Course: DevOps Name: ch . lakshmi priyanka

Module: Docker & Docker Hub Batch no: 115

Topic: SonarQube, Nexus, Tomcat, Docker Assignment no: 05

Trainer Name: Mr. Madhukar sir Date of submission: 30 – dec – 2023

Mail-ID: chlakshmipriyanka9@gmail.com

Assignment: SonarQube do code quality analysis, Store in nexus artifacts and Deploy in tomcat using project GitHub link, Build and Push Project in the DockerHub: <https://github.com/Venna12/dockerjenkin.git>

Pre-requirements:

1. Project link from GitHub (Venna12).
2. Install Java jdk-11 and maven.
3. Jenkins and setup Jenkins (Install require plugs).
4. Install SonarQube and setup environment.
5. Install Nexus Artifacts and setup environment.
6. Install Tomcat and setup environment.

----------------------------- **EC2 Connect** -----------------------------

* Launch the EC2 instances, take instance type (t2.medium or t2.large), select Ubuntu OS for the project and wait the instances to change status 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 the Google.
* Connect the cloud command line interface.
* Change normal user to root user (using sudo -i).
* Update the Linux server using command (apt update -y).

-------------------------- **Install Java jdk-11 and Maven** --------------------------

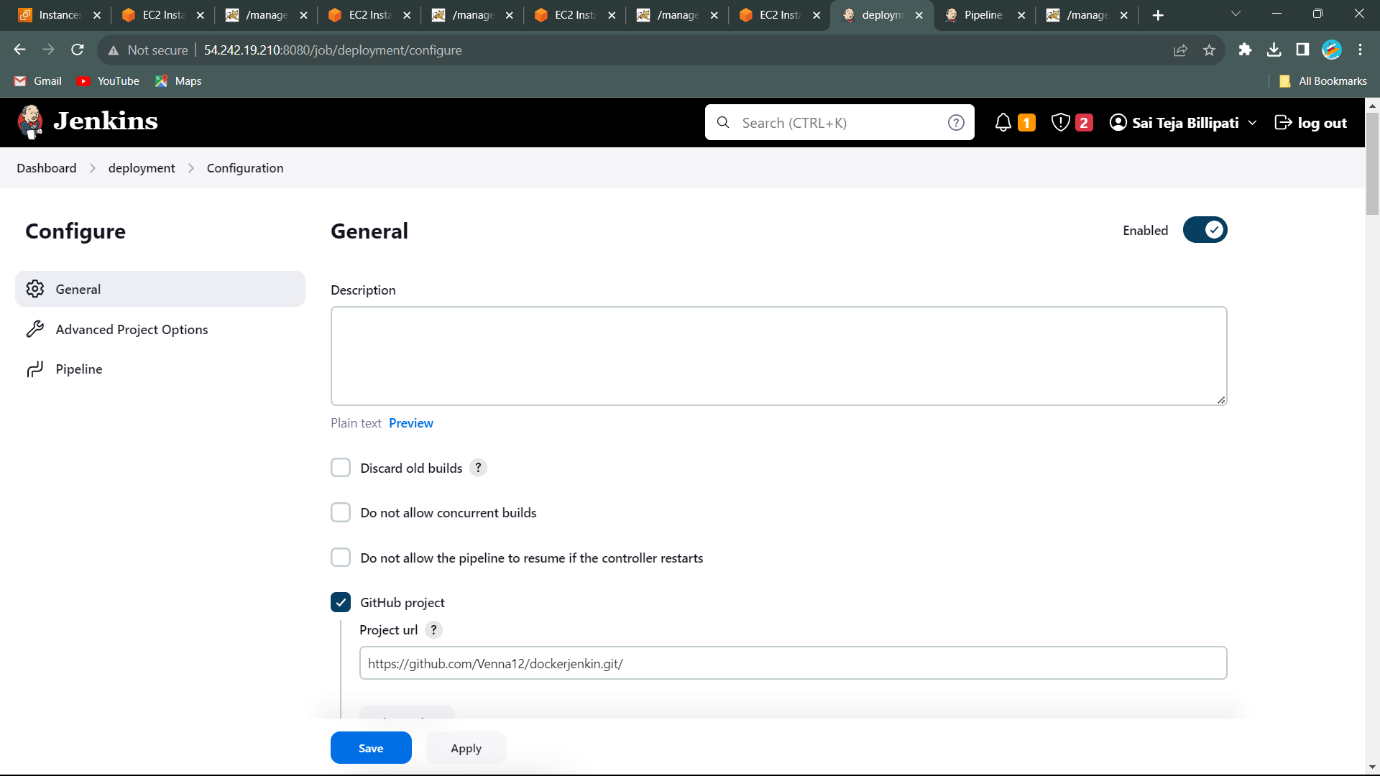
-🡪 Use the command to install Java jdk-11 in the server (sudo apt install default-jdk -y or apt install openjdk-11-jdk -y).

-🡪 Use the command to install Maven in the server (sudo apt install maven -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 pipeline. Job name Dockerassgin.
* We got configure dashboard, and click on the Git, add the clone link to given box.



**Project URL**: <https://github.com/Venna12/dockerjenkin.git>

-------------------------------- **SonarQube** --------------------------------

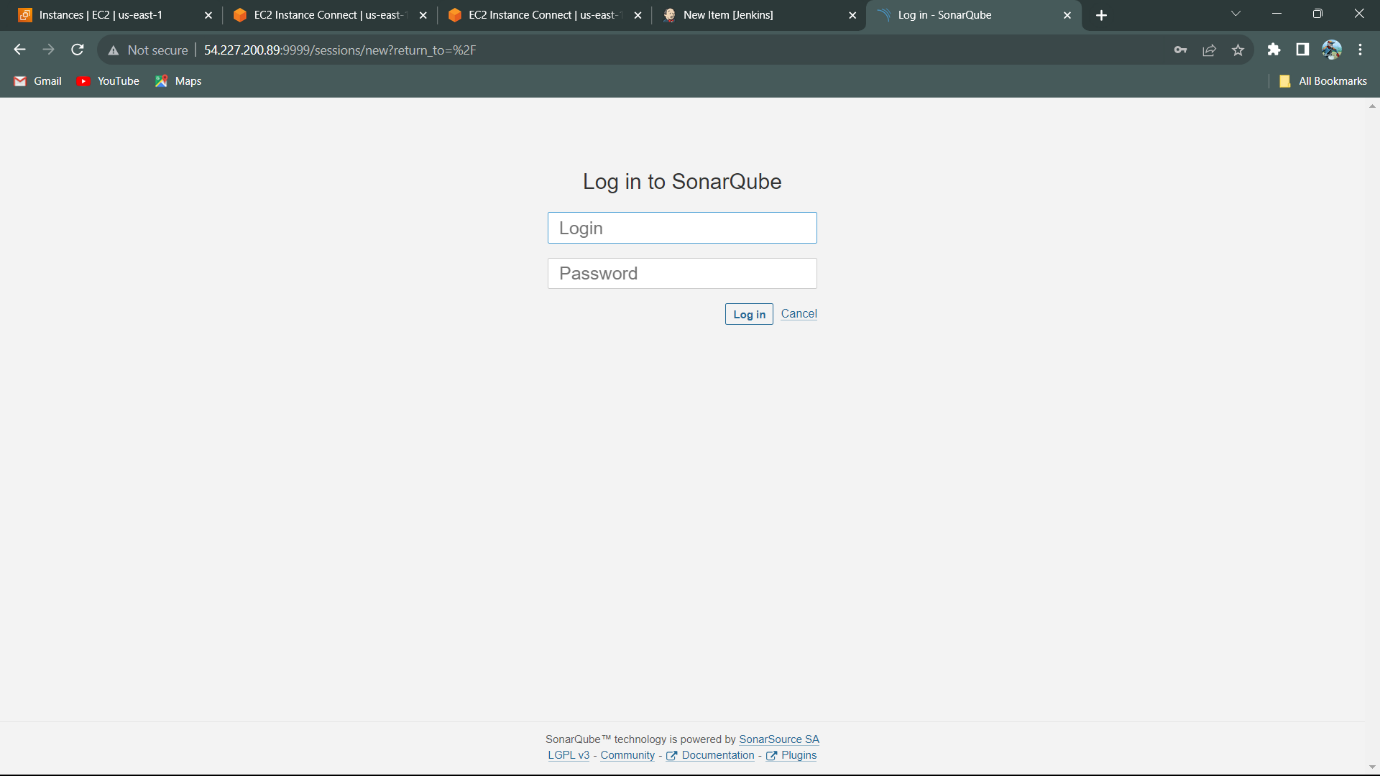
* SonarQube is a static code analysis tool, a tool that scans your code and tries to detect flaws, bugs, security vulnerabilities, etc. It can also measure test coverage of your code if provided with proper reports. All these features focus on direct code development and help developers build better 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 on 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



Interface of the SonarQube for Log in to SonarQube.

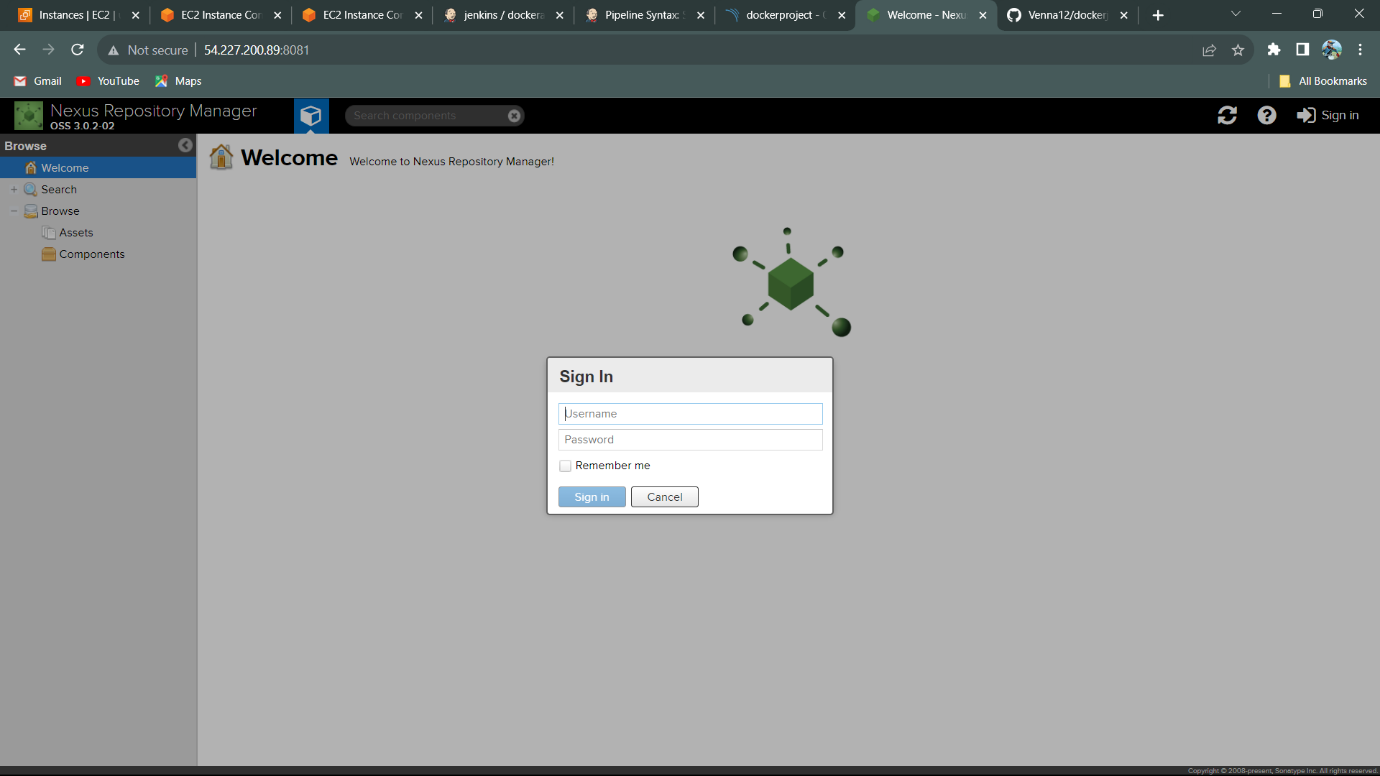
-------------------------------- **Nexus** --------------------------------

* 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 pure written in the Java. But it can store the 29 other programming languages artifacts in it.
* Nexus is run on Java jdk-8 version only.
* To run the Nexus use server IP address and Nexus Port no: 8081.

