

Course: DevOps
Module: Docker & Docker Hub
Topic: SonarQube, Nexus, Tomcat, Docker
Trainer Name: Mr. Madhukar sir
Name: Billipati Sai Teja
Batch no: 115
Assignment no: 05
Date of submission: 28th – Dec – 2023
Mail-ID: (BILLIPATISAITEJA@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).

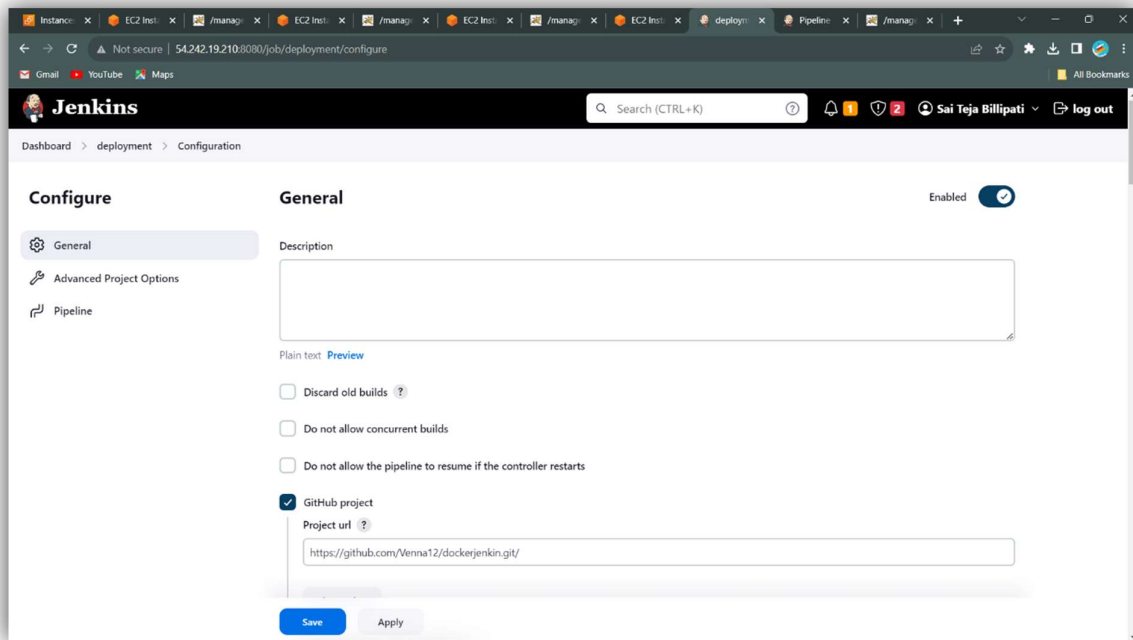
----- **Install Java jdk-11 and Maven** -----

- ➔ 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 -----

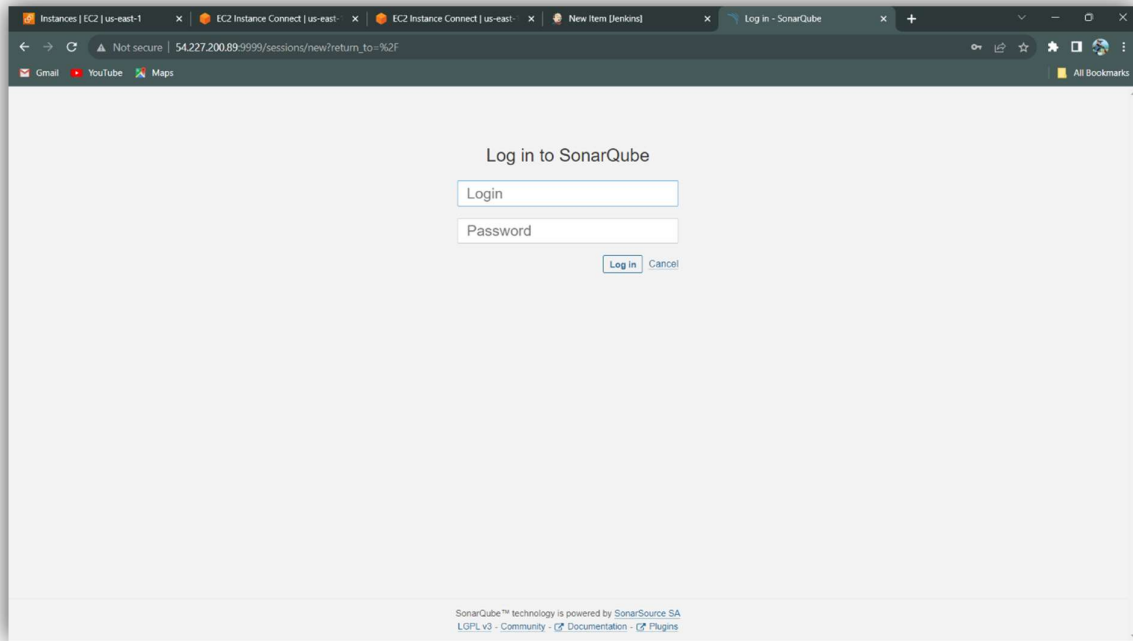
- SonarQube is a static code analysis tool, a tool that scans your code and tries to detect flaws, bugs, security vulnerabilities, etc. It can also measure test coverage of your code if provided with proper reports. All these features focus on direct code development and help developers build better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

