CI/CD Pipeline with GitHub Actions & Docker (No Cloud)

Final Project Report

Introduction

In modern software development, automation is crucial for ensuring rapid and reliable software delivery. This project aims to implement a full CI/CD (Continuous Integration and Continuous Deployment) pipeline using only local and open-source tools. The goal is to automate the process of testing, building, and deploying a containerized application without using commercial cloud platforms.

Abstract

The project showcases how to build a cloud-independent CI/CD pipeline for a Docker-based application. The process starts with writing a Dockerfile and configuring Docker Compose for local development. GitHub Actions handles the automation pipeline—running tests, building the Docker image, and pushing it to Docker Hub. The final image is then pulled and deployed locally using Minikube or a virtual machine, simulating a production environment. This approach ensures efficient local testing and deployment while adhering to best DevOps practices.

Tools Used

- **GitHub Actions**: Automates the CI/CD pipeline on code changes.
- **Docker**: Containerizes the application for portability and consistency.
- **Docker Hub**: Hosts and stores the built Docker images.
- **Minikube / Local VM**: Used for local deployment of the containerized application.
- **Docker Compose**: Facilitates local development with multi-service configuration.

Steps Involved in Building the Project

1. **Application Containerization**
    - A Dockerfile was created to package the application with all dependencies.
    - Docker Compose was used for easier local orchestration of services.

2. **CI/CD Workflow with GitHub Actions**
    - A YAML-based workflow was configured to trigger on pushes to the main branch.
    - The workflow includes steps to install dependencies, run tests, build the Docker image, and push it to Docker Hub.
    - Docker credentials were stored securely using GitHub Secrets.
3. **Image Hosting on Docker Hub**
    - Successfully built images were pushed to a public repository on Docker Hub for accessibility.
4. **Local Deployment with Minikube or VM**
    - Minikube was set up to run a lightweight Kubernetes cluster locally.
    - The deployed image was pulled from Docker Hub and exposed using Kubernetes manifests or simple Docker commands.
    - The application was accessed locally and validated for correct functionality.
5. **Verification**
    - The deployed app was tested in the browser and screenshots were taken for documentation.

Conclusion

The project successfully demonstrates a fully functional CI/CD pipeline without relying on commercial cloud infrastructure. By leveraging GitHub Actions, Docker, and local deployment tools, the process of building, testing, and deploying an application has been fully automated. This setup is ideal for local development, testing, and learning DevOps principles. It also provides a strong foundation for scaling into more complex environments or integrating cloud platforms in the future.

✅ **GitHub Repository:**
https://github.com/Cgoyal-Developer/CI-CD-Pipeline-with-GitHub-Actions-Docker.git

✅ **Docker Hub Image:**
https://hub.docker.com/repository/docker/cgoyaldeveloper/titanic-project/general