Server Ip:8081 to open Nexus on the any search engine.

Default 🡪 Username: admin

Password: admin123.



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

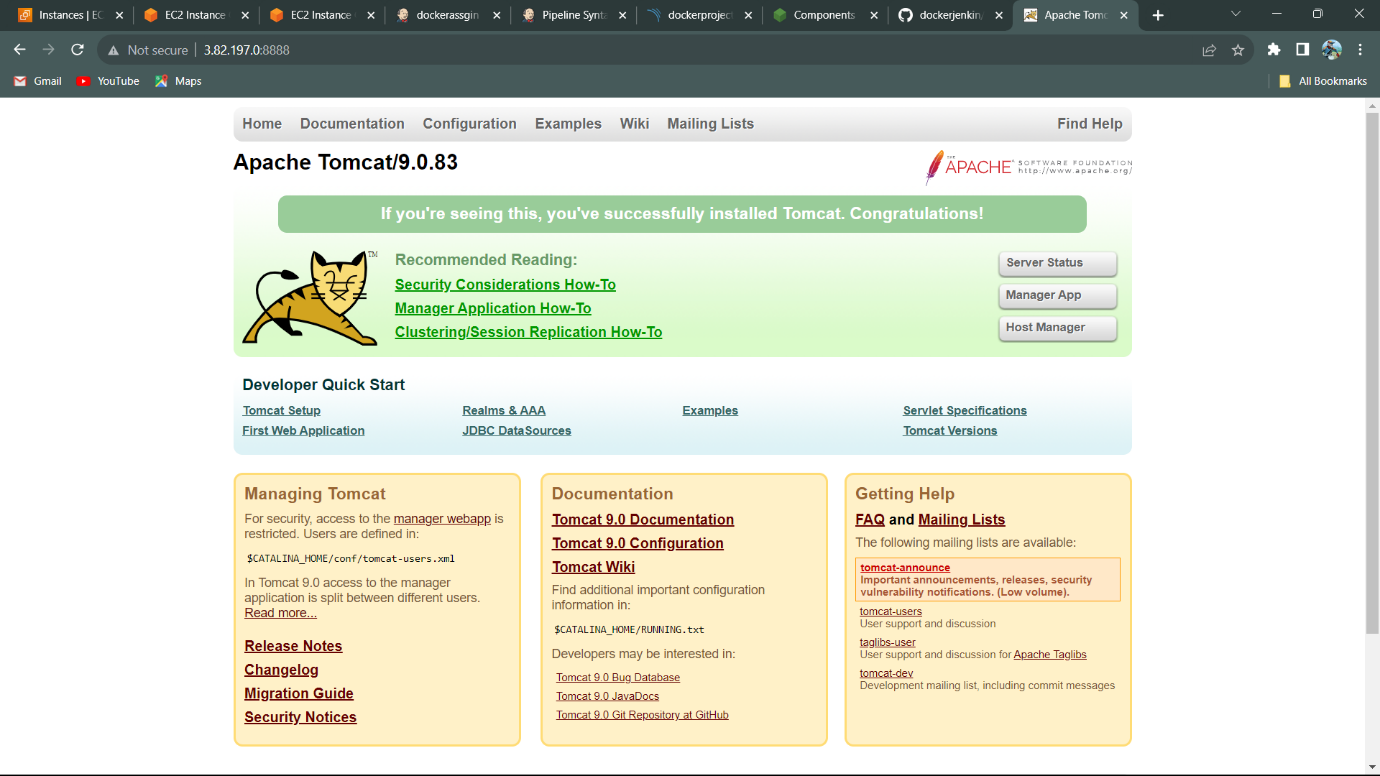
Login details of the Apache Tomcat server.

>>>Login: tomcat

>>>Password: s3cret

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

🡪 IP Address:8080 (Port no).



User Interface of the Apache Tomcat/9.0.83.

**-🡪 Write a pipeline script for the Git clone for the project using link**

**Project URL**: <https://github.com/Venna12/dockerjenkin.git>

1. Create a pipeline for Dockerjenkin:
   1. Clone the Project
   2. Validate the Project
   3. Compile the Project
   4. 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{

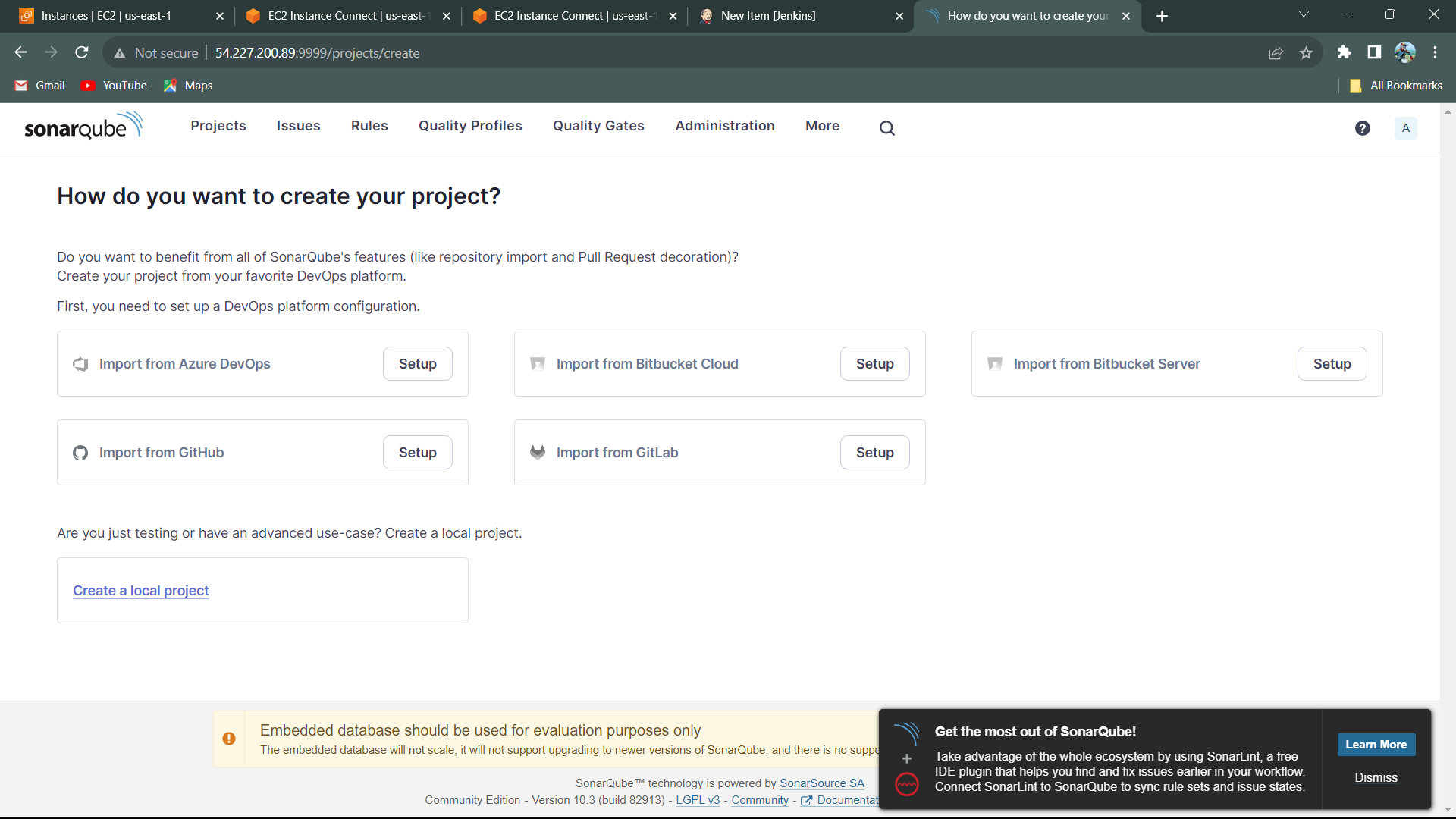
sh 'mvn test'

}

}

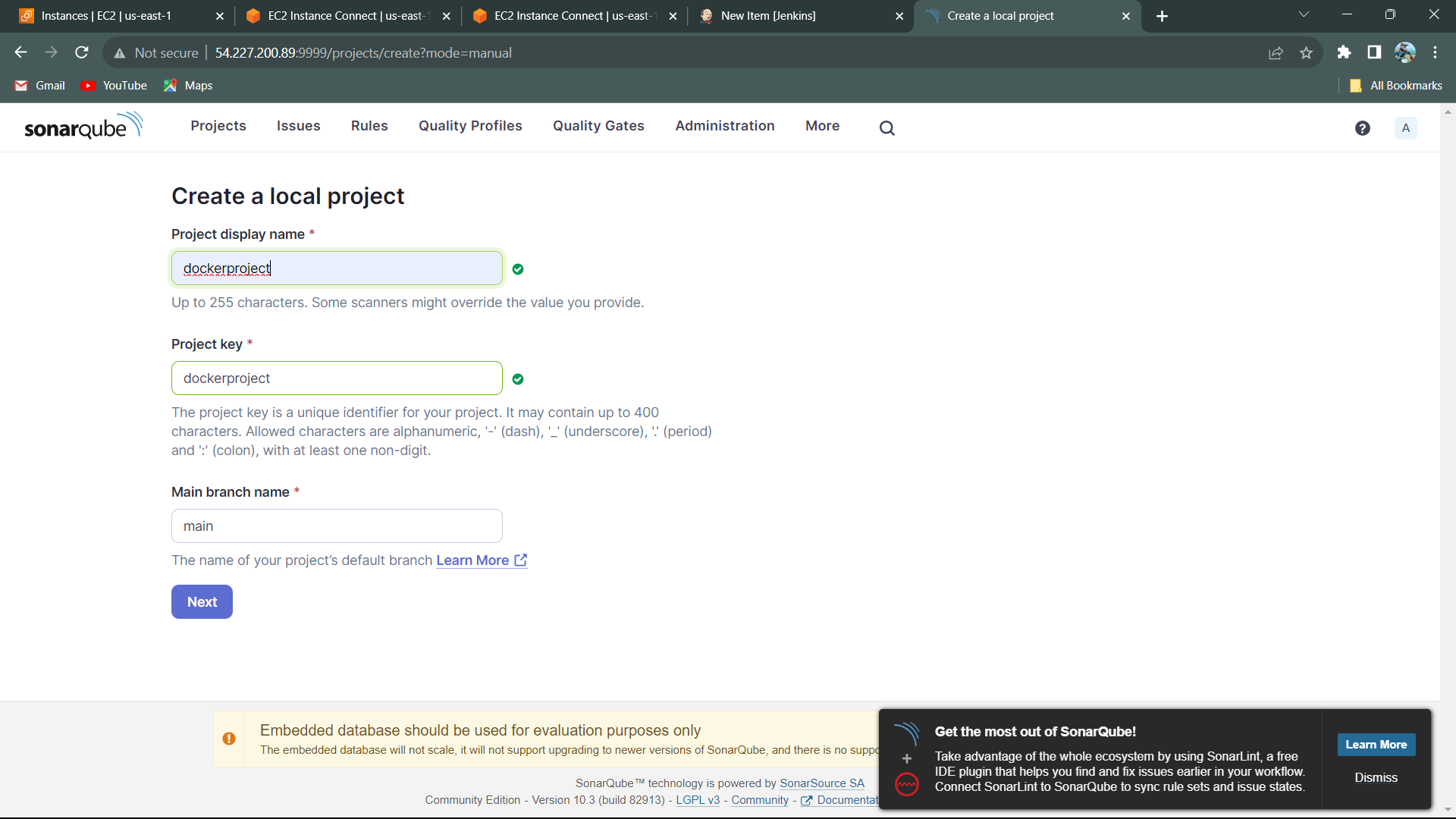
**Configuration of SonarQube to Jenkins pipeline:**