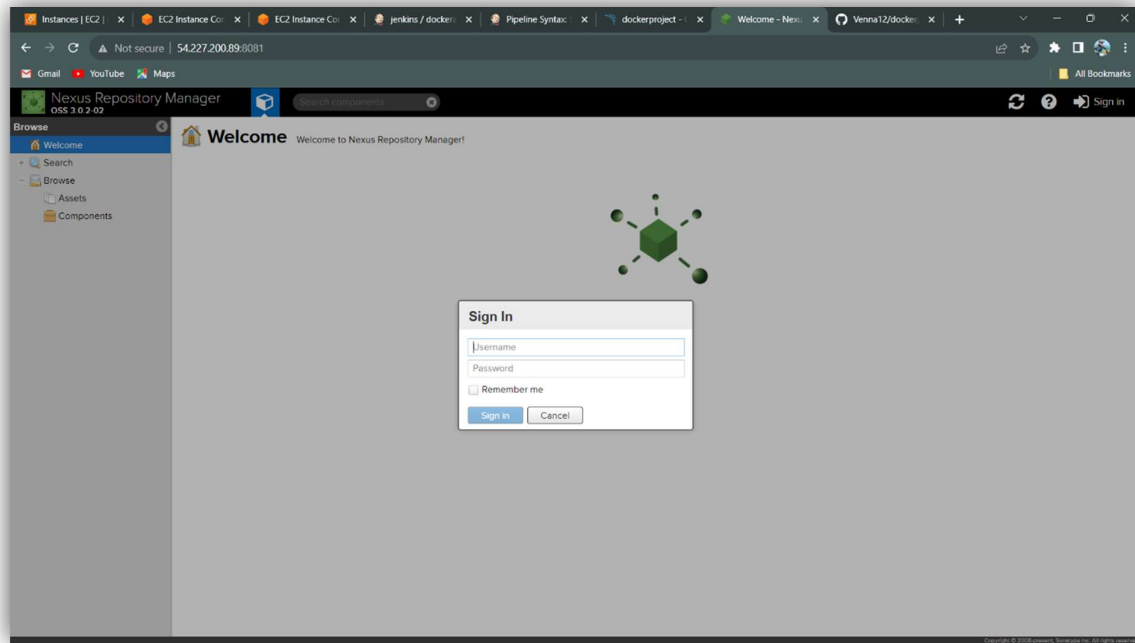
Password: admin



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

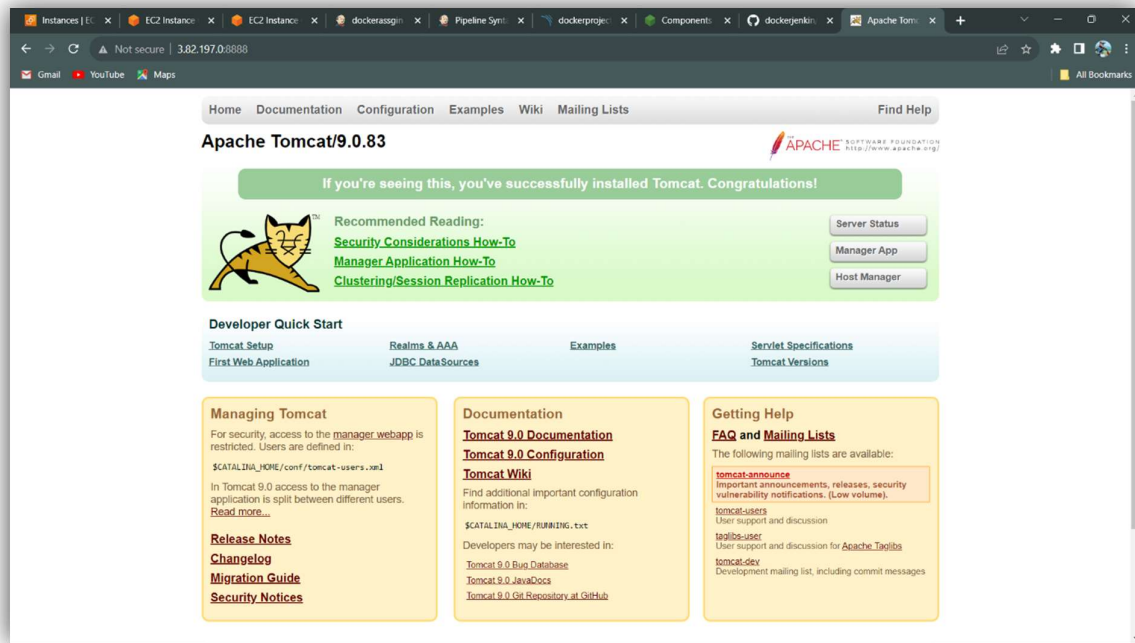
Login details of the Apache Tomcat server.

>>>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:

- 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'
      }
    }
  }
}
```

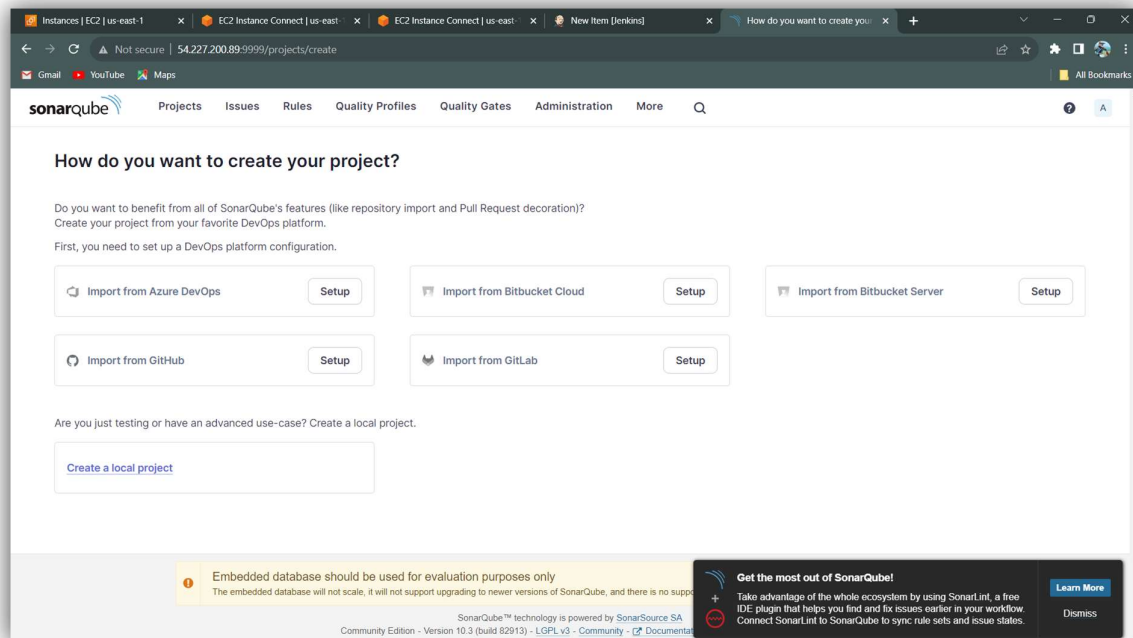
```

