Installation Guide for setting up CICD Pipeline for a Java Project

**The compete setup will run in a Dockerized Environment**

=======================================================

**Tools Used:**

**Visual Studio Code**

**Docker for Windows**

**Git bash**

**Account Created: Github , Slack**

=======================================================

* **Full Git Setup and Push to GitHub**

1. **Create a GitHub Account**
   * Sign up at GitHub.
2. **Download the Project from GitHub**
   * <https://github.com/rajsbox/my-java-project.git>
   * Click Code > Download ZIP.
   * Extract and save it to a location on your computer.
3. **Install Git Bash**
   * Download from: Git Bash.
   * Install it and right-click inside your project folder → Open Git Bash here.
4. **Initialize Git in the Project Folder**
   * Open Git Bash and run:

git status <If it gives a fatal error, continue with:>

git init

git add .

git commit -m "Initial commit"

git status

1. **Create a New Repository on GitHub**
   * Go to GitHub.
   * Click New repository.
   * Name it: my-java-project (same as your folder).
   * Make it public, click Create repository.
2. **Connect Local Project to GitHub Remote**
   * Run the following commands in Git Bash:

git remote add origin https://github.com/rajsbox/CICD\_Pipeline.git

git branch -M master

Replace your-username with your actual GitHub username.

1. **Push Your Code to GitHub**

Run: git push -u origin master

1. **Create and Switch to the New Branch**

Run: git checkout -b feature-branch

1. **(Optional) Make Some Changes**
   * If you want to make any changes to the code, do so now, then stage and commit them:

git add .

git commit -m "Made changes in feature-branch"

1. **Push the New Branch to GitHub**

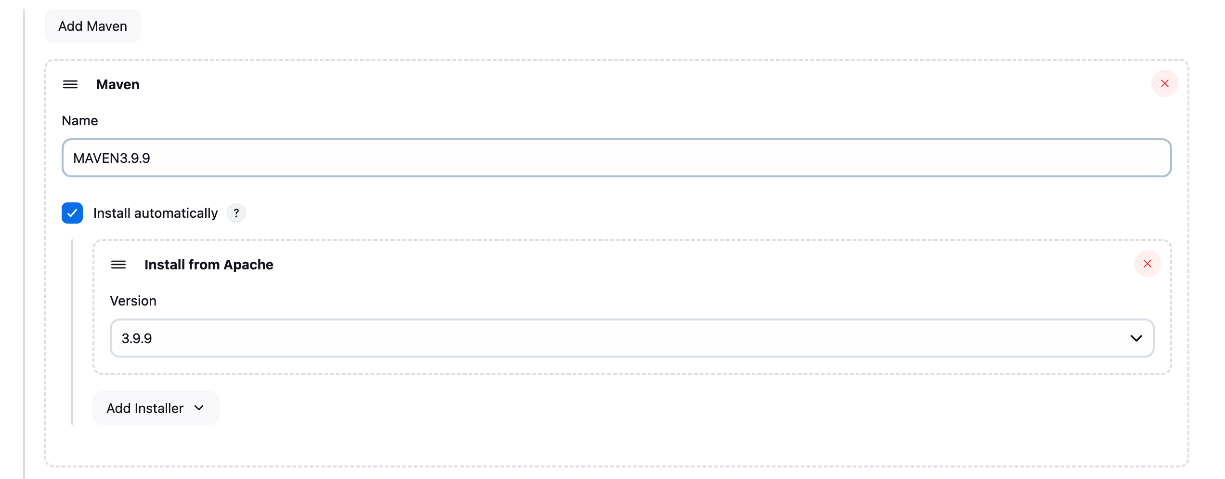
Run: bash git push -u origin feature-branch

* **Install Docker and relevant docker containers**

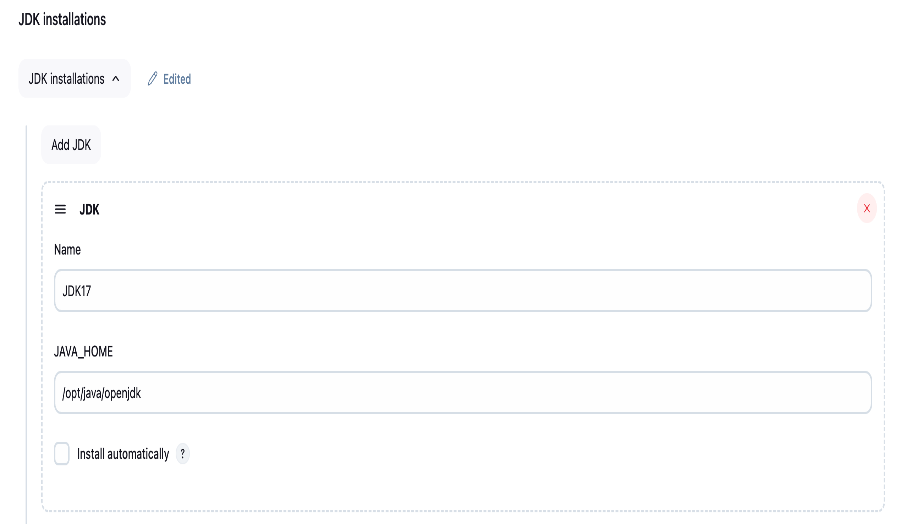
1. **Docker Desktop**
   * Ensure Docker Desktop is installed and running on your Windows machine.
   * Windows supports Linux containers (default on Docker Desktop).
2. **Create docker file with name docker-compose.yml to build three containers.**
3. Jenkins
4. SonarQube
5. Nexus Repo
6. Locate docker-compose.yml file. This is placed under my-java-project directory
7. Go to docker terminal and run following command in the same directory where docker-compose.yml file is located

docker-compose up -d

1. Wait for some time for all the containers to get up
2. Jenkins: localhost:8080
3. Setup initial password and install all default plugins
4. SonarQube: Localhost:9000 : default login admin/admin
5. Change the password and set a new password
6. Nexus repository: localhost:8081
7. Setup initial password
8. **Configure Jenkins Container**
9. Once login – Go to Manage Jenkins >tools
10. Add Maven



