**Steps to Create:**

Stage 1 : Clone Repository

Stage 2 : Maven Build

Stage 3: Code Review

Stage 4: Upload Artifact

Stage 5: Deploy Applications

**Four servers in AWS:**

Tomcat Server t2.medium

Sonar Server t2.medium

Nexus Repo Server t2.medium

Jenkins Server t2.medium

Prerequisites:

Maven, Github, Tomcat, SonarQube, Nexus Repo, Jenkins

**Jenkins Server Setup:**

T2-medium, java

apt install openjdk-11-jre-headless -y

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

sudo apt-get update

sudo apt-get install jenkins

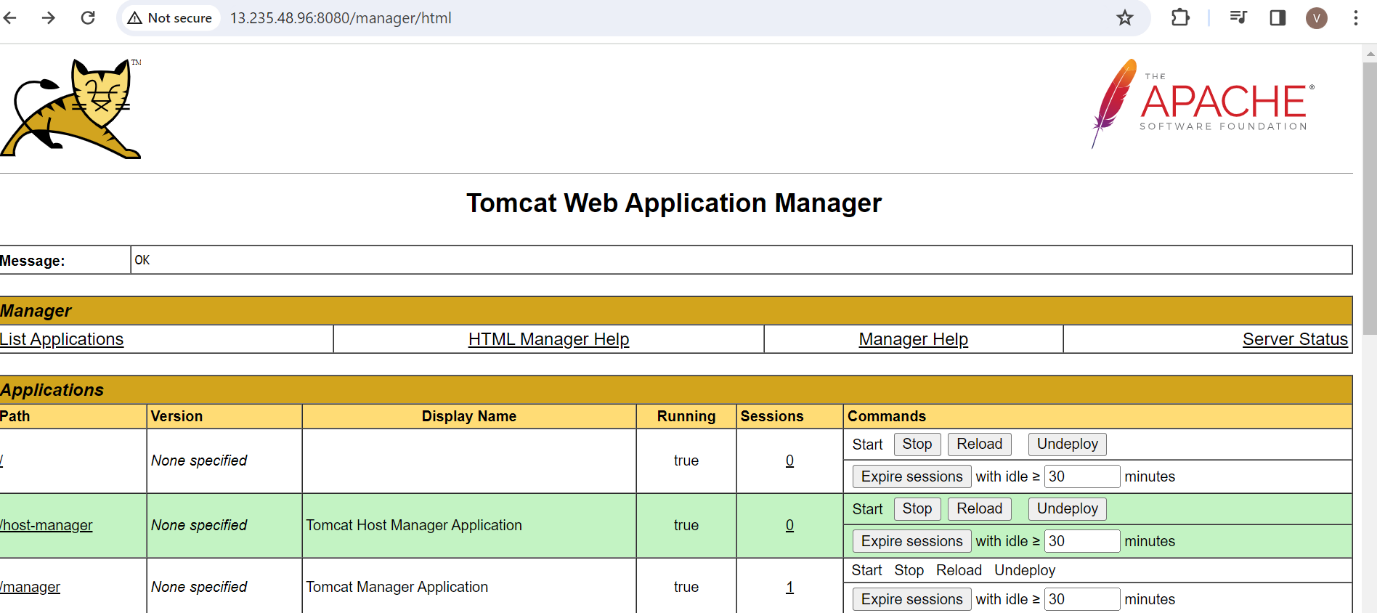
**Tomcat Server Setup:**

Prerequisites: java, ec2 t2-medium

Create EC2 machine

Run below commands:

* Apt update
* Install jdk -- apt install openjdk-11-jre-headless
* Install tomcat in the server: apt install tomcat9 tomcat9-admin -y
* Create user to access Tomcat:
* Vi /etc/tomcat9/tomcat-users.xml
* <user username="admin" password="admin" roles="manager-gui, admin-gui, manager-script"/>
* Open port 8080 in aws
* Browse to the tomcat server using ip and port



**Sonar Server Setup:**

Prerequisites:

Java , EC2 instance (t2-medium)

root@ip-172-31-5-191:~# apt update

apt install openjdk-11-jre-headless

root@ip-172-31-5-191:~# java –version

cd opt

wget https://binaries.sonarsource.com/Distribution/sonarqube/[sonarqube-7.9.zip](https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.zip)

apt install unzip

unzip sonarqube-7.8.zip

To check the port where sonar runs:

root@ip-172-31-5-191:/opt/sonarqube-7.8/conf# cat /opt/sonarqube-7.8/conf/sonar.properties | grep 9000

To check the database which sonar is using:

sonar@ip-172-31-5-191:/opt/sonarqube-7.8/conf$ cat sonar.properties | grep database

# TCP port for incoming HTTP connections. Default value is 9000.

#sonar.web.port=9000

Open the port for the instance in aws.

