Course: DevOps Name: ch. Lakshmi Priyanka
Module: Docker & Docker Hub Batch no: 115
Topic: SonarQube, Nexus, Tomcat Assignment no: 04
Trainer Name: Mr.Madhukar sir Date of submission:15-dec-2023
Mail-ID:chlakshmipriyanka9@gmail.com

Assignment: SonarQube does code quality analysis, Store in nexus artifacts, and Deploy in tomcat using the project GitHub link: https://github.com/Venna12/dockerjenkin.git

Pre-requirements:

^ . Project link from GitHub (Venna12).

^ . Install Java jdk-11 and maven.

^ . Jenkins and setup Jenkins (Install requires plugs).

^ . Install SonarQube and set up an environment.

^ . Install Nexus Artifacts and set up the environment.

^ . Install Tomcat and set up an environment.

Connect to the instance:

- Launch the EC2 instances, take instance type (t2.medium or t2.large), select Ubuntu OS for the project, and wait for the instances to change from pending to running state.
- Edit the Security Bounds and add the security bound (use port no or use All traffic and Anywhere 0.0.0./0) to run the servers in Google.
- Connect the cloud command line interface.
- Change the normal user to the root user (using sudo -i).
- Update the Linux server using a command (apt update -y).

^ . Use the command to install Java jdk-11 in the server (sudo apt install default-jdk -y or apt install openjdk-11-jdk -y).

^ . Use the command to install Maven in the server (sudo apt installmaven -y). To Check install Java and Maven: java --version, mvn --version.

Creating a Pipeline in the Jenkins for Project

Open the Jenkins using the AWS Public IPV4 and using port
no 8080.

^ . Create the Job using the pipeline. Job name Dockerassgin.

^. We configure the dashboard, clicked on Git, and added the clone link to the given box.

Project URL: https://github.com/Venna12/dockerjenkin.git

SonarQube:

SonarQube is a static code analysis tool, a tool that scans your code and tries to detect flaws, bugs, security vulnerabilities, etc. It can also measure the test coverage of your code if provided with proper reports. All these features focus on direct code development and help developers build better products. SonarQube is an open-source platform developed by Sonar Source for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs and code smells in 29 programming languages.

^ SonarQube is start using port no 9000. IP:9000. To Sonarqube software.

^ We can install SonarQube manual or Using Docker we can easily install SonarQube.

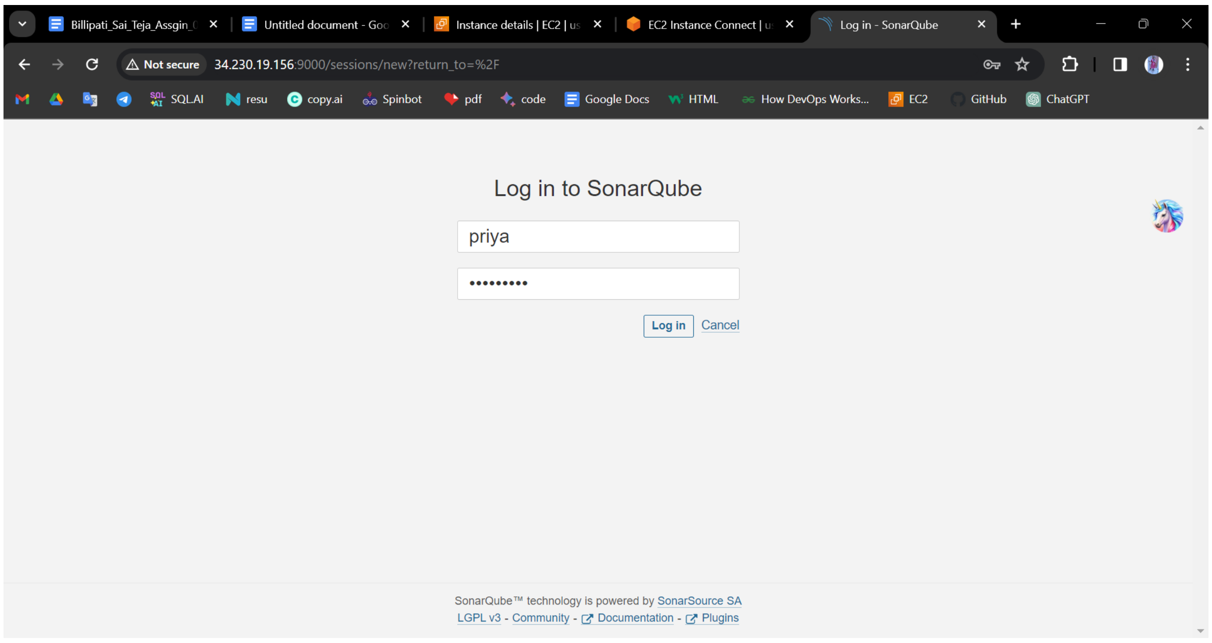
Using Docker: Install Docker: sudo apt install docker.io -y
Install SonarQube: docker run --name “Name of the container” -d -p 9000:9000 sonarqube: latest.

Default

^ Username: admin

Password: admin

The interface of the SonarQube for logging in to SonarQube.



Nexus:

^ Nexus by Sonatype is a repository manager that organizes, stores and distributes artifacts needed for development. With Nexus, developers can completely control access to, and deployment of, every artifact in an organization from a single location, making it easier to distribute software.

^ Nexus is purely written in Java. But it can store the 29 other programming language artifacts in it.

