

PROJECT-3

Course: DevOps

Name: Vanguri Raja Vamsy

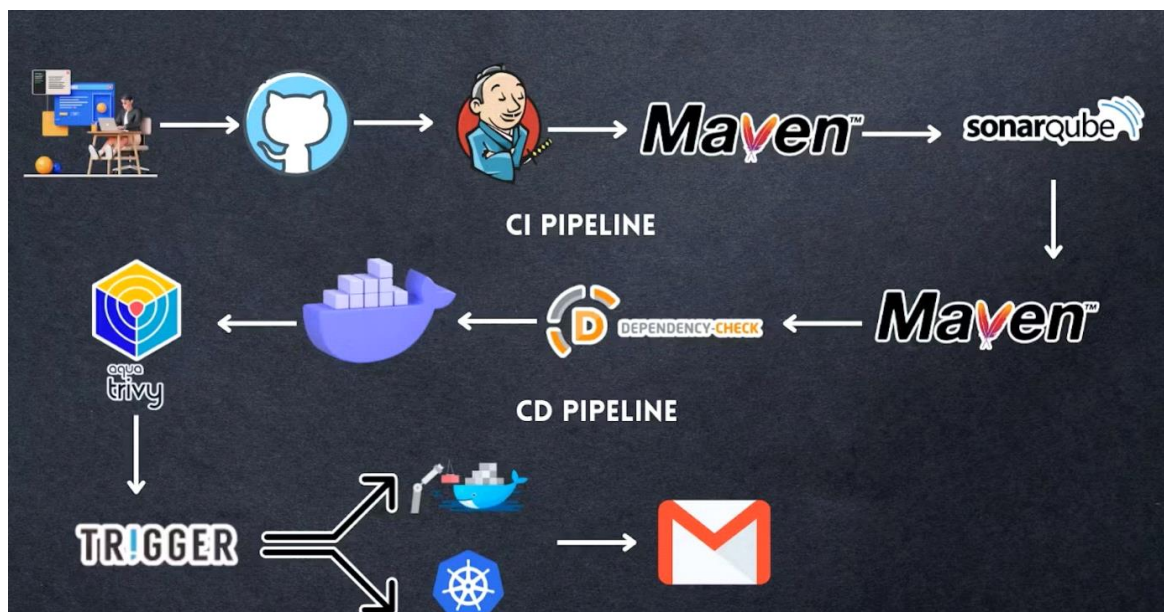
Batch No: 115 (9am-10am)

Mail id: rajavamsyvanguri@gmail.com

Trainer Name: Mr. Madhukar Reddy

Date: 22-12-23

Title: DEVSECOPS PROJECT: COMPLETE CI-CD (3 TIER APP)- DOCKERJENKINS.



ABSTRACT:

This project proposes a DevSecOps framework, seamlessly blending security into the software development and delivery pipeline. Focused on automating security testing, continuous monitoring, and threat detection, the initiative aims to proactively enhance security while accelerating time-to-market. Embracing Infrastructure as Code (IaC) security and fostering a collaborative culture, the project emphasizes shared responsibility among development, security, and operations teams. By automating incident response and promoting a unified approach, organizations can achieve a balanced synergy between speed and security, ensuring resilient software in the face of evolving cyber threats. This project provides a practical guide for implementing DevSecOps principles in the development lifecycle.

Steps: -

Step 1 — AWS

Step 2 — Server Connections and Installations.

Step 3 —Jenkins

Step 4 — Sonarqube

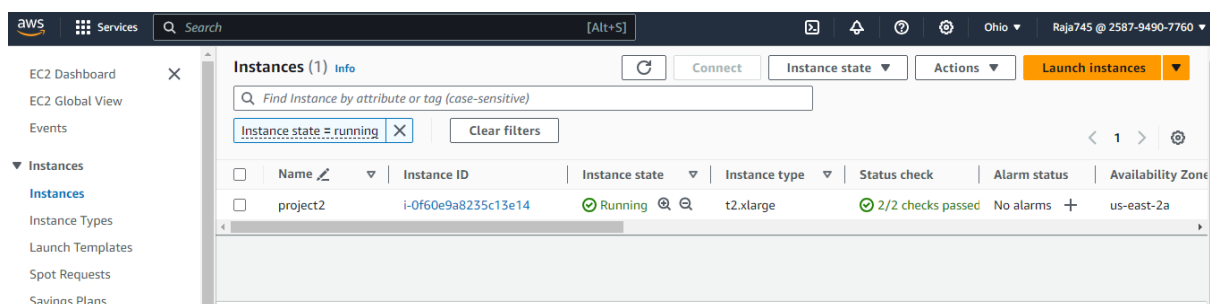
Step 5 — Install Nexus artifact.

Step 6 — Install tomcat and deploy the project in it.

Step 7 —DP-Check

Step 8 —Docker

STEP-1-AWS:



By using the AWS EC2, instance is created.

- Creating a t2.xlarge instance with the security group which is added the all traffic for inbound rules. I will allow all ports.
- Or we can specify the ports that which we need to allow.

STEP-2-SERVER CONECTIONS AND INSTALLATIONS:

- For the installation of Jenkins, we can install Jenkins through the terminal by implementing the following steps.
- We implement these commands by connecting through the console.

Installation of Jenkins:

- Execute the below command
vi jenkins.sh
- Write the code in it.

```

#!/bin/bash

sudo apt update -y #sudo apt upgrade -y

wget -O -https://packages.adoptium.net/artifactory/api/gpg/key/public | tee
/etc/apt/keyrings/adoptium.asc

echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc]
https://packages.adoptium.net/artifactory/deb ${awk -F=
'/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee
/etc/apt/sources.list.d/adoptium.list sudo apt update -y

sudo apt install temurin-17-jdk -y sudo apt install maven -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null 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 -y

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status Jenkins

```

- After writing code implement these commands to start the Jenkins.

```

sudo chmod 777 jenkins.sh

./jenkins.sh # this will install Jenkins

```
- Now copy the the Ip address and go through the Jenkins portal the interface will like shown below.

Installation of Docker:

- For the installation of the docker the following commands will be followed.

```

sudo apt-get update.

sudo apt-get install docker.io -y

```
- Now after creation of the docker we need to create a sonar container.
- The default port number of the SonarQube is 9000.

```

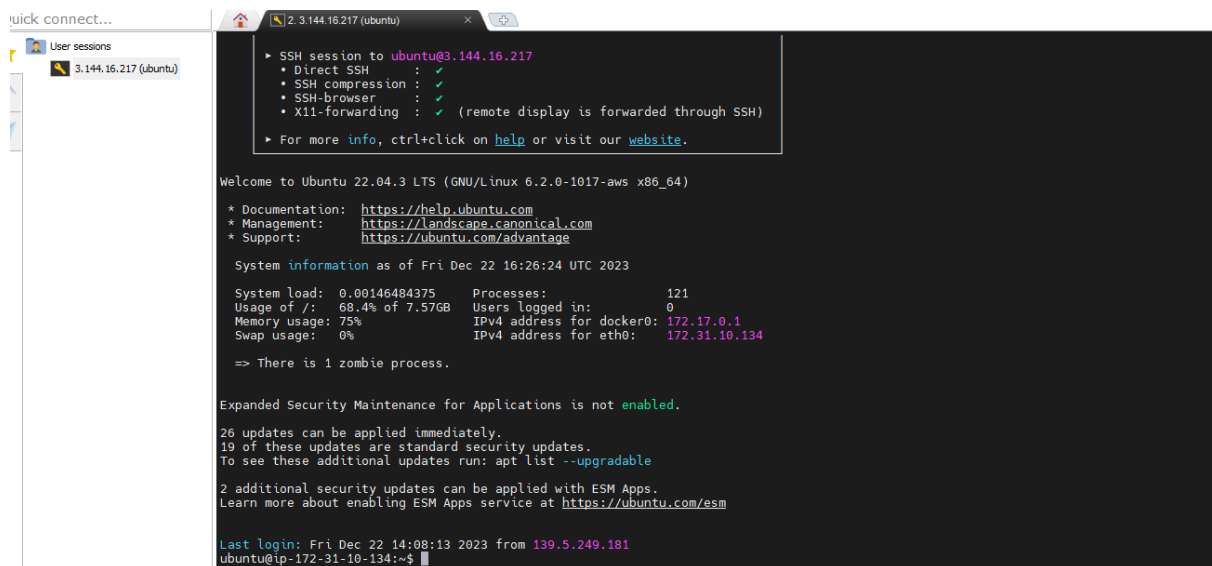
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community.

```
- To check whether the container is created or not the command is.

```

docker ps or docker ps -a

```



```
quick connect...
User sessions
3.144.16.217 (ubuntu)

SSH session to ubuntu@3.144.16.217
• Direct SSH : ✓
• SSH compression : ✓
• SSH-browser : ✓
• X11-forwarding : ✓ (remote display is forwarded through SSH)
• For more info, ctrl+click on help or visit our website.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1017-aws x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Fri Dec 22 16:26:24 UTC 2023

System load: 0.00146484375 Processes: 121
Usage of /: 68.4% of 7.57GB Users logged in: 0
Memory usage: 75% IPv4 address for docker0: 172.17.0.1
Swap usage: 0% IPv4 address for eth0: 172.31.10.134

=> There is 1 zombie process.

Expanded Security Maintenance for Applications is not enabled.

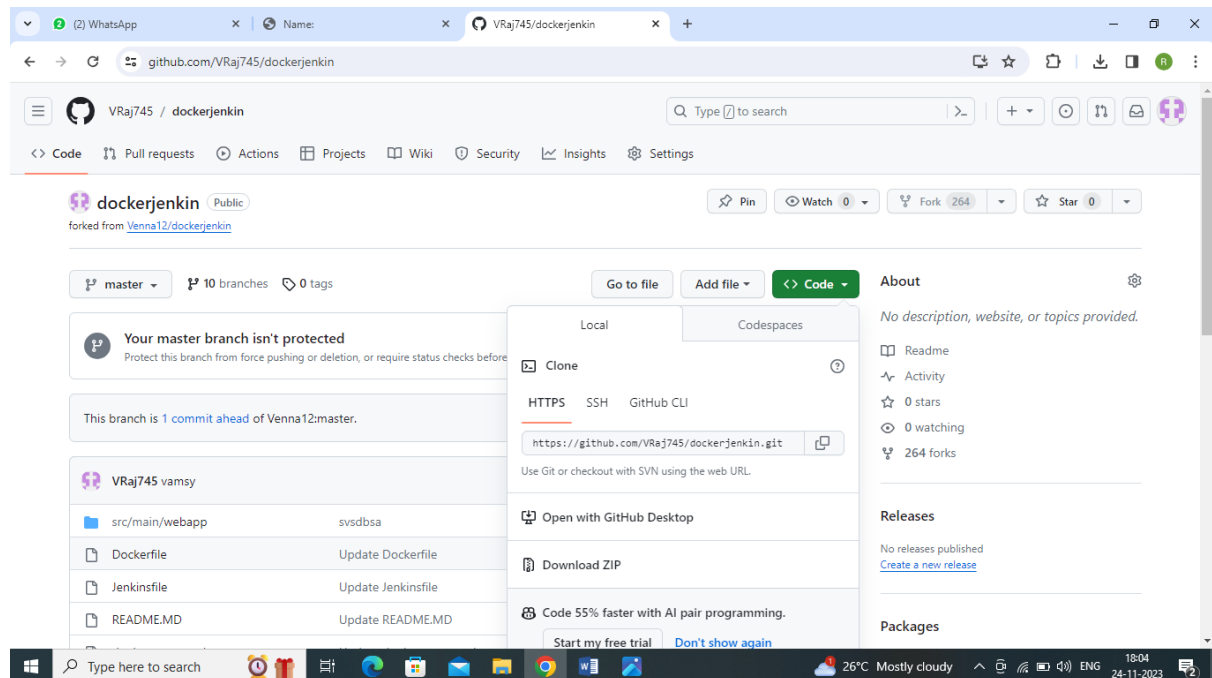
26 updates can be applied immediately.
19 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Fri Dec 22 14:08:13 2023 from 139.5.249.181
ubuntu@ip-172-31-10-134:~$
```

