CI/CD Deployment for Springboot Application

Prototype of the Application

Name : Amit Yadav

GitHub : https://github.com/amityadav872699/JAVAFSD-Project05-master

This documentation will guide you through the steps involved in setting up a CI/CD pipeline for a Spring Boot application using Jenkins as the CI/CD tool and AWS EC2 instance as the hosting server. The application code will be fetched from a GitHub repository, and the deployment will be automated using Jenkins.

Sprint Planning

The Implementation is done in Eight sprints which are mentioned below

Prerequisites:

A GitHub account to store the source code.
A Jenkins server with admin access.
An AWS account to create an EC2 instance.
Basic knowledge of Spring Boot, AWS EC2, and Jenkins.

Sprint involved:

Sprint 1: Clone the GitHub repository:

Clone the repository containing the source code of the Spring Boot application to the local system. This will be used by Jenkins to build the application and create a deployment package.

Sprint 2: Install necessary plugins:

Install the following plugins in the Jenkins server:

Maven Integration Plugin
Git plugin
These plugins will be used to build the application and fetch the source code from the GitHub repository.

Sprint 3: Create a Jenkins job:

Create a new Jenkins job by navigating to Jenkins Dashboard > New Item > Freestyle

project.

In the Source Code Management section, select Git and provide the URL of the GitHub repository where the code is stored.

In the Build section, select Invoke top-level Maven targets and provide the necessary Maven goals to build the application. For example, clean install.

Sprint 4: Configure AWS EC2 instance:

Create an EC2 instance in the AWS account and configure the security group to allow HTTP traffic on port 8080.

Install Java and Tomcat on the EC2 instance to run the Spring Boot application.

Sprint 5: Install AWS CLI:

Install the AWS CLI on the Jenkins server to automate the deployment process.

Sprint 6: Configure AWS credentials:

Add the AWS access key and secret key to the Jenkins server using the AWS CLI.

Sprint 7: Create a deployment job:

Create a new Jenkins job to deploy the Spring Boot application to the EC2 instance. In this job, configure the following steps:

Fetch the latest deployment package from the Jenkins workspace.
Copy the deployment package to the EC2 instance using the AWS CLI.
Start the Tomcat server on the EC2 instance to run the application.

Sprint 8: Test the deployment:

Run the deployment job and check if the application is deployed successfully on the EC2 instance. Access the application using the public IP address of the EC2 instance.

Conclusion:

By following the above steps, you can automate the integration and deployment of a Spring Boot application using Jenkins and AWS EC2 instance. This will reduce the manual effort required for deployment and increase the efficiency of the development process.