1. Add JDK



1. Plugins to install in Jenkins:

Nexus artifacts uploader

Sonarqube scanner

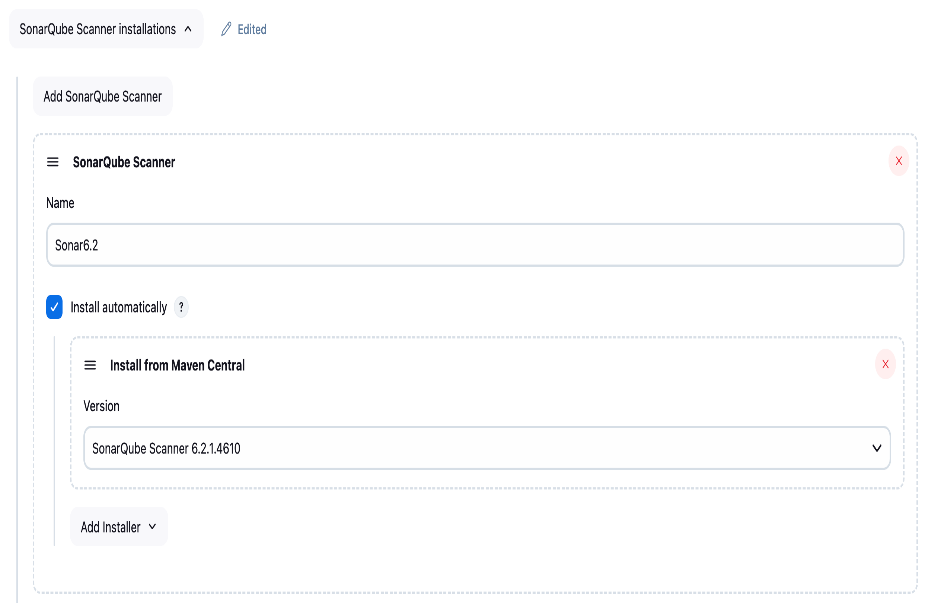
Build timestamp

Pipeline Maven integration

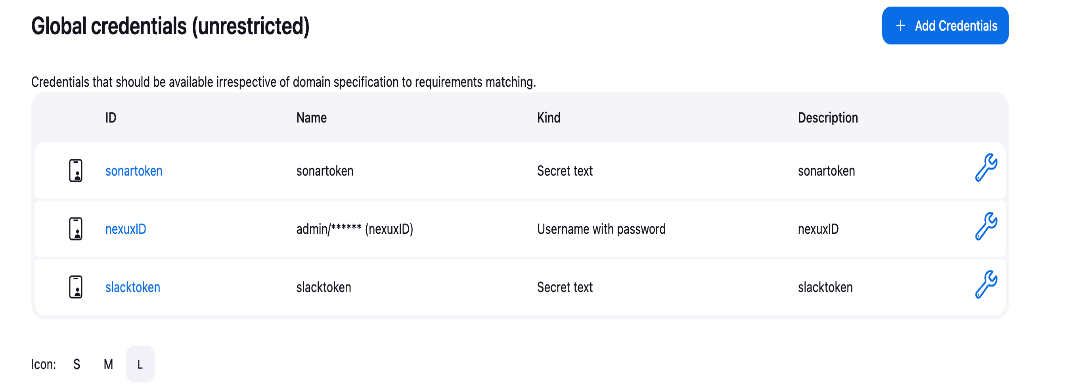
Pipeline utility steps

Slank notification plugin

1. Open vsCode and install Jenkinsfile support plugin
2. Add sonarqube tool to Jenkins : <https://www.udemy.com/course/decodingdevops/learn/lecture/32412762#overview>



1. Add nexus login credentials in Jenkins: <https://www.udemy.com/course/decodingdevops/learn/lecture/32412770#overview>



1. Configure slank -https://www.udemy.com/course/decodingdevops/learn/lecture/32412778#overview
2. All done here for CI Pipeline

* **Create a Jenkins Pipeline**

1. Now go to Jenkins > Create new item > Give a name > final pipeline
2. Select Pipeline > ok
3. Give a description name > final pipeline
4. Go down and under pipe line > select Pipeline Script>
5. Paste the code from Jenkinsfile > Located under my-java-project folder
6. And press save > start build now

* **Run the Java Web application in a docker container**
  + - 1. Copy the nexus repository and copy the latest WB-1.0.jar and paste this to /my-java-project/target directory
      2. Go to docker terminal and run following commands from /my-java-project path
      3. These commands will use Dockerfile llocated under same directory to build the webapplication

docker build -t my-java-webapp .

docker run -d -p 8083:8080 --name java-webapp my-java-webapp

* + - 1. The web application webpage can be open using localhost:8083 port.
* **Troubleshooting Steps**

1. **Steps to login inside containers** 
   1. Go to docker terminal and run following cmds

docker ps

docker exec -it jenkins bash

java -version

echo $JAVA\_HOME

exit

1. Docker not found under Jenkins container

Run following commands

docker exec -u root -it jenkins bash

apt-get update

apt-get install -y docker.io

exit

docker restart Jenkins

rerun the pipeline.