Above fig shows that the connection of instance in mobaxterm, which is used to install the jenkins, tomcat and required packages and more.

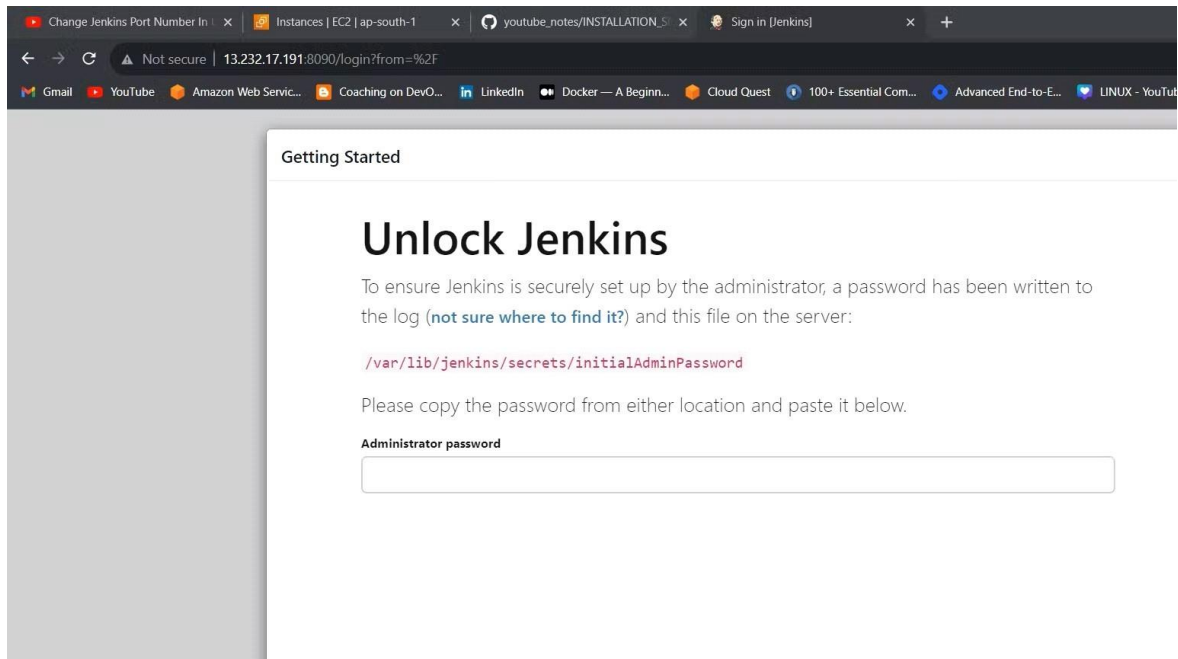
GITHUB:



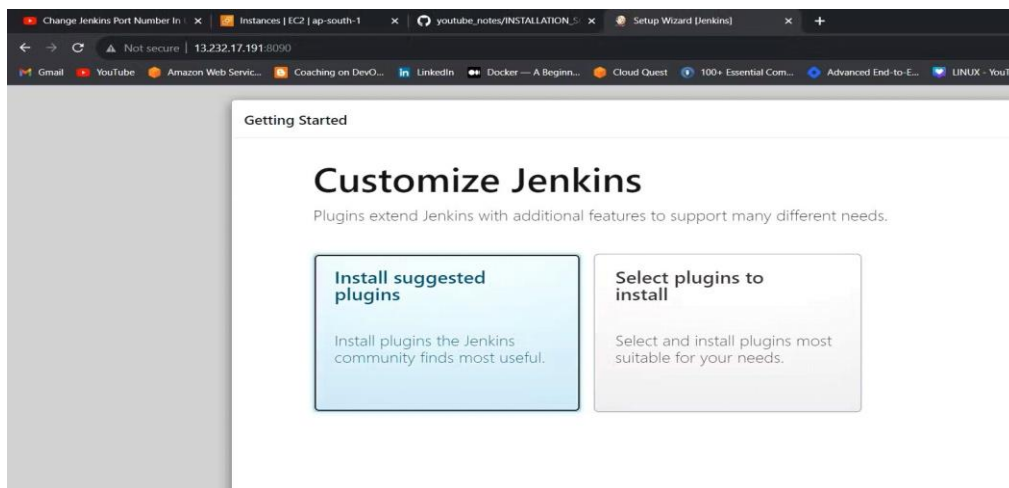
Above fig shows that the git repository that which is used to deploy the webapp in the tomcat server. It is forked by the <https://github.com/Venna12/dockerjenkin.git>.

STEP-3-JENKINS:

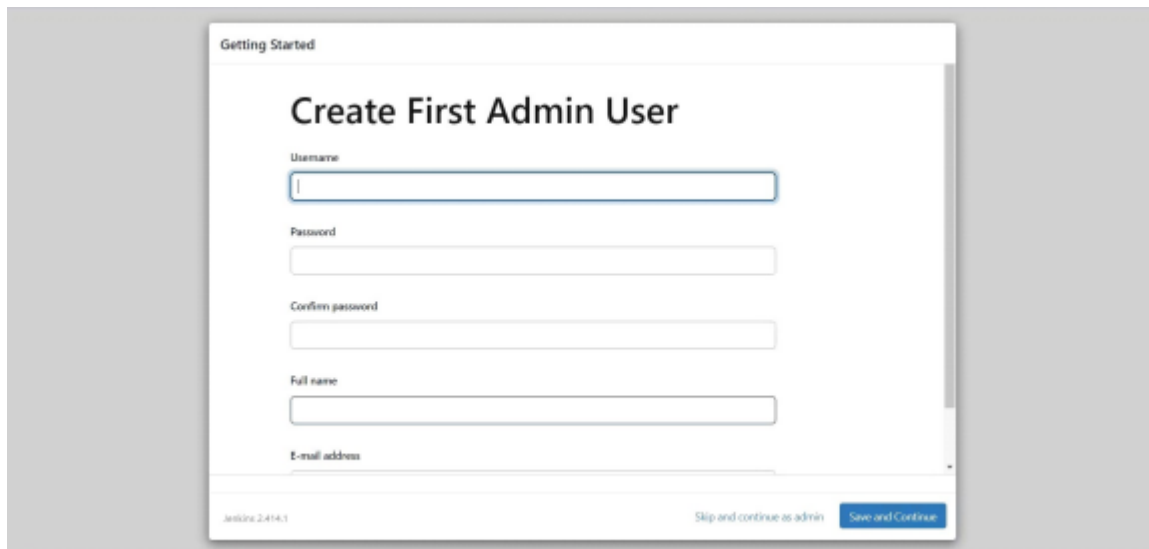
Now copy the the Ip address and go through the Jenkins portal the interface will like shownbelow



- Unlock the Jenkins by using command through the given path then we will get the passwd
cat /var/lib/Jenkins/secrets/initialAdminPassword
- Install the suggested Plugins.



- After installing the plugins, we need to create the credentials for Jenkins.

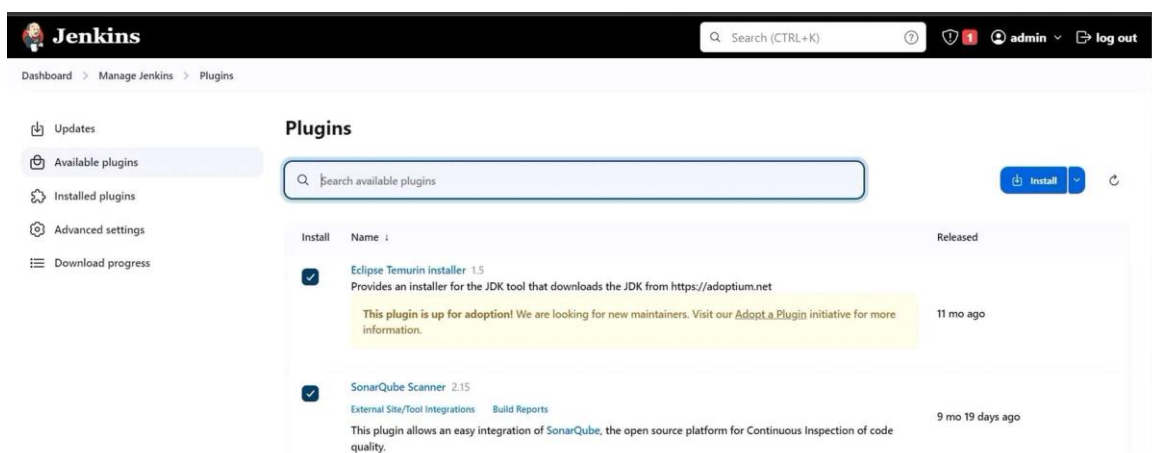


The image shows the 'Getting Started' screen for Jenkins, specifically the 'Create First Admin User' form. The form has the following fields: Username, Password, Confirm password, Full name, and E-mail address. At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 2.414.1 is displayed in the bottom left corner.

