Course: DevOps **Name: Billipati Sai Teja**
Module: DOCKER **Mail-ID: billipatisaiteja@gmail.com**
Topic: DOCKER IMAGE **Batch no: 115**
Trainer Name: Mr. Madhukar sir **Project No: 02**
Date of submission: 2nd – JAN – 2024

. . . . . .

Project Title: DEVSCOPS PROJECT – PET PROJECT

- In this project will deploy the Project using the Jekins and docker and to know the quality of the project we use SonarQube in one instance
- The following steps will be done.

Steps:-

Step 1 — Create an T2 Large Instance

Step 2 — Install Jenkins, Docker. Create a SonarQube. Create a Pipeline Project in Jenkins using a Declarative Pipeline

Step 3 — Configure Sonar Server in Manage Jenkins

Step 4 — Install OWASP Dependency Check Plugins

Step 5 — Install Nexus artifact.

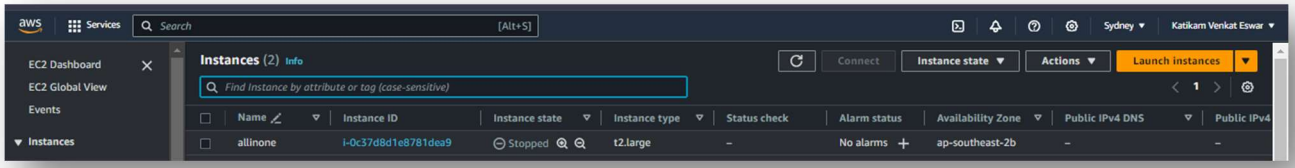
Step 6 — Install tomcat and deploy the project in it.

Step 7 —Install Docker & Build a Docker Image and Push

Step 8 — Image Push

STEP 1:

- Creating a T2 – Large instance.
- Go to security groups>edit bounds. Create a new group and give all traffic so that any port number can be run.



STEPS 2: INSTALL JEKINS AND DOCKER & REQURIED PLUGINS IN JEKNKINS

- 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.

