CI/CD Pipeline with GitHub Actions & Docker

Introduction

In modern software development, Continuous Integration and Continuous Deployment (CI/CD) are crucial for automating the process of building, testing, and deploying applications. This project demonstrates the setup of a CI/CD pipeline using GitHub Actions, Docker, and Minikube to deploy a containerized Flask application locally without requiring any cloud services.

Abstract

The project implements a full CI/CD workflow where a Python Flask application is containerized with Docker, tested using pytest, and automatically built and pushed to Docker Hub via GitHub Actions. Finally, the Docker image is deployed on a local Kubernetes cluster using Minikube. This workflow ensures automation, consistency, and efficiency in application delivery.

Tools Used

- Python & Flask → For building the sample web application.
- Pytest → For unit testing the application.
- Docker → To containerize the application.
- Docker Hub → To store and share the built images.
- GitHub Actions → For CI/CD workflow automation.
- Minikube → To deploy and test the containerized application locally on Kubernetes.
- kubectl → For managing Kubernetes resources.

Steps Involved in Building the Project

1. Application Setup
    - Created a simple Flask application (app.py) with a test file (test_app.py).
    - Added requirements.txt for Python dependencies.
2. Containerization
    - Wrote a Dockerfile to containerize the Flask app.
    - Verified the image locally with Docker.
3. CI/CD Workflow with GitHub Actions
    - Added a GitHub Actions workflow (ci-cd.yml) to:
        - Install dependencies

        - Run tests
        - Build the Docker image
        - Push the image to Docker Hub

4. Configuring GitHub Secrets
    - Stored DOCKER_USERNAME and DOCKER_PASSWORD as GitHub repository secrets.
    - Ensured secure authentication for pushing images.
5. Deployment with Minikube
    - Started Minikube cluster using minikube start.
    - Deployed the app with Kubernetes manifests (deployment.yml, service.yml).
    - Verified pods and services with kubectl get pods and kubectl get svc.
    - Accessed the running Flask app using minikube service flask-service.

Conclusion

This project successfully demonstrates the implementation of a CI/CD pipeline using GitHub Actions, Docker, and Minikube. The pipeline ensures code is automatically built, tested, and deployed whenever changes are pushed to the repository. By leveraging containerization and Kubernetes, the application can be deployed consistently across different environments. This setup provides a strong foundation for scaling to cloud platforms in future projects.

📸 Screenshots (Deliverables)

- ✅ **GitHub Actions Workflow Success**
- ✅ **Docker Hub Image**
- ✅ **Kubernetes Service Running (kubectl get svc)**
- ✅ **Application Running on Browser**

🔗 Deliverables

- **GitHub Repository:** https://github.com/adhirajsingh1/cicd-github-actions-demo
- **Docker Hub Image:** https://hub.docker.com/repository/docker/adhirajsingh1/flask-app/general
- **Workflow Run:** https://github.com/adhirajsingh1/cicd-github-actions-demo/actions/runs/17193601493