**Project Writeup: Pet Clinic CI/CD Pipeline on AWS EC2**

**Project Overview**

Dr. Shawn operates a pet clinic and needs a software solution to manage the details of pet visits, appointments, and owner information. Bella Solutions has already developed a Spring Boot-based application for managing the clinic's operations. The goal of this project is to host the software on an AWS EC2 instance, allowing remote access. The software will be containerized using Docker, and a CI/CD pipeline will be created using Jenkins for continuous integration and deployment.

**Objectives**

* **Frontend Development**: Design and develop the frontend for the pet clinic software.
* **Database Design**: Design the database schema to store pet and owner details.
* **Dockerization**: Containerize the Spring Boot application using Docker.
* **CI/CD Pipeline**: Set up Jenkins on AWS EC2 to automate the build, test, and deployment processes.

**Steps to Achieve the Objectives**

**1. Import and Configure Spring Boot Project in Eclipse**

* Download the Spring Boot project with the generated code from Bella Solutions.
* Import the project into Eclipse IDE.
* Ensure all dependencies are included and the project runs correctly within Eclipse.

**2. Configure the Project with Dockerfile**

A Dockerfile is used to containerize the Spring Boot application. This file defines the environment, dependencies, and configuration needed to run the application inside a Docker container.

* **Create Dockerfile**: In the root directory of the Spring Boot project, create a Dockerfile with the following content:
* # Use the base image as JDK 11  
  FROM openjdk:17  
    
  # Create an Directory App  
  RUN mkdir /app  
    
  # Copy contents of target directory into app  
  COPY target/ /app  
    
  COPY src/main/resources/application.properties /app/application.properties  
    
  # Set the Working Directory as app  
  WORKDIR /app  
    
    
    
  # Execute the jar file which will run the project on port 9090  
  CMD java -jar pet-clinic-1.0.0.jar --spring.config.location=application.properties
* Build the Docker image using the following command:

docker build -t pet-clinic .

**3. Configure the Project with Jenkinsfile**

The Jenkinsfile defines the pipeline for Jenkins, automating the process of building, testing, and deploying the project.

* Create a Jenkinsfile in the root directory of the project with the following content:
* pipeline {  
   agent any  
    
   environment {  
   // Set the variables for the project and Docker image  
   *DOCKER\_IMAGE\_NAME* = 'pet-clinic-app' // Name for the Docker image  
   *DOCKER\_TAG* = 'latest' // Tag for the Docker image  
   *EC2\_USER* = 'ec2-user' // SSH username for EC2 instance (Amazon Linux)  
   *EC2\_IP* = '3.95.213.130' // EC2 instance IP address  
   *PRIVATE\_KEY\_PATH* = 'pet-clinic-keypair.pem' // Path to your private SSH key  
   *MYSQL\_PASSWORD* = 'root123@' // MySQL root password  
   *MYSQL\_PORT* = '3306' // MySQL port inside the container  
   *MYSQL\_HOST* = 'mysqldb' // Host name of the MySQL container  
   *PET\_CLINIC\_PORT* = '9091' // Port for pet-clinic application  
   }  
    
    
   stages {  
   stage('Clone Repository') {  
   steps {  
   // Clone the repository from GitHub  
   echo 'Cloning the Git repository'  
   git *branch*: 'main', *credentialsId*: 'git-credentials', *url*: 'https://github.com/Tameemahmedd/pet-clinic.git' // Replace with actual repo URL  
   }  
   }  
    
   stage('Build') {  
   steps {  
   sh 'chmod +x ./mvnw'  
    
   // Run Maven to clean and compile the project  
   echo 'Building the project with Maven'  
   sh './mvnw clean compile' // Using Maven Wrapper to compile  
   }  
   }  
    
   stage('Test') {  
   steps {  
   // Run Maven tests  
   echo 'Running tests with Maven'  
   sh './mvnw test' // Run unit tests with Maven  
   }  
   }  
    
   stage('Package') {  
   steps {  
   // Package the Spring Boot application into a JAR file  
   echo 'Packaging the project with Maven'  
   sh './mvnw package -DskipTests' // Create JAR without running tests  
   }  
   }  
    
  stage('Containerize') {  
   steps {  
   echo 'Building Docker images and running containers'  
    
   script {  
   // Build the MySQL container  
   sh '''  
   docker network create bootApp || true # Create a Docker network (ignore if it already exists)  
   docker pull mysql:9.0.1 # Pull the MySQL image  
   docker run -d --name mysqldb -p 3308:3306 --network=bootApp \  
   -e MYSQL\_ROOT\_PASSWORD=${MYSQL\_PASSWORD} \  
   -e MYSQL\_DATABASE=pet\_clinic \  
   mysql:9.0.1 # Run MySQL container  
   '''  
    
   // Build the Pet Clinic application container  
   sh '''  
   docker build -t ${DOCKER\_IMAGE\_NAME}:${DOCKER\_TAG} . # Build the Docker image for the pet-clinic app  
   docker run -d --name pet-clinic -p ${PET\_CLINIC\_PORT}:9091 --network=bootApp \  
   -e MYSQL\_HOST=mysqldb \  
   -e MYSQL\_PORT=${MYSQL\_PORT} \  
   ${DOCKER\_IMAGE\_NAME}:${DOCKER\_TAG} # Run Pet Clinic application container  
   '''  
   }  
   }  
  }  
    
   stage('Deploy') {  
   steps {  
   script {  
   // Optional: Deploy the containers on EC2 instance using SSH  
   echo 'Deploying the Docker containers on EC2'  
    
