**JENKINS**

**Build Tool installation:**

**Make sure you have installed Maven and JDK8 in Jenkins. Follow below steps to install JDK8**

1. **SSH to Jenkins instance**
2. **Run commands**

* **Sudo su -**
* **Sudo apt update**
* **Sudo apt install openjdk-8-jdk -y**
* **ls /usr/lib/jvm**
* **copy the path like - /usr/lib/jvm/java-1.8.0-openjdk-amd64**

1. **Go to Jenkins dashboard**
2. **Click on “Manage Jenkins” – Click on “Global Tool Configuration”**
3. **Go to “JDK Installations” – Click Add – Give name as “OracleJDK8”**
4. **Paste the JDK8 installation path which we have copied to “JAVA\_HOME”**
5. **Now go down and select “Maven Installations”**
6. **Give name as “MAVEN3” – select the version as 3.\* and Click Save**

**Create Our First Job**

1. **Click on “Create Job”**
2. **Give Job name as “Build”**
3. **Select template “Freestyle project” and click ok**
4. **Give description as “sscademy maven build project”**
5. **Select “Git” and paste the GitHub link**
6. **Mention the Branch name as “ss-rem”**
7. **Go to “Build Steps” option – Add build step – Select “Invoke top-level Maven targets”**
8. **Select Maven version – “MAVEN3”**
9. **Type in Goals – “install”**
10. **Click on “Save”**
11. **Go to Dashboard**
12. **Go to Build Job and click on “Build Now”**
13. **You can go to “Console output” to check the logs**
14. **Go to Workspace to see the complete files Jenkins has used to Build the artifact**
15. **To save artifact we have to add Post Build action**
16. **Now Click on “Configure”**
17. **Click on “Add post-build action” – Select “Archive the artifact”**
18. **Type “ \*\*/\*.war” in Files to archive**
19. **Click Save**
20. **Run Build now again**

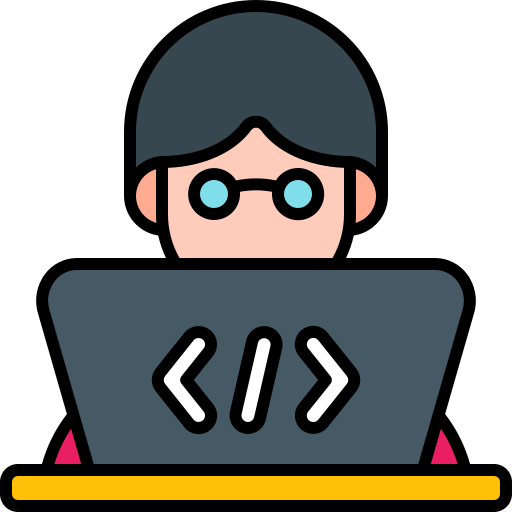
**What is MAVEN**

**Maven is a Build tool. (Refer PPT)**

**Versioning the Artifact**

**Since in each build, the artifact is getting replaced with the new artifact. Just consider if you want to keep each artifact ran on every build. To achieve this, we have to version the artifact.**

**Flow of Continuous Integration**



**Developer(git)**



**GitHub**

Fetch Code

Build

Unit Test

Code Analysis

Upload Artifact



**Steps for Continuous Integration**

1. Jenkins Setup
2. Nexus Setup
3. SonarQube Setup
4. Security Setup
5. Plugins
6. Integrate

* Nexus
* SonarQube

1. Write pipeline script
2. Set notifications

Jenkins Setup:

1. Create an EC2 instance

* Give name as you wish “Jenkins-server”
* Select AMI “Ubuntu 20.04 LTS”
* Select instance type : t2.small
* Create key pair
* Edit incoming securing group –
  + - Allow port 22 from anywhere
    - Allow port 8080 from anywhere
    - Allow port 80 from anywhere
* Copy the Jenkins setup shell script in the user data field
* Launch instance

Nexus setup

1. Create an EC2 instance for Nexus server

* Give name as “Nexus server”
* Select AMI from marketplace – CentOS 7
* Select instance type – t2.medium
* Create key pair – nexus key
* Create security group
  + - Name – nexusSG
    - Allow port 22 from anywhere
    - Allow port 8081 (nexus runs on port 8081)
* Copy paste Nexus shell script into user data field

SonarQube setup

1. Launch instance for Sonar Server

* Give name as “SonarServer”
* Use AMI as “Ubuntu 18.04 LTS”
* Instance type: t2.medium
* Create key pair – sonar key
* Create security group
  + - Name – sonarSG
    - Allow port 22 from anywhere
    - Allow port 80 from anywhere
    - Allow port 9000 from anywhere
* Copy pastes the Sonar shell script into user data field
* Launch instance

Check the services up and running

1. Login to Jenkins server and check Jenkins service is running using below command

Systemctl status Jenkins

Access from browser: Public\_ip:8080

1. Login to Sonar server and check the sonar service is running

Systemctl status sonaqube

Access from browser: Public\_ip

1. Login to Nexus server and check the nexus service is running

Systemctl status nexus

Access from browser: Public\_ip:8081

**Plugins Installation**

* Nexus
* Sonarqube
* Git
* Pipeline Maven Integration
* BuildTimeStamp

Go to Manage Jenkins 🡪 Manage Plugins 🡪 Click on Available 🡪 Search for “Nexus” and

Select “Nexus artifact uploader” 🡪 Search for “Sonarqube scanner” 🡪 Search for “Buildtimestamp” 🡪 Search for “Pipeline Maven Integration” 🡪 last one “Pipeline utility steps”