This is the Sign in page for Jenkins.

Required Plugins in The Jenkins:

- Go to Manage Jenkins → Plugins → Available Plugins → Install below plugins:
 - Eclipse Temurin Installer
 - SonarQube Scanner



The image shows the Jenkins 'Plugins' page. The left sidebar contains links to Updates, Available plugins, Installed plugins, Advanced settings, and Download progress. The main content area shows a search bar and a table of available plugins. The table has columns for 'Install', 'Name', and 'Released'. Two plugins are listed: 'Eclipse Temurin installer 1.5' and 'SonarQube Scanner 2.15'. Both have a checkmark in the 'Install' column. The 'Eclipse Temurin installer 1.5' plugin has a yellow warning box stating it is up for adoption and looking for new maintainers. The 'SonarQube Scanner 2.15' plugin has a description of its functionality and a link to 'External Site/Tool Integrations'.

Install	Name	Released
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	11 mo ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.15 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	9 mo 19 days ago

Go to Manage Jenkins → Tools → Install JDK(17.0.8.1+1) and Maven3(3.6.0) → Click on Apply and Save

Dashboard > Manage Jenkins > Tools

JDK installations

Add JDK

JDK

Name

jdk17

☒ Install automatically ?

Install from adoptium.net ?

Version ?

jdk-17.0.8.1+1

Add Installer

Dashboard > Manage Jenkins > Tools

Maven

Name

maven3

☒ Install automatically ?

Install from Apache

Version

3.6.0

Add Installer

- Create a job by taking pipeline project and name it as pet project.

Instances [EC2 | us-east-1] x EC2 Instance Connect | us-east-1 x EC2 Instance Connect | us-east-1 x New Item (Jenkins) x Log in - SonarQube x

Not secure | 3.82.197.0.8000/view/all/newJob

Jenkins

Search (CTRL+K)

Sai Teja Billipati log out

Dashboard > All >

Enter an item name

Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

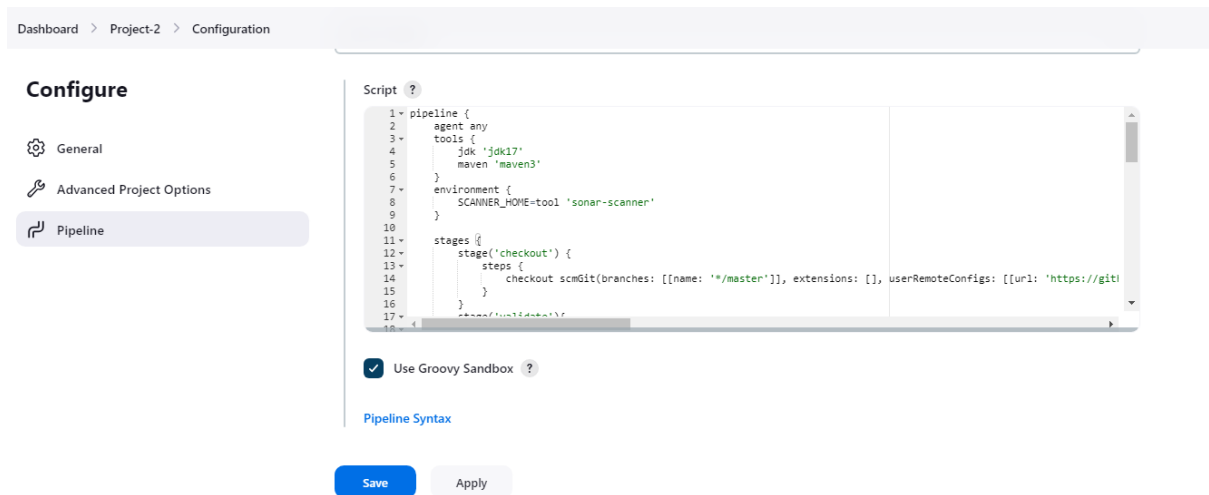
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multi-branch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Above shows the pipeline job creation in the Jenkins.

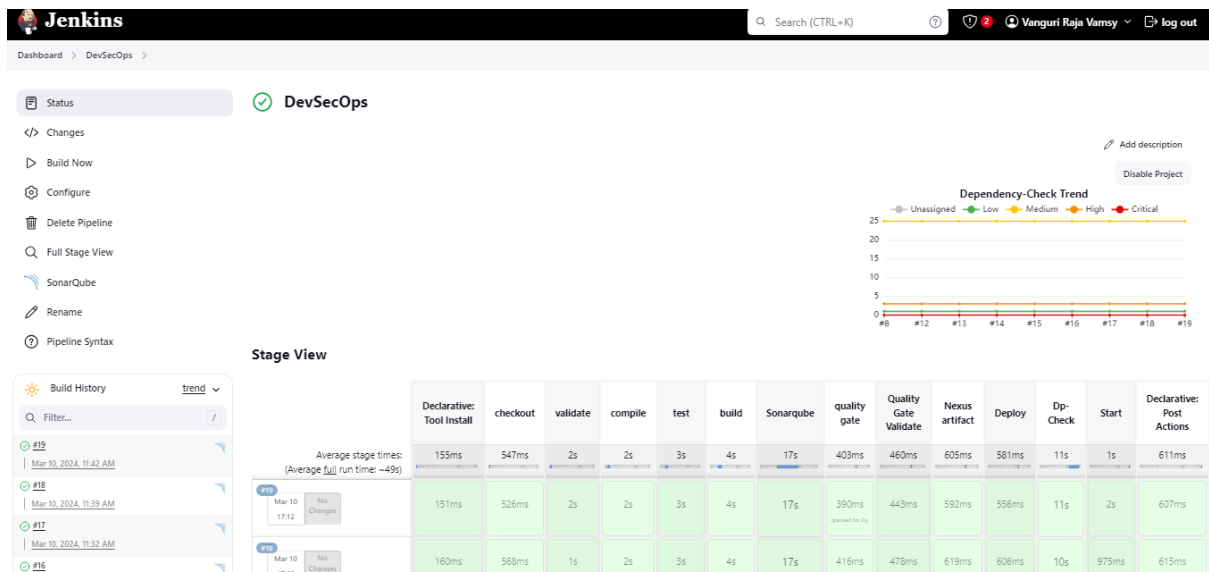
Pipeline code:

```
pipeline {
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    stages {
        stage('checkout') {
            steps {
                checkout scmGit(branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/VRaj745/dockerjenkin.git']])
            }
        }
        stage('validate'){
            steps{
                sh 'mvn validate '
            }
        }
        stage('compile'){
            steps{
```

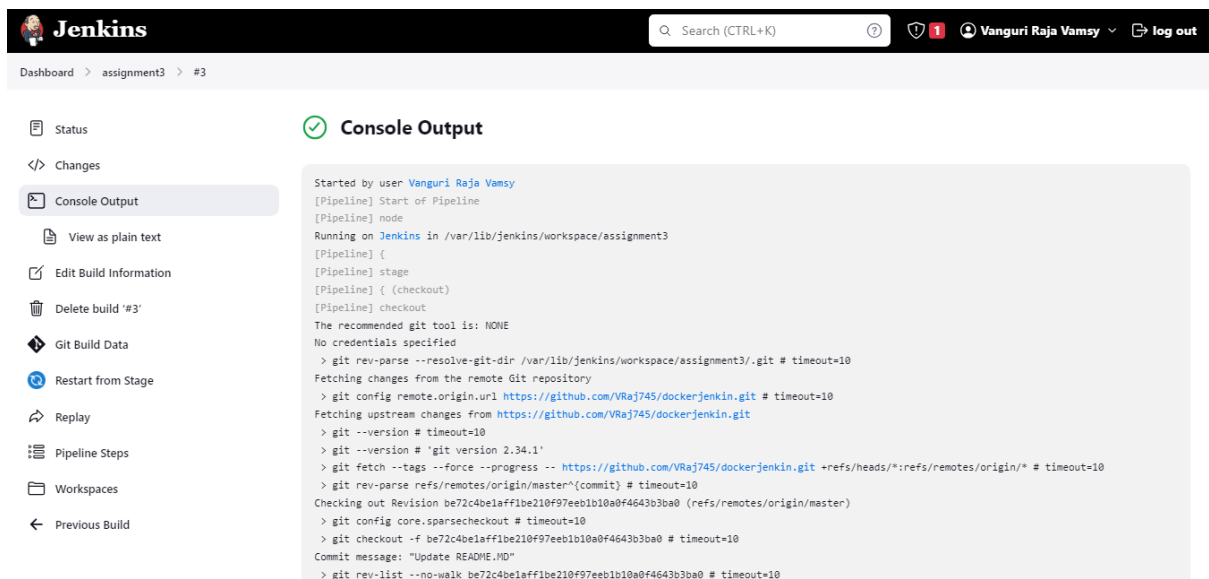


```
        sh 'mvn compile'
    }
}
stage('test'){
    steps{
        sh 'mvn test'
    }
}
stage('build'){
    steps{
        sh 'mvn package'
    }
}

stage('Dp-Check'){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format XML ',
odcInstallation: 'Dp-Check'
        dependencyCheckPublisher    pattern: '**/dependency-check-
report.xml'
    }
}
}
```



This shows the creation and building the pipeline job and it shows the stage view.