   // Copy the JAR file to EC2 instance (if necessary for your app)  
   sh """  
   scp -i ${PRIVATE\_KEY\_PATH} target/pet-clinic-1.0.0.jar ${EC2\_USER}@${EC2\_IP}:/home/${EC2\_USER}/pet-clinic/  
   """  
    
   // SSH into the EC2 instance to run the MySQL and Pet Clinic Docker containers (if not using local Docker)  
   sh """  
   ssh -i ${PRIVATE\_KEY\_PATH} ${EC2\_USER}@${EC2\_IP} 'docker network create bootApp || true' // Create network if not exists  
   ssh -i ${PRIVATE\_KEY\_PATH} ${EC2\_USER}@${EC2\_IP} 'docker pull mysql:9.0.1' // Pull the MySQL image  
   ssh -i ${PRIVATE\_KEY\_PATH} ${EC2\_USER}@${EC2\_IP} 'docker run -d --name mysqldb -p 3308:3306 --network=bootApp \  
   -e MYSQL\_ROOT\_PASSWORD=${MYSQL\_PASSWORD} -e MYSQL\_DATABASE=pet\_clinic mysql:9.0.1' // Run MySQL container on EC2  
    
   ssh -i ${PRIVATE\_KEY\_PATH} ${EC2\_USER}@${EC2\_IP} 'docker build -t ${DOCKER\_IMAGE\_NAME}:${DOCKER\_TAG} .' // Build the Pet Clinic image on EC2  
   ssh -i ${PRIVATE\_KEY\_PATH} ${EC2\_USER}@${EC2\_IP} 'docker run -d --name pet-clinic -p ${PET\_CLINIC\_PORT}:9091 --network=bootApp \  
   -e MYSQL\_HOST=mysqldb -e MYSQL\_PORT=${MYSQL\_PORT} ${DOCKER\_IMAGE\_NAME}:${DOCKER\_TAG}' // Run Pet Clinic container on EC2  
   """  
   }  
   }  
   }  
   }  
    
   post {  
   success {  
   echo 'Build and deployment successful!'  
   }  
   failure {  
   echo 'Build or deployment failed!'  
   }  
   }  
  }
* This pipeline automates:
  + Checking out the code from GitHub.
  + Building the project using Maven.
  + Dockerizing the Spring Boot application.
  + Deploying the application to an AWS EC2 instance.

**4. Create and Launch AWS EC2 Instance**

* **Launch EC2 Instance**: Go to the AWS EC2 console, create an instance using an appropriate Amazon Machine Image (AMI), and ensure it has enough resources for your application.
* **Configure Security Groups**: Open the necessary ports (e.g., 22 for SSH, 8080 for the Spring Boot application).

**5. Configure EC2 Instance with JDK 11**

* Connect to the EC2 instance using SSH.
* **Install JDK 11**:

sudo apt update

sudo apt install openjdk-11-jdk

* Verify the installation:

java -version

**6. Install Docker on EC2**

* Install Docker to containerize the application:

sudo apt-get install docker.io

sudo systemctl start docker

sudo systemctl enable docker

* Verify the installation:

docker --version

**7. Install Jenkins on EC2**

* **Install Jenkins**:

sudo apt update

sudo apt install openjdk-8-jdk

wget -q -O - https://pkg.jenkins.io/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable/ / > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins

* Start Jenkins:

sudo systemctl start jenkins

sudo systemctl enable jenkins

* Open Jenkins in your browser (http://<EC2-public-IP>:8080) and complete the initial setup.

**8. Create Admin User in Jenkins**

* In Jenkins, create an admin user to manage the pipeline and configurations.

**9. Upload the Code to GitHub**

* Create a GitHub repository.
* Push the local project (Spring Boot) to the GitHub repository.

**10. Create Jenkins Pipeline on EC2 with SCM as Git**

* In Jenkins, create a new pipeline job.
* In the pipeline configuration, set the source code management (SCM) to Git and provide the repository URL, and account credentials.
* Link the Jenkinsfile from the repository.

**11. Build the Pipeline to Dockerize the Application**

* **Create SSH Credentials**: In Jenkins, create SSH credentials for accessing the AWS EC2 instance.
* **Add SSH Agent in the Jenkinsfile**: Use the SSH agent to deploy the container to the EC2 instance as shown in the Jenkinsfile.

**12. Testing and Deployment**

* Once the pipeline is set up, test the process by running the pipeline. Ensure that:
  + The Spring Boot application is built using Maven.
  + The Docker container is created.
  + The container is successfully deployed to the AWS EC2 instance and the application is accessible on the specified port (8080).

**Conclusion**

This project successfully establishes a CI/CD pipeline for a pet clinic management system, hosting it on an AWS EC2 instance. By integrating Docker for containerization and Jenkins for automation, the process of building, testing, and deploying the application is streamlined, ensuring that future updates can be deployed efficiently with minimal manual intervention.

This setup provides Dr. Shawn with a scalable, reliable, and maintainable solution for managing the clinic’s operations remotely, with continuous updates delivered automatically through the Jenkins pipeline.