To start sonar:

Always start sonar with a user sonar else with root it will start and stop. So create username and password for user sonar

adduser sonar (give password)

username and password: sonar/sonar123

Give the sonar user the full sudo access:

Vi /etc/sudoers

sonar ALL=(ALL) NOPASSWD: ALL (not mandatory)

chown -R sonar:sonar sonarqube-7.8

root@ip-172-31-5-191:/opt# ls -l

total 200924

drwxr-xr-x 11 sonar sonar 4096 Jun 17 2019 sonarqube-7.8

su – sonar

[sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$](mailto:sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$) /opt/sonarqube-7.9/bin/linux-x86-64/sonar.sh start

Starting SonarQube...

Started SonarQube.

[sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$](mailto:sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$) /opt/sonarqube-7.9/bin/linux-x86-64/sonar.sh status

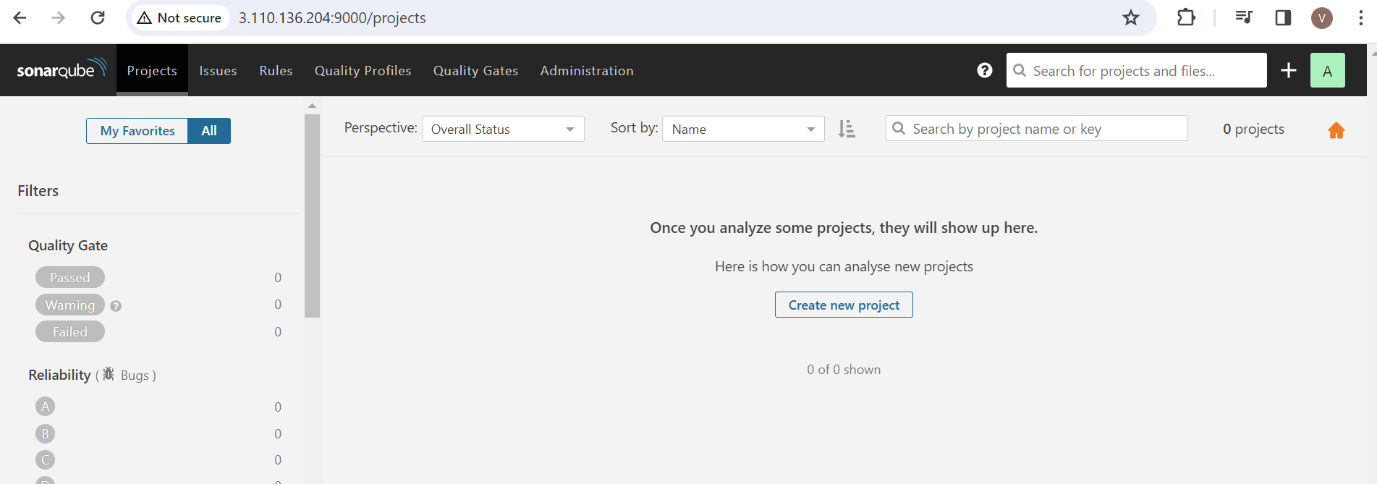
SonarQube is running (1147).

apt install net-tools

[sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$](mailto:sonar@ip-172-31-5-191:/opt/sonarqube-7.8/bin/linux-x86-64$) netstat -tulnp

tcp6 0 0 :::9000 :::\* LISTEN 1319/java

Username and password for the sonarqube UI is **admin**



**Nexus Server Setup:**

Prerequisite:

T2-medium, java (java 1.8 version required for nexus to run)

apt install openjdk-8-jre-headless -y

cd /opt/

wget <https://download.sonatype.com/nexus/3/nexus-3.65.0-02-unix.tar.gz>

tar -xvzf nexus-3.65.0-02-unix.tar.gz

Rename the folder to nexus:

mv nexus-3.65.0-02 nexus

Create user nexus:

adduser nexus

passwd: nexus

Give the nexus user the sudo privileges:

