BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DEVOPS (Merged - CSIZG514/SEZG514) (S1-25)

Assignment - 2

Assignment Type: DevOps CI/CD implementation for ACEest Fitness & Gym

Assignment Context and Rationale:

In today's technology-driven business landscape, automation, agility, and continuous delivery form the backbone of modern software engineering practices. This assignment places students in the role of a DevOps Engineer for ACEest Fitness & Gym, a dynamic fitness startup undergoing digital transformation. The organization is embracing incremental and test-driven development, aiming to streamline its end-to-end software delivery process using contemporary DevOps tools and methodologies.

Students will 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.

Learning Objectives:

By successfully completing this assignment, students will be able to:

1. Apply DevOps principles to design and implement a fully automated complete CI/CD pipeline.
2. Demonstrate proficiency in using industry-standard tools:
    - Version Control: Git and GitHub
    - Build Automation: Jenkins
    - Testing Framework: Pytest
    - Code Quality: SonarQube
    - Containerization: Docker or Podman
    - Container Registry: Docker Hub/ local registry like quay.io
    - Deployment and Orchestration: Minikube / Kubernetes (AWS, Azure, or GCP)
3. Integrate automated testing and quality validation into the build workflow.
4. Employ progressive deployment strategies ensuring zero-downtime and rollback capabilities.
5. Establish infrastructure consistency and reliability through containerized deployment and orchestration.
6. Cultivate an understanding of collaborative workflows, code versioning, and continuous improvement practices.

Expected Learning Outcomes:

Upon completion, students will have demonstrated the ability to:
Build an end-to-end automated DevOps pipeline.
Integrate testing, quality assurance, and deployment automation effectively.
Apply real-world deployment and rollback strategies.
Understand how DevOps enhances software quality, speed, and operational resilience.

Assignment Scenario: As the appointed DevOps Engineer for ACEest Fitness & Gym, you are responsible for automating the application's development and deployment lifecycle. The company's product is evolving through multiple incremental versions.

Code Versions given as a zip file: ACEest_Fitness.py, ACEest_Fitness-V1.1, ACEest_Fitness-V1.2, ACEest_Fitness-V1.2.1, ACEest_Fitness-V1.2.2, ACEest_Fitness-V1.2.3, ACEest_Fitness-V1.2.1, ACEest_Fitness-V1.3.

DevOps Assignment 2 on Taxila.zip

Your goal is to establish a **robust, test-driven, and fully automated DevOps pipeline** ensuring code quality, consistency, and reliability across multiple application versions.

Assignment Tasks and Deliverables:

1. Application Development
    - Develop a foundational Flask web application representing the core functionalities of a fitness and gym management system.
    - The provided Python file (ACEest_Fitness.py and other working version of the application) serves as the base for this phase.
    - Implement modular, maintainable code adhering to Pythonic standards and version naming conventions.
2. Version Control System Setup
    - Initialize a Git repository and link it with a remote GitHub repository.
    - Track all incremental versions, new features, bug fixes, and infrastructure changes through structured commits, branching, and tagging strategies.
3. Unit Testing and Test Automation
    - Design and implement unit tests using Pytest to verify the reliability and correctness of the Flask application.
    - Integrate automated test execution within the CI pipeline to ensure that every code change is validated before build or deployment.
4. Continuous Integration with Jenkins
    - Configure Jenkins as the build server.
    - Set up Jenkins hooks to continuously poll the Git repository and trigger builds automatically on code changes.
    - Generate build artifacts corresponding to different application versions.
5. Containerization with Docker and Podman
    - Package the Flask application into a Docker image encapsulating all dependencies.
    - Store and version-control these images using Docker Hub as the central container registry.
6. Continuous Delivery and Deployment Strategies
    - Deploy the containerized application using Kubernetes (local via Minikube or on cloud providers such as AWS, Azure, or GCP).
    - Implement advanced deployment methodologies:
        - Ensure rollback Blue-Green Deployment
        - Canary Release
        - Shadow Deployment
        - A/B Testing
        - Rolling Update
    - mechanisms are in place to revert to the last stable version in case of deployment failure.

7. Automated Build and Testing Integration

- Integrate automated build processes within Jenkins pipelines.
- Execute Pytest-based automated tests inside the containerized environment to validate each deployment.
- Incorporate SonarQube for static code analysis and quality gate enforcement.

Submission Requirements:

- Project folder containing:
    - Flask application files (ACEest_Fitness.py and versions)
    - Jenkins pipeline configuration file (Jenkinsfile)
    - Dockerfile and Kubernetes YAML manifests
    - Pytest test cases and SonarQube report
- GitHub repository link (public or with invited access)
- A short **report (2–3 pages)** summarizing:
    - CI/CD architecture overview
    - Challenges faced and mitigation strategies
    - Key automation outcomes

Submission Guidelines (On or Before) Deadline:

- Ensure your GitHub repository is public and accessible for review.
- Submit the endpoint URL of your running cluster with all types of deployment strategies by the specified deadline.
- The Jenkins workflow should demonstrate successful runs for recent commits.
- SonarQube results for the code quality
- You own docker image -repo with all versions of the application.