human error, and provides a reliable workflow following DevOps and GitOps best practices.

Features:

- Flask-based web application. - Docker containerization with DockerHub integration.

- GitHub Actions workflow with linting, build, push, and manifest update.

- Kubernetes manifests for deployment and service configuration.

- GitOps workflow with ArgoCD for automatic sync and deployment

Usage:

1. Clone the repository.

2. Build and run the app locally with Docker.

3. Push code to GitHub → workflow automatically lints, builds, and

deploys.

4. Kubernetes manifests auto-update with new image version.

5. ArgoCD syncs manifests and deploys to cluster automatically.

Conclusion

This project successfully demonstrates a complete CI/CD pipeline for a Flask application. By combining GitHub Actions, Docker, Kubernetes, and ArgoCD, it achieves end-to-end automation from code to deployment. The setup improves efficiency, reduces human error, and provides a reliable workflow following DevOps and GitOps best practices.