Continuous Integration Pipeline with Environment-based Testing in Jenkins

Iac - Jenkins - Assessment

Title: Continuous Integration Pipeline with Environment-based Testing in Jenkins

Duration

120 minutes

Type

Assessment

What you will be provided

- Pre-configured **VM** with:
  - Jenkins installed and running on port 8080
  - Java and Maven preinstalled
  - Required Jenkins plugins (Git, JUnit, Maven Integration, etc.)
- The application is available on the Desktop > Project folder.

What you need to know

- Basic Git & GitHub operations
- Maven build lifecycle (`mvn clean package`, `mvn test`)
- Jenkins Freestyle job creation
- Poll SCM scheduling syntax
- Jenkins artifact and report publishing

Scenario

You are part of a DevOps team maintaining a Java-based application that needs to be built and tested for multiple environments — **development (dev)** and **production (prod)**.

Your task is to enhance a Jenkins Continuous Integration (CI) pipeline that automates building, testing, and archiving artifacts for both environments.

You must configure the Jenkins job to accept an environment parameter and ensure the build runs successfully for both environments.

Additionally, the job should automatically trigger using **Poll SCM** every 5 minutes and store the generated .jar files as Jenkins artifacts.

What you will do

In this assessment, you will create a Jenkins Freestyle job that builds, tests, and archives Maven artifacts for multiple environments using parameters and Poll SCM automation.

Activities

Step 1: Push project to GitHub

- Install Maven in the terminal.
- Initialize Git and push the project to a new GitHub repository in your own account.

Step 2: Create Jenkins job

- In browser, browse http://localhost:8080 for Jenkins page.
- Login with details Username: jenkinsuser , password: Jenkinsuser@123
- Create a Jenkins **Freestyle** job named **EnvParameterizedJob**.
- Configure the job to pull source code from the **main** branch of your GitHub repository.

Step 3: Add build parameter

- Configure the job as parameterized with a **String Parameter:**
    - Name: ENVIRONMENT
    - Default Value: dev
    - Description: Specify environment (dev or prod)

Step 4: Build and test

- Configure the job to execute Maven build and test phases while passing the ENVIRONMENT parameter (-Denv=$ENVIRONMENT)

Step 5: Reporting and artifacts

- Configure Jenkins to:
    - Publish JUnit test reports from `target/surefire-reports/*.xml`
    - Archive the generated JAR artifact from `target/*.jar`

Step 6: Configure trigger (Poll SCM)

- Configure **Poll SCM** with the schedule H/5 * * * *

Step 7: Verify automation

- Manually trigger the job once using **Build with Parameters**, testing both environments (dev and prod).
- Push a new commit to your GitHub repository (e.g., update the README file).
- Wait for Jenkins to automatically trigger the build using **Poll SCM**.
- Confirm that the build runs successfully, executes Maven build & test steps, and archives the .jar artifact.

Expected Outcome

- A Jenkins job named **EnvParameterizedJob** exists and runs successfully.
- Maven build (`mvn clean package`) and test (`mvn test`) stages execute without errors.
- **JUnit test reports** are visible in Jenkins.
- Jenkins automatically triggers the build via **Poll SCM** within 5 minutes of a new Git commit.
- The generated `.jar` file is archived as a build artifact.
- The last triggered build is successful and triggered automatically (not manually).

Testcases

1. Check if Jenkins job 'EnvParameterizedJob' exists. (10 marks)
2. Check if job has ENVIRONMENT string parameter. (10 marks)
3. Check if job runs successfully with default ENVIRONMENT=dev. (10 marks)
4. Check if job runs successfully with ENVIRONMENT=prod. (10 marks)
5. Check if .jar artifact is archived in Jenkins (20 marks)
6. Check if JUnit test reports are published in Jenkins (10 Marks)
7. Check if Poll SCM trigger is configured for every 5 minutes (10 Marks)
8. Verify if Jenkins job was triggered automatically by Poll SCM (20 Marks)