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

```

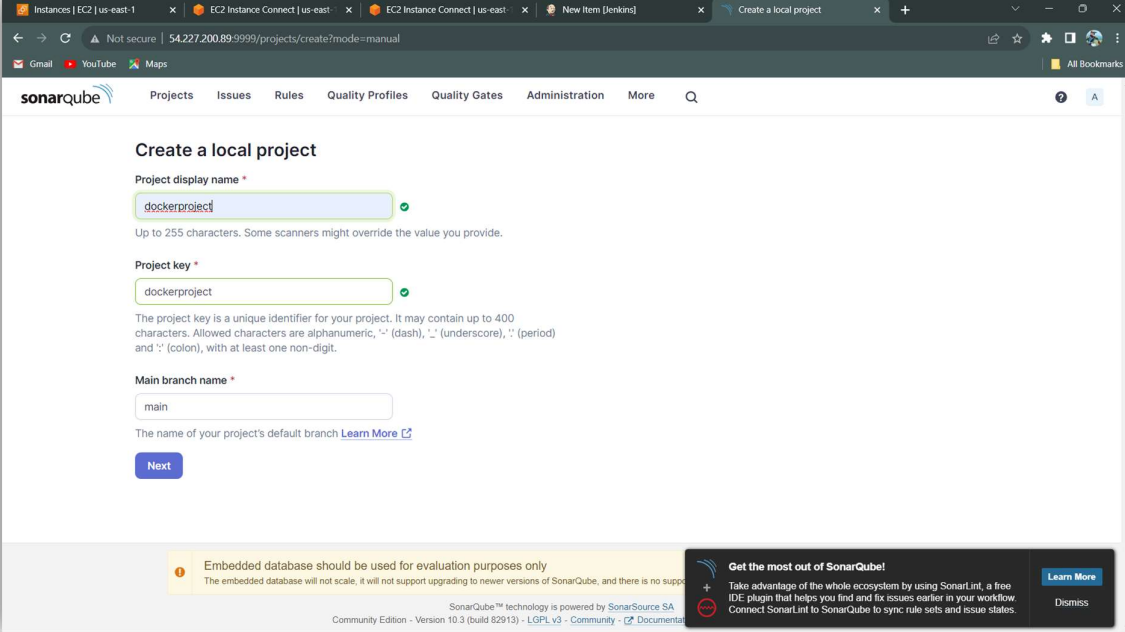
Configuration of SonarQube to Jenkins pipeline:

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.



SonarQube Dashboard

5. Click on the [Create a local project](#).
 - a. Project display name: dockerproject
 - b. Project key: dockerproject
 - c. Main branch name: main

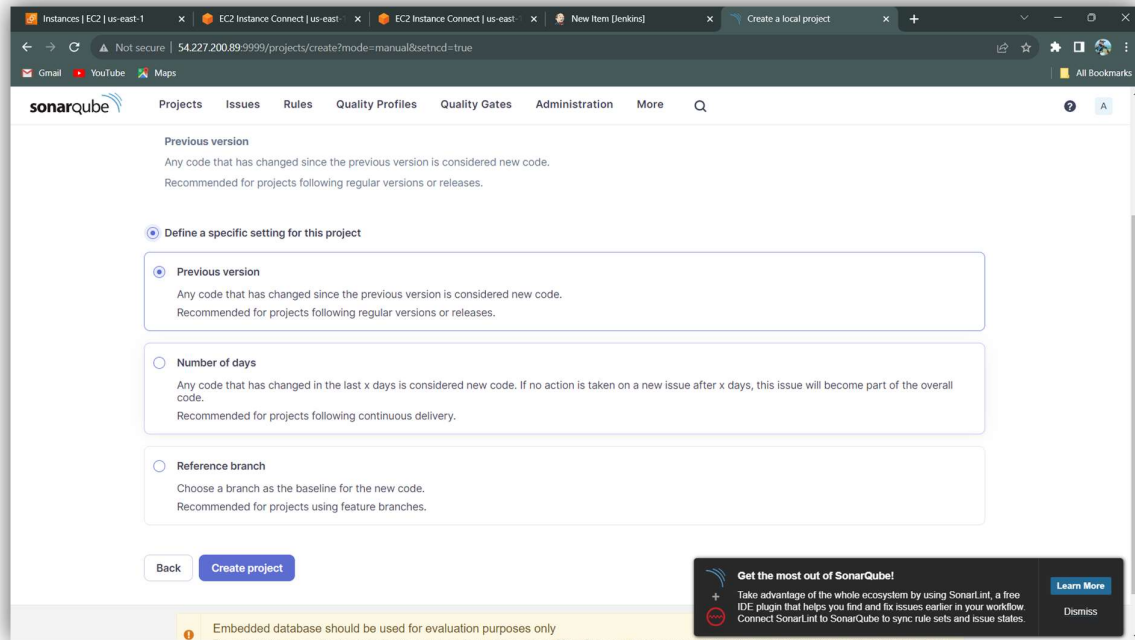


The screenshot shows a web browser window with the SonarQube interface. The page title is "Create a local project". The form contains three input fields, each with a green checkmark indicating valid input:

- Project display name ***: The input field contains "dockerproject". Below it, a note states: "Up to 255 characters. Some scanners might override the value you provide."
- Project key ***: The input field contains "dockerproject". Below it, a note states: "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 ***: The input field contains "main". Below it, a note states: "The name of your project's default branch [Learn More](#)".

At the bottom of the form is a blue "Next" button. Below the form, there is a yellow warning banner that reads: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for it." To the right of this banner is a dark blue promotional banner for SonarLint, which says: "Get the most out of SonarQube! Take advantage of the whole ecosystem by using SonarLint, a free IDE plugin that helps you find and fix issues earlier in your workflow. Connect SonarLint to SonarQube to sync rule sets and issue states." with a "Learn More" button and a "Dismiss" button.

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.



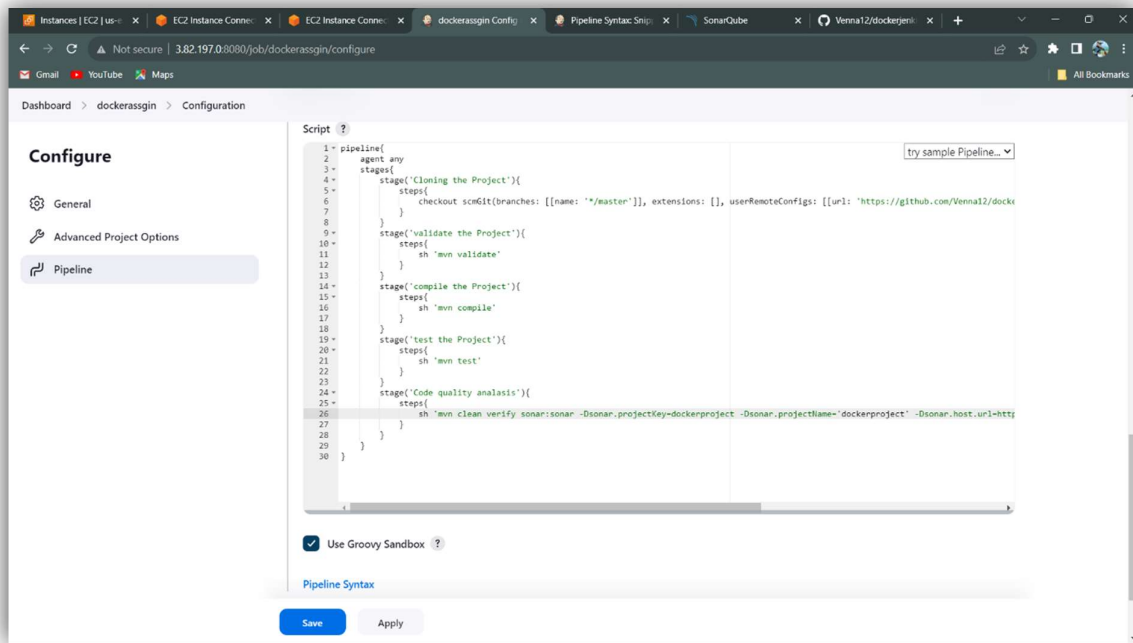
9. It generates Token name “dockerproject”, Expires in 30days, Click on Generate.

10. Run analysis on the project – select the maven it generates the maven code.

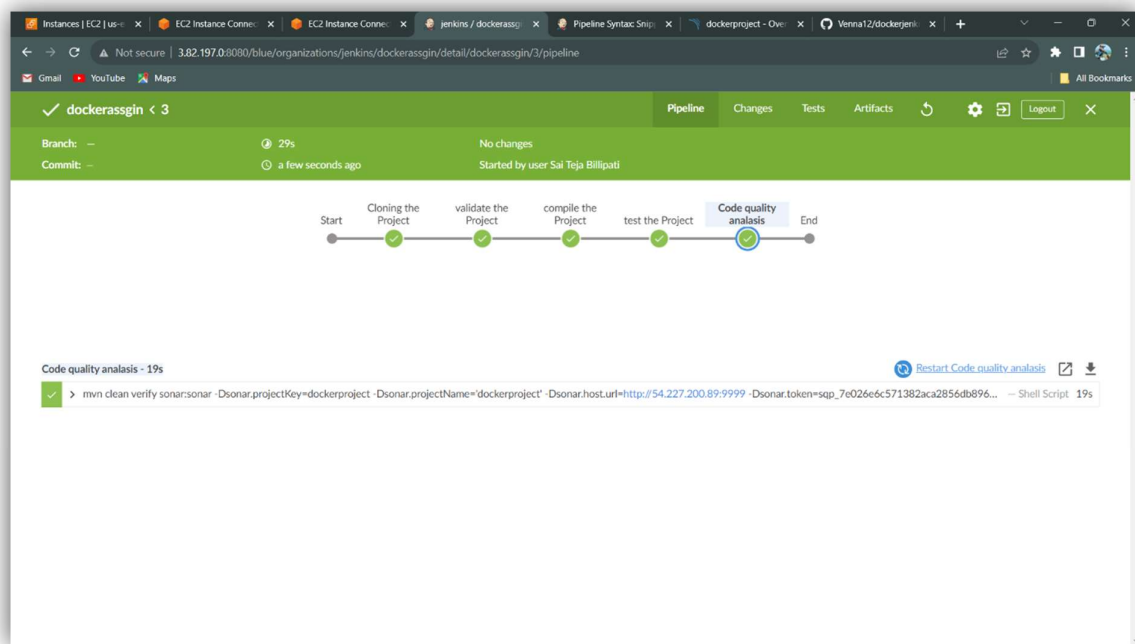
```
mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject
-Dsonar.projectName='dockerproject'
-Dsonar.host.url=http://54.227.200.89:9999
-Dsonar.token=sqp_7e026e6c571382aca2856db896ab7d99a5d9dc52
```

11. 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"
    }
}
```