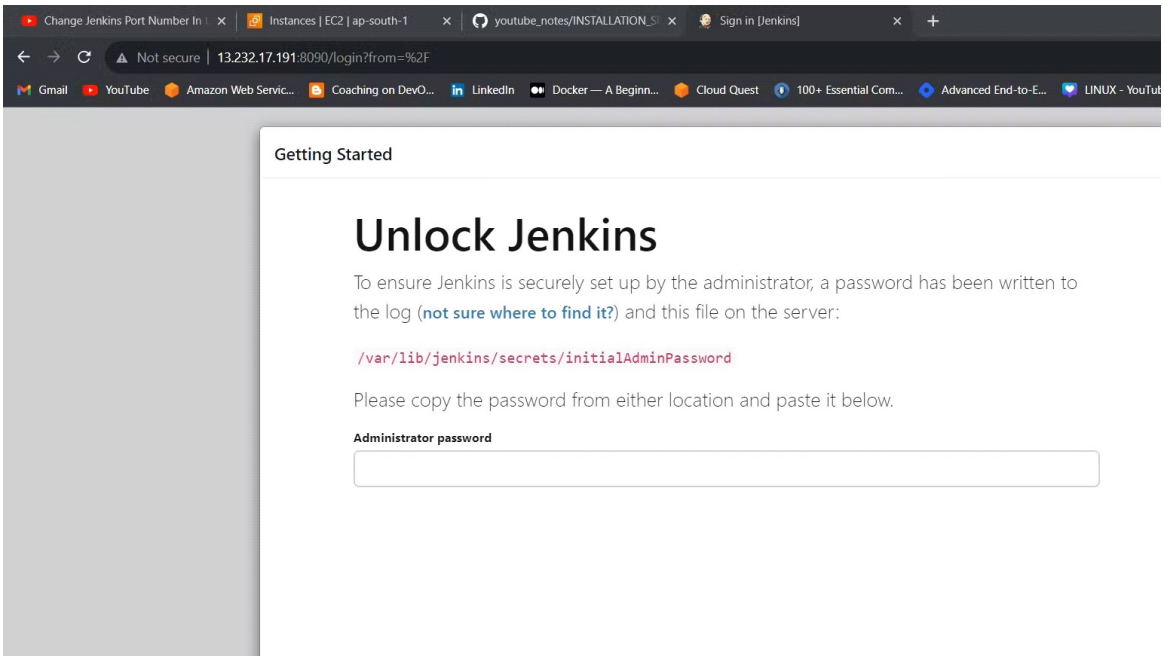
- Execute the below command
  vi jenkins.sh
- Write the code in it.

```
#!/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 installl Jenkins

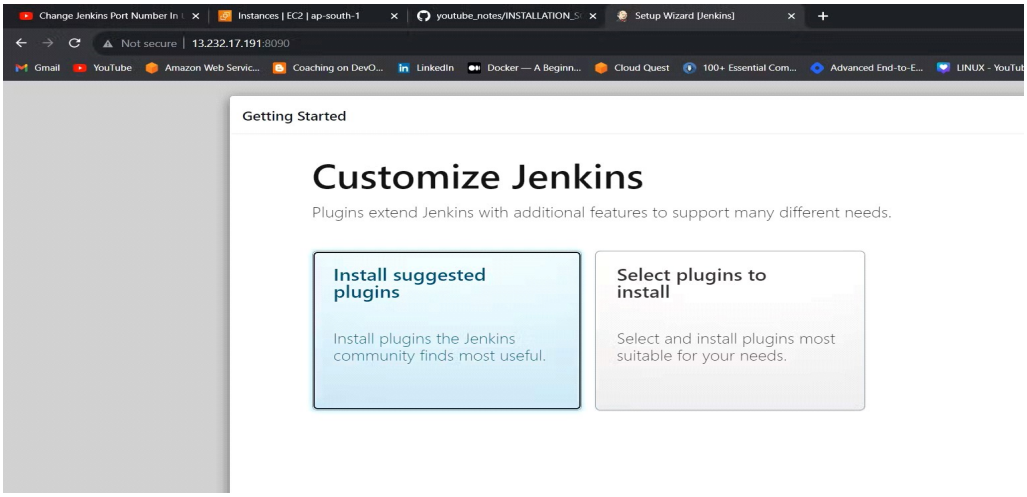
- Now copy the the Ip address and go through the Jenkins portal the interface will like shown below.



