ELEVATE LABS

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

Prepared by: Sonali Pathare

Role: DevOps Engineer / Intern

INTRODUCTION :

In today's DevOps-driven world, automation is the backbone of modern software delivery. The goal of this project is to design and implement a CI/CD pipeline that automates building, testing, and deploying applications using GitHub Actions and Docker, all within a local environment. This setup demonstrates how continuous integration (CI) and continuous deployment (CD) can be achieved effectively without cloud dependencies. It focuses on improving software quality, minimizing manual intervention, and ensuring reliable builds.

ABSTRACT :

This project implements a complete CI/CD pipeline using GitHub Actions and Docker, enabling automated build and deployment processes directly from a GitHub repository.

The pipeline performs the following actions:

1. Checks out the latest code from GitHub.
2. Builds a Docker image using a Dockerfile.
3. Runs unit tests to verify the application functionality.
4. Pushes the verified image to Docker Hub using stored credentials.
5. Deploys the image locally via Docker Desktop or a local virtual machine.

This approach ensures that every code update triggers a standardized process, providing faster feedback and maintaining code stability through automated testing and deployment.

TOOLS USED :

- GitHub Actions - Workflow automation and CI/CD orchestration
- Docker - Containerization platform for building and deploying applications
- Docker Hub - Image registry for storing and sharing built images
- GitHub - Version control system for managing source code
- VS Code / Terminal - For local development and configuration

- Docker Desktop / Local VM - For running and validating deployments

STEPS INVOLVED IN BUILDING THE PROJECT :

1. Application Setup:
   - Created a sample web application (Python or Node.js).
   - Initialized a GitHub repository and pushed source code.
2. Docker Configuration:
   - Wrote a Dockerfile defining image build instructions.
   - Added a docker-compose.yml to manage containers easily.
   - Verified build and run locally using docker build and docker run.
3. CI/CD Workflow Setup:
   - Created a .github/workflows/main.yml file for automation.
   - Configured GitHub Actions jobs to build, test, and push images.
   - Integrated Docker Hub credentials using GitHub Secrets (DOCKER_USERNAME, DOCKER_TOKEN).
4. Automated Image Push to Docker Hub:
   - Successfully pushed the built Docker image to a Docker Hub repository.
5. Local Deployment and Testing:
   - Pulled the image from Docker Hub using docker pull.
   - Deployed locally using Docker Desktop or a VM.
   - Verified deployment accessibility via http://localhost:8080.
6. Pipeline Validation:
   - Confirmed workflow automation by pushing code changes and observing successful builds and deployments in the GitHub Actions tab.

CONCLUSION :

The CI/CD pipeline with GitHub Actions and Docker successfully automated the entire software delivery process from build to deployment, without any cloud dependency.

It demonstrated the power of integrating version control, automation, and containerization to achieve a consistent and reliable deployment cycle.

This project improved understanding of:

- Continuous Integration / Deployment workflows
- Docker container management

- Secure credential handling using GitHub Secrets
- Local deployment automation

The outcome reinforces DevOps principles of automation, consistency, and faster delivery, serving as a foundation for scaling pipelines to cloud or Kubernetes-based environments.

DELIVERABLES :

- GitHub Repository: github.com/sonalipathare05/cicd--without-cloud-vm
- Workflow File: .github/workflows/main.yml
- Dockerfile & docker-compose.yml
- Docker Image: Available on Docker Hub
- Screenshots: CI/CD build logs and deployed app verification