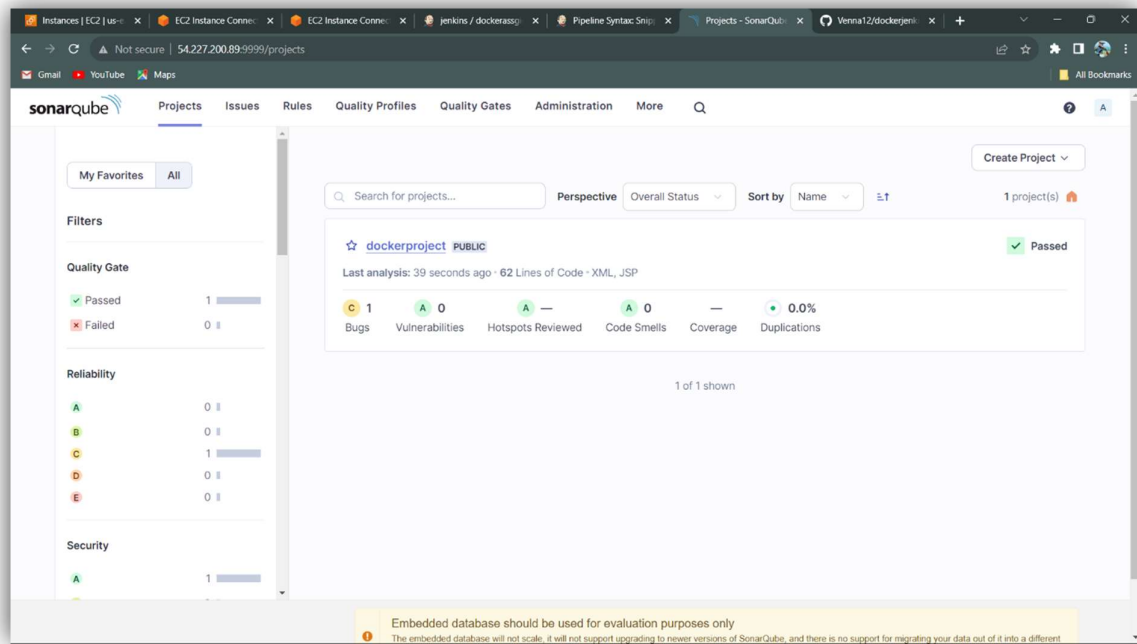



12. Apply and save the pipeline. Build the Pipeline.



13. Code Quality analysis is build Successfully with the SonarQube.

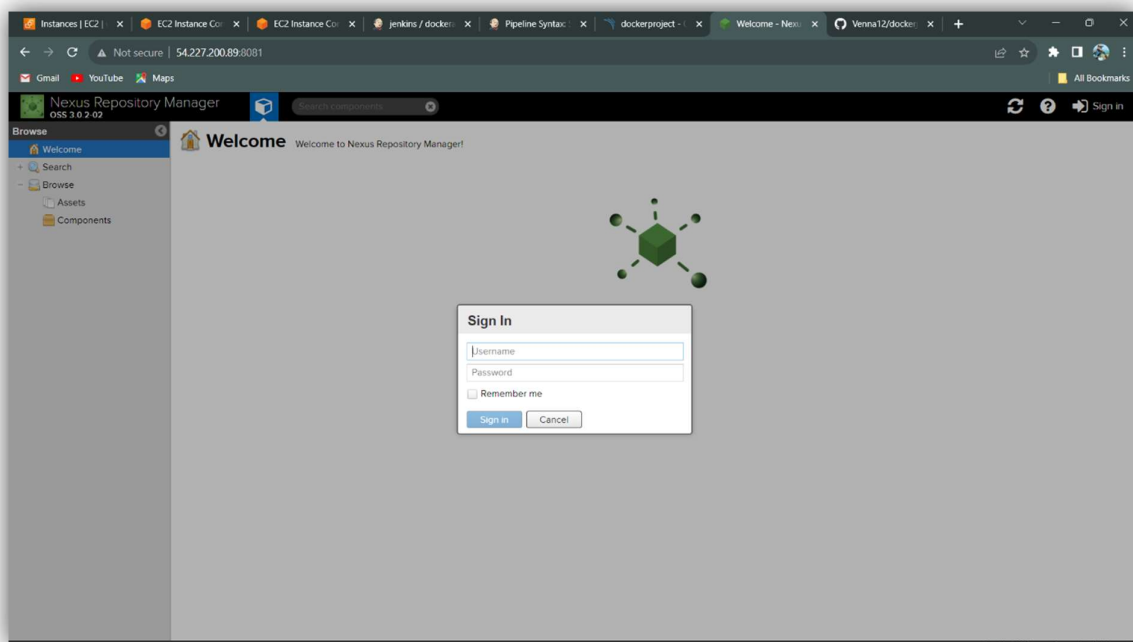
14. It shows the Detect flaws, Bugs, Security Vulnerabilities, Code coverage, in the SonarQube.



