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

1. Introduction

In contemporary software development, Continuous Integration and Continuous Deployment pipelines are crucial for the fast, consistent, and automated delivery of applications. This project is concerned with designing and creating a full CI/CD pipeline using GitHub Actions, Docker, and Docker Hub. The pipeline should be designed to create Docker images, execute automated tests, and deploy locally without the need for external cloud services.

2. Objectives

• Automate the build, test, and deployment process with GitHub Actions.

• Develop a Dockerfile and docker-compose.yml for containerization.

• Push Docker images to Docker Hub for versioned container management.

•Deploy and run the application locally with Minikube or a VM.

3. Tools and Technologies

The following tools and platforms were used in this project:

• GitHub Actions, for setting up automated workflows.

• Docker, to containerize the Flask application.

• Docker Hub, for storing and sharing container images.

• Minikube/Local VM, for local deployment and testing.

4. Methodology

The project utilizes a systematic process to execute the CI/CD pipeline:

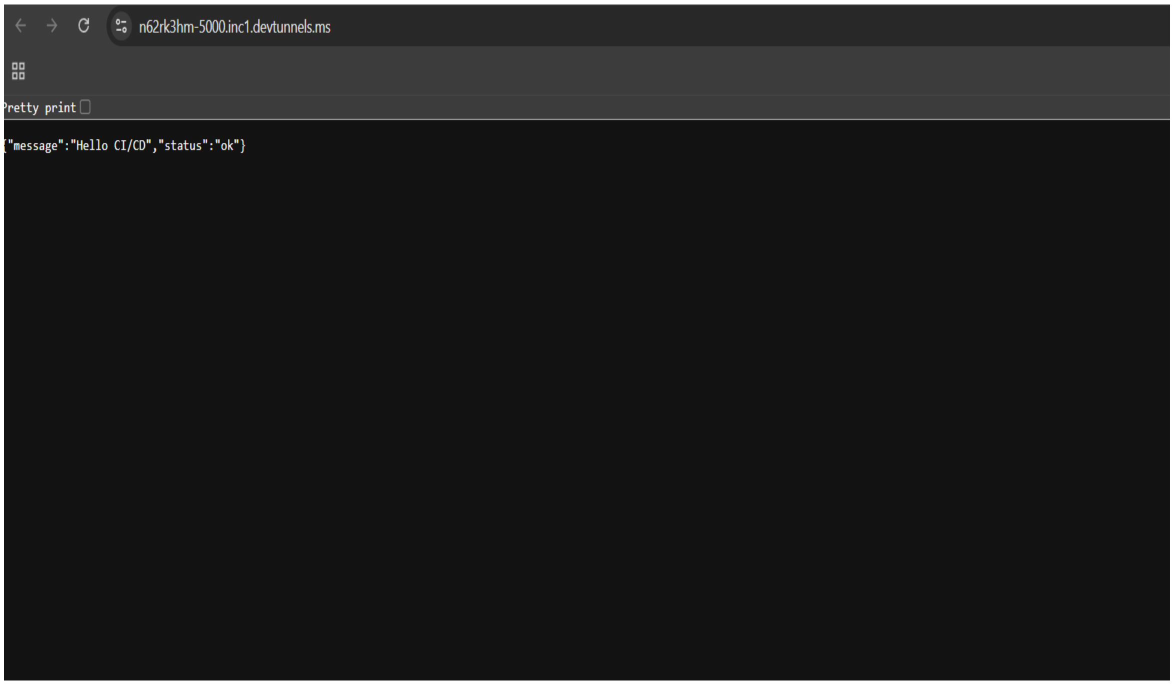
- Create a basic Flask application (app.py).
- Declare project dependencies in requirements.txt.
- Create Dockerfile and docker-compose.yml for containerizing.
- Create automated unit tests (test_app.py) with Pytest.
- Create GitHub Actions workflow (ci.yml) to build, test, and push Docker image.
- Deploy and run the Docker image locally with Minikube or Docker CLI.

5. Deliverables

The project delivers the following outputs:

- A GitHub repository with all source code and workflow files.
- A Docker Hub repository containing the built Docker images.
- CI/CD workflow logs from GitHub Actions.
- Screenshots and logs of the deployed Flask application.

6. Output



```
{"message":"Hello CI/CD","status":"ok"}
```

6. Conclusion

This project effectively shows how a CI/CD pipeline with GitHub Actions and Docker can be implemented without the use of cloud services. The pipeline tests, builds, and deploys automatically, hence providing quicker and more consistent software delivery. The method can be applied to more sophisticated applications and production-level deployments in the future