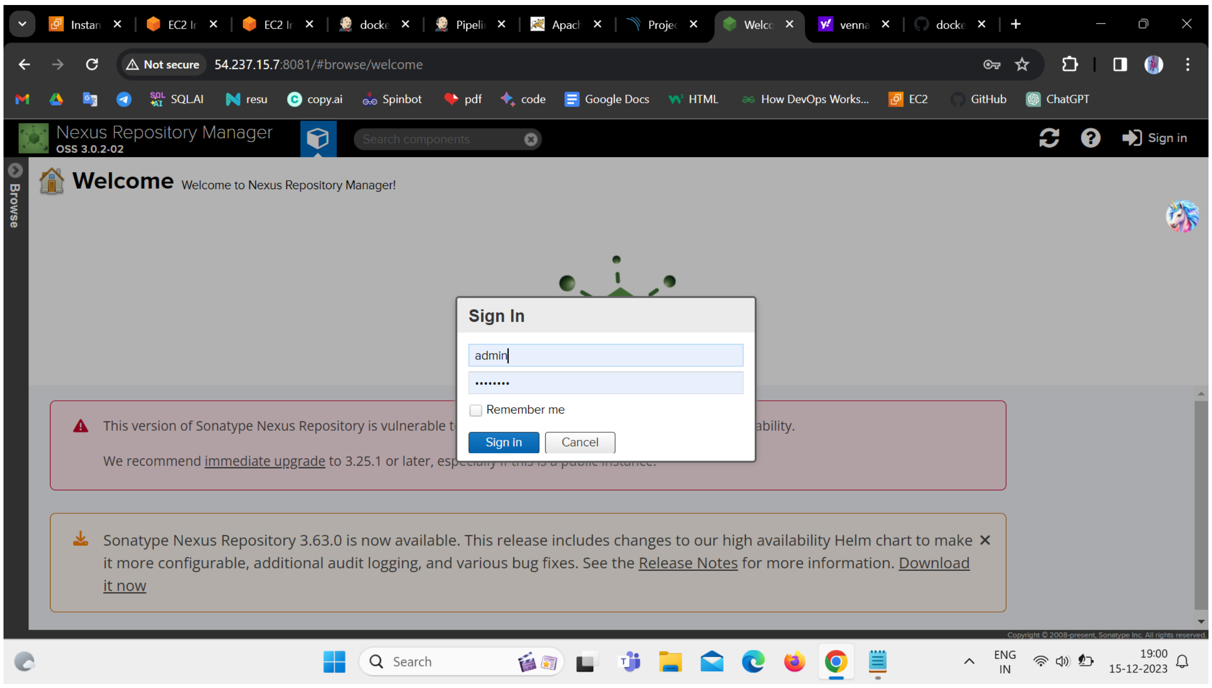
^ Nexus is run on Java jdk-8 version only.

^ To run the Nexus use the server IP address and Nexus Port no: 8081. Server Ip:8081 to open Nexus on any search engine.

Default

☾ Username: admin

Password: admin123.



Welcome Welcome to Nexus Repository Manager!

This version of Sonatype Nexus Repository is vulnerable t ability.

We recommend immediate upgrade to 3.25.1 or later, esp

Sonatype Nexus Repository 3.63.0 is now available. This release includes changes to our high availability Helm chart to make it more configurable, additional audit logging, and various bug fixes. See the Release Notes for more information. Download it now

User interface of the Nexus.

Apache Tomcat

Tomcat Apache Tomcat is a free and open-source implementation of the Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies. It provides a "pure Java" HTTP web server environment in which Java code can also run. Thus, it is a Java web application server, although not a full JEE application server.

^ By default, Tomcat starts up on HTTP connector 8080. If another application on the install machine is already using port 8080 (for example, if you have another instance of Tomcat on the machine), then change the default startup port by modifying the conf/server. xml file.

To sign in the Tomcat, we are config the details of the user and password.

Username: deployer

Password: deployer

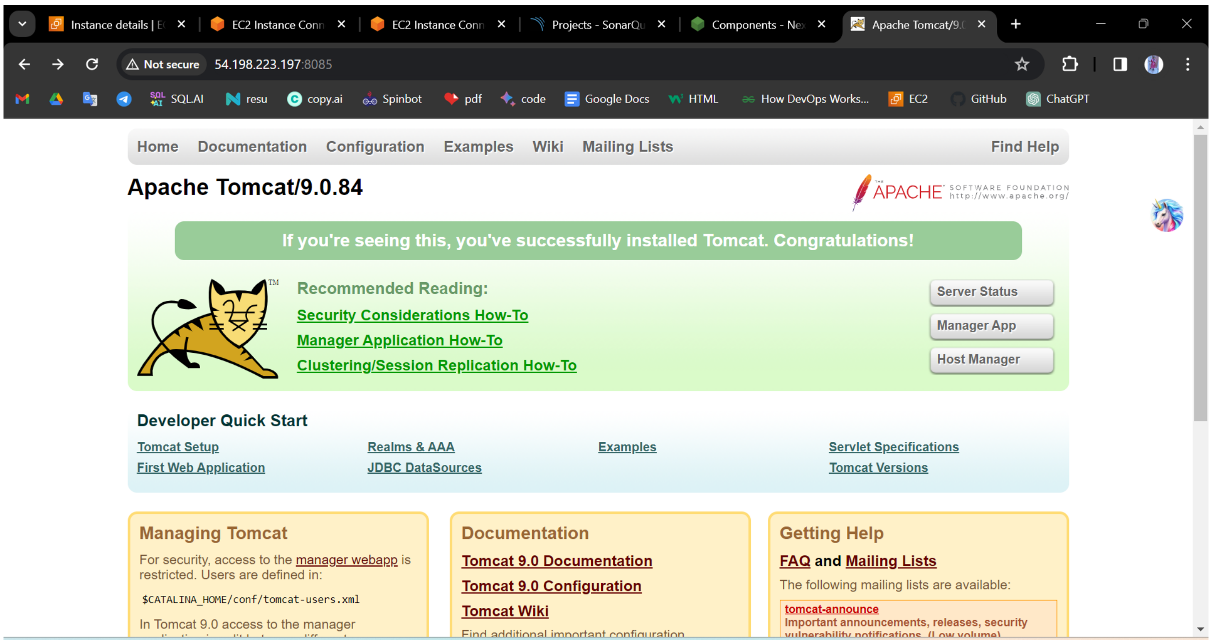
Login details of the Apache Tomcat server.

Login: tomcat

Password: s3cret

To open the Tomcat server use the IP address and port of Tomcat.

☾ IP Address:8080 (Port no).



Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/9.0.84

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Server Status
Manager App
Host Manager

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 9.0 access to the manager

Documentation
Tomcat 9.0 Documentation
Tomcat 9.0 Configuration
Tomcat Wiki
Find additional important configuration

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

User Interface of the Apache Tomcat/9.0.83.