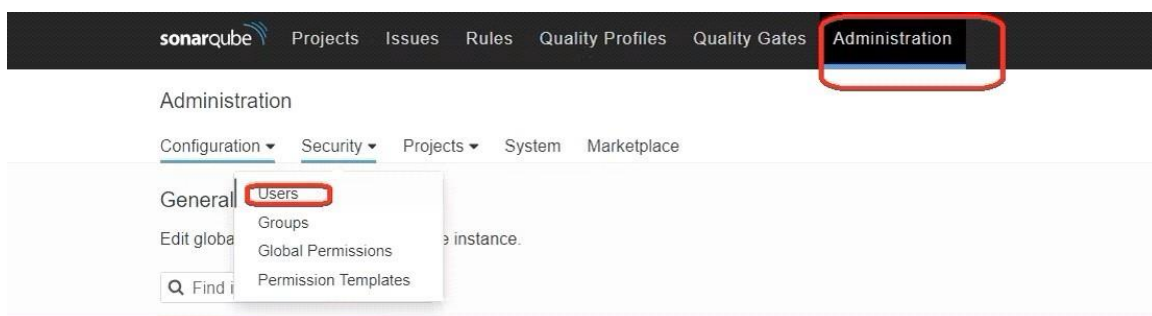
```
Dashboard > assignment3 > #3

Downloading: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/maven-metadata.xml
100 % completed (609 B / 609 B).
Downloaded: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/maven-metadata.xml (609 B at
2.6 kB/s)
Uploading: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/java-tomcat-maven-example-1.0-
20231222.153603-2.war
Uploaded: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/java-tomcat-maven-example-1.0-
20231222.153603-2.war (2.1 kB at 10 kB/s)
Uploading: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/maven-metadata.xml
Uploaded: http://3.144.16.217:8081/repository/maven-snapshots/com/example/java-tomcat-maven-example/1.0-SNAPSHOT/maven-metadata.xml (609 B at 5.5
kB/s)
Uploading artifact java-tomcat-maven-example.war completed.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] deploy
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/assignment3/target/java-tomcat-maven-example.war to container Tomcat 9.x Remote with
context assignment3
[/var/lib/jenkins/workspace/assignment3/target/java-tomcat-maven-example.war] is not deployed. Doing a fresh deployment.
Deploying [/var/lib/jenkins/workspace/assignment3/target/java-tomcat-maven-example.war]
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Above figures shows the **console output** of the pipeline job.

STEP-4-SONARQUBE:

- Create a token with a name and generate.



- copy the Token.

Tokens of Administrator

Generate Tokens

Name Expires in

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	<input type="button" value="Revoke"/>

- Go to Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this. In that secret paste the token that we copied.

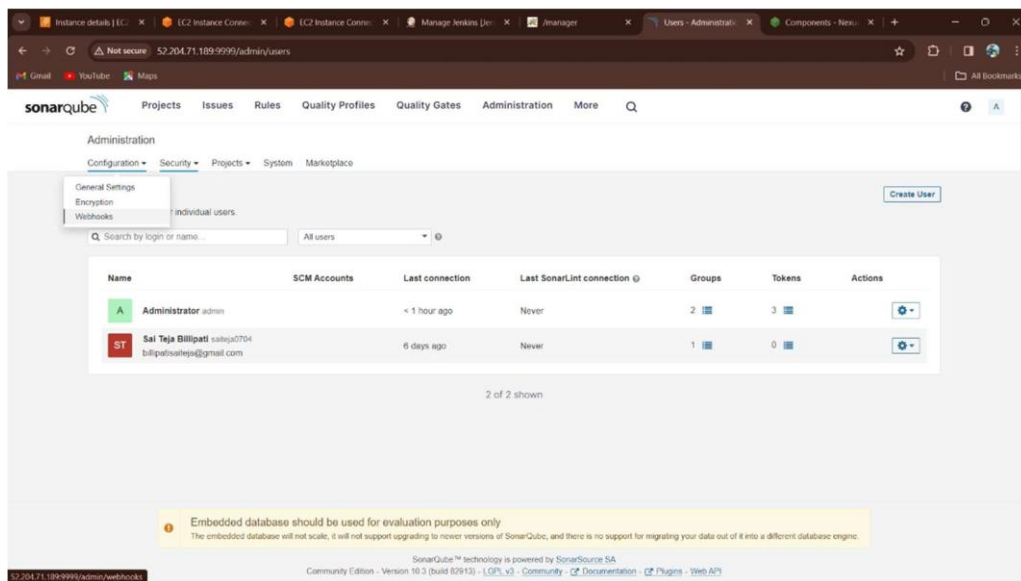
- After the creation of credentials, the interface will be like this.

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
Sonar-token	sonar	Secret text	sonar

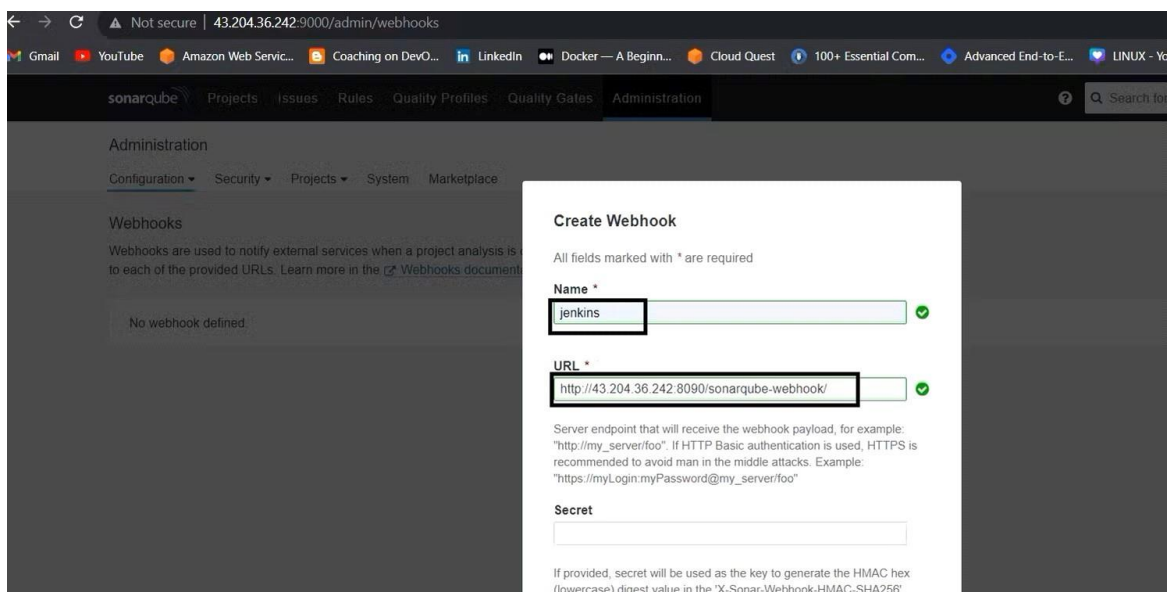
- Now, go to Dashboard → Manage Jenkins → System and Add then click on apply and save.

- In the SonarQube Dashboard add a quality gate also Administration--> Configuration--->Webhooks.



- Add details #in URL section of quality gate.

<<http://jenkins-public-ip:8080>>/sonarqube-webhook/



- Let's go to our Pipeline and add SonarQube Stage in our Pipeline Script.

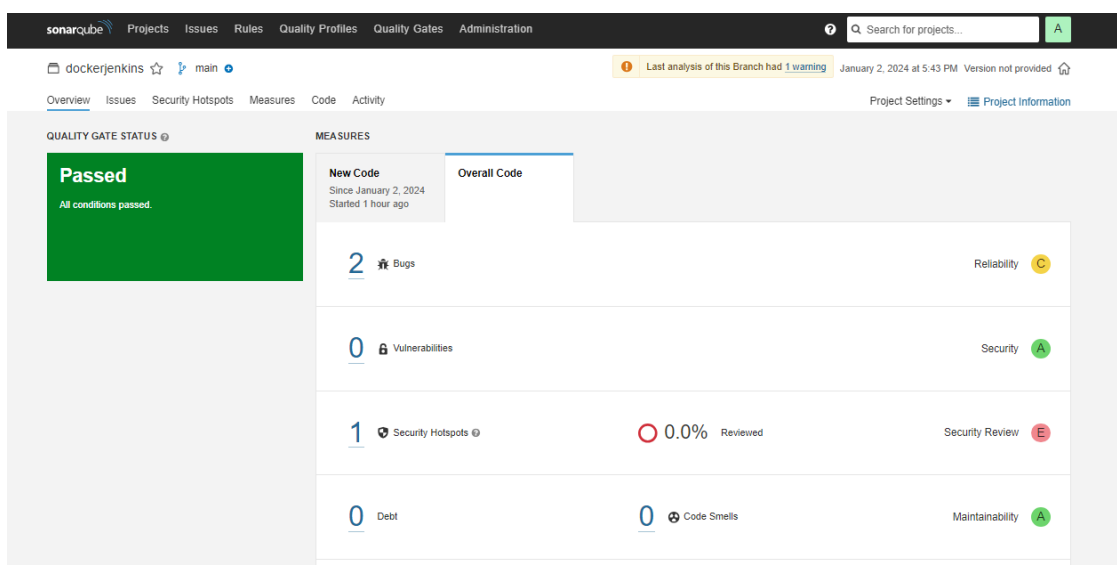
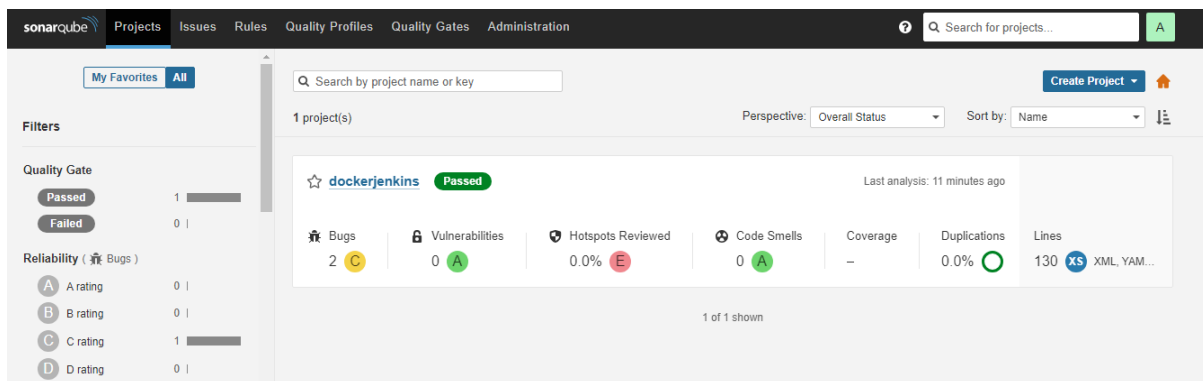
```
environment {
    SCANNER_HOME=tool 'sonar-scanner'
}
stage('Sonarqube'){
    steps{
```