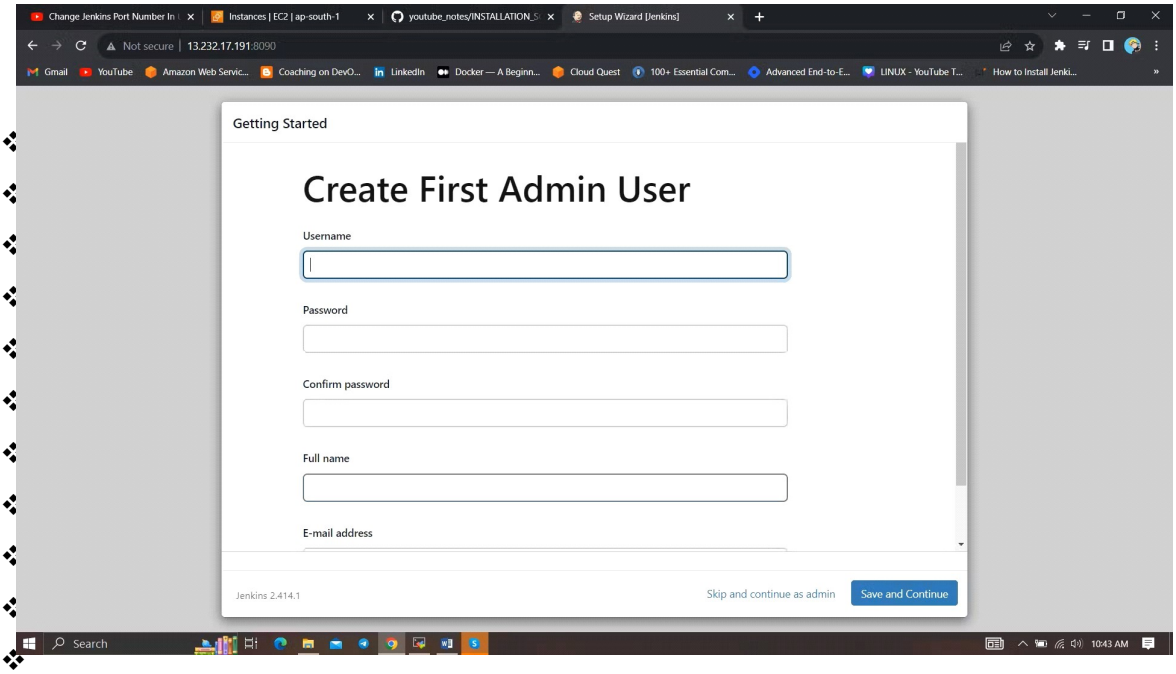
- Unlock the Jekins by using command through the given path then we will get the passwd.

  - ✓ **cat /var/lib/Jenkins/secrets/initialAdminPassword**

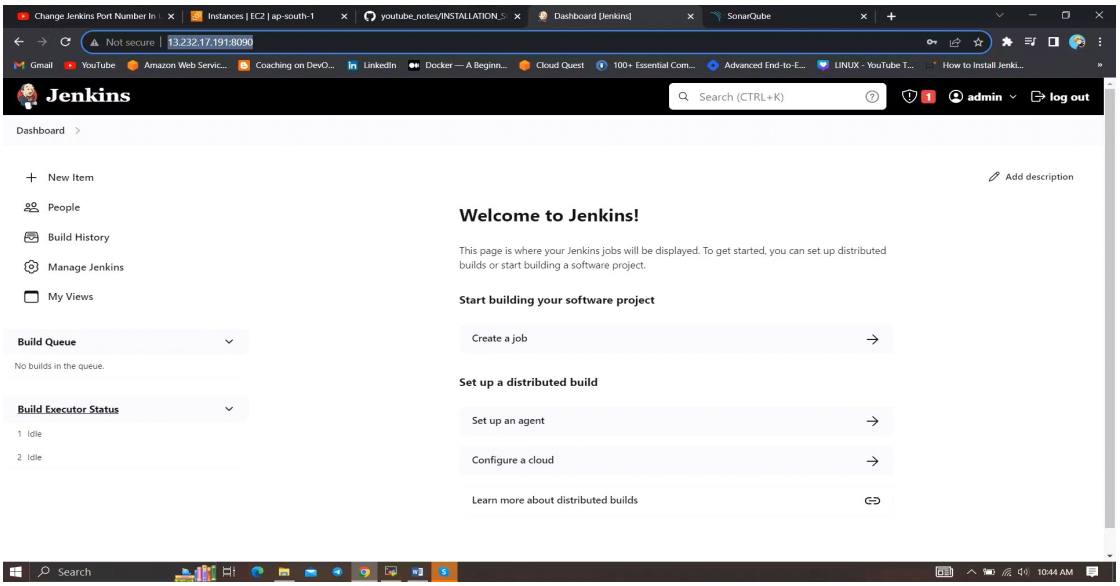
- Install the suggested Plugins.



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



