Project Report

Name: Harshad Dhane

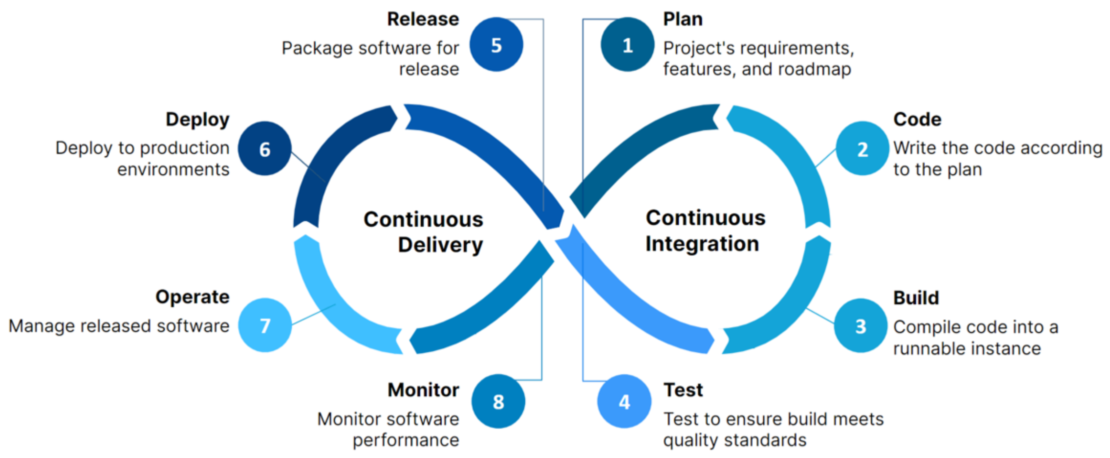
ID: 2024HT66510

Introduction to DEVOPS

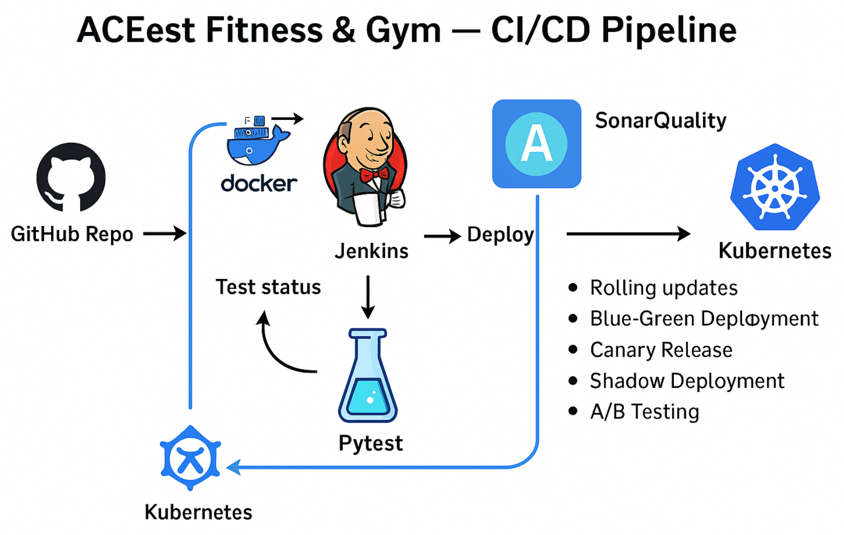
(Merged - CSIZG514/SEZG514) (S1-25)

Github Link: https://github.com/Harshad141/ACEest_Fitness/tree/main

CI/CD Architecture Overview


Release
5
Package software for release
Plan
1
Project's requirements, features, and roadmap
Deploy
6
Deploy to production environments
Code
2
Write the code according to the plan
Continuous Delivery
Continuous Integration
Operate
7
Manage released software
Build
3
Compile code into a runnable instance
Monitor
8
Monitor software performance
Test
4
Test to ensure build meets quality standards

The project transformed a Tkinter desktop app into a modular Flask web application and deployed it through a fully automated CI/CD pipeline. The architecture simulates a real-world DevOps environment for ACEest Fitness & Gym.
We had to simulate an industry-grade DevOps environment, implementing a fully automated Continuous Integration and Continuous Delivery (CI/CD) pipeline. The assignment emphasizes hands-on engagement with leading DevOps tools while reinforcing conceptual understanding of automation, testing, version control, and deployment strategies in real-world cloud and containerized environments.



