JavaFullStackDev.in
Posted on May 26

💖 29

A Step-by-Step Tutorial on Setting Up Jenkins for Spring Boot Development

#webdev #programming #tutorial #devops

Introduction

Jenkins is an open-source automation server that enables developers to build, test, and deploy their software efficiently. It is widely used in Continuous Integration (CI) and Continuous Deployment (CD) pipelines, making it a crucial tool in modern software development. In this blog, we will explore Jenkins, its features, and how to integrate it with a Spring Boot application for a seamless CI/CD pipeline.

What is Jenkins?

Jenkins automates parts of the software development process, including building, testing, and deploying code. It helps teams integrate changes more frequently and ensures that software is always in a deployable state.

Key Features of Jenkins:

- **Easy Installation**: Jenkins can be installed via native system packages, Docker, or standalone Java application.

- **Extensible**: It has a vast library of plugins that integrate with various tools and platforms.
- **Distributed Builds**: Jenkins can distribute build tasks across multiple machines, improving performance and reliability.
- **Pipeline as Code**: Using Jenkinsfile, you can define your CI/CD pipelines as code, making them version-controllable and reproducible.

Setting Up Jenkins

Prerequisites:

- Java Development Kit (JDK) installed on your system.
- A Spring Boot application to work with.

Steps to Install Jenkins:

1. **Download and Install Jenkins**:

    - **On Windows**:
        - Download the Jenkins installer from the official website.
        - Run the installer and follow the setup wizard.
    - **On Linux**:

    ```
    wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key
    sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/
    sudo apt-get update
    sudo apt-get install jenkins
    ```

- **Using Docker**:

    ```
    docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts
    ```

1. **Start Jenkins**:

    - **Windows**: Jenkins starts automatically after installation.
    - **Linux**: Use `sudo systemctl start jenkins`.

2. **Access Jenkins**:

    - Open your browser and go to `http://localhost:8080`.
    - Follow the on-screen instructions to unlock Jenkins using the initial admin password found in `/var/lib/jenkins/secrets/initialAdminPassword`.

Integrating Jenkins with Spring Boot

1. Create a Spring Boot Project

If you don't have a Spring Boot project, you can generate one using [Spring Initializr].

2. Set Up Jenkins Job

1. **Create a New Job**:

    - Go to Jenkins dashboard.
    - Click on "New Item" and enter a name for your job.
    - Choose "Freestyle project" and click "OK".

2. **Configure Source Code Management**:

    - In the job configuration page, under "Source Code Management", select "Git".
    - Enter the repository URL and credentials if necessary.

3. **Configure Build Triggers**:

    - Under "Build Triggers", you can choose "Poll SCM" or "GitHub hook trigger for GITScm polling" to automate the build process.

4. **Add Build Steps**:

    - Under "Build", click "Add build step" and select "Invoke Gradle script" or "Invoke top-level Maven targets" depending on your build tool.
    - For Gradle, specify `build` as the Tasks.
    - For Maven, specify `clean install`.

5. **Save and Apply** the configuration.

3. Pipeline as Code with Jenkinsfile

Instead of a freestyle project, you can define your pipeline in a `Jenkinsfile` and store it in your repository.

Example Jenkinsfile for a Spring Boot Application:

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/your-repo/spring-boot-app.git'
            }
        }
        stage('Build') {
            steps {
                sh './gradlew clean build' // Use 'mvn clean install' if us
            }
        }
        stage('Test') {
            steps {
                sh './gradlew test' // Use 'mvn test' if using Maven
            }
        }
        stage('Package') {
            steps {
                sh './gradlew bootJar' // Use 'mvn package' if using Maven
            }
        }
        stage('Deploy') {
            steps {
                // Add your deployment steps here, e.g., using SCP, SSH, Do
                sh 'scp build/libs/*.jar user@server:/path/to/deploy'
            }
        }
    }

    post {
        success {
            echo 'Build and Deploy succeeded!'
        }
        failure {
            echo 'Build or Deploy failed!'
        }
    }
}
```

4. Run the Pipeline

- Commit and push your `Jenkinsfile` to your repository.
- Create a new pipeline job in Jenkins.
- In the job configuration, point to your repository where the `Jenkinsfile` is located.
- Jenkins will automatically detect and execute the pipeline defined in the `Jenkinsfile`

`Jenkinsfile`.

Conclusion

Integrating Jenkins with a Spring Boot application streamlines the development process by automating builds, tests, and deployments. This guide provides a solid foundation for setting up and using Jenkins to enhance your CI/CD pipeline. By following these steps, you can ensure that your software is always in a deployable state, improving both development speed and product quality.

Top comments (0)

Code of Conduct • Report abuse