^ Write a pipeline script for the Git clone for the project using the link Project URL: https://github.com/Venna12/dockerjenkin.git

1. Create a pipeline for Dockerjenkin
    a. Clone the Project
    b. Validate the Project
    c. Compile the Project
    d. Test the Project
2. Pipeline script for the Project

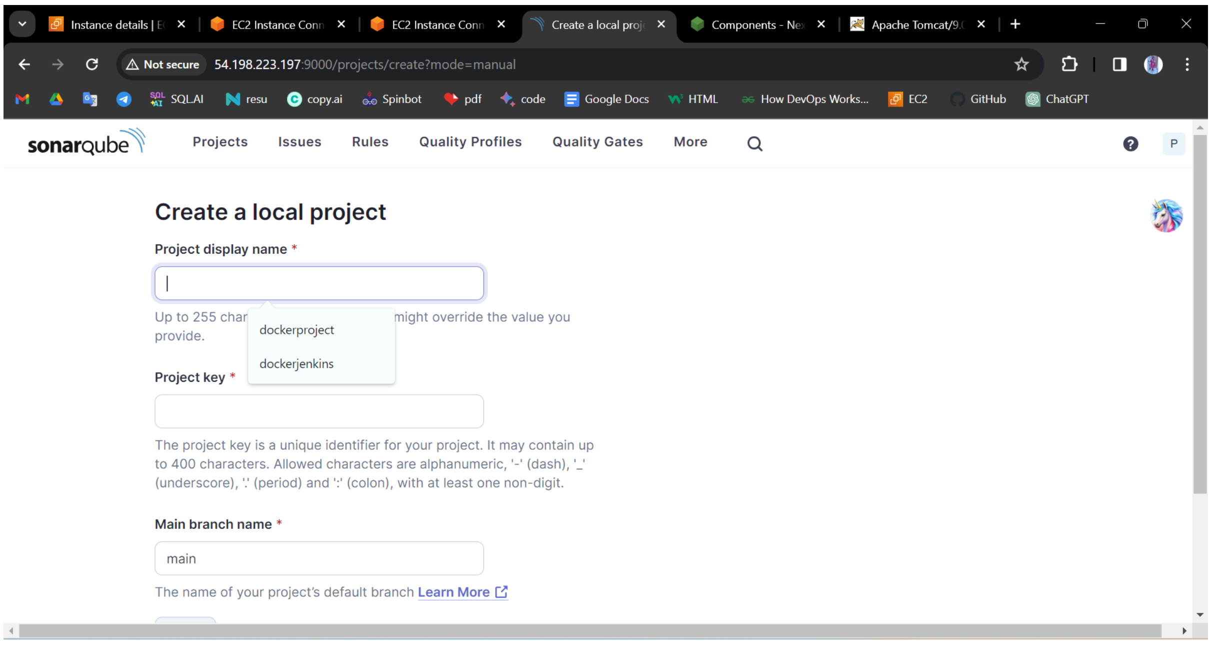
Script:

```
pipeline{
 agent any
 stages{
 stage('Cloning the Project'){
 steps{
 checkout scmGit(branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/Venna12/dockerjenkin.git']])  }
 }
 stage('validate the Project'){
 steps{
 sh 'mvn validate'
 }
 }
 stage('compile the Project'){
```

```
steps{
sh 'mvn compile'
}
}
stage('test the Project'){
steps{
sh 'mvn test'
}
}
```

Configuration of SonarQube to Jenkins pipeline:

1. Open the SonarQube using Server IP address with port no.
    a. Server Ip:port no (9000)
2. Login to SonarQube using default credentials.
    a. Username: admin
    b. Password: admin
3. Once login to SonarQube to change default password.
4. It redirects to SonarQube Dashboard.


54.198.223.197:9000/projects/create?mode=manual

Create a local project

Project display name *

Up to 255 char... might override the value you provide.

dockerproject
dockerjenkins

Project key *

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

main

The name of your project's default branch Learn More

SonarQube Dashboard

5. Click on the Create a local project.
    a. Project display name: dockerproject
    b. Project key: dockerproject
    c. Main branch name: main
6. It shows different options for the options

   a. Previous version

   b. Number of days

   c. Reference branch

7. Select the Previous version for this project

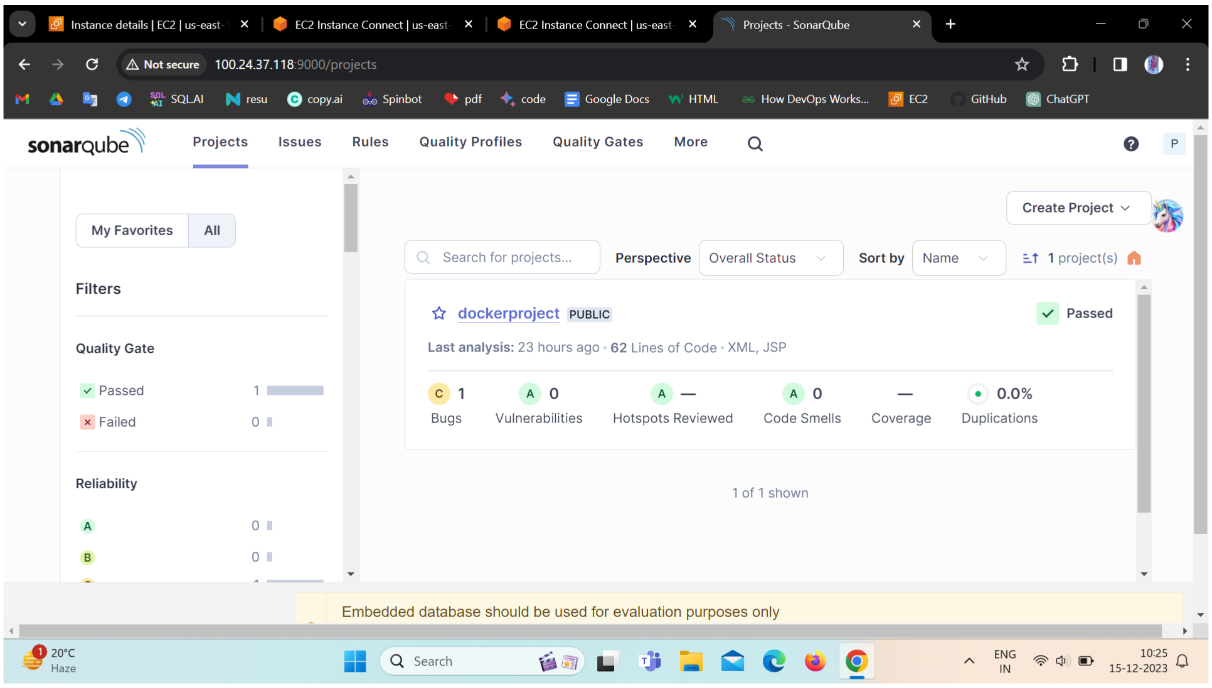
8. Click on the create project.