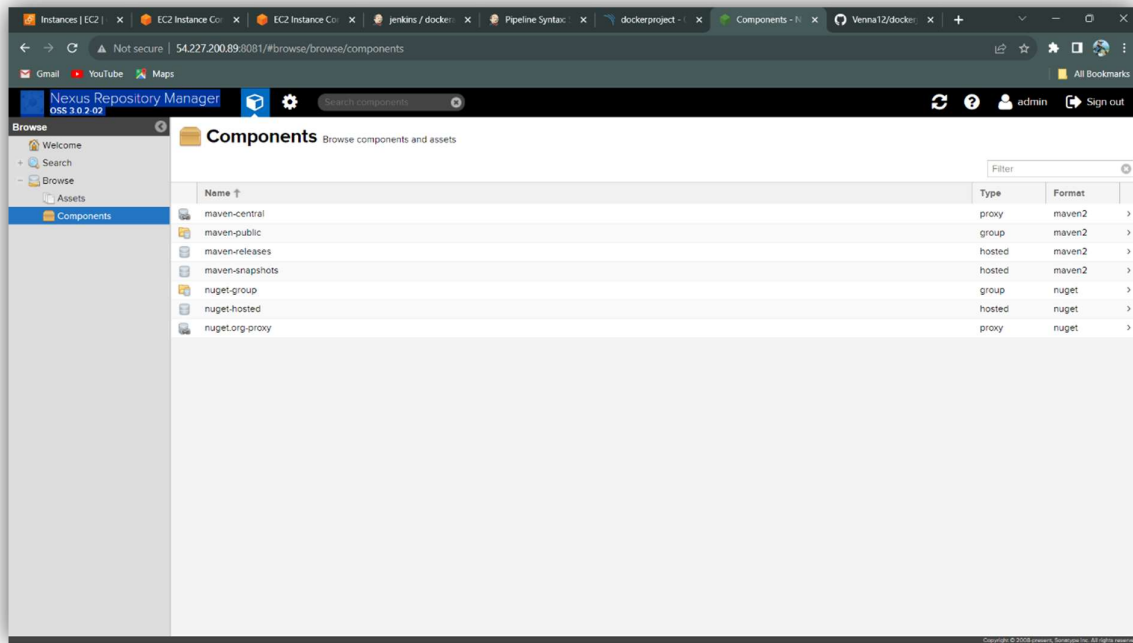
Code Quality Analysis with SonarQube.

Configuration of Nexus to Jenkins pipeline:

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



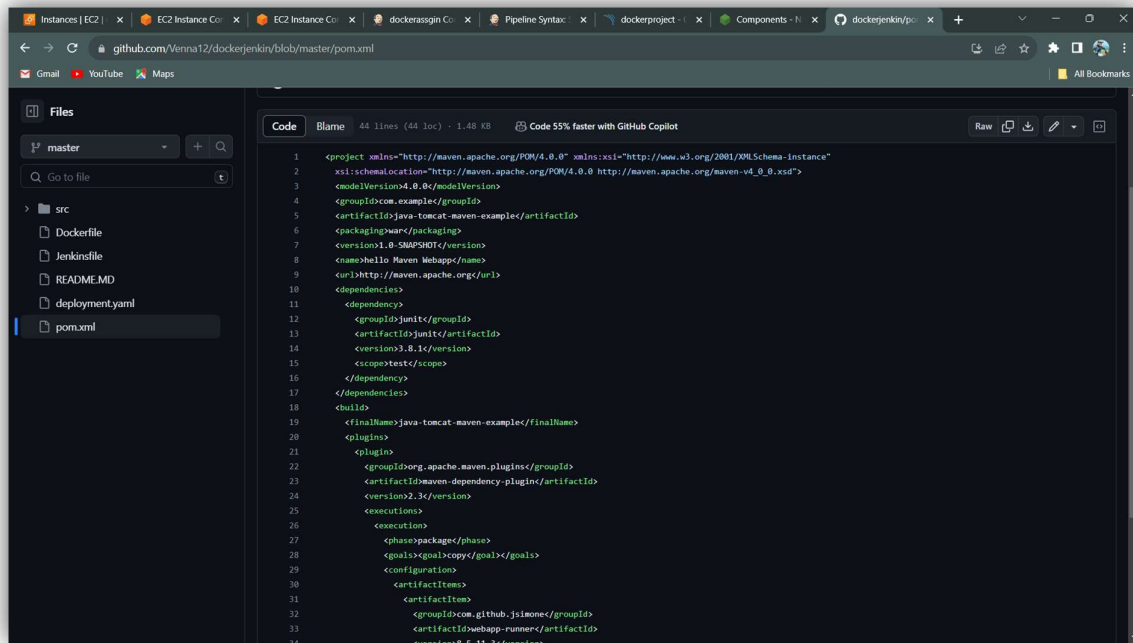
3. Open Components and Click on Maven-Snapshots.
4. Open Maven-snapshots repository.



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.



Pom.xml file

12. Open pipeline syntax.
 - a. Sample step
 - i. nexusArtifactUploader: Nexus Artifact Uploader
 - b. Nexus Version
 - i. NEXUS 3
 - c. Protocol
 - i. HTTP
 - d. Nexus URL
 - i. 54.227.200.89:8081
 - e. Credentials
 - i. Create Jenkins for credentials using
 1. Username: admin
 2. Password: admin123
 - f. GroupId
 - i. Com.example
 - g. Version
 - i. 1.0-SNAPSHOT
 - h. Repository
 - i. Maven-snapshots (Repository from the Nexus)
 - i. Add Artifact
 - j. Artifact Id
 - i. Java-tomcat-maven-example
 - k. Type
 - i. War
 - l. File
 - i. /var/lib/jenkins/workspace/dockerassgin/target/java-tomcat-maven-example.war (Building war from console output)
13. Generate Pipeline and copy the code