```

        withSonarQubeEnv('sonar-server') {
            sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=dockerjenkins \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=dockerjenkins '''
        }
    }
    stage('quality gate'){
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-
token'
            }
        }
    }
}

```

By using the sonarqube we done the code quality analysis. It says us about the details about code which shows the vulnerabilities, bugs etc.

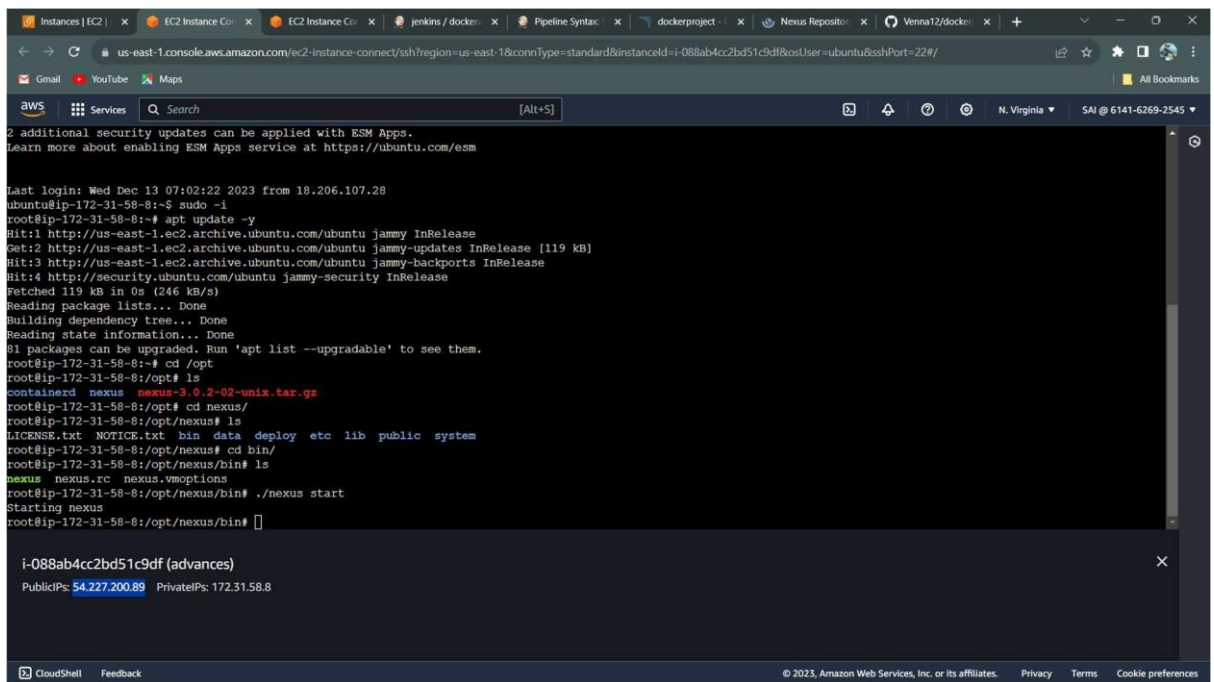


The above fig shows that the Analysis of code.

STEP-5-NEXUS:

Nexus is a Sonatype Artifactory repository manager [OSS]. It allows you to store, distribute, and retrieve build artifacts whenever it's required. Using Nexus, developers can easily access and deploy build artifacts in an organization from a single location.

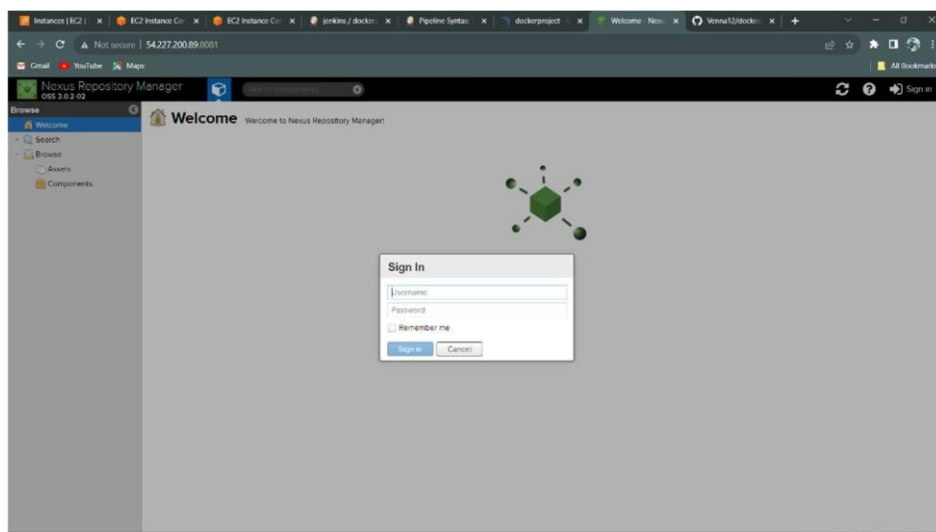
- Install the nexus and un-tar it as below process



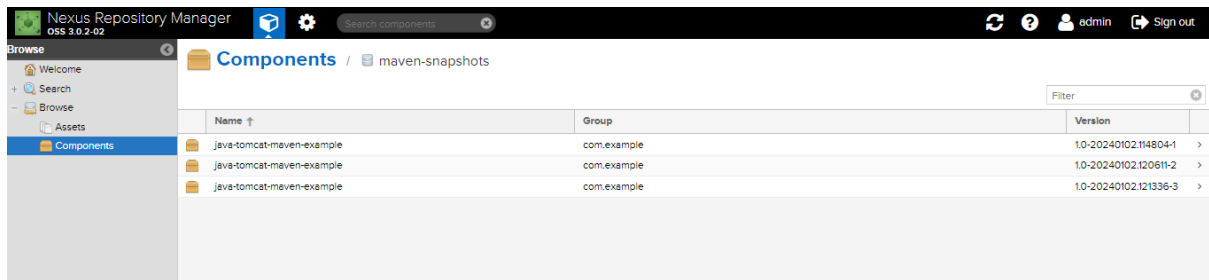
```
2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Dec 13 07:02:22 2023 from 10.206.107.28
ubuntu@ip-172-31-58-8:~$ sudo -i
root@ip-172-31-58-8:~# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Fetched 119 kB in 0s (246 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
81 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-58-8:~# cd /opt
root@ip-172-31-58-8:/opt# ls
contained nexus nexus-3.0.2-02-unix.tar.gz
root@ip-172-31-58-8:/opt# cd nexus/
root@ip-172-31-58-8:/opt/nexus# ls
LICENSE.txt NOTICE.txt bin data deploy etc lib public system
root@ip-172-31-58-8:/opt/nexus# cd bin/
root@ip-172-31-58-8:/opt/nexus/bin# ls
nexus nexus.rc nexus.vmoptions
root@ip-172-31-58-8:/opt/nexus/bin# ./nexus start
Starting nexus
root@ip-172-31-58-8:/opt/nexus/bin#
```

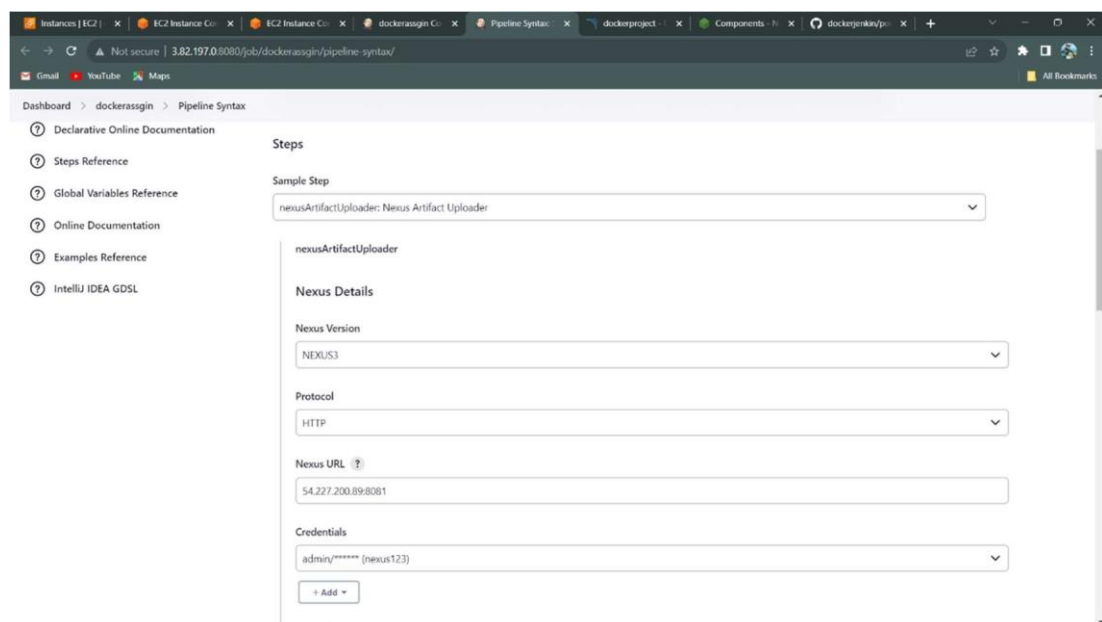
- And login through the nexus **USER:admin & PASSWD:admin.**



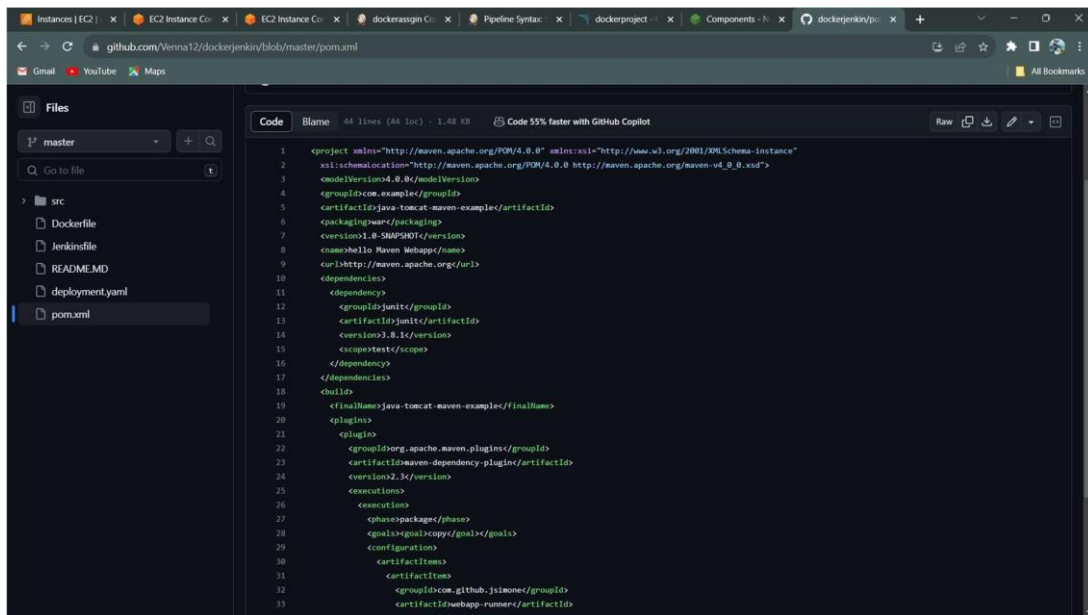
- Go to > Components > maven-Snapshots!



- Now go through the Jenkins and in the pipeline, syntax select the **Nexus artifact Uploader**.
- Fill it in accordance through that required one with the current Ip address of the server

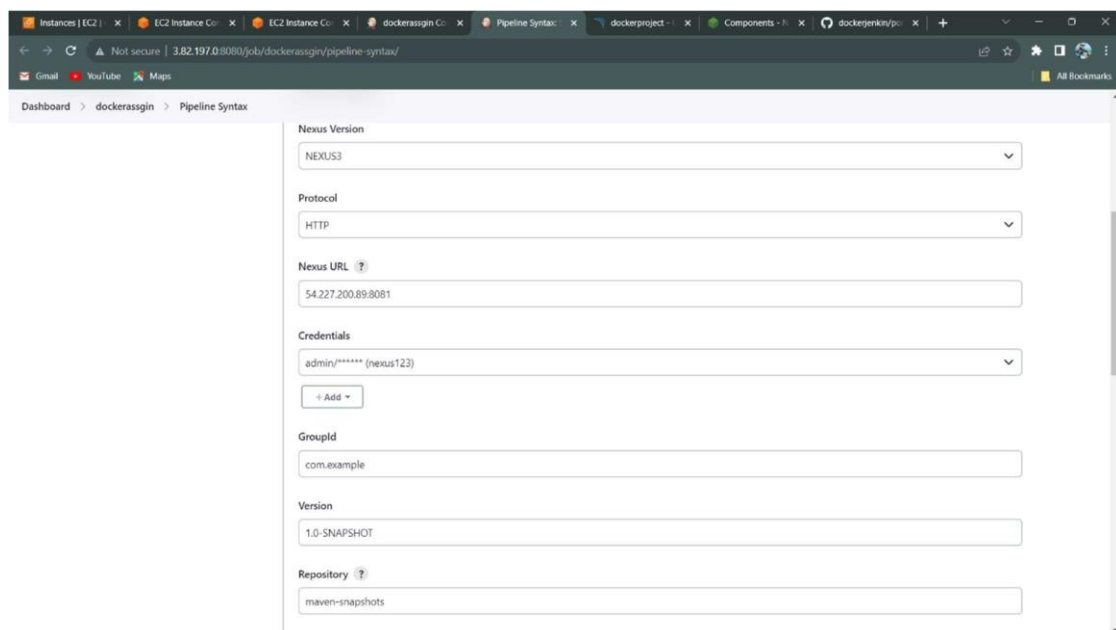


- For the Group-id, versions we can see that in the project of the **pom.xml**.