It generates Token name “dockerproject”, Expires in 30days, Click on Generate. 10. Run analysis on the project – select the maven it generates the maven code. mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject

Copy the maven script and add to Jenkins Pipeline.

```
pipeline{
    agent any
    stages{
        stage('clone'){
            steps{
                checkout scmGit(branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/Venna12/dockerjenkin.git']])
            }
        }
        stage('validate'){
            steps{
                sh 'mvn validate'
            }
        }
        stage('compile'){
            steps{
                sh 'mvn test'
            }
        }
        stage('package'){
            steps{
                sh 'mvn package'
            }
        }
        stage('SonarQube'){
            steps{
                sh "mvn clean verify sonar:sonar -Dsonar.projectKey=docker
-Dsonar.projectName='docker' -Dsonar.host.url=http://54.83.95.72:9000
-Dsonar.token=sqp_a103669de8c2ffb345368ffe04d1835455391934"
            }
        }
        stage('nexus'){
            steps{
                nexusArtifactUploader artifacts: [[artifactId:
'java-tomcat-maven-example', classifier: '', file:
'/var/lib/jenkins/workspace/dockerjenkin/target/java-tomcat-maven-examp
le.war', type: 'war']], credentialsId: 'nexus', groupId: 'com.example',
nexusUrl: '54.237.15.7:8081', nexusVersion: 'nexus3', protocol: 'http',
repository: 'maven-snapshots', version: '1.0-SNAPSHOT'
            }
        }
```

Code Quality analysis is build Successfully with the SonarQube.
It shows the Detect flaws, Bugs, Security Vulnerabilities, Code coverage, in the SonarQube.



Quality Gate

Passed	1
Failed	0

Reliability

A	0
B	0

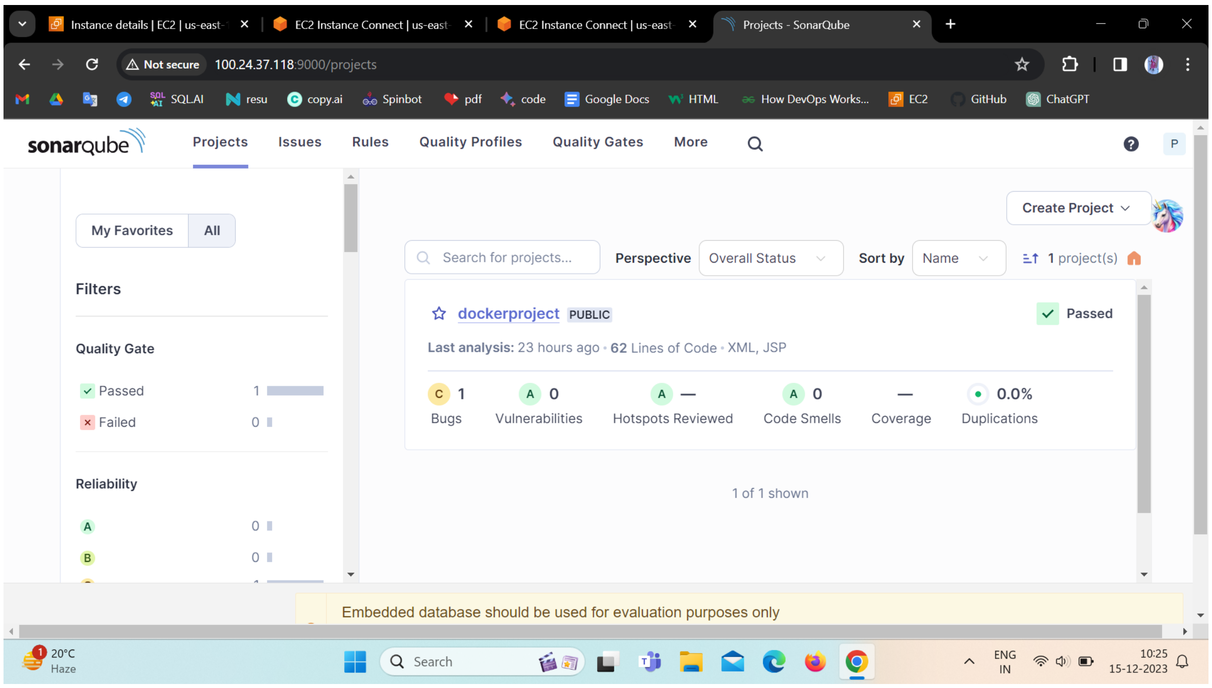
dockerproject PUBLIC — Passed

Last analysis: 23 hours ago • 62 Lines of Code • XML, JSP

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications
C 1	A 0	A —	A 0	—	0.0%

