Main Test DevOps

Implement a CI/CD pipeline using Jenkins multi-branch pipelines, SonarQube for code quality, Docker for Containerization, GitHub webhooks and Kubernetes for Deployment.

Scenario:

You are part of a DevOps team working on a Java-based microservice.The application is managed in a GitHub repository with multiple branches, each serving a different purpose.You need to create a Jenkins multi-branch pipeline that:

1)Triggers automatically using a GitHub webhook.

2)Builds and tests the Java application using Maven.

3)Performs SonarQube code quality analysis.

4)Packages the application into a Docker container and pushes it to an image registry.

5)Deploys the application to a Kubernetes cluster using YAML manifests.

6)Implements a Git branching strategy to manage different types of deployment work.

I. Branching Strategy and Execution Rules

Branch Type Stages to Execute

Feature/\*(Feature Branch) Build, Test, SonarQube

Develop (Development Branch) Build, Test, SonarQube, Docker Build, Push to Registry

II. Webhook Integration

a. Configure a GitHub webhook that triggers Jenkins when

i) commit is pushed to any branch

ii) new Pull Request is created

III. Building Java Code with Maven

a. Compile the application using Maven.

b. Generate a .jar file artifact.

IV. SonarQube Code Quality Analysis

a. Integrate SonarQube into the Jenkins pipeline.

b. Ensure the build fails if the SonarQube quality gate fails.

V. Docker Containerization and Image Push

a. Write Dockerfile with the following requirements:

i.Base image: openjdk-11-jre-slim

ii.Copies the application JAR from the build output.

iii.Exposes port 8080 (the application port).

iv.Uses ENTRYPOINT ["java", "-jar", "app.jar"].

VI. Kubernetes Deployment

a. Write Kubernetes manifest files that include:

i.deployment.yaml (Defines the app deployment and replicas).

ii.service.yaml (Exposes the application internally).

b. Add Kubernetes apply command to Jenkinsfile to enable deployment.