**CI/CD Pipeline with GitHub Actions and Docker**

**Introduction**

Continuous Integration and Continuous Deployment (CI/CD) pipelines are essential for modern software development. They automate the build, test, and deployment processes, ensuring faster delivery of high-quality applications. This project demonstrates setting up a full CI/CD pipeline using GitHub Actions, Docker, and a local deployment environment (Minikube), without relying on cloud services. This local setup is cost-effective and particularly useful for testing and learning purposes without external dependencies.

Abstract The objective of this project was to design and implement a complete CI/CD pipeline. The pipeline automatically builds a Docker image of an application, runs basic tests to verify its functionality, pushes the image to Docker Hub, and deploys the application locally using Minikube or a local virtual machine. By automating these stages, manual intervention is minimized, enhancing deployment reliability and speed. This project highlights the fundamental DevOps principles of continuous integration, continuous delivery, and containerization.

**Tools Used**

* GitHub Actions – For automating CI/CD workflows, integrating smoothly with GitHub repositories.
* Docker – To create, build, and manage containerized applications.
* Docker Hub – A public registry for storing and sharing container images.
* Minikube – A tool that runs a single-node Kubernetes cluster locally for testing deployments.
* Linux Shell – For managing deployments and cluster operations via CLI commands.
* Python Flask – Used for creating a lightweight, easily deployable web application.

**Steps Involved in Building the Project**

1. Application Setup A simple Python Flask web application was developed. The application includes a single endpoint that returns a "Hello, World!" message, ensuring simplicity and focus on the CI/CD process.
2. Dockerization A Dockerfile was created to containerize the Flask application. A docker-compose.yml file was also written to simplify local testing and development.
3. GitHub Actions Workflow A GitHub Actions workflow was created under .github/workflows/ci-cd.yml to:
   * Checkout code from the GitHub repository.
   * Set up Docker Buildx to support building and pushing multi-platform Docker images.
   * Log in to Docker Hub securely using GitHub secrets.
   * Build and push the Docker image automatically on every push to the main branch.
4. Local Deployment After a successful build and push, the image was pulled onto a local Minikube Kubernetes cluster:
   * Deployment and Service YAML manifests were created and applied.
   * Alternatively, Docker Compose was used to spin up the service locally for testing.
5. Testing and Validation The application was tested by accessing the service URL provided by Minikube, confirming the deployment was successful and the application was running as expected.

**Conclusion**

This project demonstrated a practical, complete lifecycle of an automated CI/CD pipeline using free and open-source tools. Implementing CI/CD locally teaches essential practices such as code automation, containerization, and service deployment without relying on paid cloud services. This pipeline can be expanded with additional testing stages, multi-environment deployments (e.g., staging and production), and monitoring integrations to emulate real-world DevOps environments. Overall, it emphasizes the efficiency, reliability, and speed achievable through automation and modern development practices.