Vi /etc/sudoers

nexus ALL=(ALL) NOPASSWD: ALL

Change the ownership of directories to nexus user:

root@ip-172-31-40-192:/opt# ls -l

total 226440

drwxr-xr-x 10 root root 4096 Feb 9 07:24 nexus

-rw-r--r-- 1 root root 231860503 Feb 6 08:22 nexus-3.65.0-02-unix.tar.gz

drwxr-xr-x 3 root root 4096 Feb 9 07:24 sonatype-work

chown -R nexus:nexus nexus

chown -R nexus:nexus sonatype-work

root@ip-172-31-40-192:/opt# ls -l

total 226440

drwxr-xr-x 10 nexus nexus 4096 Feb 9 07:24 nexus

-rw-r--r-- 1 root root 231860503 Feb 6 08:22 nexus-3.65.0-02-unix.tar.gz

drwxr-xr-x 3 nexus nexus 4096 Feb 9 07:24 sonatype-work

root@ip-172-31-40-192:/opt/nexus/bin# cat /opt/nexus/bin/nexus.rc

run\_as\_user="nexus"

To check and configure the port details:

cat /opt/nexus/etc/nexus-default.properties

**systemd**

This example is a script that uses systemd to run the repository manager service. Create a file called nexus.service. Add the following contents, then save the file in the /etc/systemd/system/ directory:

**[Unit]**

**Description=nexus service**

**After=network.target**

**[Service]**

**Type=forking**

**LimitNOFILE=65536**

**ExecStart=/opt/nexus/bin/nexus start**

**ExecStop=/opt/nexus/bin/nexus stop**

**User=nexus**

**Restart=on-abort**

**TimeoutSec=600**

**[Install]**

**WantedBy=multi-user.target**

vi /etc/systemd/system/nexus.service (copy the above module to this file)

sudo systemctl enable nexus.service

sudo systemctl start nexus.service

sudo systemctl status nexus.service

Open port 8081 in aws:

<http://3.109.48.198:8081/>

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

UI username and password for nexus:

Admin

Password: admin123

**Jenkins Pipeline :**

**Stage 1: Git Clone**

Setup Github Credentials in jenkins

A screenshot of a computer

Description automatically generated

**Stage 2: Maven Build**

Maven configuration in global tool so that maven will be installed automatically. No need to install in machine. Go to Global Tools and under Maven Installations:

A screenshot of a computer

Description automatically generated

pipeline {

agent any

environment {

mavenHome = tool name: "Maven-3.9.6",type: "maven"

mavenCMD = "${mavenHome}/bin/mvn"

}

stages {

stage('Fetch Code') {

steps {

git credentialsId: 'Github\_credential', url: 'https://github.com/vijeshnair89/MavenBuild.git'

}

}

stage('Maven Build') {

steps {

sh '${mavenCMD} clean package'

}

}

}

}

**Stage 3: Code Review**

Config sonar with Jenkins.

Install sonarqube scanner plugin in Jenkins:

A screenshot of a computer

Description automatically generated

Generate sonar token and add it as secret text in Jenkins credentials:

To generate token, go to sonar GUI:

Go to Myaccount -> security -> Generate tokens (give any name)

A screenshot of a computer

Description automatically generated

Token : c8600b4b73f7ecb4b23dfab2c807928184c10ea7

Add it as secret text:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Configure sonarqube server in Jenkins under system:

A screen shot of a computer

Description automatically generated

A close-up of a paper

Description automatically generated

A screenshot of a computer

Description automatically generated

The sonar server ip is dynamic In real world we go for elastic ip which is static.

Use pipeline syntax for code review pipeline script:

A screenshot of a computer

Description automatically generated

stage('Code review') {

steps {

withSonarQubeEnv('sonar-server') {

sh '${mavenCMD} sonar:sonar'

}

}

}

A screenshot of a computer

Description automatically generated

A yellow text on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

Stage 4: Upload artifact in nexus server: (nexus or jfrog)

Install nexus artifact uploader plugin:

A screenshot of a computer

Description automatically generated

Add nexus creds to Jenkins:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a line

Description automatically generated

Create a repository in nexus server for storing artifact:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Use pipeline syntax to create pipeline script for nexus:

A screenshot of a computer

Description automatically generated

A white lined paper with black lines

Description automatically generated

Version will be there in pom.xml

Repository will be the one we created above in the nexus server.

A screenshot of a computer

Description automatically generated

ArtifactId will be in pom.xml

Generate syntax:

A close-up of a message

Description automatically generated

stage('Upload Artifact') {

steps {

nexusArtifactUploader artifacts: [[artifactId: 'java-example', classifier: '', file: 'target/java-example.war', type: 'war']], credentialsId: 'nexus-creds', groupId: 'in.vijesh', nexusUrl: '54.165.72.80:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'vijesh-repository', version: '1.0-SNAPSHOT'

