**DevOps Internship Project Report**

**Project 4: CI/CD Pipeline with GitHub Actions & Docker**

**Name:** MALLANGI HARI NATH REDDY  
**Date of Submission:** 9-09-2025

**1. Introduction**

This project demonstrates the implementation of a complete CI/CD (Continuous Integration and Continuous Deployment) pipeline for a Dockerized Flask web application using GitHub Actions. The pipeline automates the process of testing, building Docker images, and pushing them to Docker Hub registry on every code commit, ensuring consistent and reliable deployments.

**2. Abstract**

The goal was to create an automated workflow that:

* Triggers on code changes in GitHub repository
* Runs automated tests to validate code quality
* Builds a Docker container image of the application
* Pushes the image to Docker Hub with version tags
* Simulates deployment processes

The implementation uses GitHub Actions for CI/CD orchestration, Docker for containerization, Docker Hub as the container registry, and pytest for testing framework. This setup provides a foundation for modern DevOps practices that can be extended to actual production deployments.

**3. Tools Used**

| Tool | Purpose | Version |
| --- | --- | --- |
| **GitHub Actions** | CI/CD Pipeline Orchestration | - |
| **Docker** | Containerization Platform | 20.10+ |
| **Docker Hub** | Container Image Registry | - |
| **Python** | Application Runtime | 3.9 |
| **Flask** | Web Framework | 2.3.3 |
| **Pytest** | Testing Framework | 7.4.0 |
| **Docker Compose** | Local Development | 2.0+ |

**4. Steps Involved in Building the Project**

**Step 1: Application Development**

* Created a simple Flask web application (app.py)
* Added unit tests for application validation (test\_app.py)
* Defined Python dependencies (requirements.txt, requirements-test.txt)

**Step 2: Containerization**

* Created Dockerfile for building application image
* Configured docker-compose.yml for local development
* Tested container build and run process locally

**Step 3: CI/CD Pipeline Configuration**

* Created GitHub Actions workflow (.github/workflows/ci-cd-pipeline.yml)
* Configured triggers for push and pull request events
* Set up job sequencing: test → build → push → deploy simulation

**Step 4: Docker Hub Integration**

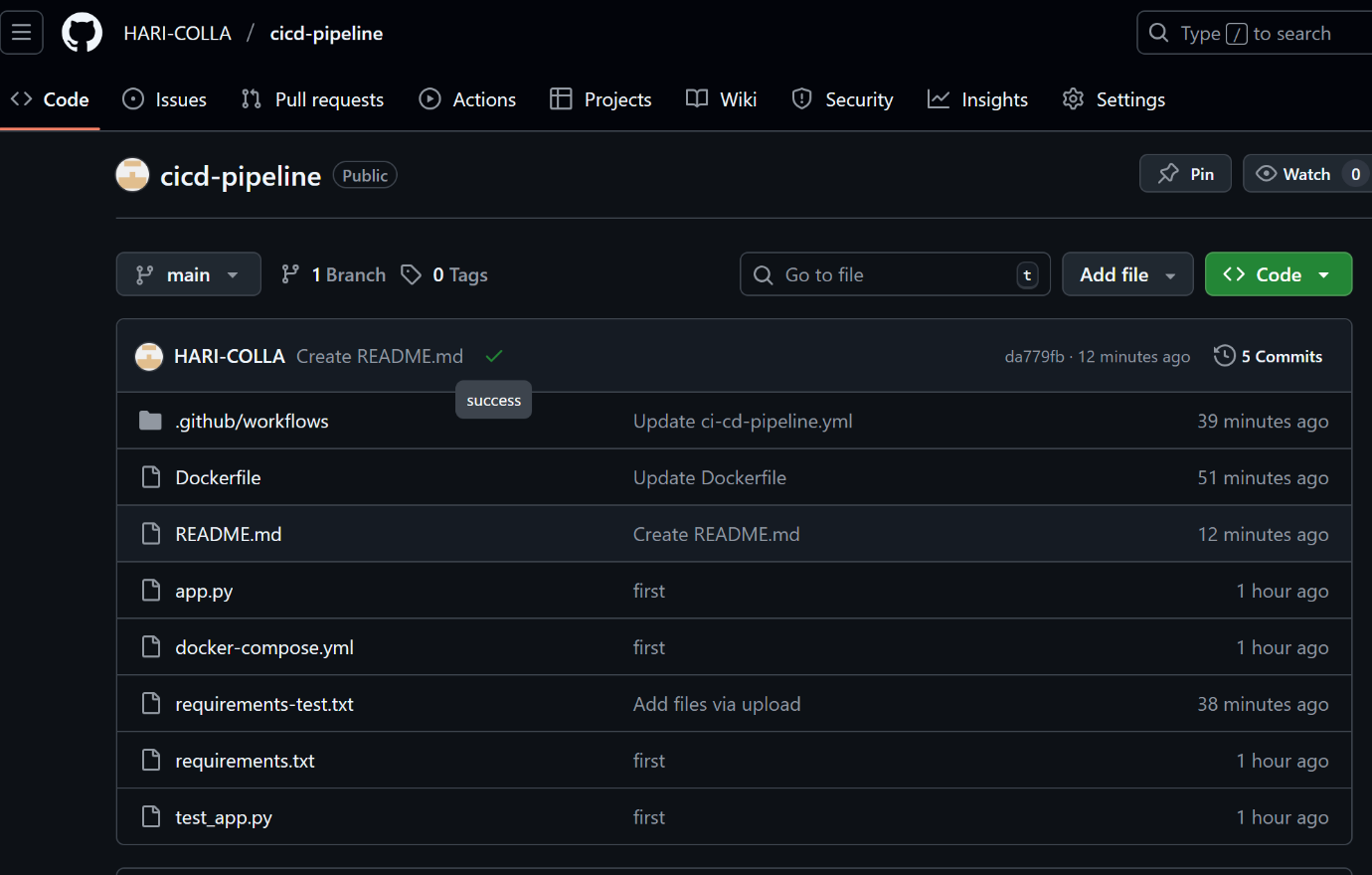
* Created Docker Hub account and repository
* Configured GitHub Secrets for secure authentication:
  + DOCKERHUB\_USERNAME
  + DOCKERHUB\_TOKEN (access token)