- After the creation of credentials, the the Jenkins dashboard will be open.



INSTALLAZATION OF DOCKER :

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

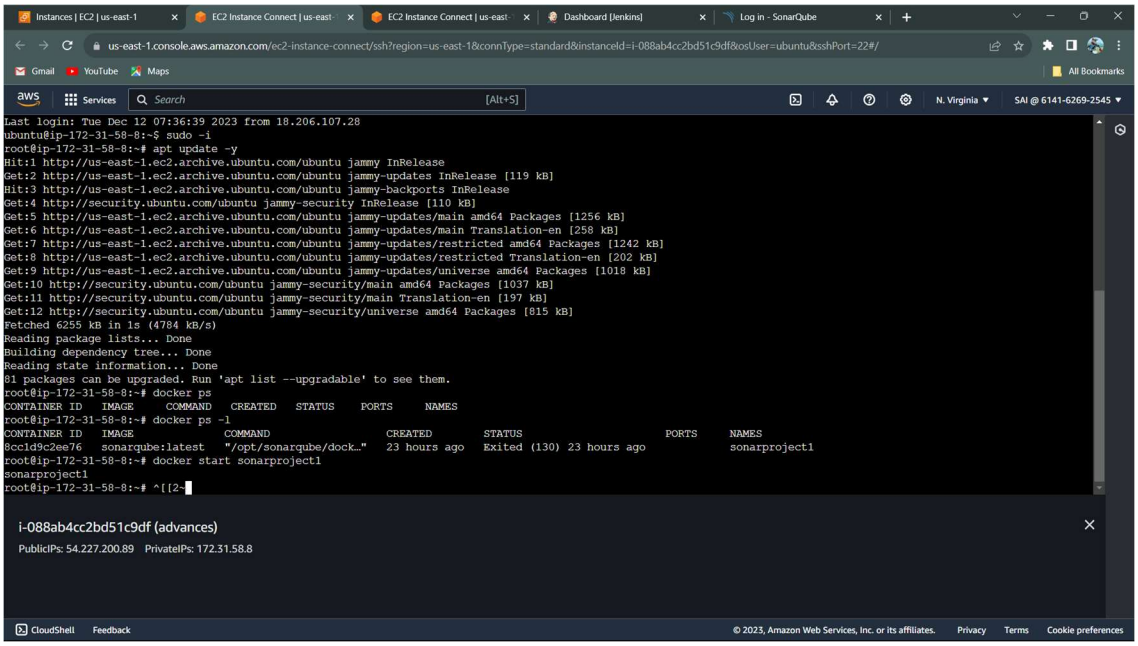
  sudo apt-get update.
  sudo apt-get install docker.io -y

- 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.

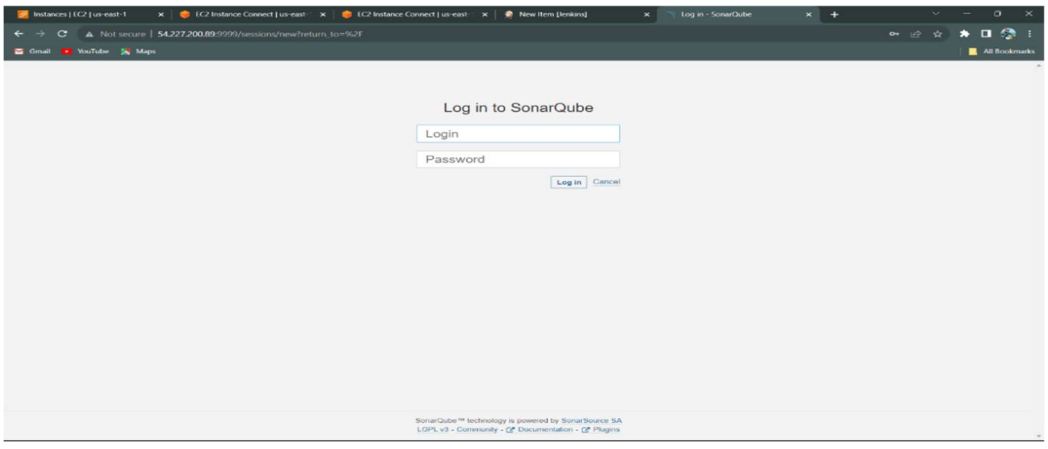