}

}

A screenshot of a computer

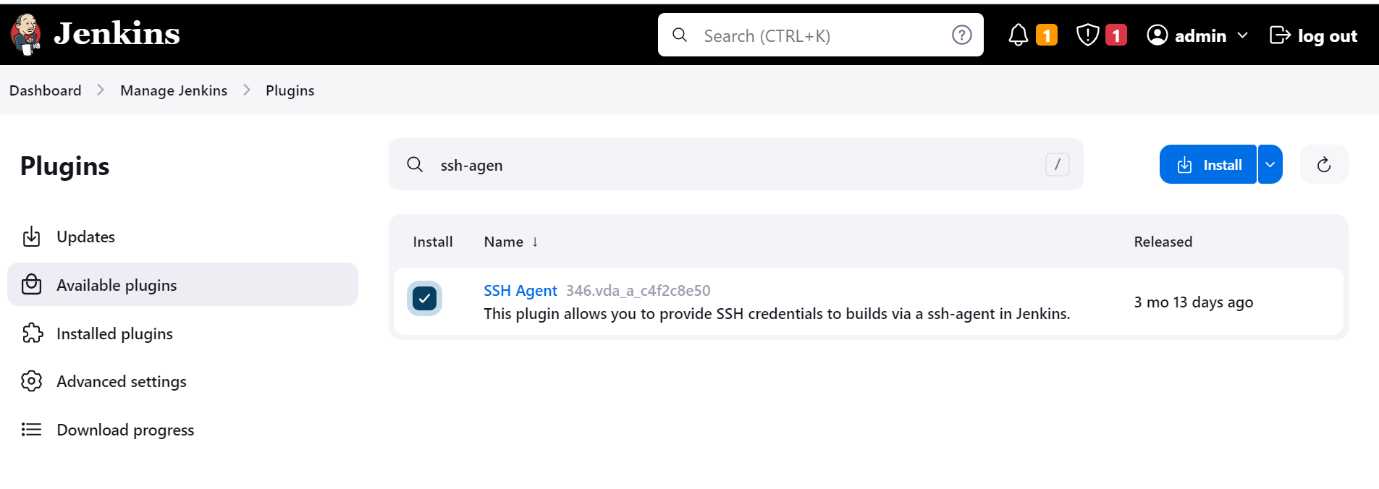
Description automatically generated

A screenshot of a computer

Description automatically generated

Stage 5: Deploy the application.

Install ssh-agent plugin in Jenkins to copy the code from Jenkins to tomcat server on a different machine.



Use pipeline syntax to create the pipeline for ssh agent deploy:

A screenshot of a computer

Description automatically generated

Add the tomcat server credentials using ssh with keys:

A screenshot of a computer

Description automatically generated

A white paper with black lines

Description automatically generated

A screenshot of a computer

Description automatically generated

Key is .pem file contents.

stage('Deploy Code') {

steps {

sshagent(['Tomcat-Server']) {

sh 'scp -o StrictHostKeyChecking=no target/java-example.war ubuntu@3.80.64.59:/home/ubuntu'

}

}

}

Scp -o StrictHostKeyChecking=no target/java-example.war ubuntu@3.80.64.59: /var/lib/tomcat9/webapps

Configure a node to be in sync with Jenkins to login to node and copy the final war file to the root tomcat directory.

A screenshot of a computer

Description automatically generated

**Final Pipeline (Working)**

pipeline {

agent any

environment {

mavenHome = tool name: "Maven-3.9.6",type: "maven"

mavenCMD = "${mavenHome}/bin/mvn"

}

stages {

stage('Fetch Code') {

steps {

git credentialsId: 'Github\_credential', url: 'https://github.com/vijeshnair89/MavenBuild.git'

}

}

stage('Maven Build') {

steps {

sh '${mavenCMD} clean package'

}

}

stage('Code review') {

steps {

withSonarQubeEnv('sonar-server') {

sh '${mavenCMD} sonar:sonar'

}

}

}

stage('Upload Artifact') {

steps {

nexusArtifactUploader artifacts: [[artifactId: 'java-example', classifier: '', file: 'target/java-example.war', type: 'war']], credentialsId: 'nexus-creds', groupId: 'in.vijesh', nexusUrl: '184.73.146.198:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'vijesh-repo', version: '1.0-SNAPSHOT'

}

}

stage('Move Code to Tomcat') {

steps {

sshagent(['Tomcat-Server']) {

sh 'scp -o StrictHostKeyChecking=no target/java-example.war ubuntu@3.80.64.59:/home/ubuntu'

}

}

}

stage('Deploy code') {

agent { label 'node1' }

steps {

sh 'sudo mv /home/ubuntu/java-example.war /var/lib/tomcat9/webapps'

}

}

}

}

A screenshot of a computer

Description automatically generated

**Browse the tomcat server for the application:**

A screenshot of a computer

Description automatically generated