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

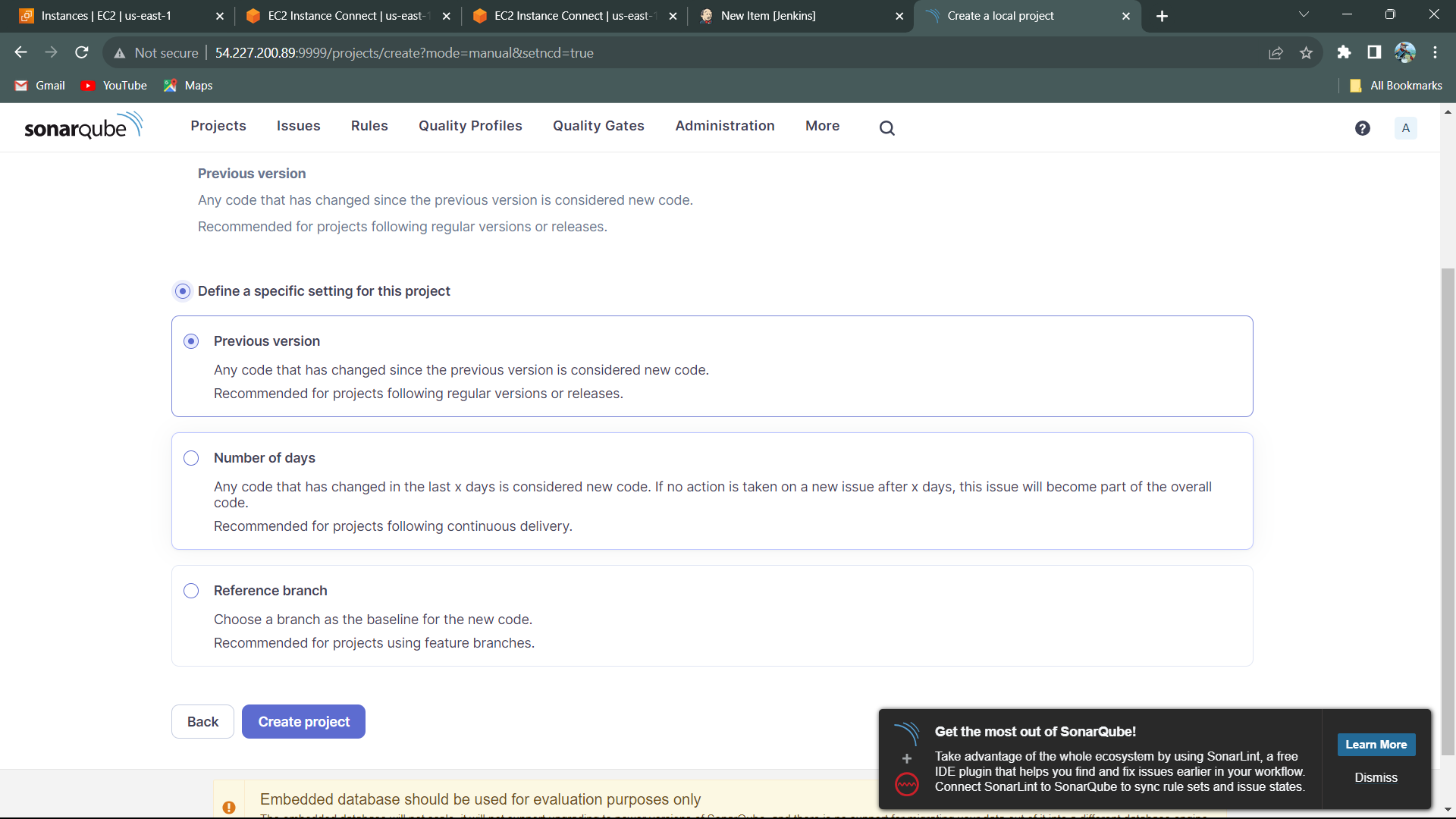


SonarQube Dashboard

1. Click on the Create a local project.
   1. Project display name: dockerproject
   2. Project key: dockerproject
   3. Main branch name: main



1. It shows different options for the options
   1. Previous version
   2. Number of days
   3. Reference branch
2. Select the Previous version for this project
3. Click on the create project.



1. It generates Token name “dockerproject”, Expires in 30days, Click on Generate.
2. Run analysis on the project – select the maven it generates the maven code.

mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject

-Dsonar.projectName='dockerproject'

-Dsonar.host.url=http://54.227.200.89:9999

-Dsonar.token=sqp\_7e026e6c571382aca2856db896ab7d99a5d9dc52

1. Copy the maven script and add to Jenkins Pipeline.

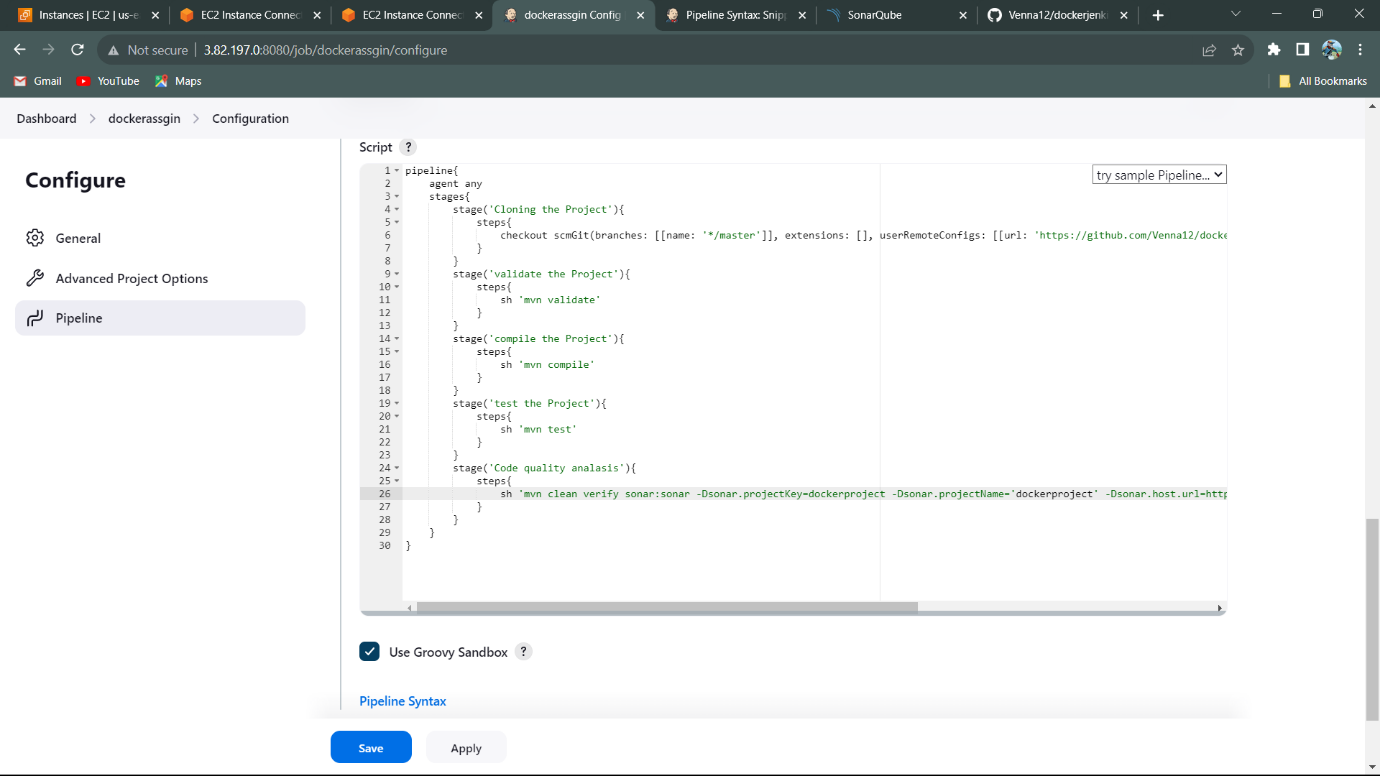
stage('Code quality analasis'){

steps{

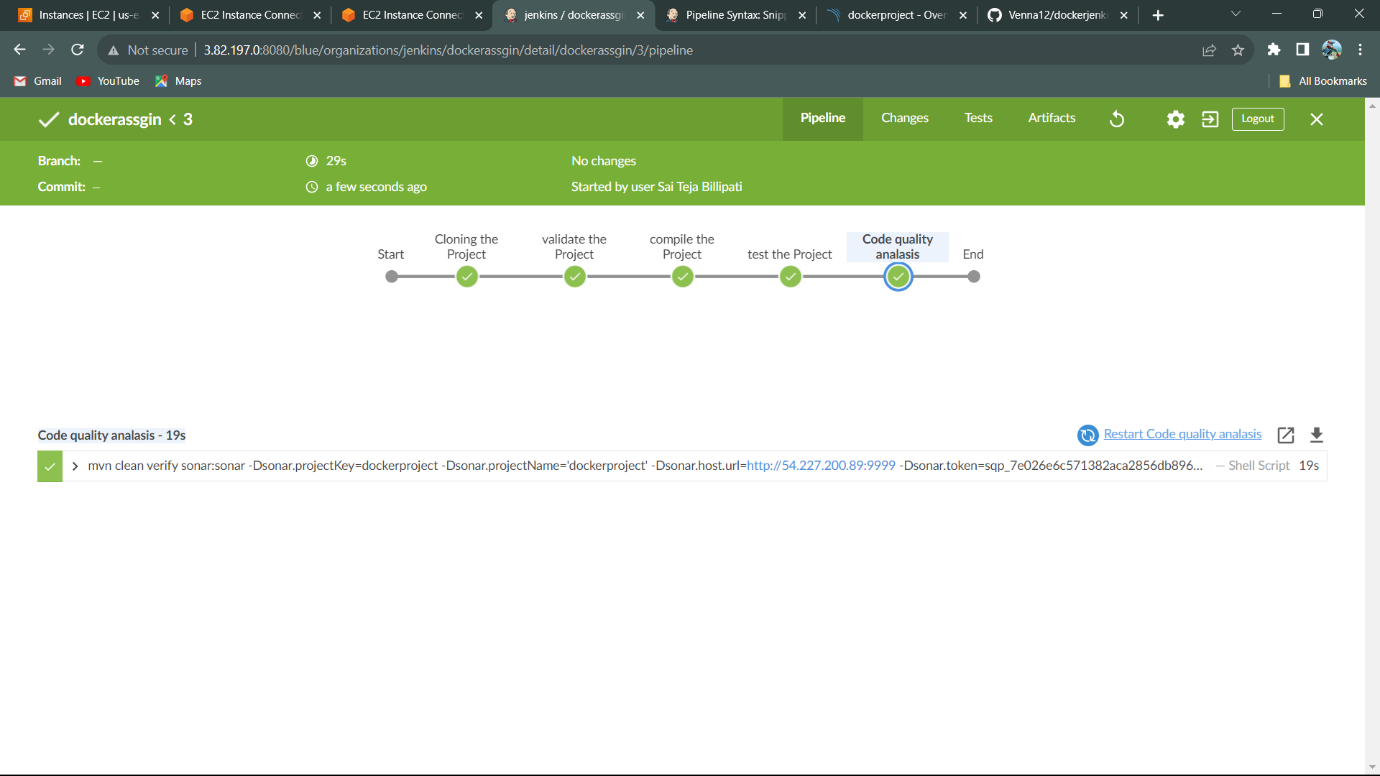
sh "mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject -Dsonar.projectName='dockerproject' -Dsonar.host.url=http://54.227.200.89:9999 -Dsonar.token=sqp\_7e026e6c571382aca2856db896ab7d99a5d9dc52"