  docker PS

  docker PS -l

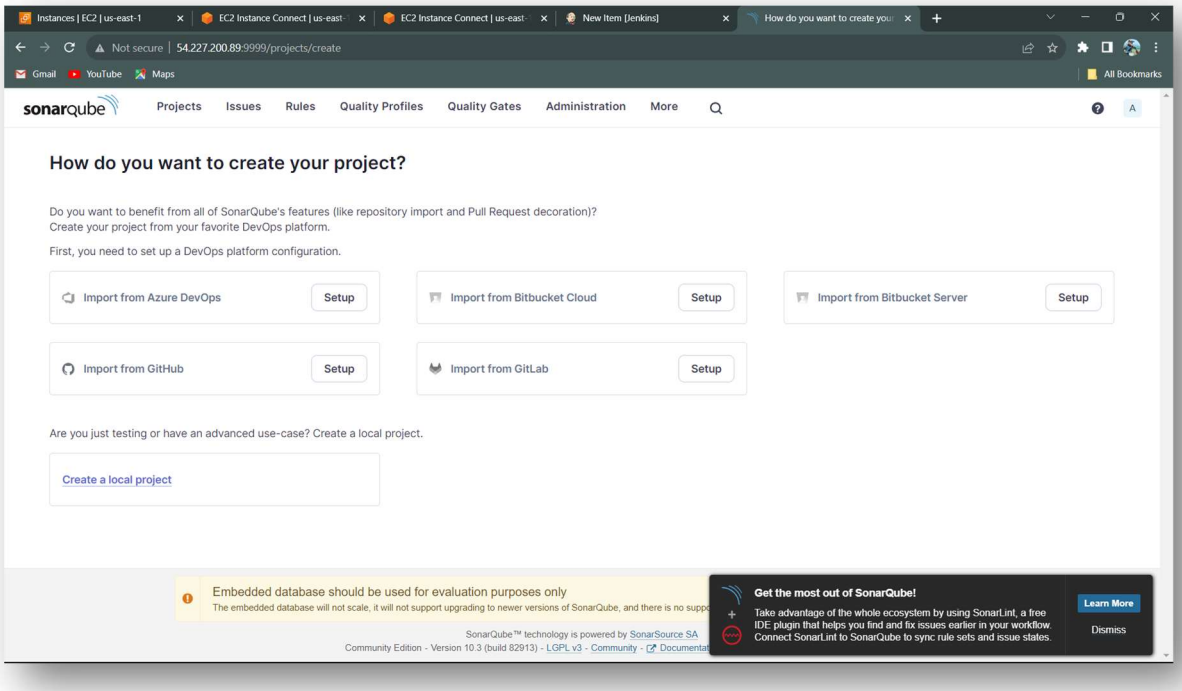


- N

  ow the SonarQube will be running.



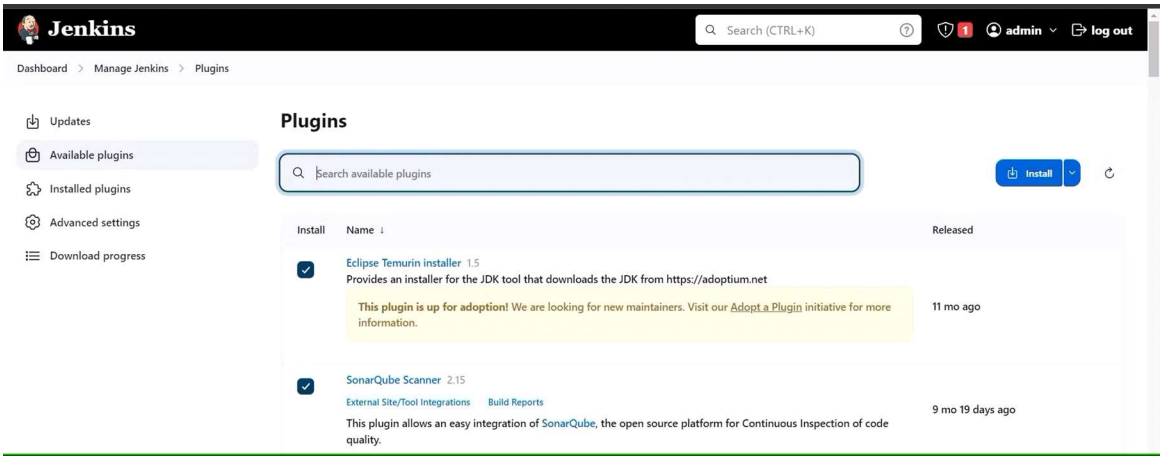
❖ Create the credentials and the sonarqube dashboard will be like this.



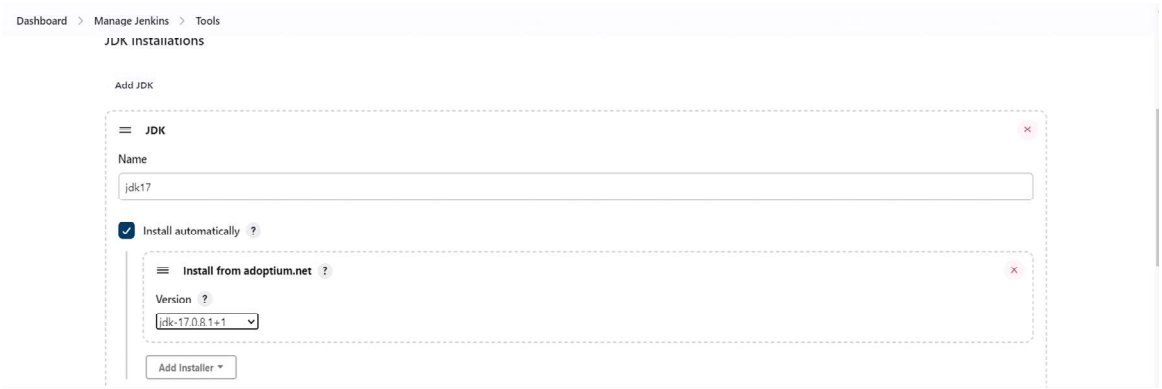
REQUIRED PLUGINS IN THE JENKINS:

❖ Go to Manage Jenkins →Plugins → Available Plugins → Install below plugins :

