- Setup a Git Repository:
- Install Jenkins:
- Create a Jenkins Job:
- Configure Jenkins
- Automate the Build:
- Pipelines

Prerequsites

- Ubuntu machine(publically available ip)
- Working Github account set up

1. Create a "Hello World" Java Application

- **Create a new directory:**
    - Open your terminal or command prompt.
    - Create a new directory for your project: `mkdir hello-world-java`
    - Navigate into the directory: `cd hello-world-java`
- **Create a Java file:**
    - Create a new file named `HelloWorld.java`: `vim HelloWorld.java`
- **Write the Java code:**
    - Open the `HelloWorld.java` file in a text editor and add the following code:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

- **Compile the code:**
    - Open your terminal and compile the code using the Java compiler (javac):
        - `javac HelloWorld.java`
- **Run the code:**
    - Run the compiled code: `java HelloWorld`
        - You should see the output "Hello, World!" printed to the console.

2. Initialize a Git Repository

- **Initialize Git:**
    - Open your terminal and initialize a Git repository in the current directory: `git init`
        - This creates a `.git` hidden directory in your project, which holds all the necessary files for Git to track changes.
- **Stage the initial commit:**
    - Add all the files in the current directory to the staging area: `git add .`

- **Commit the initial changes:**
    - Commit the changes with a descriptive message: `git commit -m "Initial commit"`

3. Push to GitHub (or a local repository)

- **Create a GitHub repository:**
    - If you want to push to GitHub:
        - Create a new repository on GitHub.
        - Copy the remote repository URL.
- **Add a remote:**
    - Add the remote repository URL to your local Git repository:
        - `git remote add origin <remote_repository_url>`
            - Replace `<remote_repository_url>` with the actual URL.
- **Push to the remote:**
    - Push your local repository to the remote: `git push -u origin main`
        - This pushes your initial commit to the `main` branch of the remote repository.

4. (Optional) Push to a local repository

- If you're working with a local Git repository:
    - You can skip the GitHub steps.
    - You can clone this repository to another location on your machine for backup or collaboration:
        - `git clone /path/to/your/local/repository`

---

5. Install Jenkins

- Updates package lists and upgrades packages

`sudo apt update -y && sudo apt upgrade -y`

- Installs OpenJDK 21

`sudo apt install openjdk-21-jdk -y`

- Shows the version of the installed Java

`java -version`

- Downloads key, adds repository source

`sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key && echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null`

- Updates package lists including newly added repository

`sudo apt update -y`

- Installs Jenkins

`sudo apt install jenkins -y`

- Starts and enables Jenkins service for automatic startup

`sudo systemctl start jenkins && sudo systemctl enable jenkins`

- Shows the status of Jenkins service (running, stopped, etc.)

`sudo systemctl status jenkins`

6. Start Jenkins

- Once the installation is complete, Jenkins will start automatically.

5. Access Jenkins

- Open a web browser and go to http://localhost:8080 (or the port you specified during installation).

6. Unlock Jenkins

- You will be prompted to unlock Jenkins. Find the initial administrator password in the specified file (usually under C:\Program Files\Jenkins\secrets\initialAdminPassword).
- Enter the password and click "Continue."

7. Install Suggested Plugins

- Choose to "Install suggested plugins." And Wait for the plugins to install.

8. Create an Admin User

- Create an admin user with a username and password.
- Click "Save and Finish."

9. Start Using Jenkins

You will be redirected to the Jenkins dashboard. You can now start creating jobs, configuring plugins, and managing your Jenkins server.

---

1. Create a New Job

- In the Jenkins dashboard, click "New Item".
- Enter a descriptive name for your job (e.g., "Hello World Job").
- Select "Freestyle project" and click "OK".

2. Configure Source Code Management

- In the "Source Code Management" section, select "Git".
- Enter the URL of your Git repository (e.g., `git@github.com:your-username/hello-world-java.git`).
- If your repository requires authentication, specify your credentials (username/password or SSH key).

3. Add Build Steps

- In the "Build Triggers" section, you can configure how Jenkins should trigger builds:

    - **Poll SCM:** Schedule periodic checks for changes in the Git repository (e.g., every 5 minutes: `H/5 * * * *`).
    - **GitHub hook trigger for GITScm polling:** Trigger builds automatically whenever changes are pushed to the Git repository (requires setting up a webhook in your GitHub repository).
- In the "Build" section, add build steps:
    - **Execute shell:** Add a shell script to compile your Java code:

      ```
      javac HelloWorld.java
      ```

    - **Execute shell:** Add a shell script to run your tests (if applicable):

      ```
      java -jar your-test-runner.jar
      ```

4. Save the Job

- Click "Save" to save the job configuration.

5. Test the Job

- Trigger a manual build by clicking "Build Now" on the job's page.
- Observe the build process in the console output.
- Check the build results to ensure the compilation and tests (if any) were successful.

Creating a pipeline

```
pipeline {
    agent any

    stages {
        stage('Clone') {
            steps {
                git branch: 'main', url: 'https://github.com/your-username/your-
repo.git'
            }
        }
        stage('Build') {
            steps {
                sh 'javac HelloWorld.java'
            }
        }
        stage('Deploy') {
            steps {
                sh 'java HelloWorld'
            }
        }
    }
}
```

```
pipeline {
    agent any

    stages {
        stage('Clone') {
            steps {
                git branch: 'main', url: 'https://github.com/your-username/your-
repo.git'
            }
        }
        stage('Prepare') {
            steps {
                sh 'Set up and config'
            }
        }
        stage('Build') {
            steps {
                sh 'Build/compile'
            }
        }
        stage('Test') {
            steps {
                sh 'RUN tests'
            }
        }
        stage('Deploy') {
            steps {
                sh './deploy'
            }
        }
    }

    post {
        success {
            mail to: 'your_email@example.com',
                   subject: 'Build Success: ${JOB_NAME}',
                   body: "Build of ${JOB_NAME} was successful!"
        }
        failure {
            mail to: 'your_email@example.com',
                   subject: 'Build Failure: ${JOB_NAME}',
                   body: "Build of ${JOB_NAME} has failed!"
        }
        always {
            archiveArtifacts artifacts: 'target/*.jar'
        }
    }
}
```