}

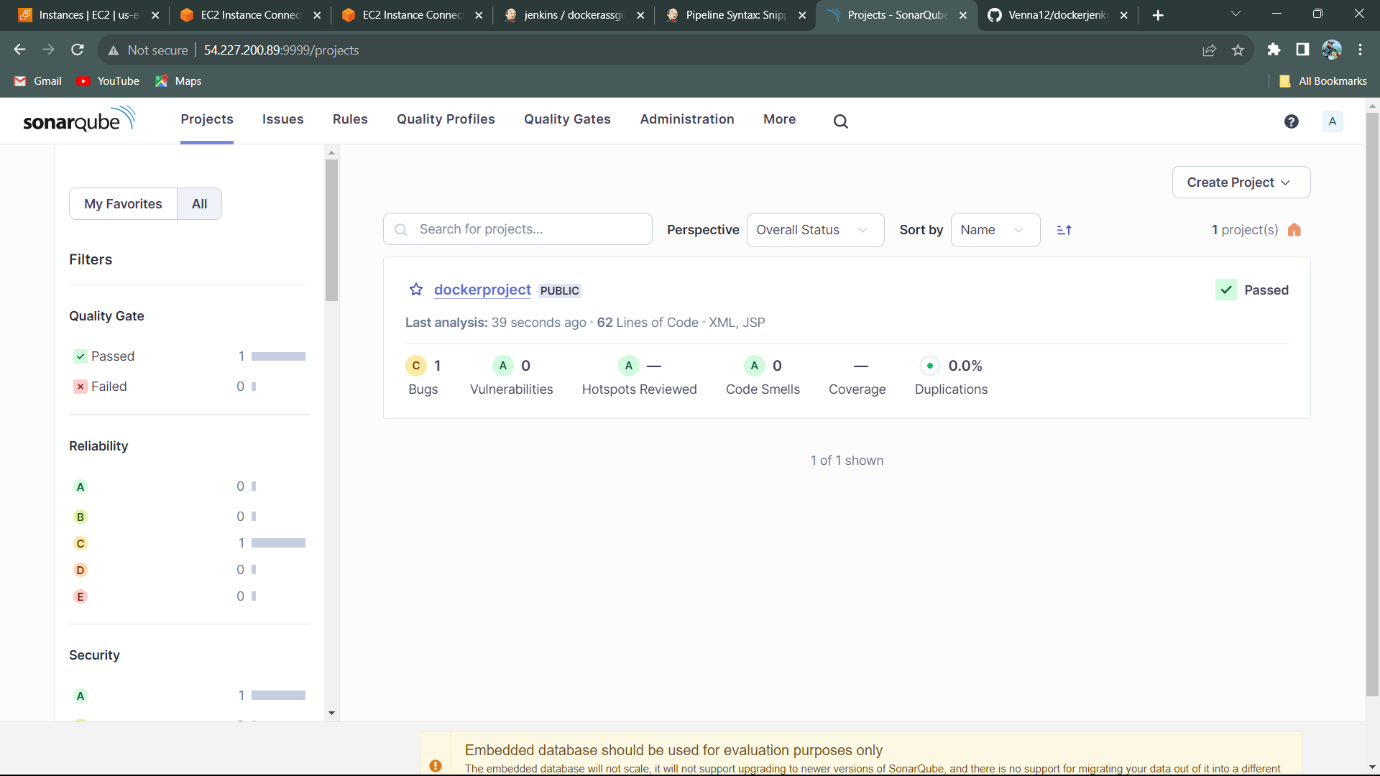
}



1. Apply and save the pipeline. Build the Pipeline.



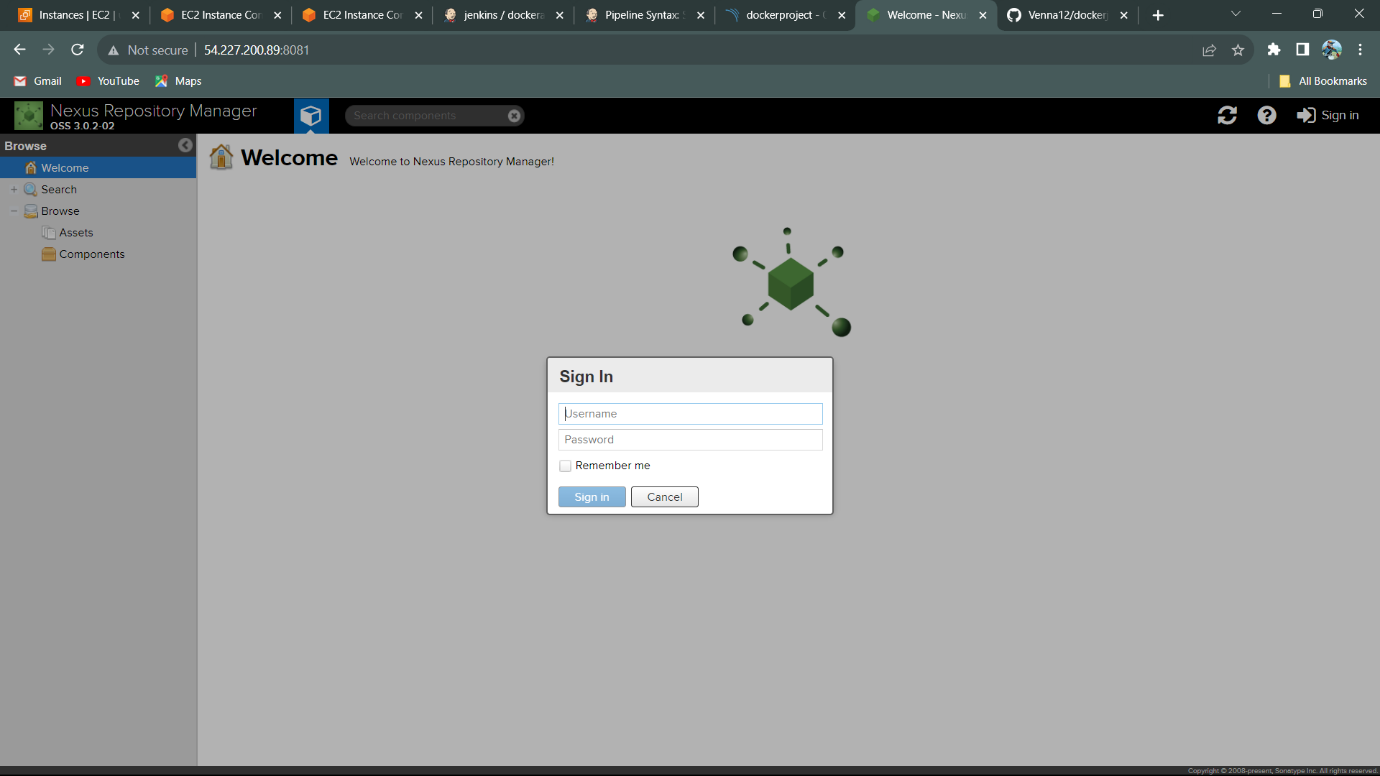
1. Code Quality analysis is build Successfully with the SonarQube.
2. It shows the Detect flaws, Bugs, Security Vulnerabilities, Code coverage, in the SonarQube.

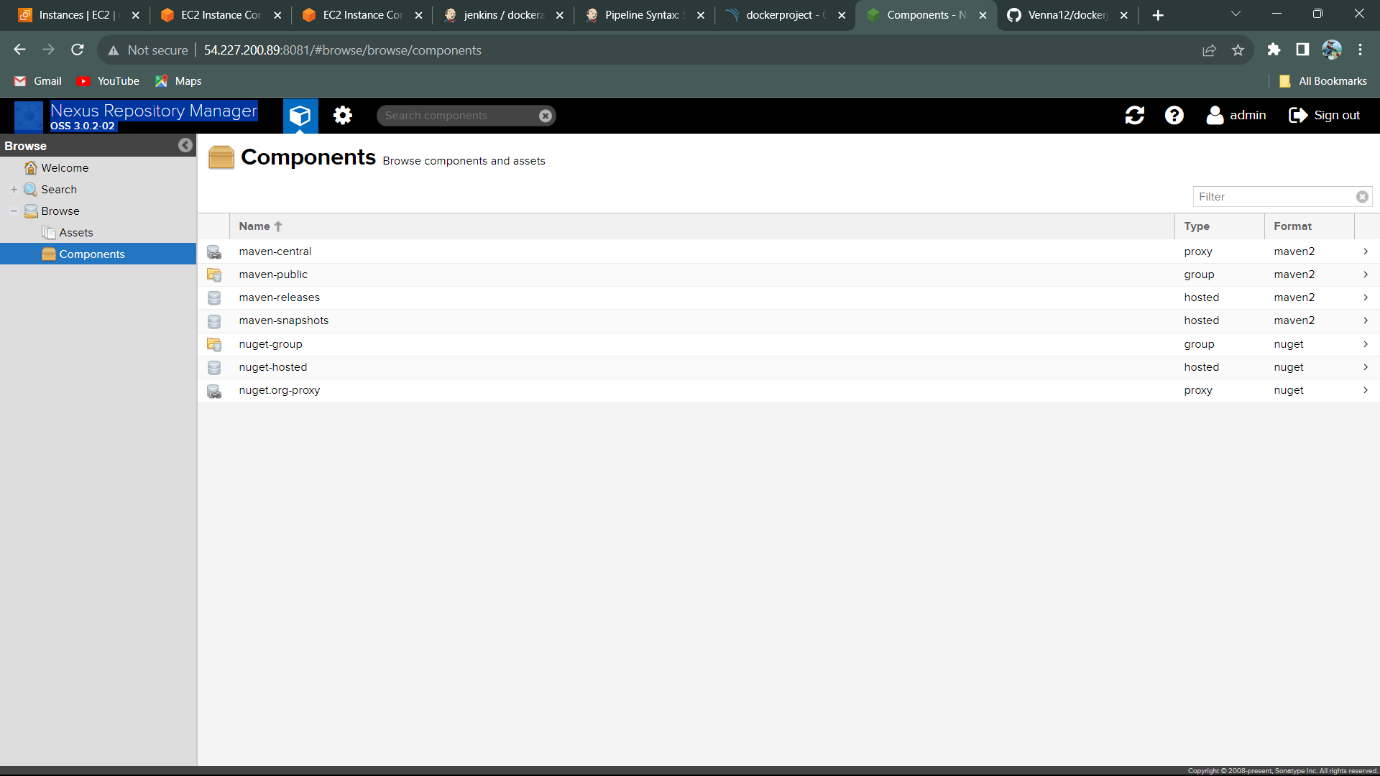


Code Quality Analysis with SonarQube.

**Configuration of Nexus to Jenkins pipeline:**

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



1. Open Components and Click on Maven-Snapshots.
2. Open Maven-snapshots repository.

1. Open Jenkins pipeline and pipeline script.
   1. Add script, to pipeline after SonarQube.
2. Add script of maven to pipeline using stage.

stage('Package'){

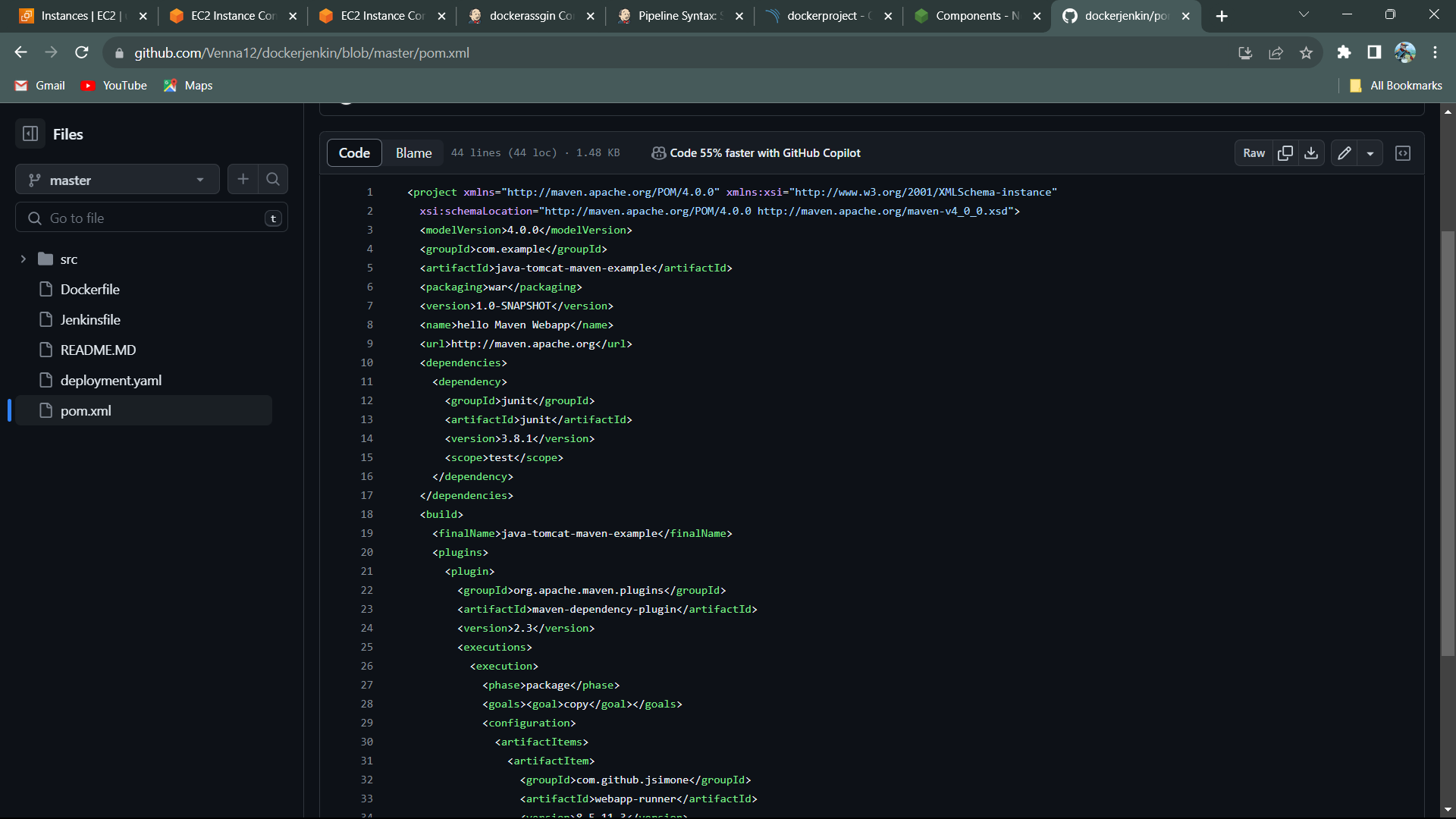
steps{

sh 'mvn package'

