**1. Automating CI/CD Pipeline with Groovy (Jenkins Pipeline Scripting)**

**Objective:**

Automate the Continuous Integration and Continuous Deployment pipeline for software projects using Jenkins and Groovy scripting.

**Steps to Implement:**

1. **Set up Jenkins:**
   * Install Jenkins on your server or use a Jenkins cloud service (like Jenkins on AWS or GCP).
   * Install necessary plugins (Git, Docker, Kubernetes, etc.) from Jenkins' plugin manager.
2. **Create Jenkins Pipeline Job:**
   * Go to Jenkins Dashboard → "New Item" → Select "Pipeline".
   * Name the pipeline and click OK.
3. **Write Jenkinsfile (Groovy Pipeline Script):**
   * Define the stages (Checkout, Build, Test, Deploy).
   * Use the sh step for shell commands to interact with the system.
4. **Configure Git:**
   * In the Checkout stage, use git to clone your repository.
5. **Build and Test:**
   * For Build, use Maven (mvn clean package).
   * For Test, run your tests with Maven (mvn test).
6. **Deploy:**
   * In the Deploy stage, use kubectl apply -f deployment.yaml to deploy the application to Kubernetes.
7. **Post-build Actions:**
   * Set up email notifications for successful or failed builds using the mail step.

**Groovy Jenkinsfile Implementation:**

groovy

Copy

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/example/repo.git'

}

}

stage('Build') {

steps {

sh 'mvn clean package'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

sh 'kubectl apply -f deployment.yaml'

}

}

}

post {

success {

mail to: 'team@example.com', subject: 'Build Success', body: 'Deployment successful!'

}

}

}

**Documentation:**

* **Overview:** Automates code checkout, build, test, and deployment using Jenkins pipeline.
* **Technologies Used:** Groovy, Jenkins, Git, Maven, Kubernetes.
* **Steps:** Detailed explanation of each pipeline stage with code snippets.
* **Impact:** Improved deployment speed and reduced manual errors.

**2. Automated REST API Testing with Groovy & Rest Assured**

**Objective:**

Automate REST API testing for an e-commerce platform using Groovy and Rest Assured.

**Steps to Implement:**

1. **Set up Groovy & Rest Assured:**
   * Make sure Groovy and Rest Assured dependencies are added to your project.
   * Example for Maven (or Gradle) dependency for Rest Assured:

xml

Copy

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>rest-assured</artifactId>

<version>4.3.3</version>

<scope>test</scope>

</dependency>

1. **Write Test Script:**
   * Use Groovy to write the test cases for various REST API operations (GET, POST, PUT, DELETE).
   * Use Rest Assured’s fluent API for validation.
   * Example: Test for a GET request on /users/1 endpoint.
2. **Integrate with JUnit:**
   * Add JUnit dependency to execute tests and assert API responses.
3. **Run the Test:**
   * Run the tests via command line or integrate with Jenkins for continuous testing.

**Groovy API Test Script:**

groovy

Copy

import io.restassured.RestAssured

import static io.restassured.RestAssured.\*

import static org.hamcrest.Matchers.\*

RestAssured.baseURI = "https://api.example.com"

given()

.header("Content-Type", "application/json")

.when()

.get("/users/1")

.then()

.statusCode(200)

.body("name", equalTo("John Doe"))

**Documentation:**

* **Overview:** Automates the validation of various REST API endpoints using Groovy and Rest Assured.
* **Technologies Used:** Groovy, Rest Assured, JUnit.
* **Steps:** Details on setting up dependencies, writing API tests, and running them.
* **Impact:** Reduced manual API testing time by 60%.