```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.example</groupId>
5   <artifactId>java-tomcat-maven-example</artifactId>
6   <packaging>war</packaging>
7   <version>1.0-SNAPSHOT</version>
8   <name>Hello Maven Webapp</name>
9   <url>http://maven.apache.org/</url>
10  <dependencies>
11    <dependency>
12      <groupId>junit</groupId>
13      <artifactId>junit</artifactId>
14      <version>3.8.1</version>
15      <scope>test</scope>
16    </dependency>
17  </dependencies>
18  <build>
19    <finalName>java-tomcat-maven-example</finalName>
20    <plugins>
21      <plugin>
22        <groupId>org.apache.maven.plugins</groupId>
23        <artifactId>maven-dependency-plugin</artifactId>
24        <version>2.1</version>
25        <executions>
26          <execution>
27            <phase>package</phase>
28            <goals>goal:copy</goals>
29            <configuration>
30              <artifactItems>
31                <artifactItem>
32                  <groupId>com.github.jsimone</groupId>
33                  <artifactId>webapp-runner</artifactId>
```

- Now we have a below Artifact we have group-id, version, and repository all we can by the port.xml



Dashboard > dockerassign > Pipeline Syntax

Nexus Version
NEXUS3

Protocol
HTTP

Nexus URL ?
54.227.200.89:8081

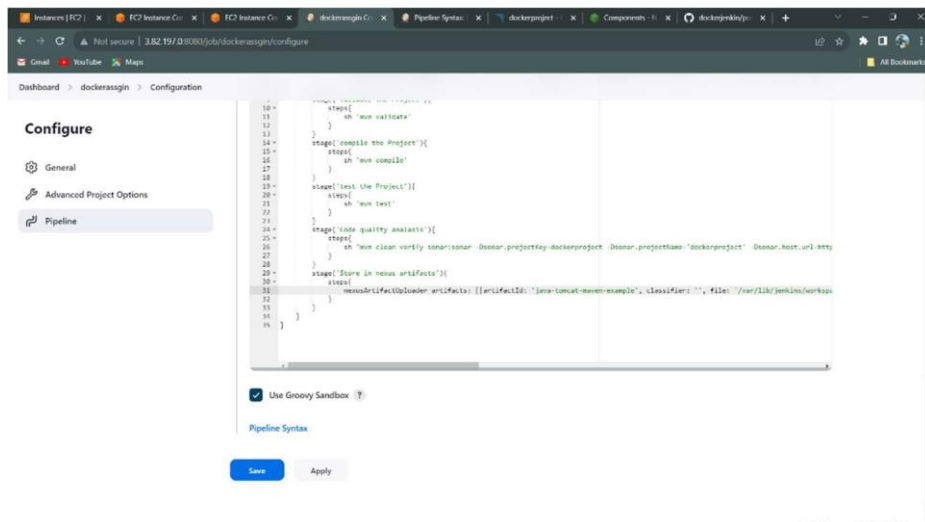
Credentials
admin/nexus123
+ Add

GroupId
com.example

Version
1.0-SNAPSHOT

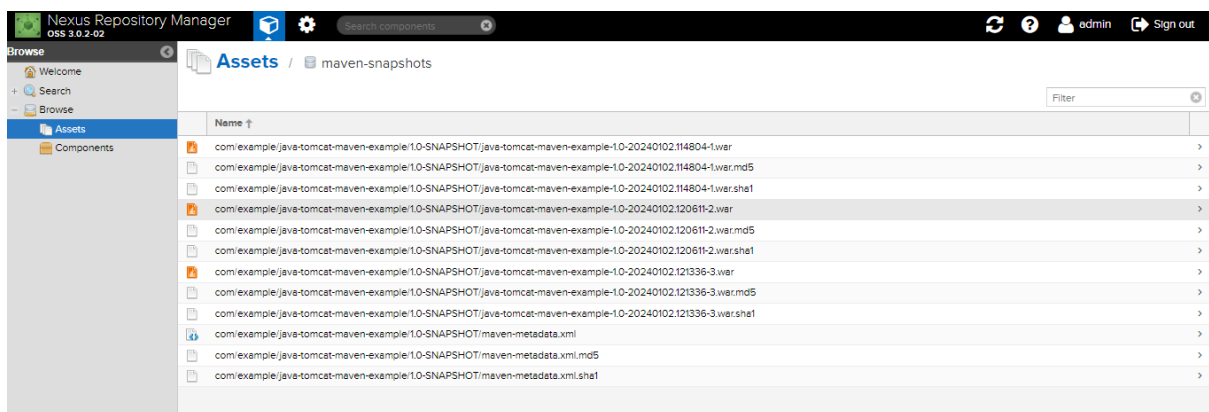
Repository ?
maven-snapshots

- Write the pipeline syntax for the nexus artifact.



```
stage('Nexus artifact'){
    steps{
        nexusArtifactUploader artifacts: [[artifactId: 'java-tomcat-maven-
example', classifier: '', file: '/var/lib/jenkins/workspace/Project-
2/target/java-tomcat-maven-example.war', type: 'war']], credentialsId:
'nexus123', groupId: 'com.example', nexusUrl: '13.59.143.13:8081',
nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots',
version: '1.0-SNAPSHOT'
    }
}
```

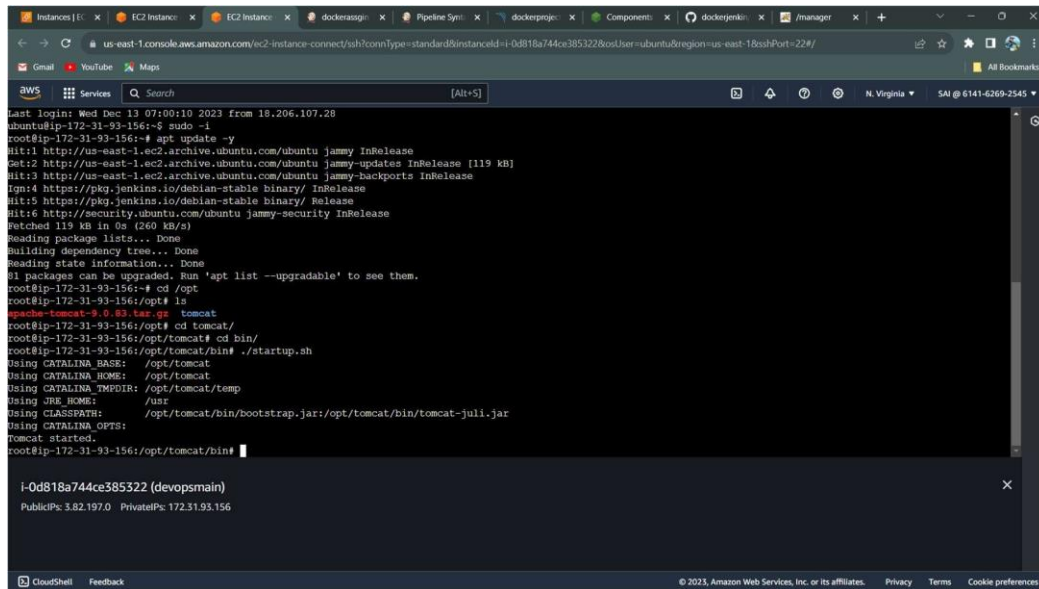
- And build it.
- The required artifact will be generated.



Above fig shows the nexus artifacts.

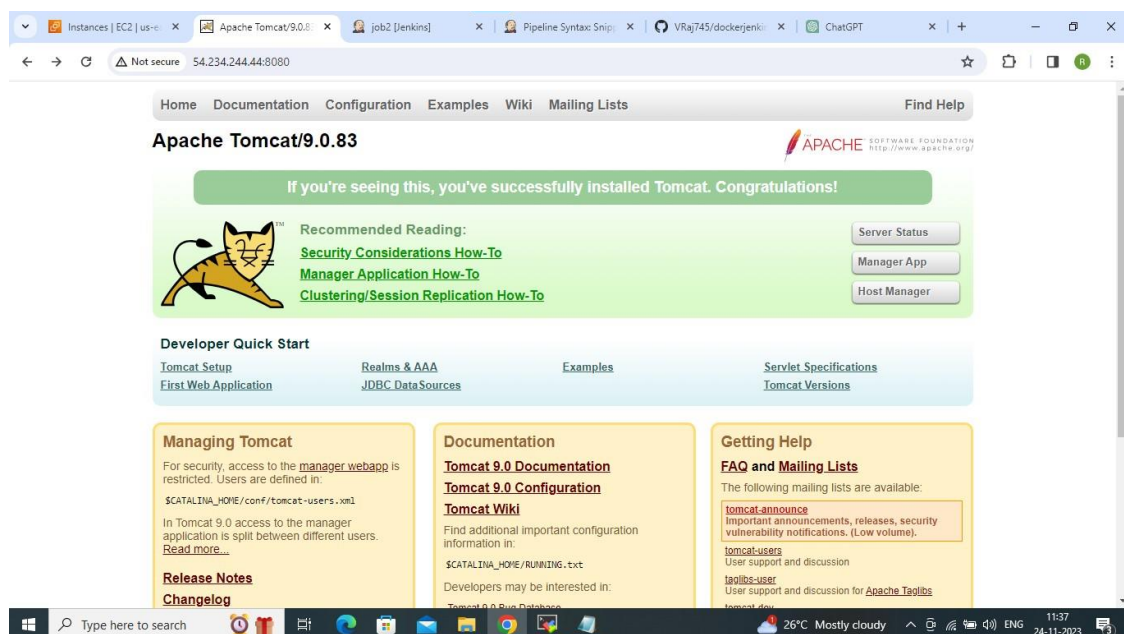
STEP-6-TOMCAT:

- Installing the tomcat in CD/OPT. Un-taring the tomcat and moving the file to the tomcat file and run the tomcat.