}

}

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



Pom.xml file

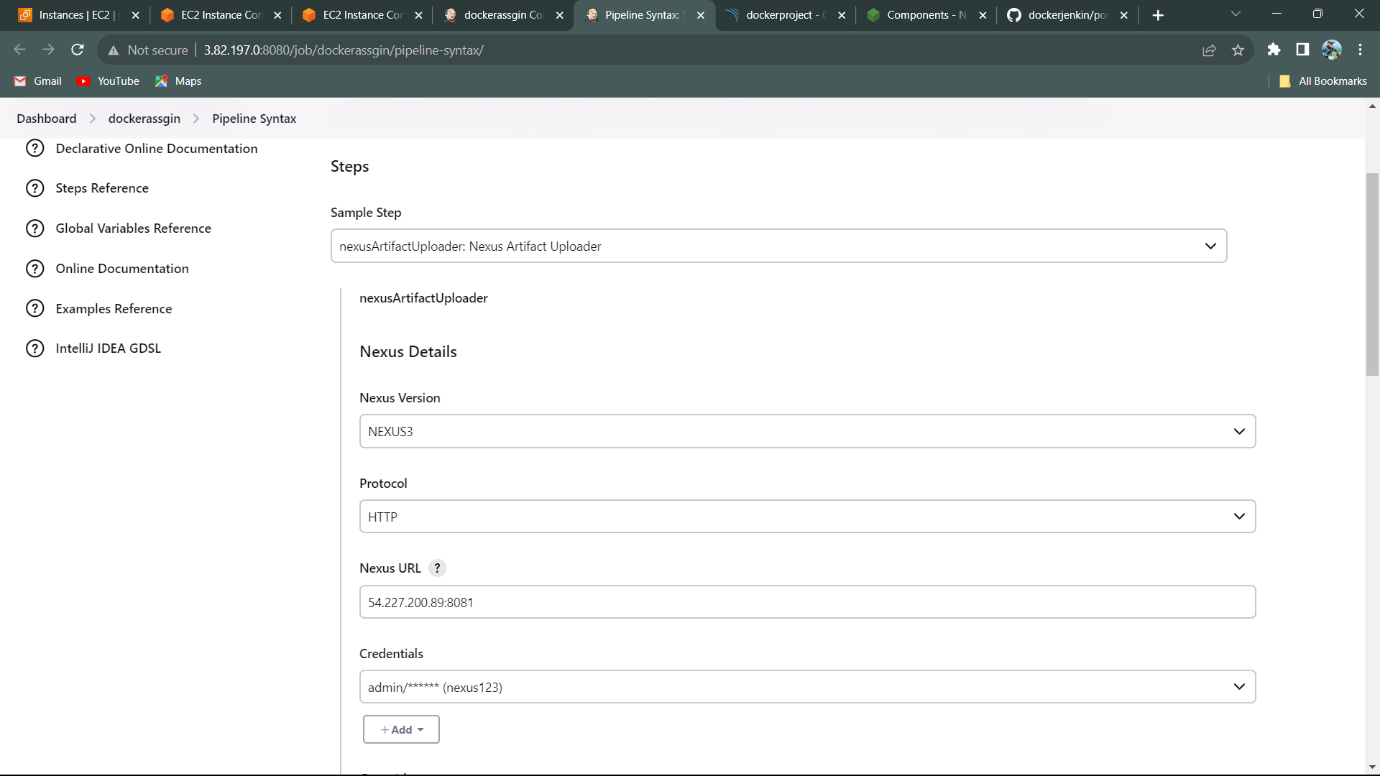
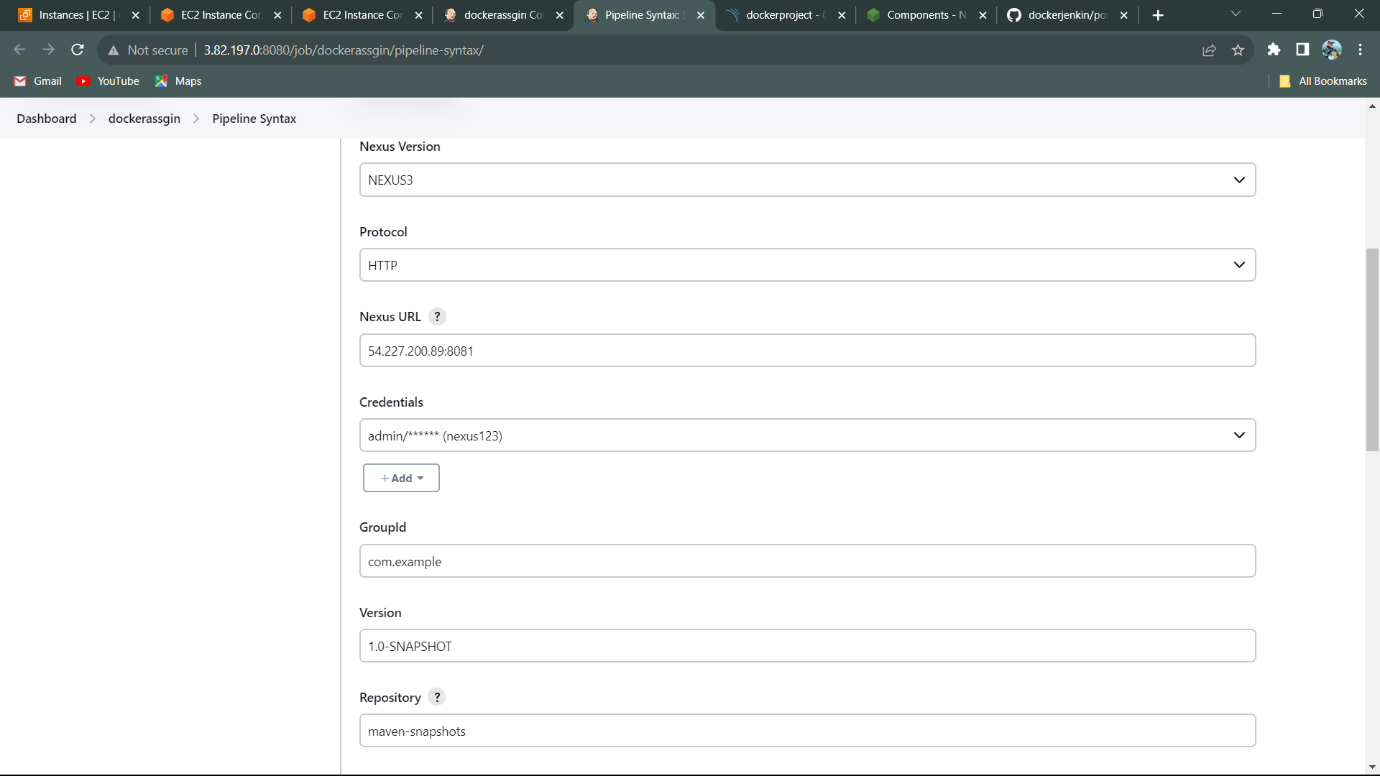
1. Open pipeline syntax.
   1. Sample step
      1. nexusArtifactUploader: Nexus Artifact Uploader
   2. Nexus Version
      1. NEXUS 3
   3. Protocol
      1. HTTP
   4. Nexus URL
      1. 54.227.200.89:8081
   5. Credentials
      1. Create Jenkins for credentials using
         1. Username: admin
         2. Password: admin123
   6. GroupId
      1. Com.example
   7. Version
      1. 1.0-SNAPSHOT
   8. Repository
      1. Maven-snapshots (Repository from the Nexus)
   9. Add Artifact
   10. Artifact Id
       1. Java-tomcat-maven-example
   11. Type
       1. War
   12. File
       1. /var/lib/jenkins/workspace/dockerassgin/target/java-tomcat-maven-example.war (Building war from console output)
2. Generate Pipeline and copy the code
3. Paste the generate pipeline syntax script to pipeline.