14. 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'
    }
}
```

The screenshot shows the Jenkins Pipeline Syntax configuration page for the 'nexusArtifactUploader' step. The left sidebar contains a navigation menu with links: Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDLS. The main content area is titled 'Steps' and shows a 'Sample Step' dropdown set to 'nexusArtifactUploader: Nexus Artifact Uploader'. Below this, the 'Nexus Details' section contains several fields: 'Nexus Version' (dropdown set to 'NEXUS3'), 'Protocol' (dropdown set to 'HTTP'), 'Nexus URL' (text input with '54.227.200.89:8081'), and 'Credentials' (dropdown set to 'admin/***** (nexus123)'). There is an '+ Add' button below the credentials field.

This screenshot shows the same Jenkins Pipeline Syntax configuration page, but with additional fields visible below the 'Credentials' section. The 'GroupId' field is a text input containing 'com.example'. The 'Version' field is a text input containing '1.0-SNAPSHOT'. The 'Repository' field is a text input containing 'maven-snapshots'. The '+ Add' button is still present below the credentials field.

Instances | EC2 | x EC2 Instance Co x EC2 Instance Co x dockerassgin Co x Pipeline Syntax x dockerproject x Components - N x dockerjenkin/po x +

Not secure | 3.82.197.0.8080/job/dockerassgin/pipeline-syntax/

Gmail YouTube Maps All Bookmarks

Dashboard > dockerassgin > Pipeline Syntax

ARTIFACTS

Artifact

ArtifactId

java-tomcat-maven-example

Type ?

war

Classifier ?

File ?

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

File must not be empty

Add

Generate Pipeline Script

checkout scmGit[branches: [[name: */master]], extensions: [], userRemoteConfigs: [[url: 'https://github.com/Venna12/dockerjenkin.git']]]

Instances | EC2 | x EC2 Instance Co x EC2 Instance Co x dockerassgin Co x Pipeline Syntax x dockerproject x Components - N x dockerjenkin/po x +

Not secure | 3.82.197.0.8080/job/dockerassgin/configure

Gmail YouTube Maps All Bookmarks

Dashboard > dockerassgin > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