```
us-east-1-console.aws.amazon.com/ec2-instance-connect/sh?connType=standard&instanceId=i-0d818a744ce385322&osUser=ubuntu&region=us-east-1&sshPort=22#/
Last login: Wed Dec 13 07:00:10 2023 from 18.206.107.28
ubuntu@ip-172-31-93-156:~$ sudo -i
root@ip-172-31-93-156:~# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
Fetched 119 kB in 0s (260 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
51 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-93-156:~# cd /opt
root@ip-172-31-93-156:/opt# ls
apache-tomcat-9.0.83.tar.gz tomcat
root@ip-172-31-93-156:/opt# cd tomcat/
root@ip-172-31-93-156:/opt/tomcat# cd bin/
root@ip-172-31-93-156:/opt/tomcat/bin# ./startup.sh
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
root@ip-172-31-93-156:/opt/tomcat/bin#
```

- The tomcat interface will be like this when we run it.



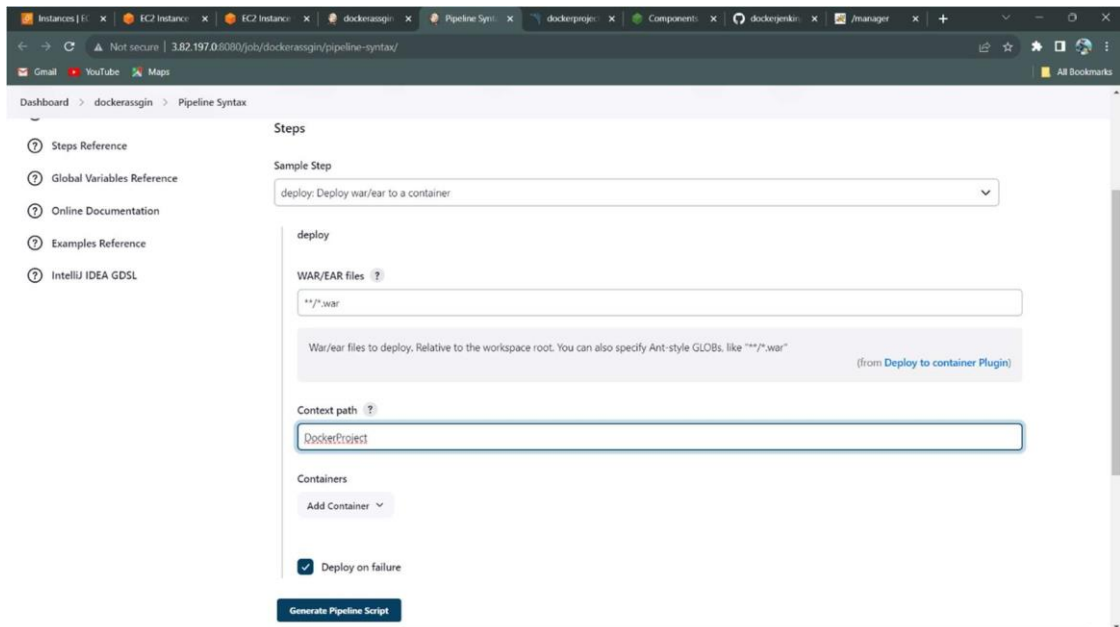
- Above fig shows that the open page of the TOMCAT server. By clicking on the manager app button, it takes us into the below view of the web page.



Tomcat Web Application Manager

Message:		OK			
Manager					
List Applications		HTML Manager Help		Manager Help	
Server Status					
Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/PROJECT-3	None specified	Archetype Created Web Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/docs	None specified	Tomcat Documentation	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/examples	None specified	Servlet and JSP Examples	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/host-manager	None specified	Tomcat Host Manager Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/manager	None specified	Tomcat Manager Application	true	1	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>

- Now go to Jenkins pipeline syntax and select the deploy the container and the context pathname it as project-3.



- Now go through the add container select the USERNAME AND PASSWORDS and give the tomcat credentials add and save.

Dashboard > dockerassgin > Pipeline Syntax

Context path ?

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
deployer

☐ Treat username as secret ?

Password ?

ID ?

Description ?

☒ Deploy on failure

Generate Pipeline Script

- Select the TOMCAT9 container. add the credential's and the URL of the tomcat.

Dashboard > dockerassgin > Pipeline Syntax

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "*/**/*.war" (from Deploy to container Plugin)

Context path ?
/

Containers

Tomcat 9.x Remote

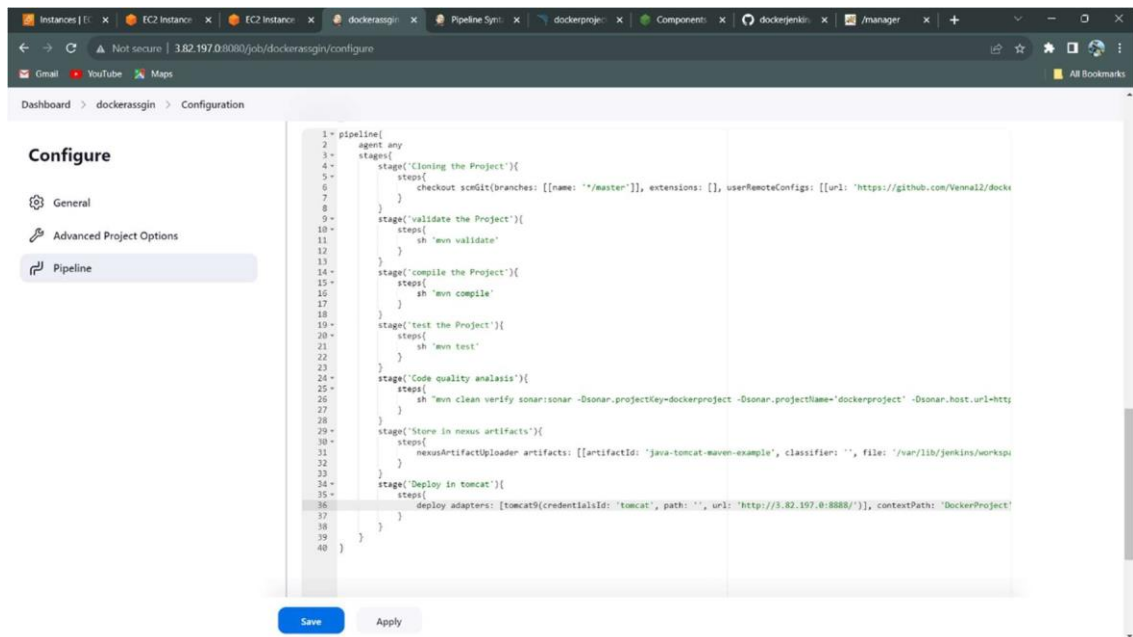
Credentials
deployer/***** (tomcat)

Tomcat URL ?
http://3.82.197.0.8888/

Advanced

Add Container

- Generate the pipeline for the deployment of tomcat.