Tools Integrated

- **Git & GitHub** for version control
- **Jenkins** for CI/CD orchestration
- **Pytest** for automated testing
- **SonarQube** for code quality and coverage analysis
- **Docker** for containerization
- **Kubernetes (Minikube)** for deployment and scaling
- **VS Code** for modular development environment

Pipeline Flow

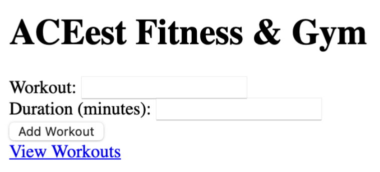
1. Developer commits code to GitHub
2. Jenkins builds Docker image
3. Pytest runs inside container
4. SonarScanner sends results to SonarQube
5. Deployment to Minikube cluster
6. Advanced rollout strategies: rolling updates, blue-green, canary,

Pipeline flow diagram

```
[ Developer ]
     |
     v
[ GitHub Commit ]
     |
     v
[ Jenkins Build ]
     |
     v
[ Docker Image ]
     |
     v
[ Pytest Tests ]
     |
     +--> [ SonarScanner → SonarQube Analysis ]
     |
     v
[ Kubernetes (Minikube) Deployment ]
     |
     v
[ Advanced Rollout Strategies ]
    ├── Rolling Updates
    ├── Blue-Green Deployment
    ├── Canary Release
    ├── Shadow Deployment
```

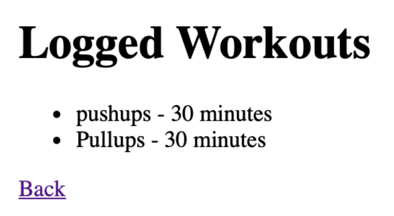
Application Deployment

Application deployment for the python application was done using Flask
Attached output screenshots below:



ACEest Fitness & Gym

Workout:
Duration (minutes):



Logged Workouts

- pushups - 30 minutes
- Pullups - 30 minutes

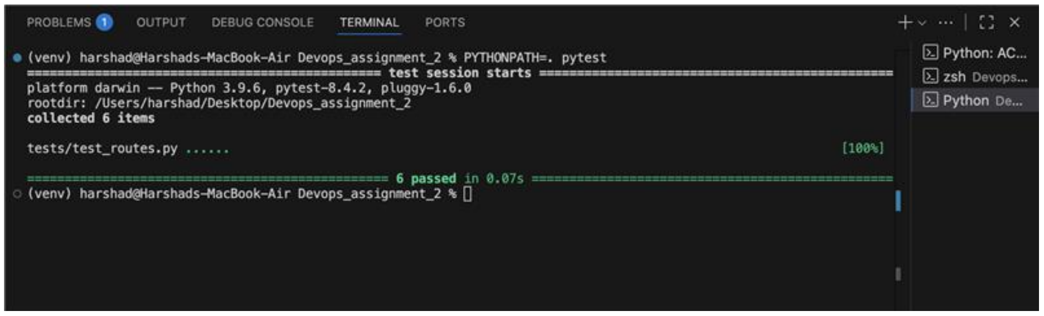
Back

Test Cases using Pytest

Please refer the code test_routes.py to understand the funtions

test_home_page: Verify returns 200 and shows "ACEest Fitness & Gym".
test_add_workout: Verify posting a valid workout adds it and returns 200.
test_view_workouts: Verify lists workouts after adding one.
test_add_workout_empty_fields: Verify empty fields show "Please enter both workout and duration."
test_add_workout_invalid_duration: Verify non-numeric duration shows "Duration must be a number.
test_multiple_workouts: Verify multiple workouts added

Output:



```
● (venv) harshad@Harshads-MacBook-Air Devops_assignment_2 % PYTHONPATH=. pytest
=============================================== test session starts ===============================================
platform darwin -- Python 3.9.6, pytest-8.4.2, pluggy-1.6.0
rootdir: /Users/harshad/Desktop/Devops_assignment_2
collected 6 items

tests/test_routes.py ......                                                                                 [100%]

================================================ 6 passed in 0.07s ================================================
○ (venv) harshad@Harshads-MacBook-Air Devops_assignment_2 % []
```

Challenges Faced & Mitigation Strategies

- **Jenkins & Java Setup** →
  Jenkins failed due to missing Java runtime.

Solution: Installed OpenJDK via Homebrew, configured `JAVA_HOME`, and registered Java 17.

- **SonarQube Integration** → SonarScanner couldn't locate coverage reports.

Solution: Generated `coverage.xml` via Pytest, updated `sonar-project.properties`, and integrated scanner into Jenkins pipeline.

- **Docker & Minikube Compatibility** → Minikube couldn't access local Docker images.

Solution: Used `minikube image load` to push images into Minikube's internal registry.

Key Automation Outcomes

CI/CD Pipeline Success

- Jenkins automates build, test, and analysis
- Pytest ensures stability
- SonarQube enforces quality gates and tracks coverage

Containerization & Deployment

- Docker ensures consistent environments
- Minikube enables Kubernetes deployment
- Advanced strategies implemented: rolling updates, rollback, blue-green, canary, shadow, A/B testing

Modular Codebase

- Flask app structured for maintainability
- Tests isolated in `tests/` directory

- CI/CD logic centralized in `Jenkinsfile` and `k8s/` manifests

Personal Feedback

This project taught me to think like a DevOps engineer — building systems that are testable, observable, and deployable. I learned to troubleshoot integration issues, automate quality checks, and deploy confidently using Kubernetes.

It was challenging but rewarding, and I now feel prepared to design CI/CD pipelines from scratch, integrate quality gates, and deploy applications using modern DevOps practices.