```
10-      steps{
11-        sh 'mvn validate'
12-      }
13-    }
14-    stage('compile the Project'){
15-      steps{
16-        sh 'mvn compile'
17-      }
18-    }
19-    stage('test the Project'){
20-      steps{
21-        sh 'mvn test'
22-      }
23-    }
24-    stage('Code quality analysis'){
25-      steps{
26-        sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject -Dsonar.projectName=dockerproject -Dsonar.host.url=http://localhost:9000'
27-      }
28-    }
29-    stage('Store in nexus artifacts'){
30-      steps{
31-        nexusArtifactUploader artifacts: [[artifactId: 'java-tomcat-maven-example', classifier: '', file: '/var/lib/jenkins/workspace/dockerassgin/target/java-tomcat-maven-example.war']]
32-      }
33-    }
34-  }
35-}
```

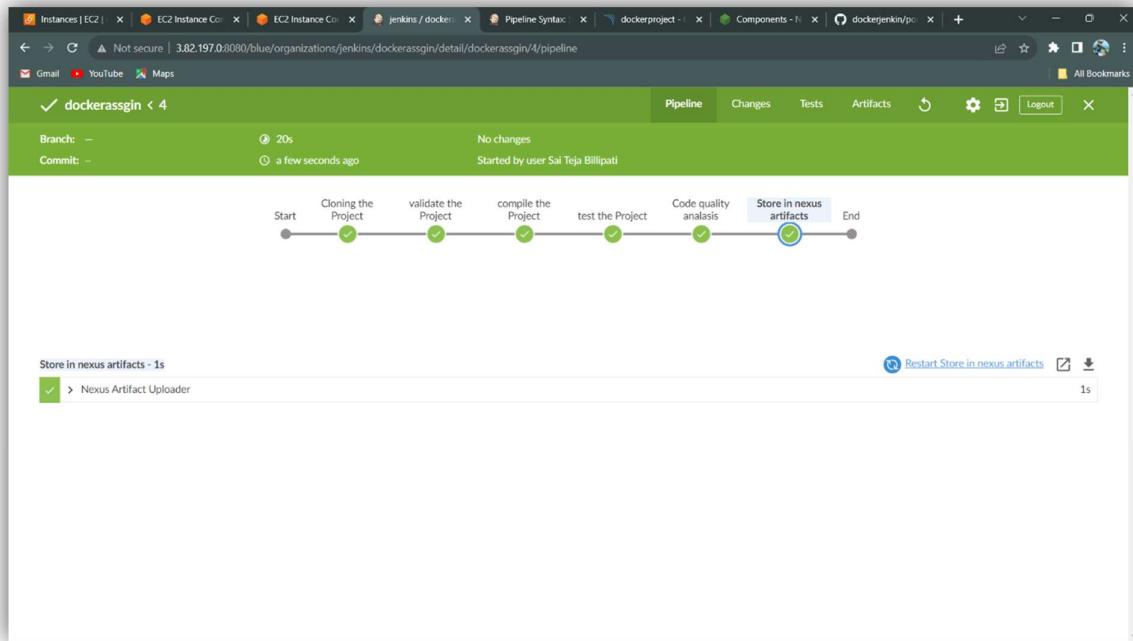
☒ Use Groovy Sandbox ?

Pipeline Syntax

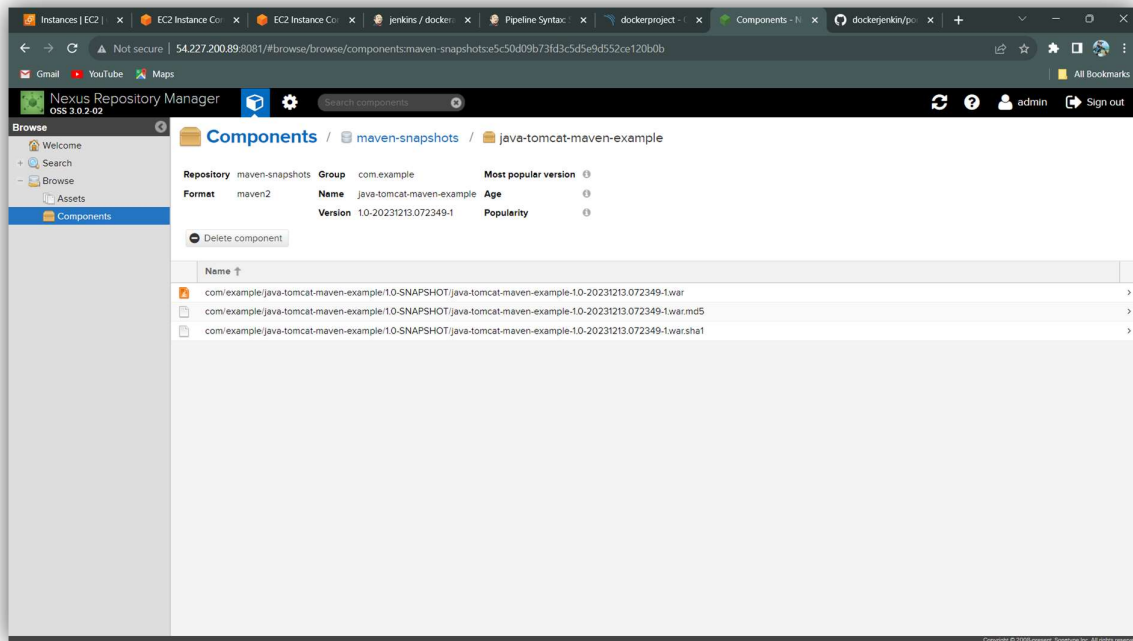
Save Apply

REST API Jenkins 2.426.1

Pipeline Script for the Nexus Artifact in Jenkins.



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



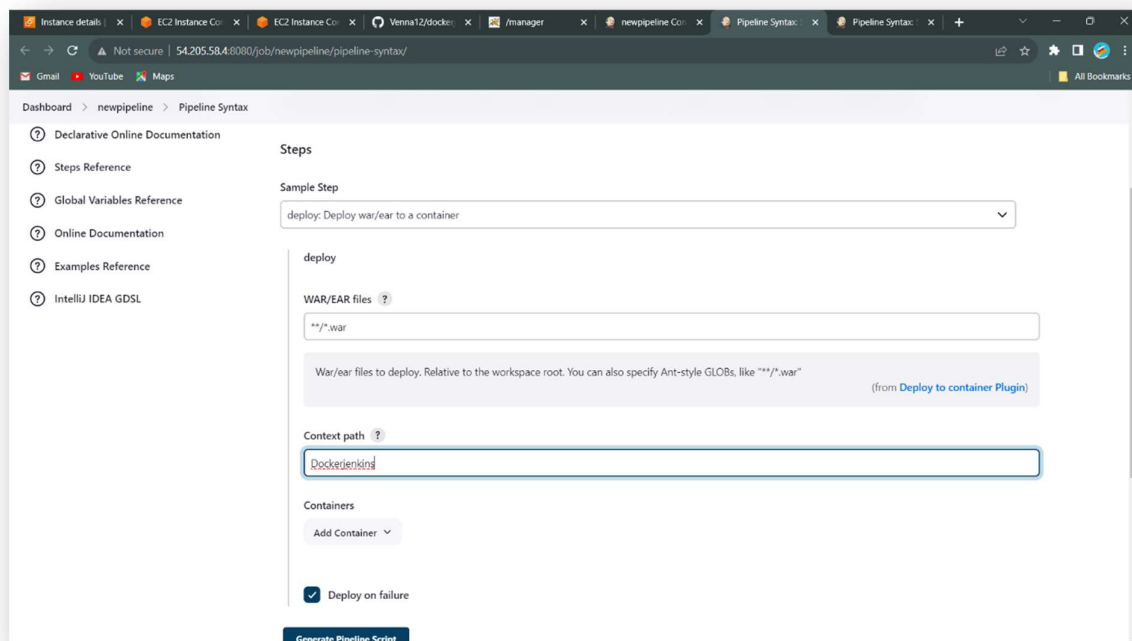
Artifact is stored in the maven-snapshots as java-tomcat-maven-example.

Deploying project into Apache Tomcat using Jenkins:

1. Open the Apache Tomcat Server using IP Address with Port no.
 - a. IP address:Port no (8080).
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.

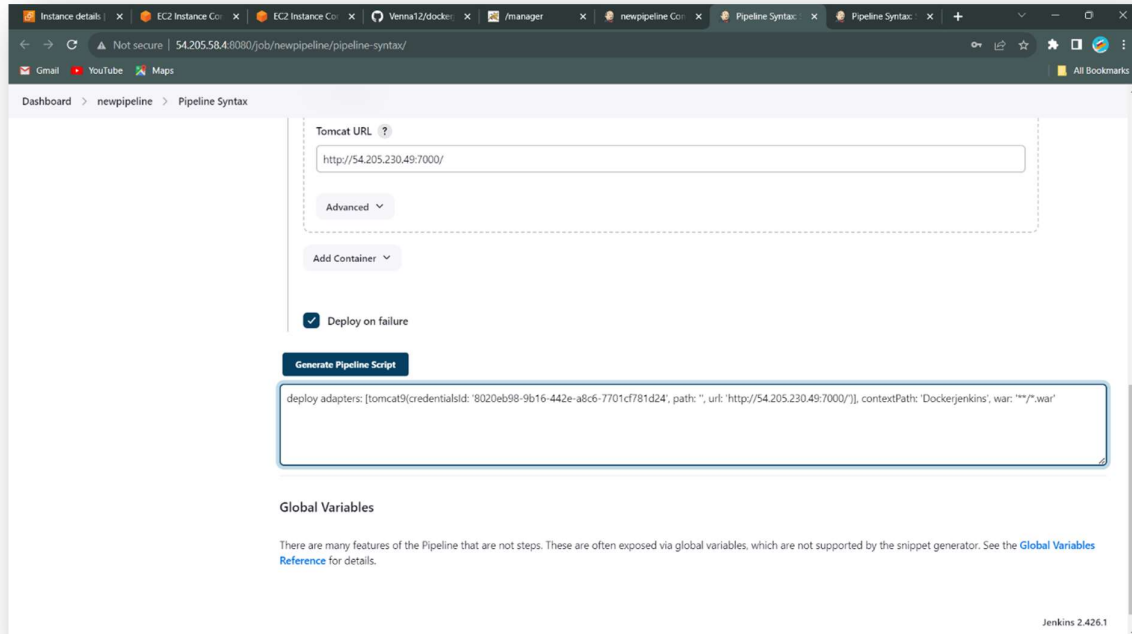
The screenshot shows a web browser window with the Jenkins Pipeline Syntax page. A modal titled "Jenkins Credentials Provider: Jenkins" is open, displaying the "Add Credentials" form. The form includes fields for Domain (Global credentials (unrestricted)), Kind (Username with password), Scope (Global (Jenkins, nodes, items, all child items, etc)), Username, and Password. There is a checkbox for "Treat username as secret" and a "Generate Pipeline Script" button at the bottom.