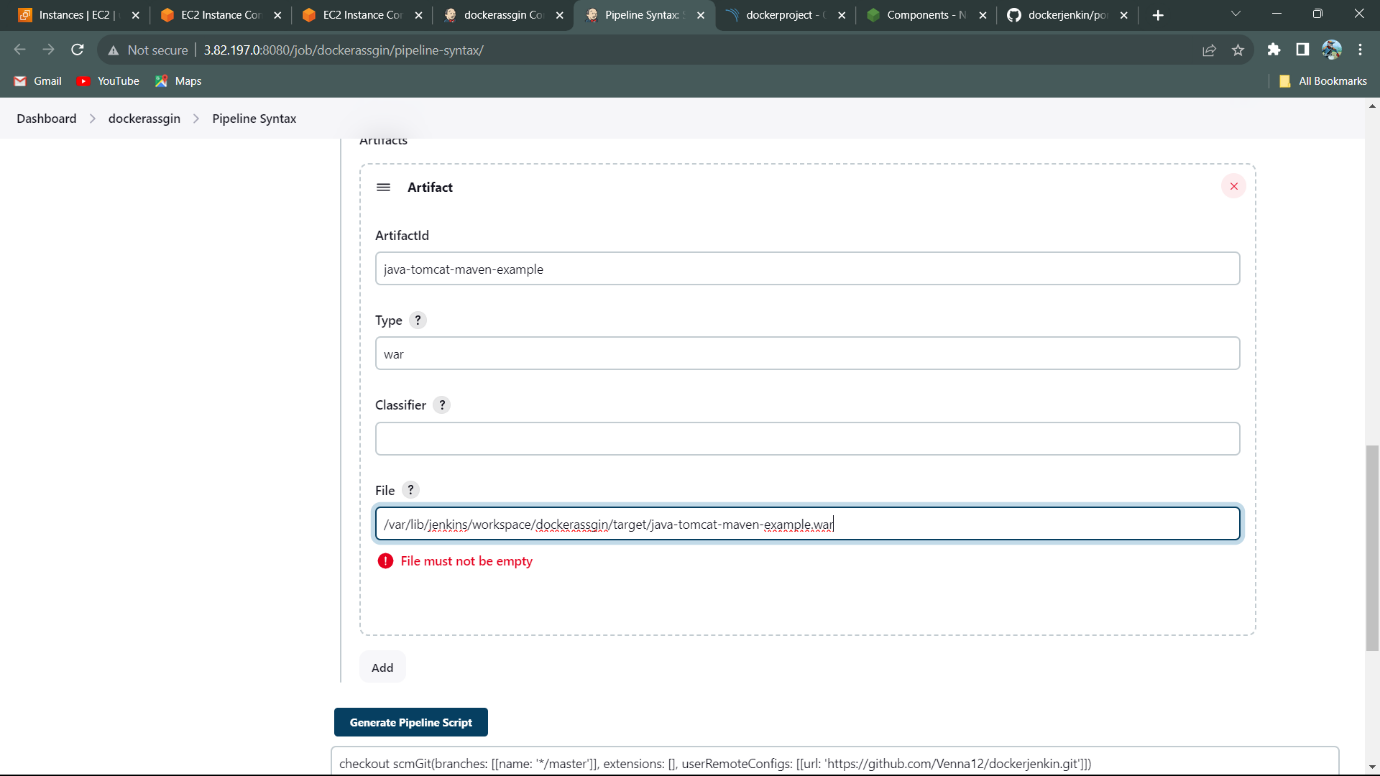
stage('Store in nexus artifacts'){

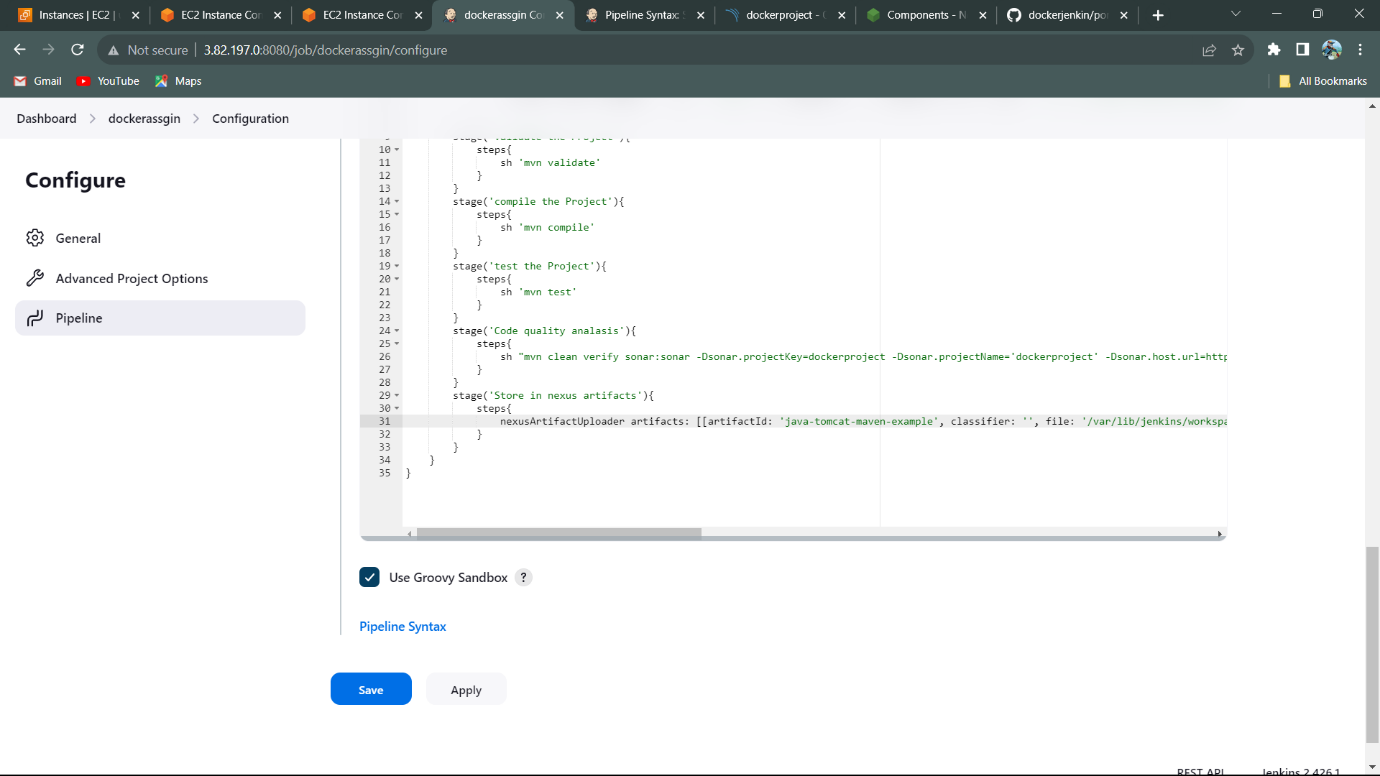
steps{

nexusArtifactUploader artifacts: [[artifactId: 'java-tomcat-maven-example', classifier: '', file: '/var/lib/jenkins/workspace/dockerassgin/target/java-tomcat-maven-example.war', type: 'war']], credentialsId: 'nexus123', groupId: 'com.example', nexusUrl: '54.227.200.89:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots', version: '1.0-SNAPSHOT'

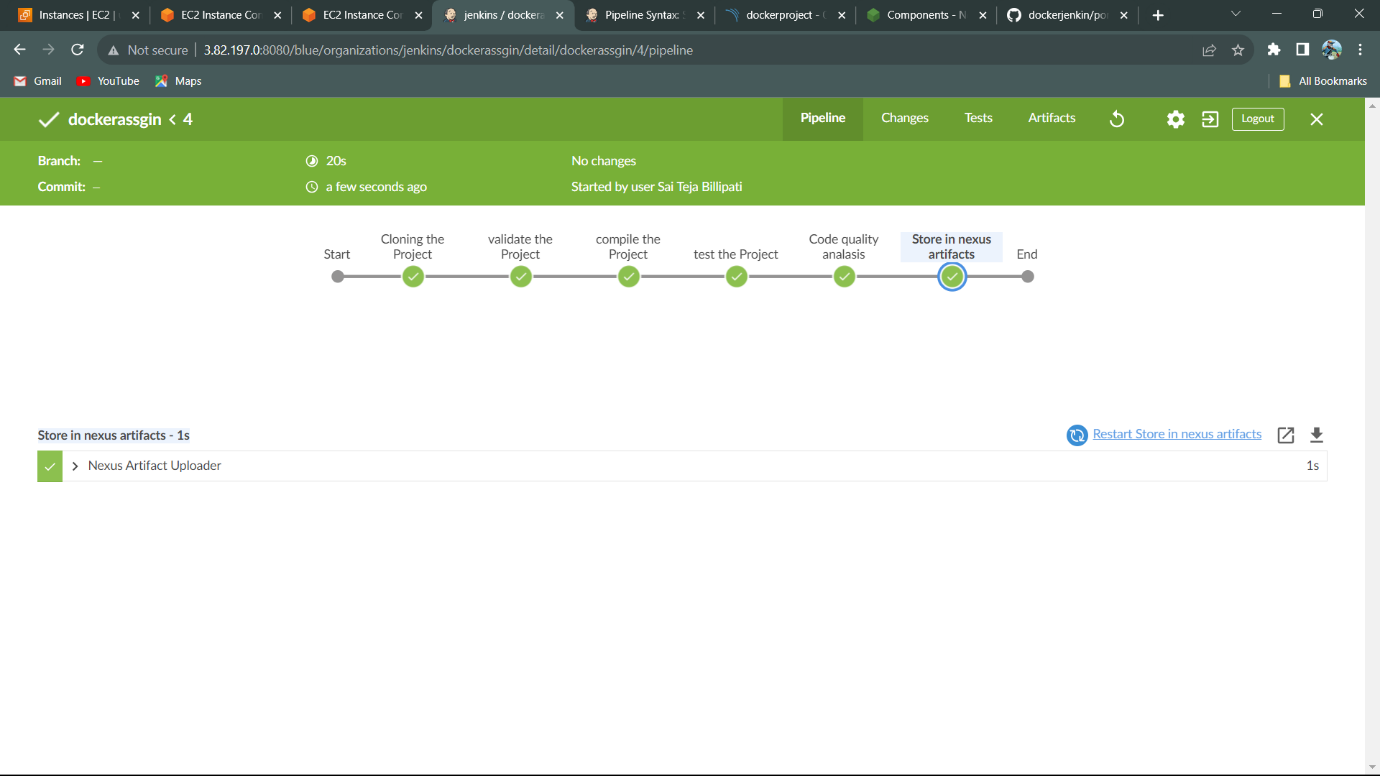
}

 }

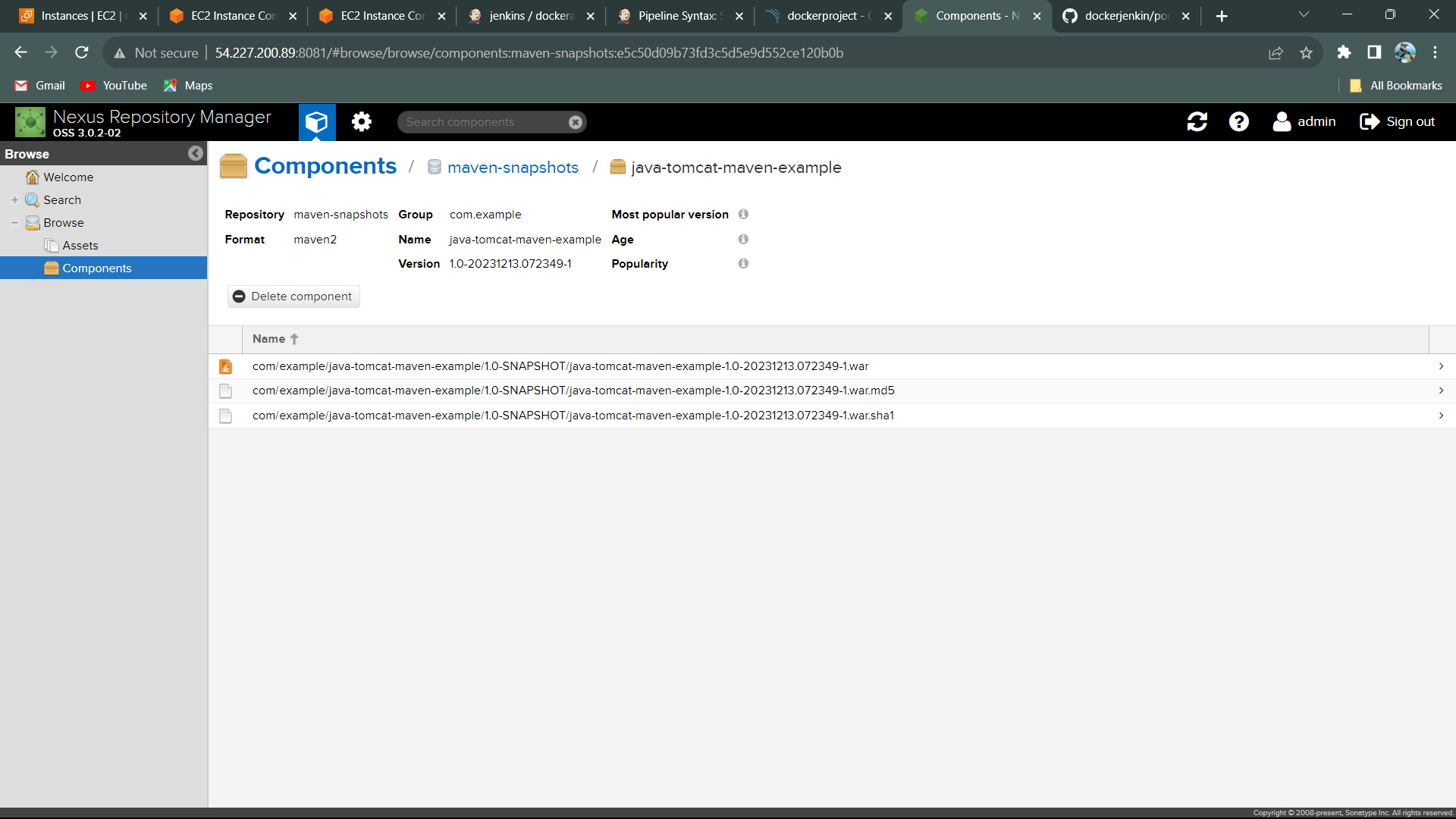




Pipeline Script for the Nexus Artifact in Jenkins.



The Pipeline is Successfully Build and The Artifact is stored in the Nexus Repository.