1 of 1 shown

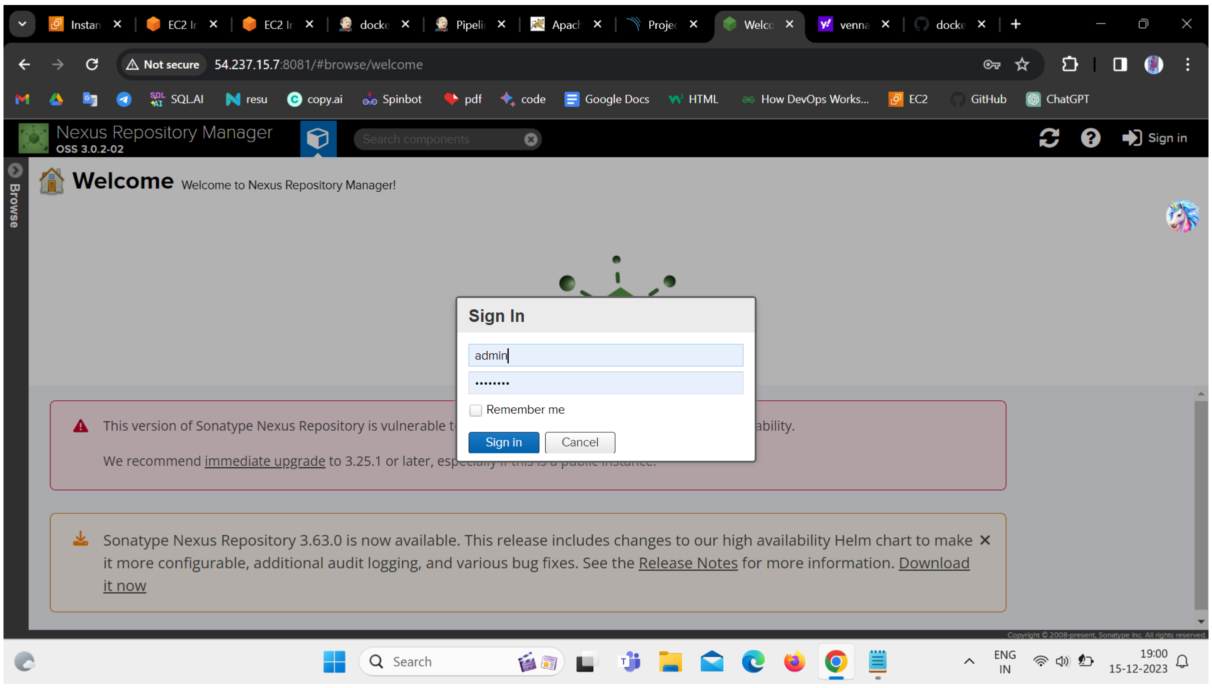
Embedded database should be used for evaluation purposes only



Code Quality Analysis with SonarQube.

Configuration of Nexus to Jenkins pipeline:

1. Open Nexus server using IP address with Port no.
   a. IP address:Port no (8081).
2. Login nexus server using admin credentials.
   a. Username: admin
   b. Password: admin123



This version of Sonatype Nexus Repository is vulnerable t ability.
We recommend immediate upgrade to 3.25.1 or later, especially if this is a public instance.

Sonatype Nexus Repository 3.63.0 is now available. This release includes changes to our high availability Helm chart to make it more configurable, additional audit logging, and various bug fixes. See the Release Notes for more information. Download it now

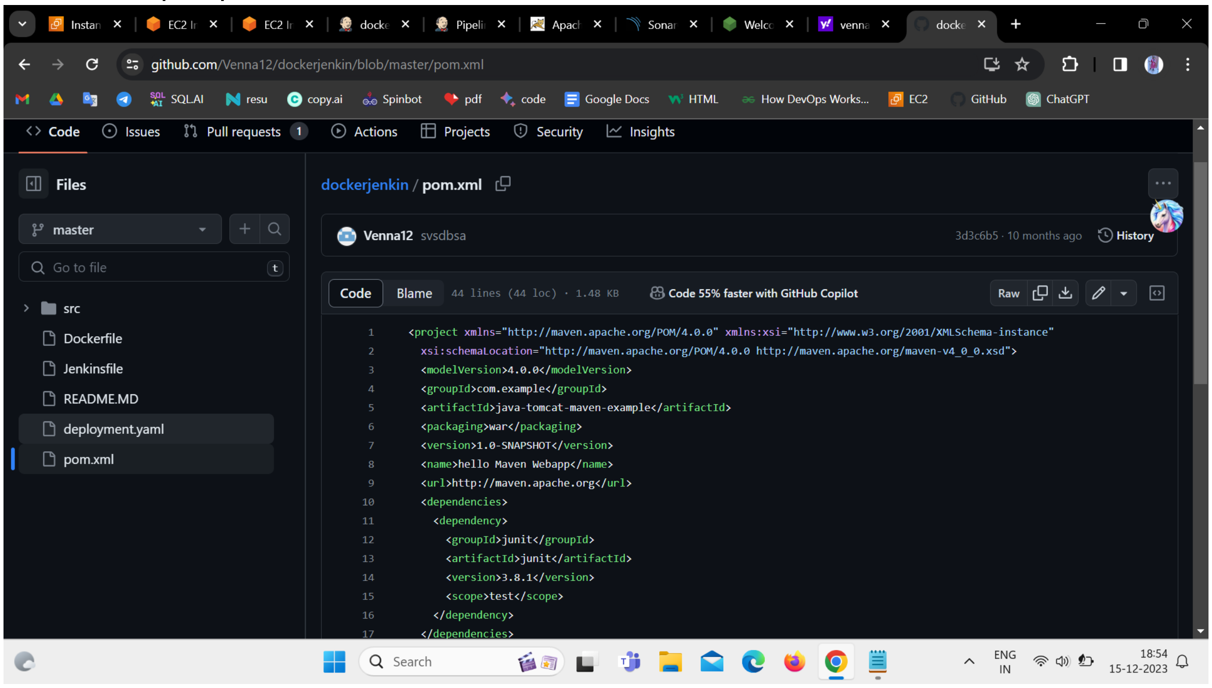
3. Open Components and Click on Maven-Snapshots.