```

stage('Deploy'){
    steps{
        deploy adapters: [tomcat9(credentialsId: '52c24633-b9e7-4df0-b8aa-0fb09b4aa831', path: '', url: 'http://13.59.143.13:8086/'),
        contextPath: 'PROJECT-3', war: '**/*.war'
    }
}

```



Tomcat Web Application Manager

Message:

OK

Manager

List Applications

HTML Manager Help

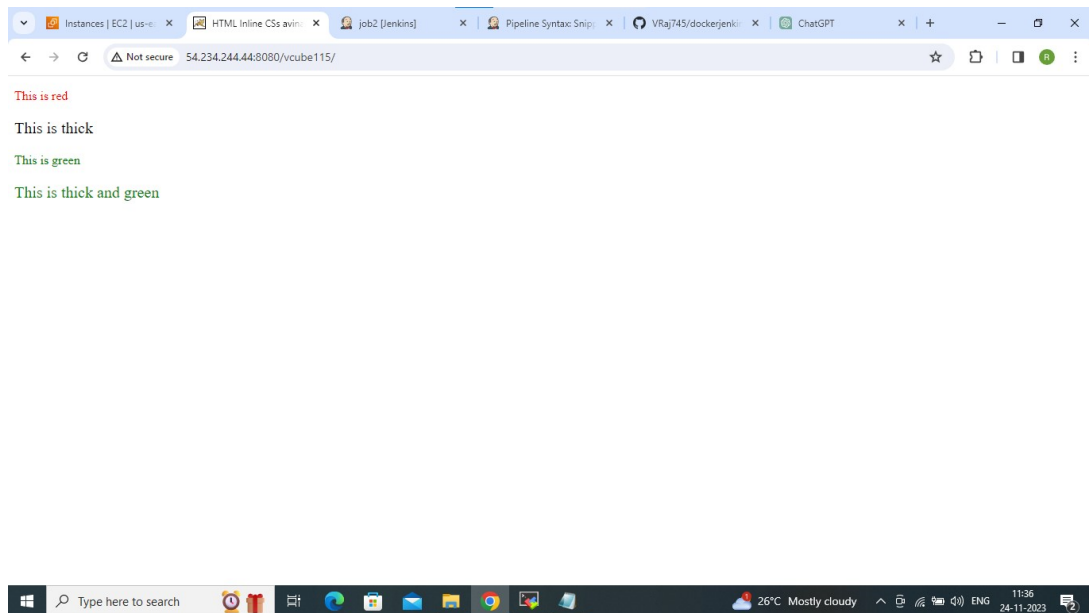
Manager Help

Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/PROJECT-3	None specified	Archetype Created Web Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/docs	None specified	Tomcat Documentation	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/examples	None specified	Servlet and JSP Examples	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/host-manager	None specified	Tomcat Host Manager Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/manager	None specified	Tomcat Manager Application	true	1	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>

- In this page we can see the deployed webapp in the list as 'Project-3'.



- This is the Final output of the Project.

STEP-7-DP-CHECK:

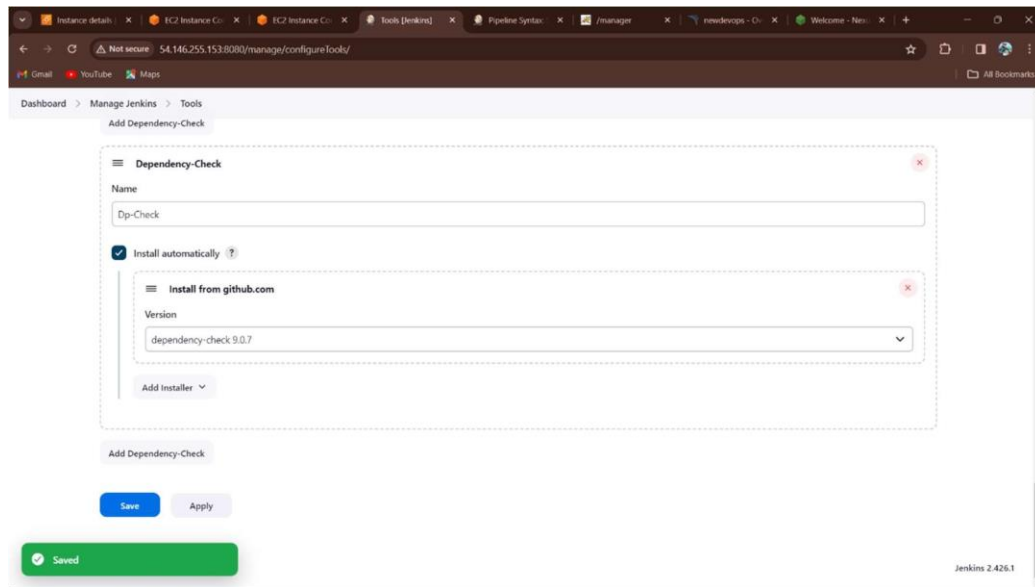
Dependency-check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities. This tool can be part of the solution to the OWASP top 10 2017: a9 - using components with known vulnerabilities.

Install OWASP Dependency Check Plugins:

Go to Dashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it.



- Go to Dashboard → Manage Jenkins → Tools → Click on Apply and Save here.



- Now go configure → Pipeline and add this stage to your pipeline and build

```
stage("OWASP Dependency
Check"){steps{
```

```
dependencyCheck additionalArguments: '--scan ./ --
format XML ',odcInstallation: 'DP-Check'
```

```
dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
```

```
}
```

```
}
```


Jenkins

Search (CTRL+K)

Vanguri Raja Vamsy log out

Dashboard > Project-2 > #7 > Dependency-Check

Dependency-Check Results

SEVERITY DISTRIBUTION

File Name	Vulnerability	Severity	Weakness
+ java-tomcat-maven-example.war	NVD CVE-2000-1210	Medium	NVD-CWE-Other
+ java-tomcat-maven-example.war	NVD CVE-2001-0590	Medium	NVD-CWE-Other
+ java-tomcat-maven-example.war	NVD CVE-2002-0493	High	CWE-254
+ java-tomcat-maven-example.war	NVD CVE-2005-4838	Medium	CWE-79
+ java-tomcat-maven-example.war	NVD CVE-2006-7196	Medium	CWE-79
+ java-tomcat-maven-example.war	NVD CVE-2007-1358	Low	CWE-79
+ java-tomcat-maven-example.war	NVD CVE-2007-2449	Medium	NVD-CWE-Other
+ java-tomcat-maven-example.war	NVD CVE-2008-0128	Medium	CWE-16
+ java-tomcat-maven-example.war	NVD CVE-2009-2696	Medium	CWE-79
+ java-tomcat-maven-example.war	NVD CVE-2013-2185	High	CWE-20

STEP-8-DOCKER:

We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins.

- Docker
- Docker commons
- Docker pipeline
- Docker API
- Docker-Build-Step.

Dashboard > Manage Jenkins > Plugins

Installed plugins

Advanced settings

Download progress

Search: docker

- ☒ **Docker 1.5**
Cloud Providers Cluster Management docker
This plugin integrates Jenkins with Docker
This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.
- ☒ **Docker Commons 439.va_3cb_0a_6a_fb_29**
Library plugins (for use by other plugins) docker
Provides the common shared functionality for various Docker-related plugins.
- ☒ **Docker Pipeline 572.v950f58993843**
pipeline DevOps Deployment docker
Build and use Docker containers from pipelines.
This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

- Now, go to Dashboard → Manage Jenkins → Tools →

Dashboard > Manage Jenkins > Tools

Docker installations

Add Docker

≡ Docker

Name

docker

☒ Install automatically ?

≡ Download from docker.com

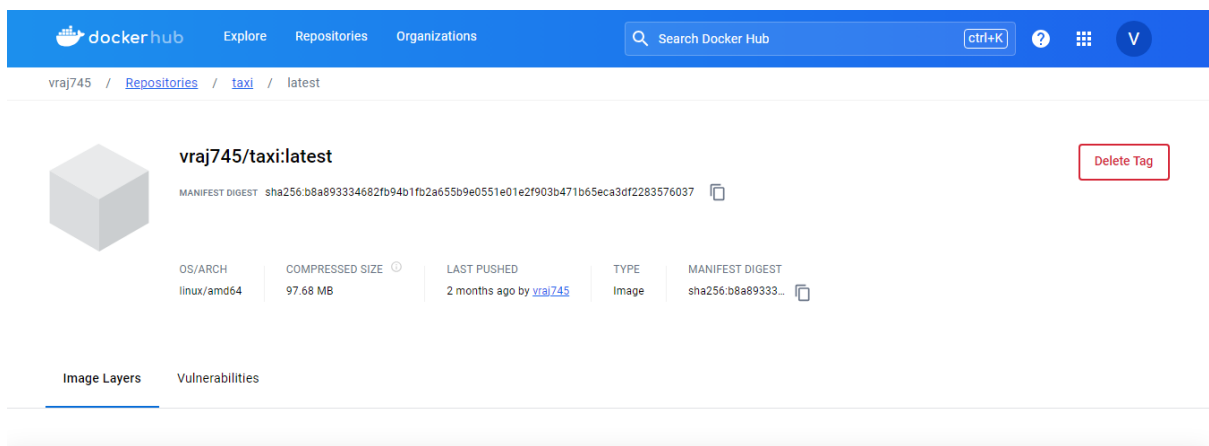
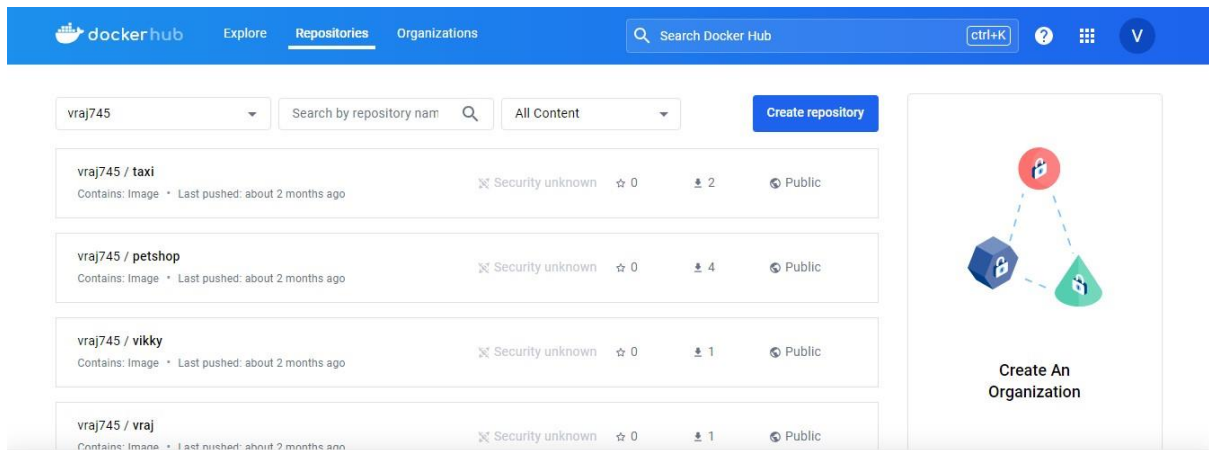
Docker version ?

latest

- Add Docker Hub Username and Password under Global Credentials
- Add this stage to Pipeline Script


```
stage ('Build and push to docker hub'){
    steps{
        script{
            withDockerRegistry(credentials: 'docker', toolName: 'docker') {
                sh "docker build -t taxi."
                sh "docker tag taxi vraj745/taxi:latest"
                sh "docker push vraj745/taxi:latest"
            }
        }
    }
}

stage ('Deploy to container'){
    steps{
        sh 'docker run -d --name pet1 -p 8080:8080 vraj745/taxi:latest'
    }
}
```
- When we login in the Docker we can see a new image has been added.



Now the Final Output will be seen by using the

<Ec2-public-ip:8082/taxi>