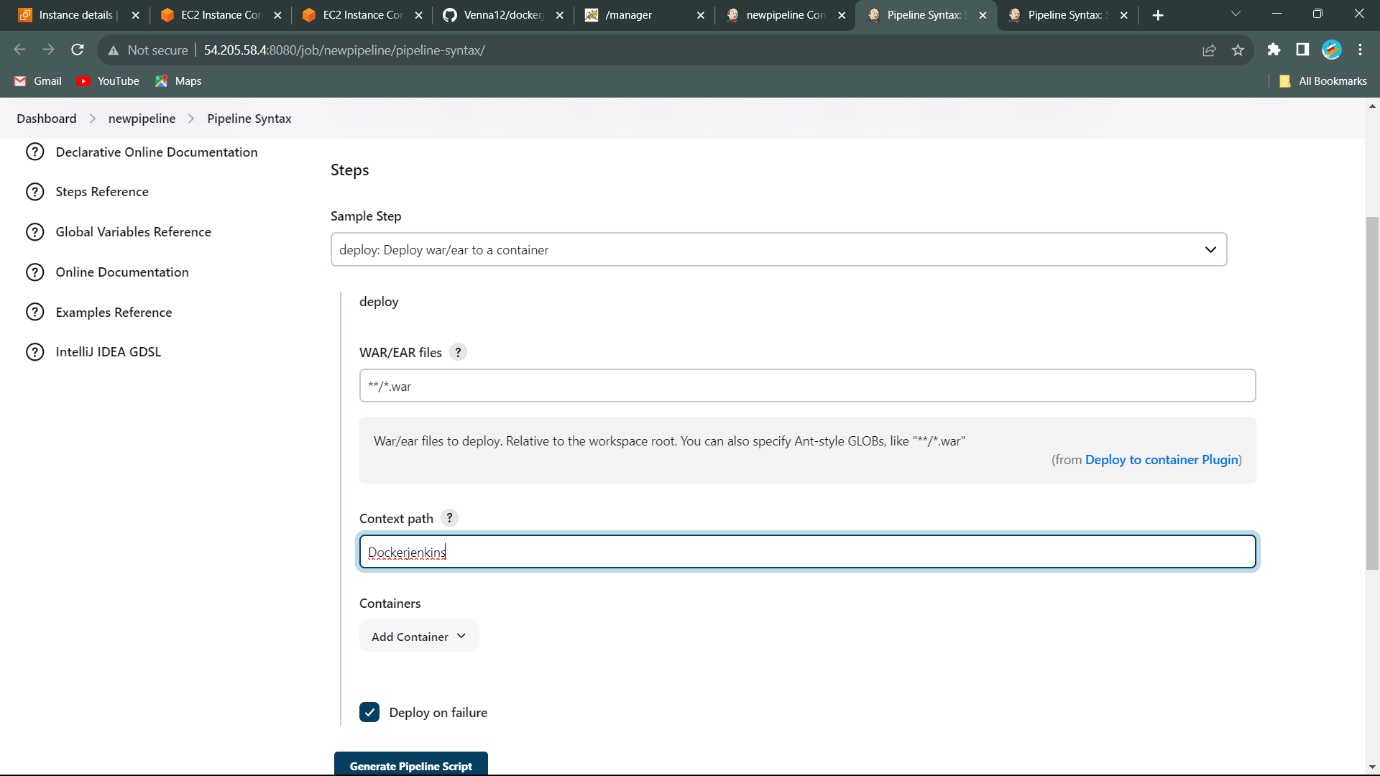
Artifact is stored in the maven-snapshots as java-tomcat-maven-example.

**Deploying project into Apache Tomcat using Jenkins:**

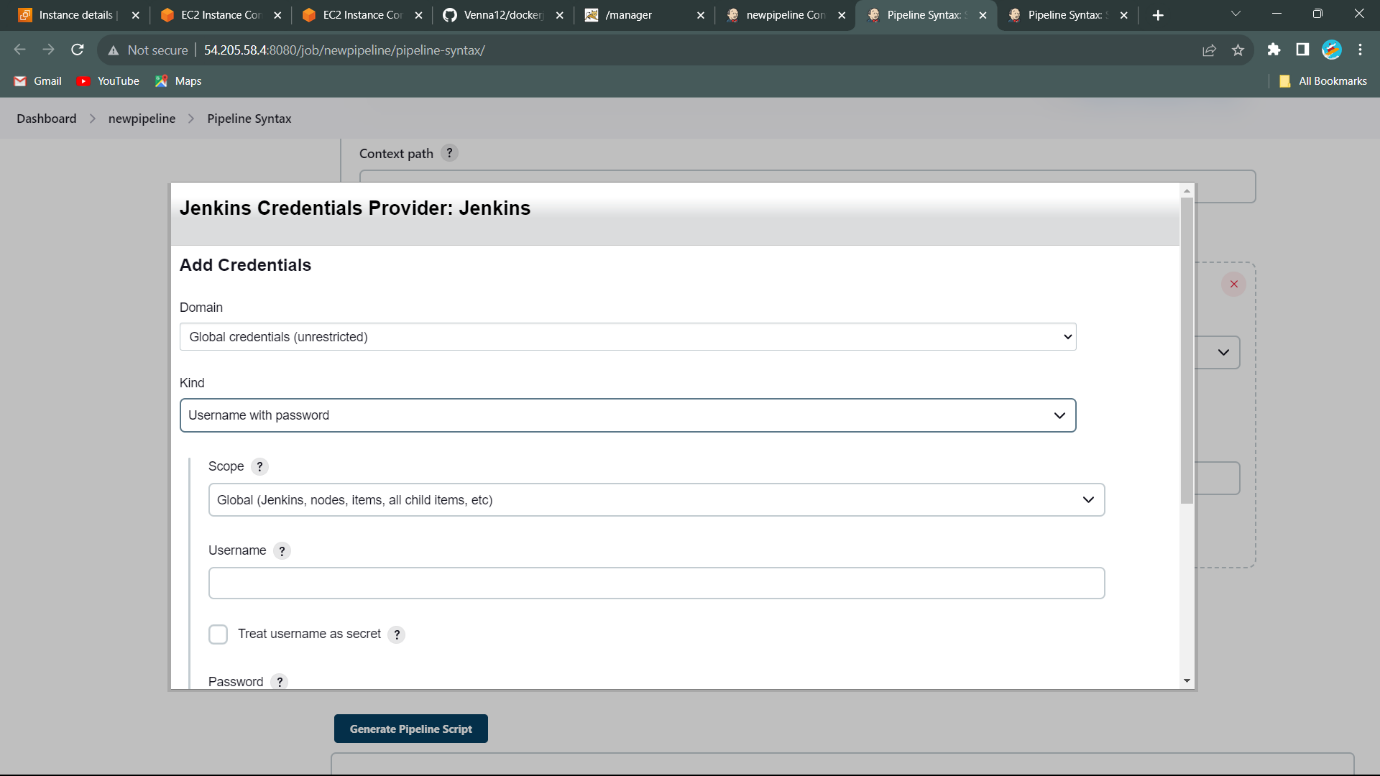
1. Open the Apache Tomcat Server using IP Address with Port no.
   1. IP address:Port no (8080).
2. Login to Tomcat using
   1. Username: Tomcat
   2. Password: s3cret
3. Install plugin for the deployment in the tomcat.

* Deploy Container

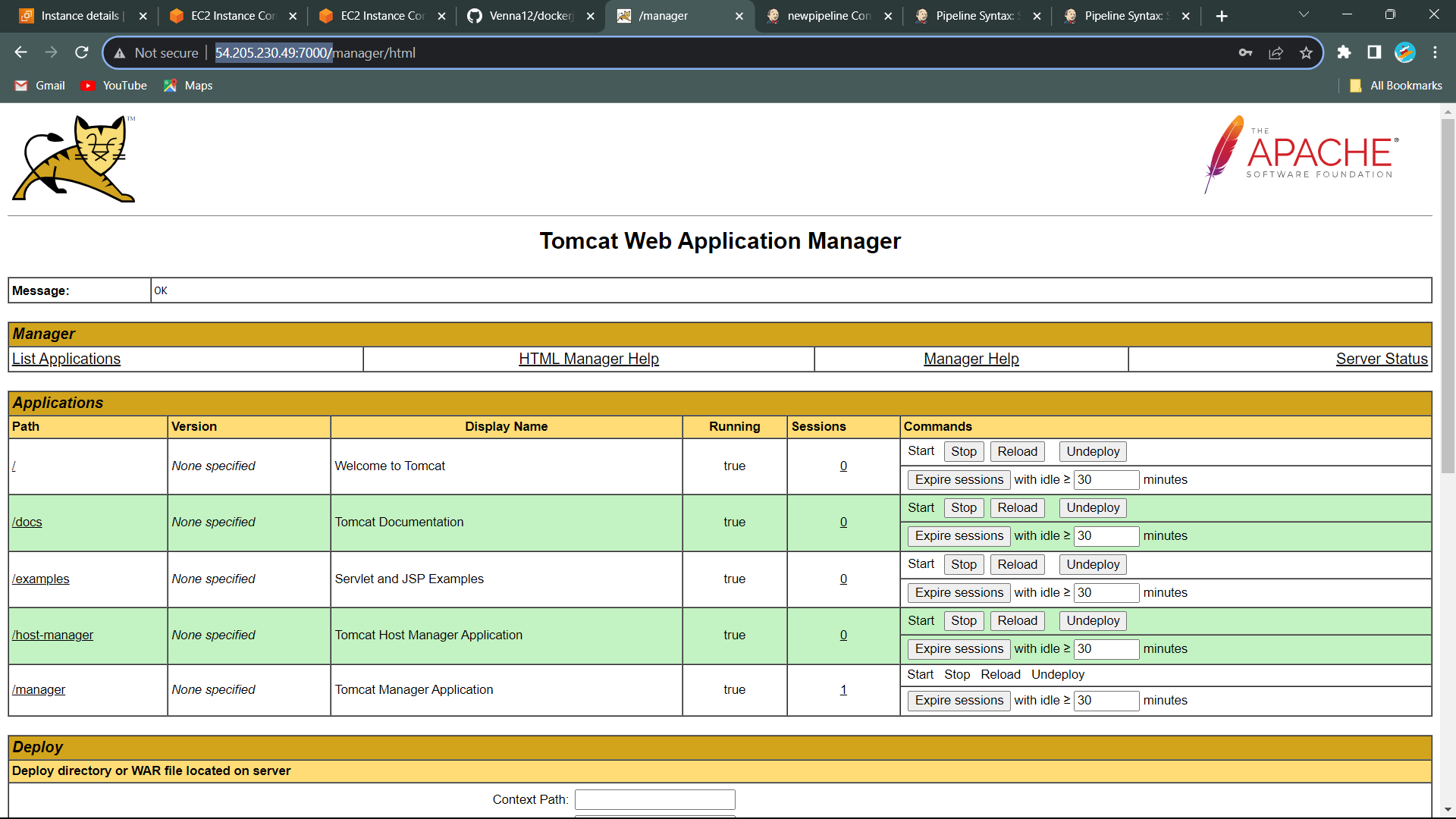
To get tomcat script, we use pipeline syntax



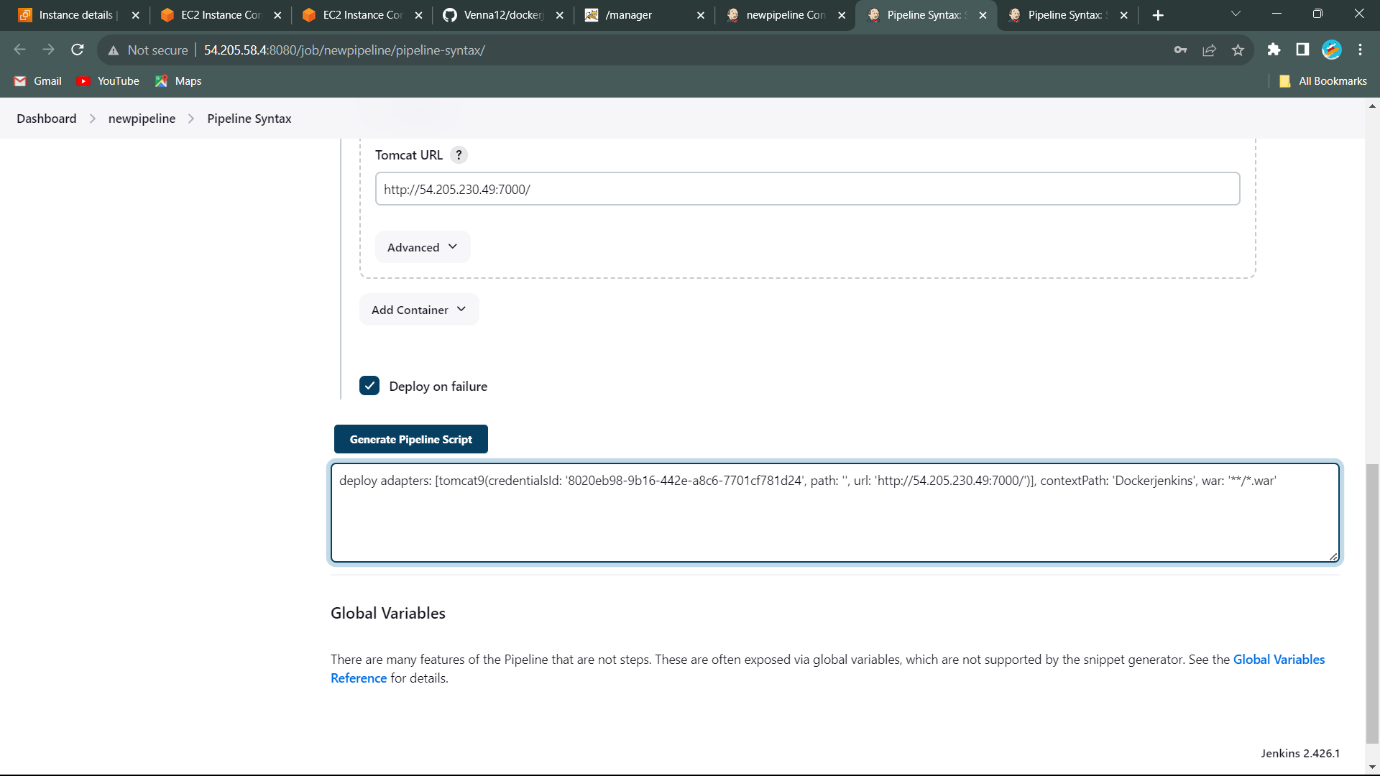
1. Create Jenkins id and password, it is add to the container.