**Step 5: Testing and Validation**

* Tested pipeline execution with code changes
* Verified automated testing functionality
* Confirmed successful image pushing to Docker Hub
* Validated local deployment simulation

**5. Screenshots Evidence**

**Screenshot 1: GitHub Repository Structure**

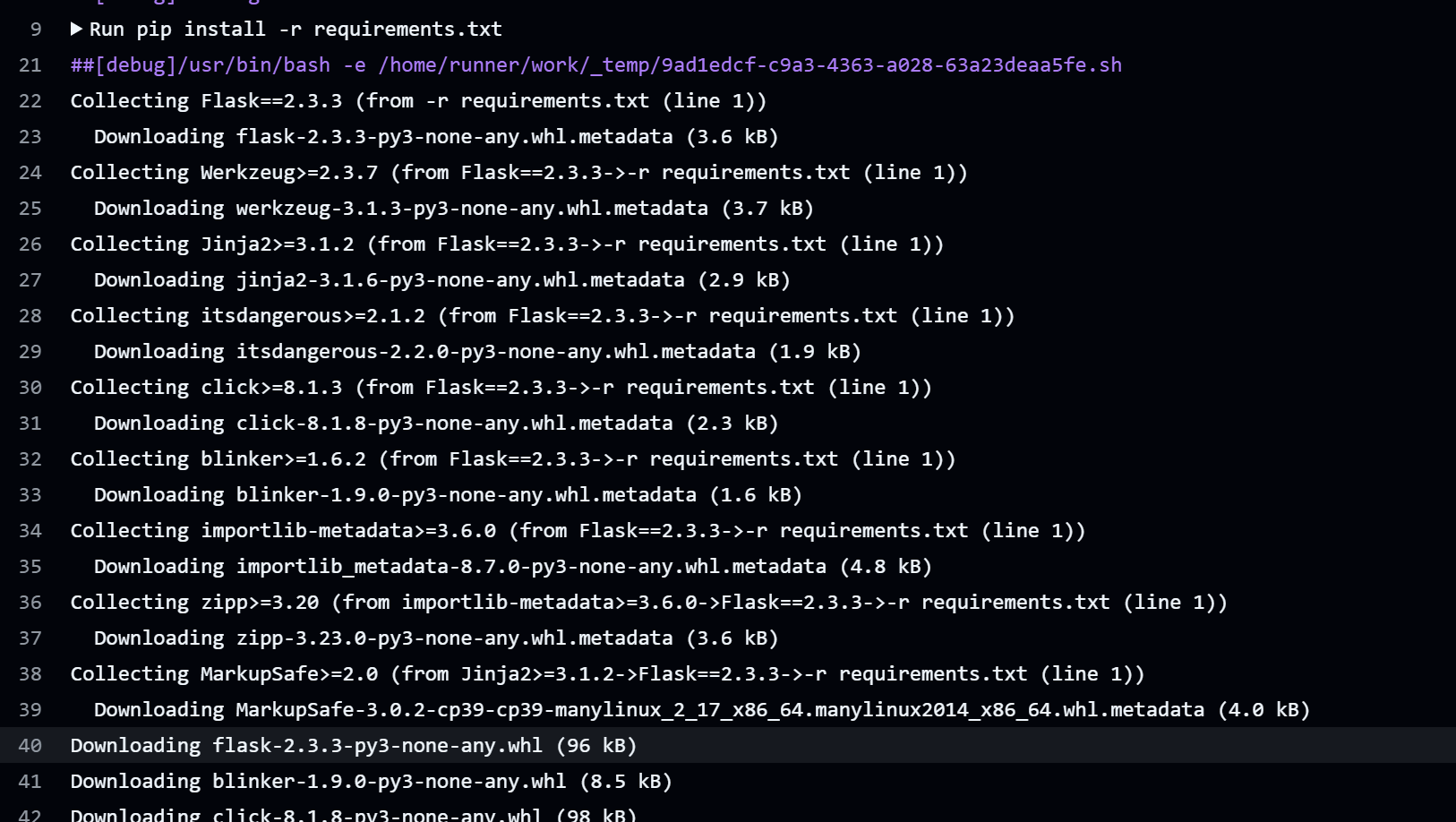


**Screenshot 2: GitHub Actions Workflow Success**

*A screenshot of a computer

AI-generated content may be incorrect.*

**Screenshot 3: Test Execution Results**

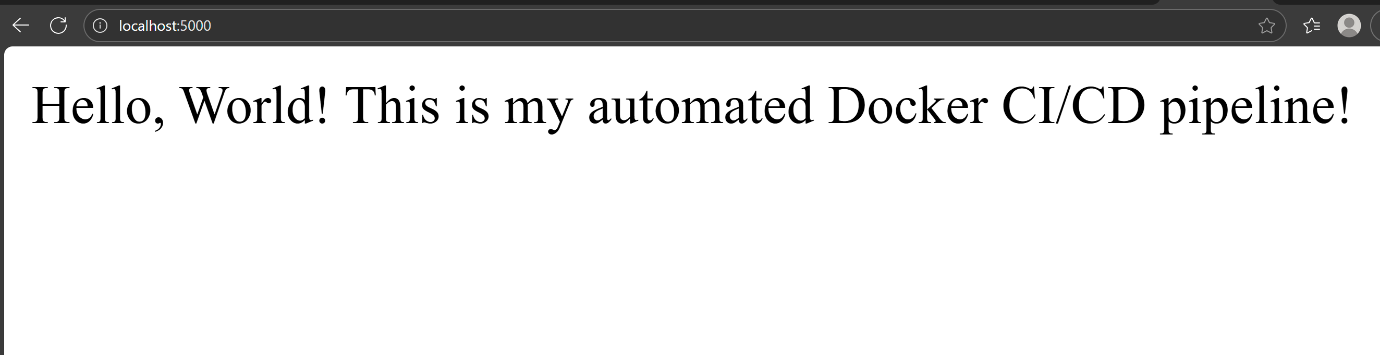


**Screenshot 4: Docker Hub Repository with Images**

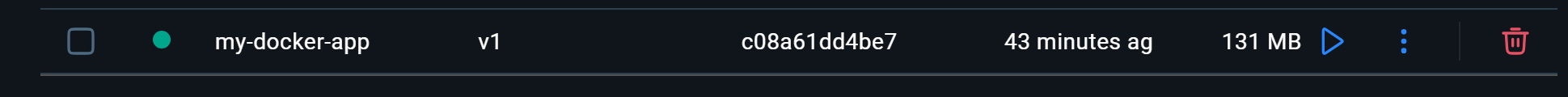
A screenshot of a computer

AI-generated content may be incorrect.

**Screenshot 5: Local Application Running**

****

**Screenshot 6: Docker Containers Running Locally**

**

**6. Challenges Faced and Solutions**

1. **Challenge:** No module named pytest error in GitHub Actions  
   **Solution:** Created separate requirements-test.txt and installed test dependencies explicitly
2. **Challenge:** Docker Hub authentication failure  
   **Solution:** Used Docker Hub access token instead of password in GitHub Secrets
3. **Challenge:** Port conflicts during local testing  
   **Solution:** Used different host ports and verified port availability

**7. Learning Outcomes**

* Gained hands-on experience with GitHub Actions CI/CD pipelines
* Learned Docker image building and container management
* Understood secure credential management with GitHub Secrets
* Practiced automated testing integration in CI pipelines
* Developed troubleshooting skills for pipeline failures

**8. Conclusion**

This project successfully demonstrates a modern CI/CD pipeline implementation using free-tier tools. The automated workflow ensures that every code change is properly tested, packaged, and made ready for deployment. This foundation can be extended to include actual deployment to cloud platforms, more comprehensive testing, security scanning, and monitoring integration.

The skills developed through this project are directly applicable to real-world DevOps roles and provide a solid understanding of essential CI/CD concepts and practices.

**9. References**

* [GitHub Actions Documentation](https://docs.github.com/en/actions)
* [Docker Documentation](https://docs.docker.com/)
* [Flask Documentation](https://flask.palletsprojects.com/)
* [Pytest Documentation](https://docs.pytest.org/)