4. Open Maven-snapshots repository.

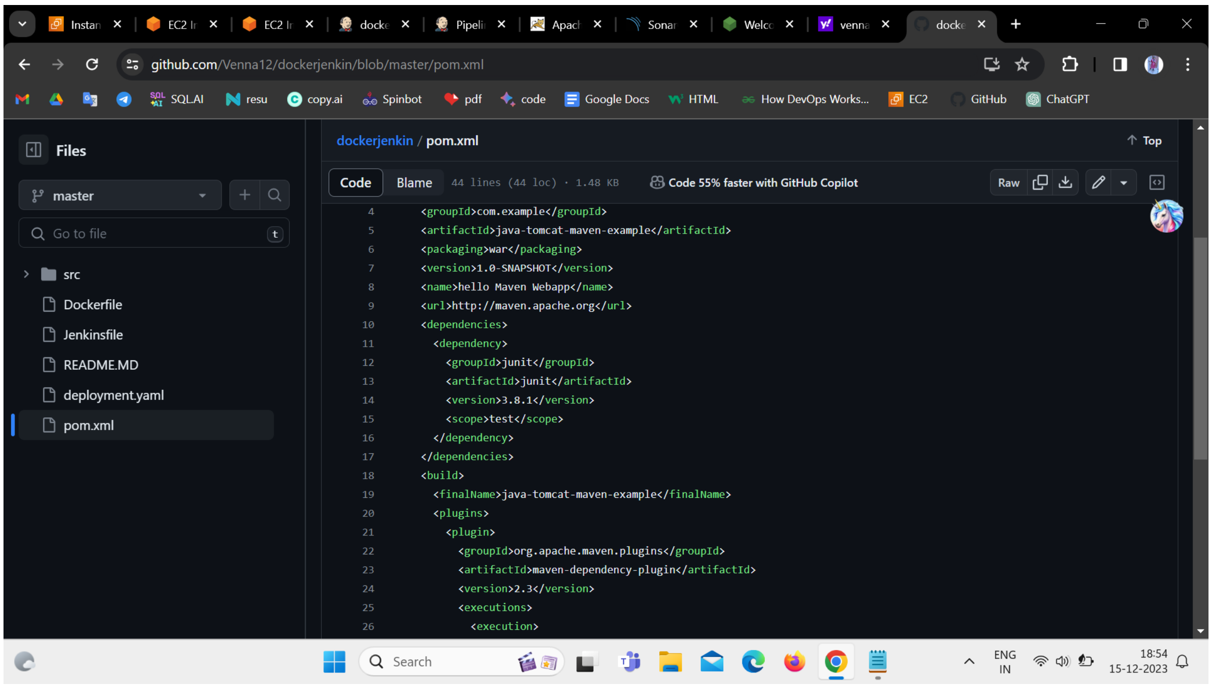
5. Open Jenkins pipeline and pipeline script.
   a. Add script, to pipeline after SonarQube.
6. Add script of maven to pipeline using stage.

```
stage('Package'){

 steps{

 sh 'mvn package'

 }

 }
```

7. Go to console output, after build success.
8. Copy Building war file from the console output.
   a.
      /var/lib/jenkins/workspace/dockerassgin/target/java-tomcat-maven example.war
9. Save it to notepad and open Jenkins pipeline.
10. Open pipeline syntax on other tab.
11. Open pom.xml file for the artifact information.



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>java-tomcat-maven-example</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>hello Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
```



dockerjenkin / pom.xml

