**CI/CD Documentation for VeroDigit Project**

**Introduction**

In modern software development, Continuous Integration (CI) and Continuous Deployment (CD) play a crucial role in automating the build, testing, and deployment processes. This document outlines the CI/CD pipeline configuration for the VeroDigit project, which integrates a Spring Boot backend and an Angular frontend. Implementing CI/CD ensures faster iterations, consistent deployments, and improved software quality.

**Objective**

The goal of this CI/CD pipeline is to automate the following tasks:

* Clone the project repository.
* Build and package the backend Spring Boot application.
* Build and deploy Docker containers for both the backend and database.
* Run automated tests (unit tests and integration tests).
* Deploy the application for staging or production environments.

**Tools and Technologies**

* **Git** for version control.
* **Jenkins** (or any preferred CI/CD tool) for pipeline orchestration.
* **Maven** for building the Spring Boot backend.
* **Docker** for containerizing the application.
* **Docker Compose** for managing multi-container setups.
* **MySQL** as the relational database.

**CI/CD Pipeline Overview**

The CI/CD process is divided into stages:

1. **Clone**: Fetch the latest code from the Hamda branch.
2. **Build**: Use Maven to compile and package the backend application.
3. **Test**: Run unit and integration tests.
4. **Docker Build**: Build Docker images for the backend.
5. **Deploy**: Run Docker containers for both the application and database.

**Dockerfile Explanation**

The Dockerfile for the Spring Boot application is configured as follows:

FROM eclipse-temurin:17-jdk-alpine

VOLUME /tmp

ARG JAR\_FILE

EXPOSE 9071

COPY target/\*.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

* **Base Image**: Uses eclipse-temurin:17-jdk-alpine for lightweight JDK 17 support.
* **Volume**: VOLUME /tmp enables temporary file storage.
* **Port**: Exposes port 9071 for external access.
* **Entry Point**: Executes the Spring Boot application jar.

**Docker Compose Configuration**

version: "3"

services:

springboot-app:

image: veroDigit:v1

restart: always

build: .

ports:

- 9090:8080

environment:

MYSQL\_HOST: mysqldb

MYSQL\_USER: root

MYSQL\_PASSWORD: root

MYSQL\_PORT: 3306

SPRING\_PROFILES\_ACTIVE: prod

mysqldb:

container\_name: mysqldb

image: mysql

volumes:

- ./mysql:/var/lib/mysql

ports:

- 3306:3306

environment:

MYSQL\_DATABASE: veroDigit

MYSQL\_ROOT\_PASSWORD: root

**Jenkinsfile for CI/CD Pipeline**

pipeline {

agent any

tools {

dockerTool 'docker'

maven 'maven'

}

stages {

stage('Clone') {

steps {

git branch: 'Hamda', url: 'https://github.com/yedeshamda/veroDigit\_Project.git'

}

}

stage('Build Database') {

steps {

sh 'docker container run --name mysqldb --network veroDigitnetwork -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=veroDigit -v /mysql:/var/lib/mysql -d mysql:8'

sh 'sleep 30'

}

}

stage('Maven Install') {

steps {

sh 'mvn install -DskipTests'

}

}

stage('Docker Build') {

steps {

sh 'docker build -t veroDigit .'

}

}

stage('Docker Run') {

steps {

sh 'docker container run --network veroDigitnetwork --name veroDigitcontainer -p 8082:8080 -d veroDigit'

}

}

}

}

**Best Practices for CI/CD**

1. **Security**:
   * Use environment variables or secret management tools for sensitive data.
   * Avoid hardcoding passwords in Docker Compose or Jenkinsfiles.
2. **Testing**:
   * Add a dedicated testing stage to run unit and integration tests before building Docker images.
   * Example: sh 'mvn test' for unit tests.
3. **Automation**:
   * Automate deployment to different environments (staging, production) with branching strategies.
4. **Monitoring and Logging**:
   * Integrate monitoring tools like Prometheus and Grafana or logging systems like ELK for real-time insights.

**Conclusion**

Implementing this CI/CD pipeline provides a robust mechanism for automating the build, test, and deployment of the VeroDigit project. Following best practices and leveraging modern tools will ensure efficient development cycles and high-quality deployments.