Click on “Install without restart”

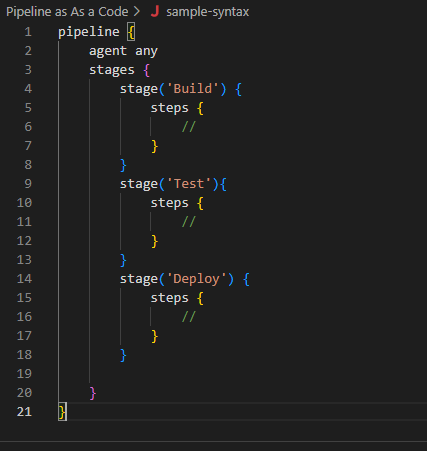
**Pipeline As a Code**

* Automate pipeline setup with Jenkinsfile
* Jenkinsfile defines stages in CI/CD Pipeline
* Jenkinsfile is a text file with Pipeline DSL Syntax
* Similar to Groovy
* Two Syntax
  + Scripted
  + Declarative

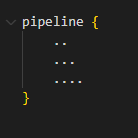
**Pipeline Concept**

* Pipeline
* Node/Agent
* Stage
* Step

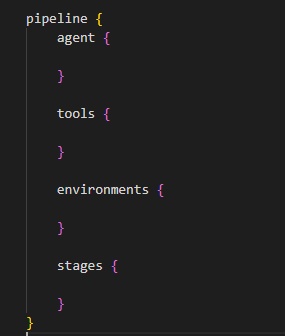
**Sample pipeline syntax**



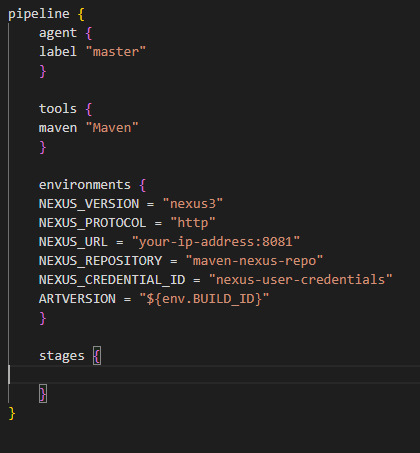
**Pipeline block**



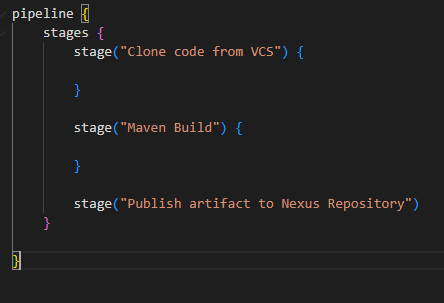
**Pipeline components**

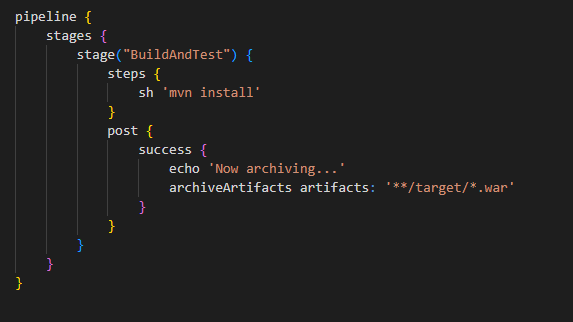


**Pipeline components with details**



**Stages**





<https://www.jenkins.io/doc/book/pipeline/>

**Code Analysis**

* Detects vulnerabilities and functional errors
* Best practices
* Vulnerabilities in the code (top 10 OWASP vulnerabilities)
* Functional Errors before deployment

**Tools in the market for Code analysis**

* Checkstyle
* Cobertura
* Mstest
* Owasp
* SonarQube Scanner
* Etc.…

**SonarQube Tools installation in Jenkins**

* Go to “Manage Jenkins” 🡪 “ go to “Global Tool Configuration”
* Go to ‘Sonarqube scanner” option 🡪 Click “Add Sonarqube scanner”
* Give name “sonar4.7” 🡪 save it

**Integrate SonarQube with Jekins**

* Go to “Manage Jenkins” 🡪 Click on “Configure system” 🡪 Go to “SonarQube Servers” 🡪 Select “Add SonarQube server” 🡪 check mark “Environmet variables” 🡪 give a name “sonar” 🡪 type the sonarqube server IP address 🡪

<http://172.88.90.22> & save it

**Now create a token for authentication**

* Go to SonarQube server URL
* Login there
* Click on Profile logo
* Go to “My Account”
* Go to security
* Give a name for token
* Click on generate
* Copy the generated token

Now got “Configure system” in Jenkins and go to SonarQube server & add the credentials by selecting “secret text” & give the token ID as “sonartoken”

<https://www.jenkins.io/doc/pipeline/steps/sonar/>

**Quality Gates in SonarQube**

* Lets create gate for our SonarQube project
* Login to SonarQube server
* Go to Quality Gates option on the top
* Create a Quality Gate – give any name e.g: “sscademy-QG”
* Click “Add condition” 🡪 select “On overall Code”
* Select metrics 🡪 “Bugs” 🡪 Give “60”
* Click Add condition
* Come to “Project” option on SonarQube
* Click on our project
* Go to “Project settings”
* Click on “Quality gate”
* Select the newly created Quality Gate
* Click again on “Project settings”
* Click on “Webhooks”
* Click “Create”
* Give name as “Jenkins-ci-webhooks”
* Type URL as 🡪 <http://jenkins_public_ip:8080/sonarqube-webhook>
* Click Create