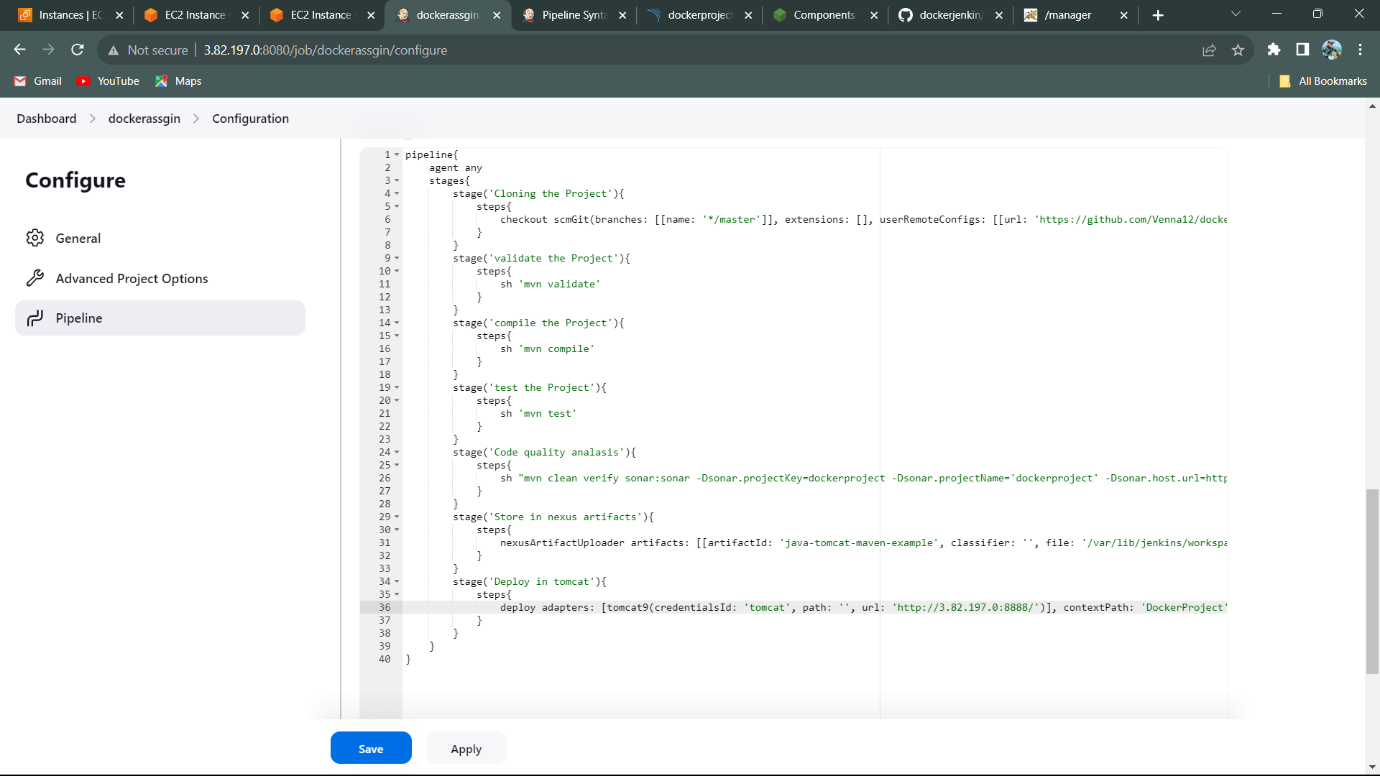
1. Open the Tomcat and host manager. It seems like that



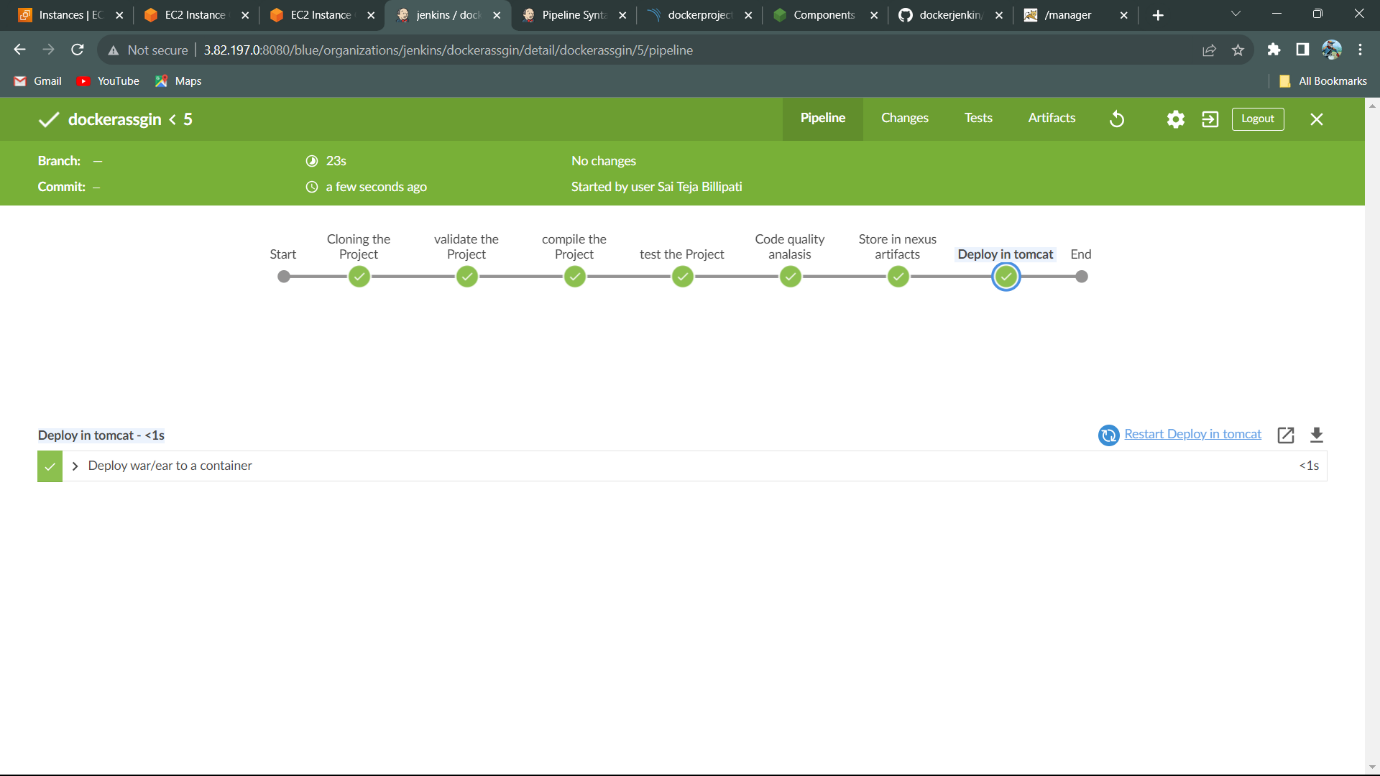
1. Copy the URL of the Tomcat with Ip address and port number. Copy the link address to the given box and we got script and copy the script the and add to the script to configure pipeline.



1. Script add to pipeline script

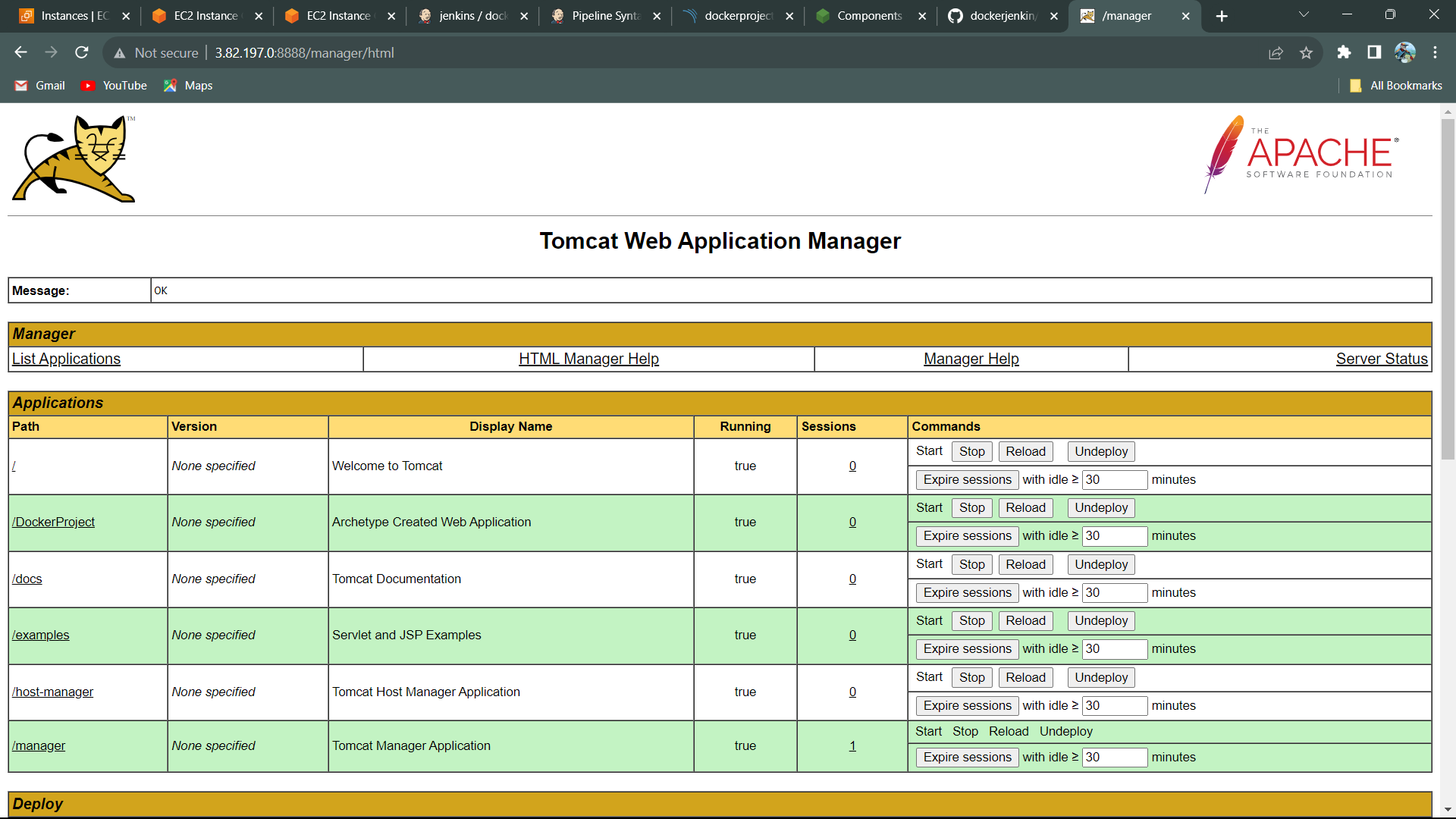


1. Build the Project.



Project is Successfully Deployed in the tomcat.

1. Open Tomcat server, we can see the DockerProject in the Tomcat Dashboad.



**Build and Push Docker Image to Docker Hub:**

1. Install Docker using Command:
   1. Sudo apt install docker.io -y
2. For Build the file, Create the Dockerfile.
3. Write the script in Dockerfile using docker keywords.

FROM nginx: latest

LABEL maintainer: Saiteja

RUN apt install nginx

CMD echo “Latest Project”

USR root

EXPOSE 80

1. Save the file.
2. To Build the Dockerfile.
   1. Use command (docker build -t foldername . )
3. Covert the Docker image to Docker Container by using command.

🡪 docker run –name nginxproject -d -p 88: 80 nginx: latest

1. It creates the Docker Container in the nginxproject.
2. To Push the Local to Remote repository, we need to tagname for the image.
   1. docker tag image id/image name username/name: version
3. Use command for the push file to Docker hub.
   1. Docker push tagname.