```
<groupId>com.example</groupId>
<artifactId>java-tomcat-maven-example</artifactId>
<packaging>war</packaging>
<version>1.0-SNAPSHOT</version>
<name>hello Maven Webapp</name>
<url>http://maven.apache.org</url>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <finalName>java-tomcat-maven-example</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>2.3</version>
      <executions>
        <execution>
```

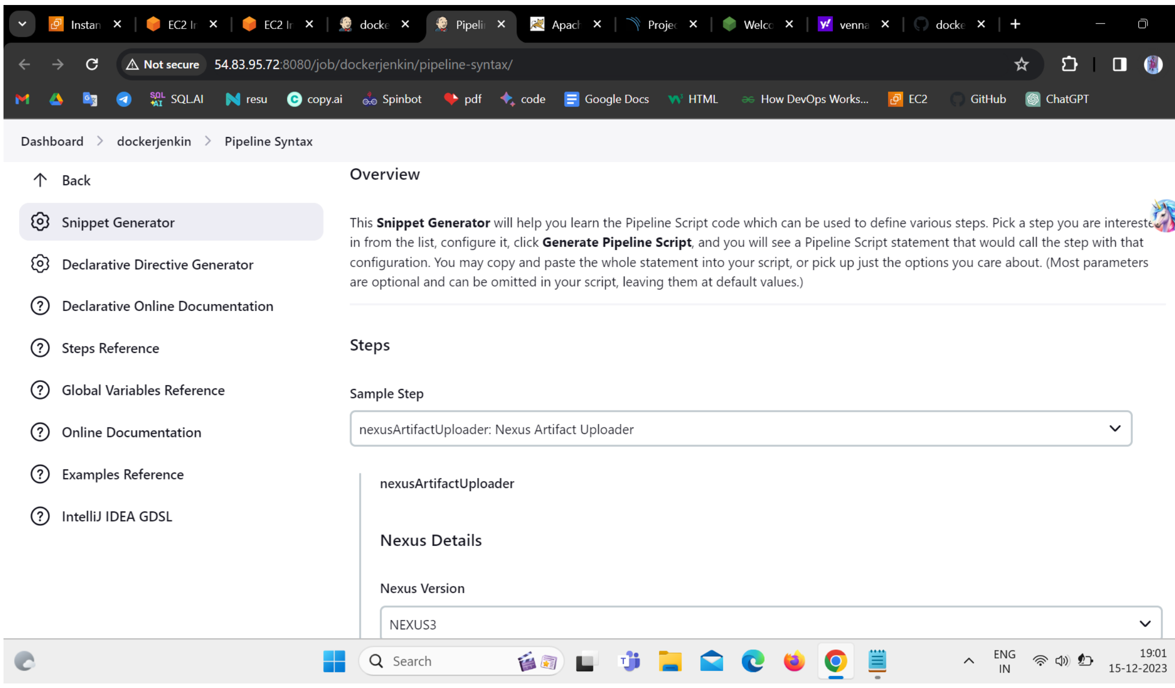
Generate Pipeline and copy the code

. Paste the generated pipeline syntax script to the pipeline.

stage('Store in nexus artifacts'){

steps{

nexusArtifactUploader artifacts: [[artifactId: 'java-tomcat-maven example', classifier: '', file:



Dashboard > dockerjenkin > Pipeline Syntax

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

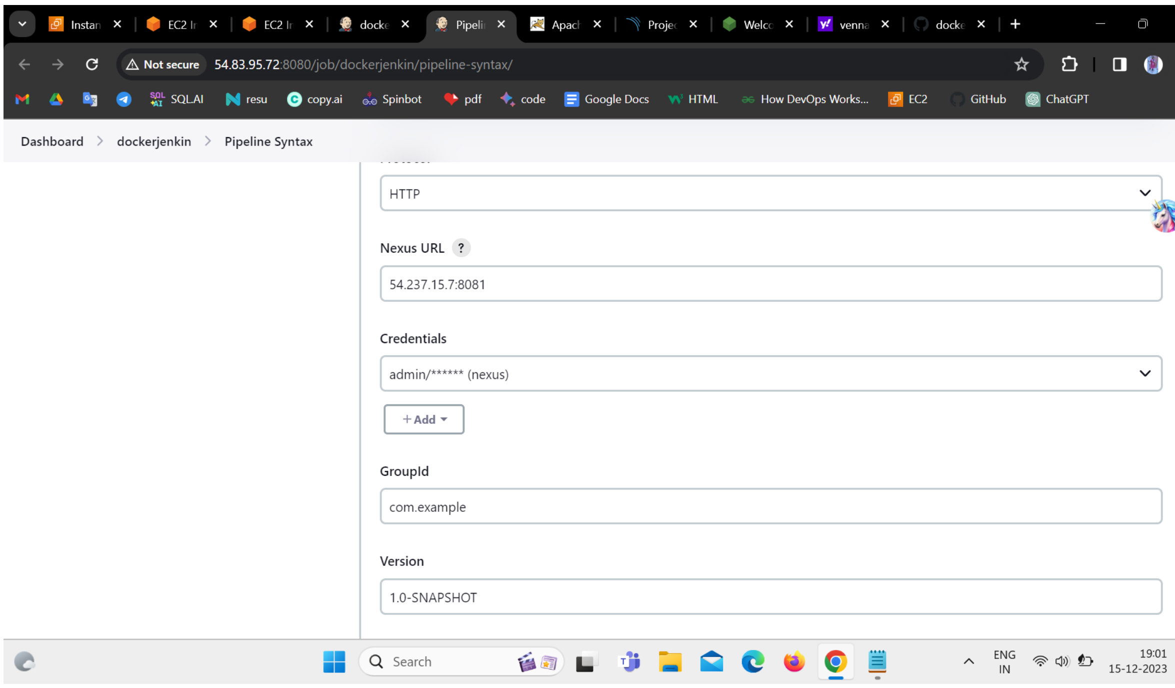
Steps

Sample Step: nexusArtifactUploader: Nexus Artifact Uploader

nexusArtifactUploader

Nexus Details

Nexus Version: NEXUS3



Dashboard > dockerjenkin > Pipeline Syntax

HTTP

Nexus URL ?

54.237.15.7:8081

Credentials

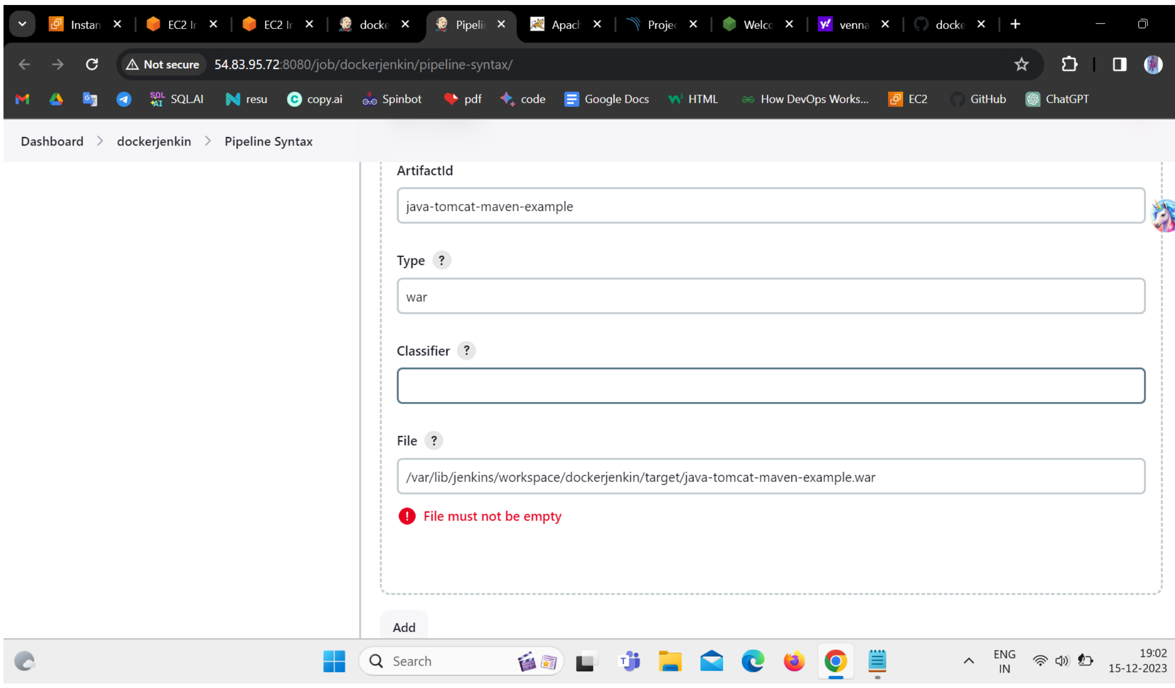
admin/****** (nexus)

GroupId

com.example

Version

1.0-SNAPSHOT



Dashboard > dockerjenkin > Pipeline Syntax

ArtifactId

java-tomcat-maven-example

Type ?

war

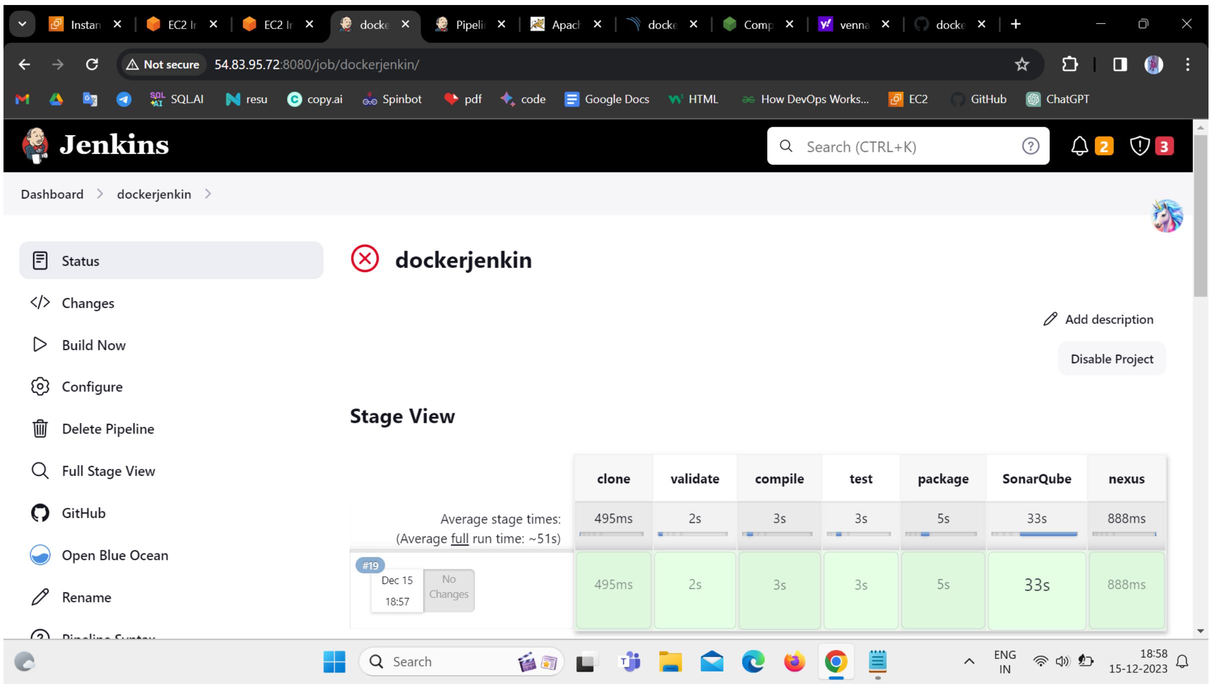
Classifier ?

File ?

/var/lib/jenkins/workspace/dockerjenkin/target/java-tomcat-maven-example.war

File must not be empty

Pipeline Script for the Nexus Artifact in Jenkins.



dockerjenkin

