**PROJECT REPORT - 2**

**Objective: Implemented CI/CD Pipeline with Github Actions & Docker**

Implemented a CI/CD Pipeline solution that automates fetches code commit from git, run tests, builds Docker image and pushes to docker hub. The Software process leverages Github Actions for orchestration, pytest for test cases and docker containerization to build docker image maintaining software automation delivery practice

Project Repository: <https://github.com/3lton007/CI-CD-Pipeline-with-GitHub-Actions-Docker>

Docker Image - eltonaloys/flask-ci-cd-app

**Tools Used**

* **Source Control:** Github Repository
* **CI/CD Engine:** Github Actions
* **Containerization:** Docker
* **Registry:** Docker Hub
* **Application Testing:** Automated test cases with Pytest

**Implementation:**

Sample Route based application was implemented, with py test cases to implement CI/CD pipeline using Github Actions. Application Project was committed to Github Repository, where using the cicd.yml script, Github Actions initiated the automation process of running the test cases for the application. Once the test cases were approved, the project was built into a docker image and deployment was pushed to Docker Hub. The Newly build image was pulled locally using “docker pull” to test in local environment.

**Process Workflow:**

1) Pushing route based flask application to github repository. Code commit triggers the Github Action Workflow.

2) Initial enviroment setup, Pytest execution and validation

3) Docker Image is built using the Dockerfile, and pushed to Docker Hub.

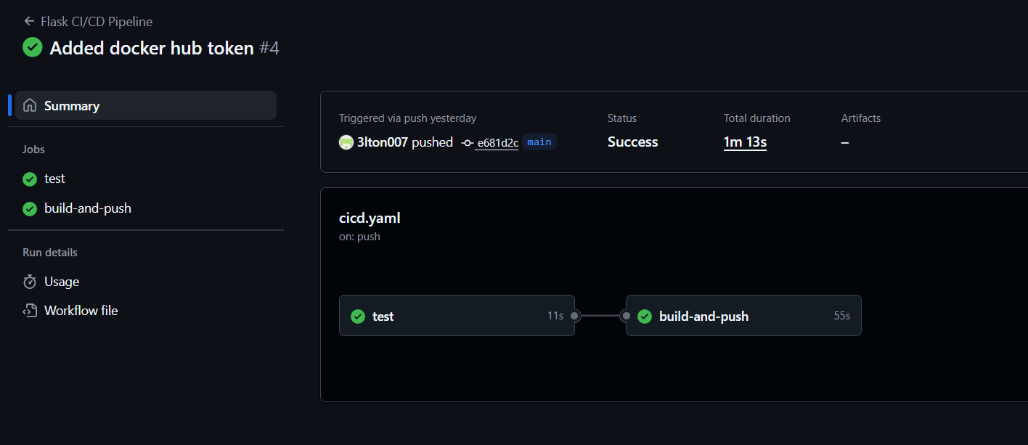
4) Verified Newly build Docker Image locally using docker pull

**Analysis & Insights:**

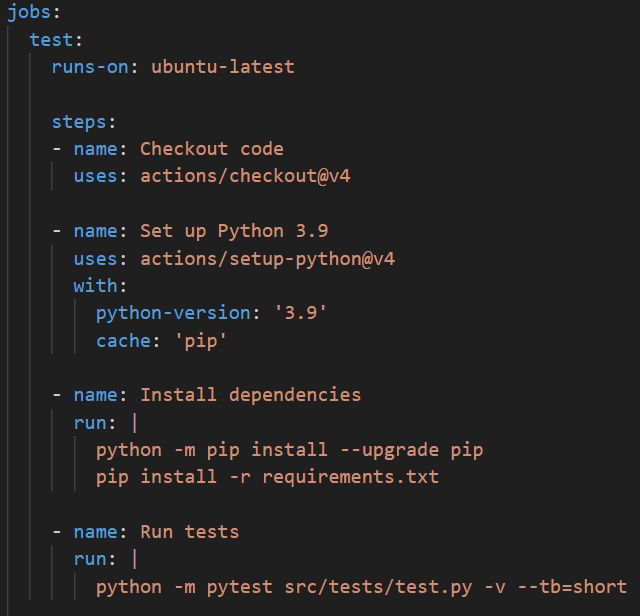
Understanding the core concept of continuous integration and continuous deployment process through github action pipeline. Some of the Challenges faced were regarding writing yaml script for automation, indentation, syntaxes etc. The Build Process performance took between 30 seconds to 1 minute. Docker Container size was around 100mb. The Deployment time was reduced and test coverage completed 100% for critical endpoints.

**Screenshots:**

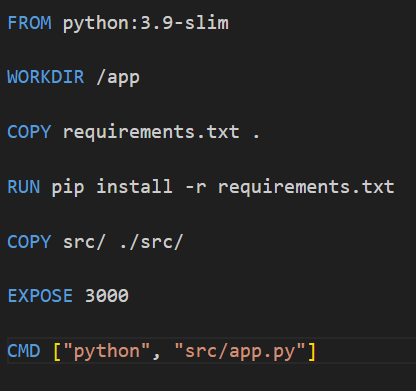
**Github Actions CI/CD Pipeline**

****

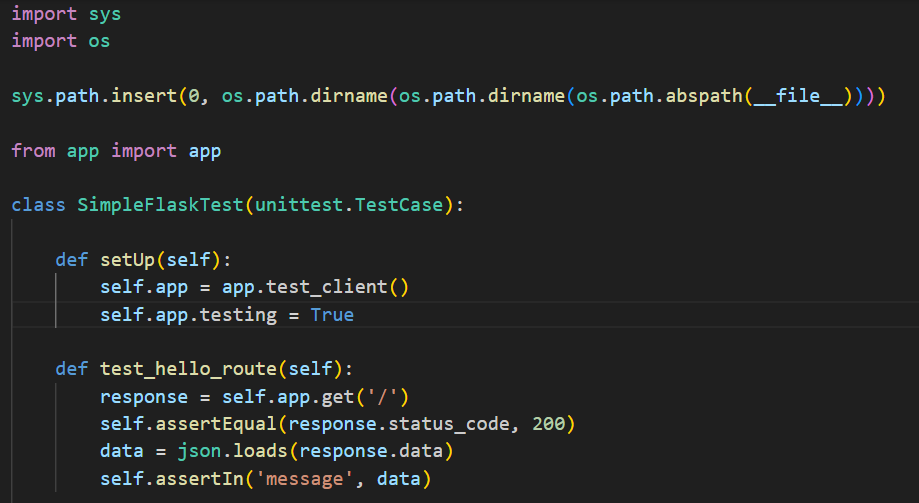
**Github Action YAML Script**

****

**Dockerfile**

****

**Flask Python Test Cases**

****

**Conclusion:** Successfully implemented CI/CD Pipeline (without Cloud) that runs tests, creates Docker images and that can be deployed locally.