Project Report

Topic: CI/CD Pipeline with GitHub Actions & Docker

By: Vinodhini V

Date: 08 September 2025

Introduction

Continuous Integration and Continuous Deployment (CI/CD) are essential practices in modern software development. They automate the process of testing, building, and deploying applications, ensuring faster delivery and higher software quality. This project focuses on building a CI/CD pipeline for a simple Flask-based To-Do application. The pipeline uses GitHub Actions to run automated tests, build Docker images, and push them to Docker Hub. This allows developers to ensure code quality and deployable artifacts without manual intervention.

Abstract

This project implements a CI/CD pipeline for a Python Flask To-Do application. The workflow includes automated testing with Pytest, containerization with Docker, and automated builds and deployments using GitHub Actions. The Docker image is pushed to Docker Hub upon successful tests, ensuring reproducibility and portability. The project demonstrates the end-to-end DevOps life cycle with version control (Git), CI/CD automation (GitHub Actions), containerization (Docker) and deploy it to the local Virtual Machine (VM).

Tools Used

- Programming Language: Python (Flask framework)
- Database: SQLite (lightweight relational database)
- Version Control: Git & GitHub
- Testing Framework: Pytest
- Containerization: Docker & Docker Compose
- CI/CD Platform: GitHub Actions
- Secrets Management: GitHub Secrets (for Docker Hub credentials)
- Deployment: Local Virtual Machine (VM)

Steps Involved in Building the Project

- Setup Flask Application - Created a basic Flask API (`app.py`) to manage To-Do tasks. Used SQLite as the database for storing tasks.
- Write Unit Tests - Implemented test cases using `pytest`. Configured tests to run against an in-memory SQLite database.

- Dockerize the Application - Wrote a `Dockerfile` to containerize the Flask application. Created a `docker-compose.yml` file for local testing and multi-container setups.
- Configure GitHub Actions Workflow- Defined .github/workflows/ci-cd.yml file.
    - Job 1: Install dependencies and run tests.
    - Job 2: Build and push Docker image to Docker Hub. Used GitHub Secrets (`DOCKER_USERNAME`, `DOCKER_PASSWORD`) for authentication.
- Version Control with Git & GitHub - Resolved Git conflicts using `git pull --rebase`. Pushed code, Dockerfile, tests, and workflow files to GitHub.
- Run CI/CD Pipeline - Verified pipeline execution under the GitHub Actions tab. Confirmed successful test runs and Docker image pushed to Docker Hub.

URL

- Github repo: https://github.com/Vinodhini02/todo-ci-cd.git
- Docker: docker pull vinu2/todo-flask:local

Conclusion

This project highlights the importance of DevOps practices in modern software engineering. Containerization and CI/CD pipelines significantly improve reliability, scalability, and developer productivity. The same approach can be extended to cloud platforms and production-grade deployments in the future.