Abstract: CI/CD Pipeline Project

Project Title: End-to-End CI/CD Pipeline Using GitHub Actions, Docker, and Kubernetes

Abstract:

This project demonstrates the implementation of a complete CI/CD (Continuous Integration and Continuous Deployment) pipeline using modern DevOps tools such as GitHub Actions, Docker, and Kubernetes. It serves as a foundational example for deploying a containerized Python application through automated workflows and infrastructure orchestration.

The core application is a lightweight Python-based web service located in the `app/` directory. It includes a `Dockerfile` for building a container image and a `requirements.txt` file to manage dependencies. This containerized app is suitable for scalable deployment across different environments.

The `docker-compose.yml` file allows local development and testing by setting up the application environment using Docker Compose. It provides an easy way to run and test the application before pushing changes to the source repository.

For the CI/CD pipeline, GitHub Actions has been integrated through the `.github/workflows/docker-build.yml` file. This workflow automates the build and push of Docker images to a container registry (e.g., Docker Hub) whenever changes are pushed to the repository. This ensures that the latest version of the application is always available in a reproducible format.

The project also includes Kubernetes configuration (`kubernetes/deployment.yaml`) for orchestrating

Abstract: CI/CD Pipeline Project

the deployment of the containerized application in a production-like environment. The YAML file defines the deployment strategy, container specs, and replicas, enabling high availability and scalability.

This project is a practical demonstration of DevOps principles, showcasing continuous delivery pipelines, containerization, and orchestration. It is ideal for learners or professionals looking to understand the process of building, deploying, and scaling applications using open-source tools and cloud-native technologies.

Key Technologies Used:

- Python (Flask or similar micro-framework)
- Docker & Docker Compose
- GitHub Actions
- Kubernetes
- YAML for configuration

Learning Outcomes:

- Build a Docker image from a Python application
- Automate builds and deployments with GitHub Actions
- Run containerized services locally using Docker Compose
- Deploy and manage applications on Kubernetes using declarative YAML files
- Understand DevOps practices for continuous integration and delivery

This project provides a complete DevOps lifecycle implementation, from development to production, promoting best practices in software development and system operations.