Stage View

	clone	validate	compile	test	package	SonarQube	nexus
Average stage times: (Average full run time: ~51s)	495ms	2s	3s	3s	5s	33s	888ms
#19 Dec 15 18:57 No Changes	495ms	2s	3s	3s	5s	33s	888ms

Successfully build:

Artifact is stored in the maven-snapshots as java-tomcat-maven-example.

Deploying project into Apache Tomcat using Jenkins:

1. Open the Apache Tomcat Server using IP Address with Port no.
   a. IP address:Port no (8085).
2. Login to Tomcat using
   a. Username: Tomcat
   b. Password: s3cret
3. Install plugin for the deployment in the tomcat.

   ^ Deploy Container

To get tomcat script, we use pipeline syntax

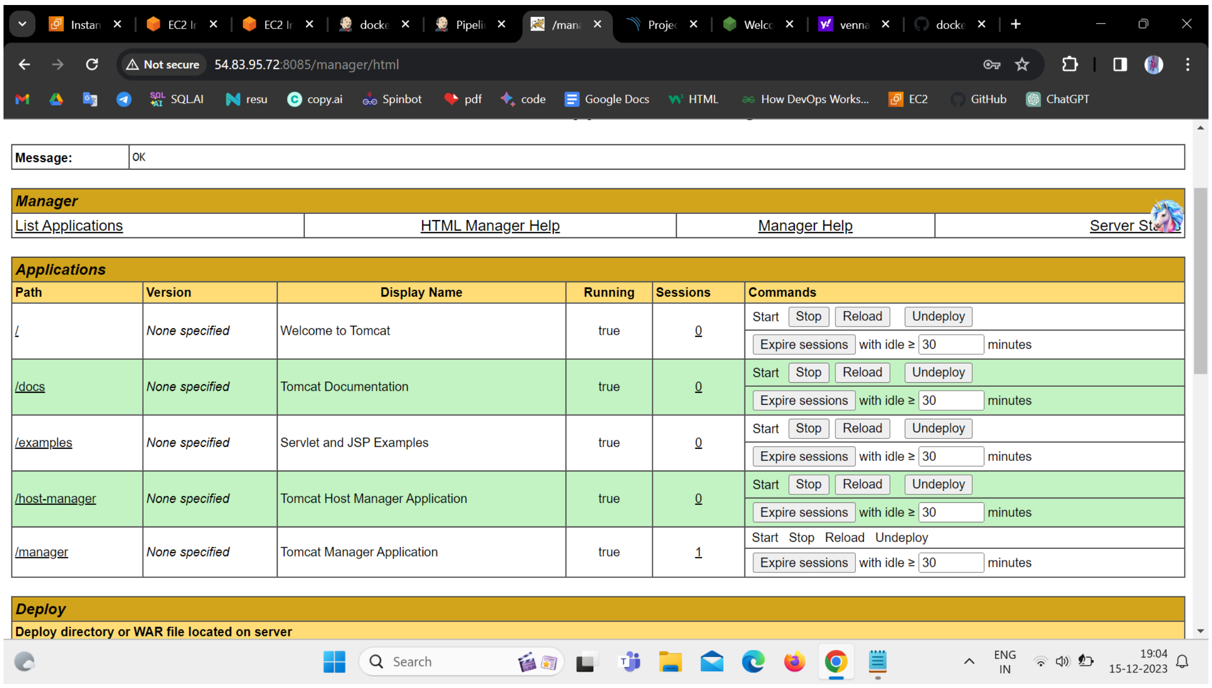
4. Create Jenkins id and password, it is add to the container.

5. Open the Tomcat and host manager. It seems like that

6. Copy the URL of the Tomcat with Ip address and port number. Copy the link address to the given box and we got the script and copy the script the and add to the script to configure pipeline.

The project is Successfully Deployed in the tomcat.

9. Open the Tomcat server, we can see the DockerProject in the Tomcat Dashboard.



Not secure 54.83.95.72:8085/manager/html
SQL.AI resu copy.ai Spinbot pdf code Google Docs HTML How DevOps Works... EC2 GitHub ChatGPT
Search
ENG IN
19:04 15-12-2023

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server St

Applications

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

Deploy

Deploy directory or WAR file located on server

THE -END