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

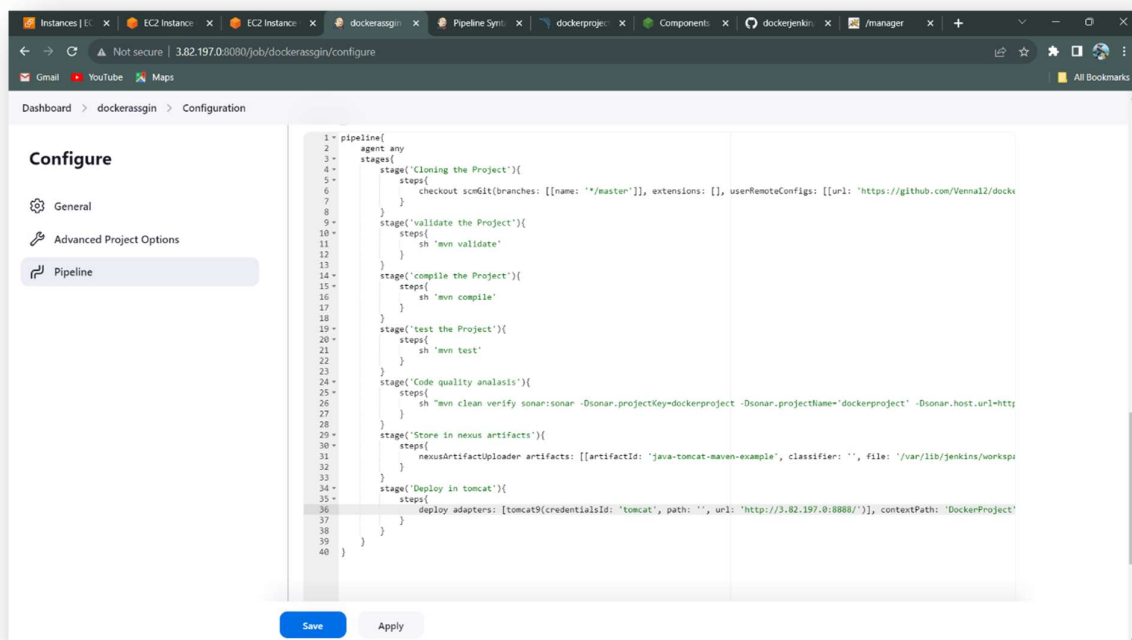
The screenshot shows the Tomcat Web Application Manager interface. It features the Tomcat logo and the Apache Software Foundation logo. Below the logos, there is a "Tomcat Web Application Manager" title. A message box is visible. The main content area is divided into sections: "Manager" with links for "List Applications", "HTML Manager Help", "Manager Help", and "Server Status"; "Applications" with a table listing various applications; and "Deploy" with a section for "Deploy directory or WAR file located on server" and a "Context Path" input field.

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

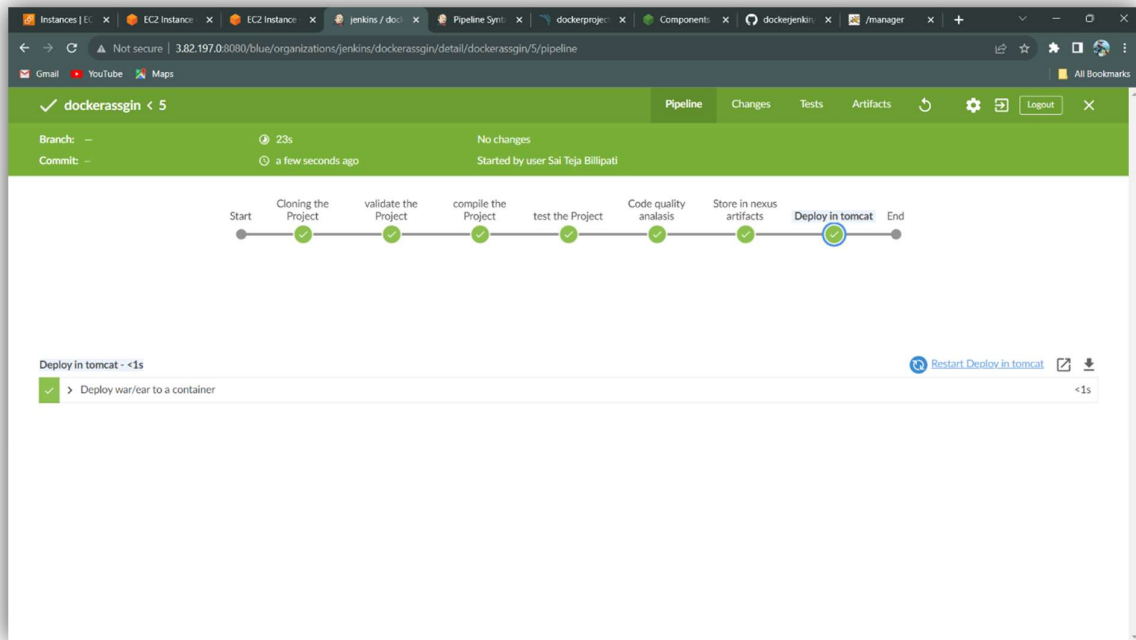
- Copy the URL of the Tomcat with Ip address and port number. Copy the link address to the given box and we got script and copy the script the and add to the script to configure pipeline.



- Script add to pipeline script

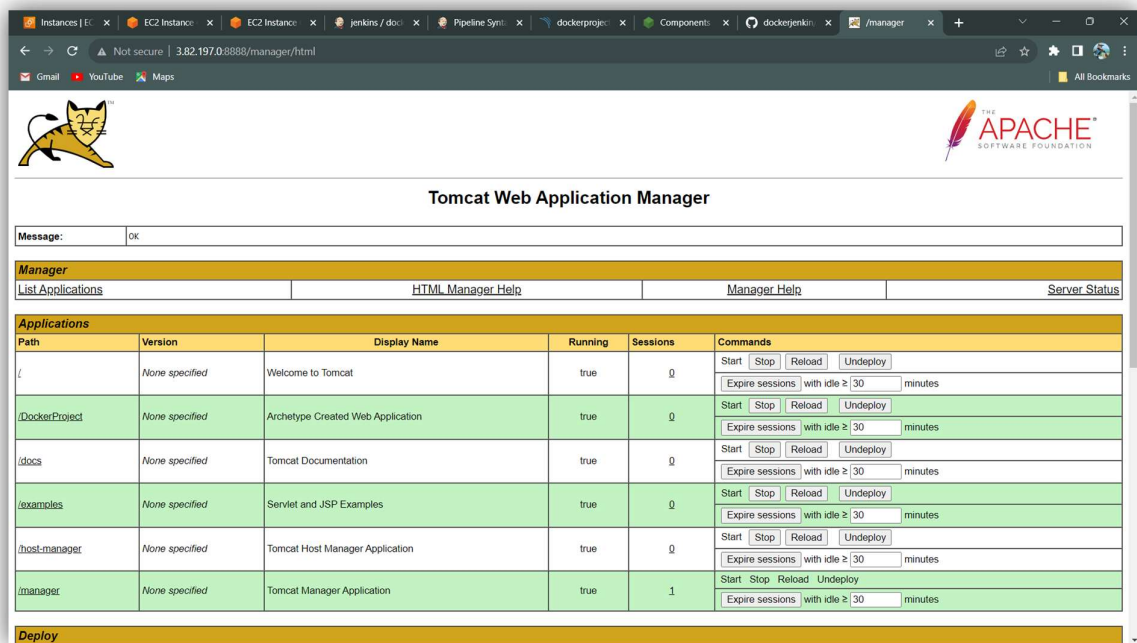


8. Build the Project.



Project is Successfully Deployed in the tomcat.

9. 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:
 - a. 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
```

4. Save the file.
5. To Build the Dockerfile.
 - a. Use command (docker build -t foldername .)
6. Covert the Docker image to Docker Container by using command.
→ docker run --name nginxproject -d -p 88: 80 nginx: latest
7. It creates the Docker Container in the nginxproject.
8. To Push the Local to Remote repository, we need to tagname for the image.
 - a. docker tag image id/image name username/name: version
9. Use command for the push file to Docker hub.
 - a. Docker push tagname.