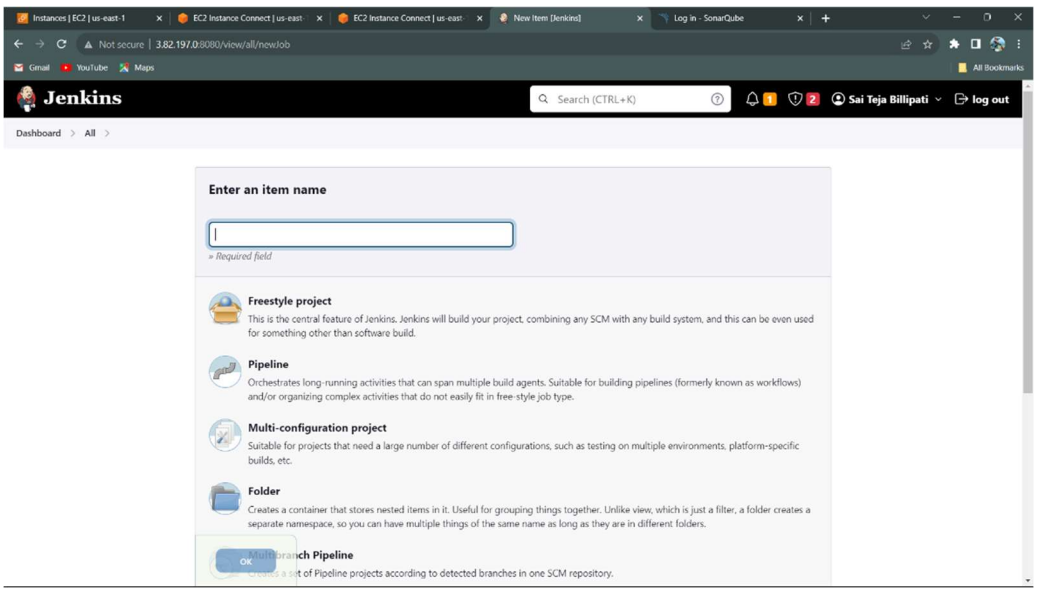
- ✓ Eclipse Temurin Installer
- ✓ SonarQube Scanner



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



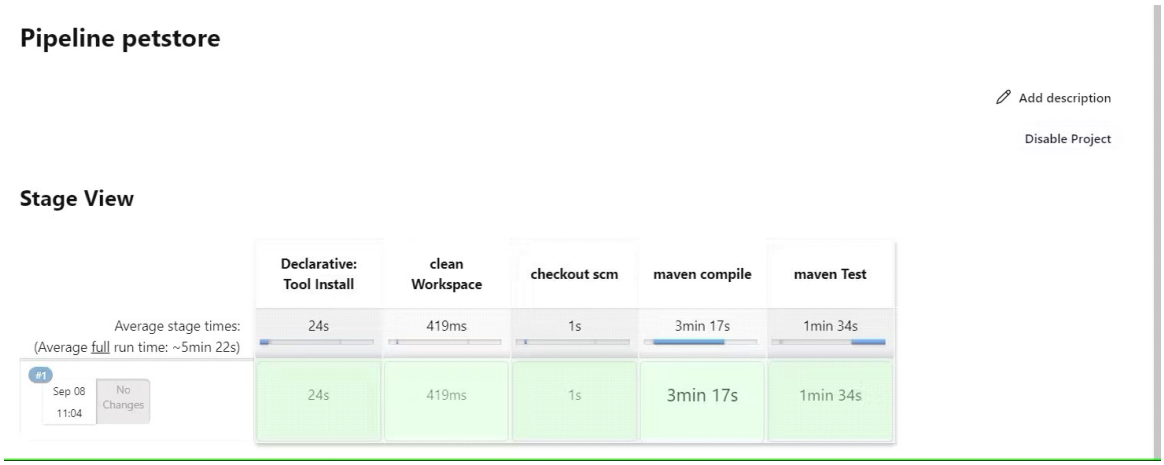
❖ Create a job by taking pipeline project and name it as pet project.



❖ Enter the pipeline script as below.

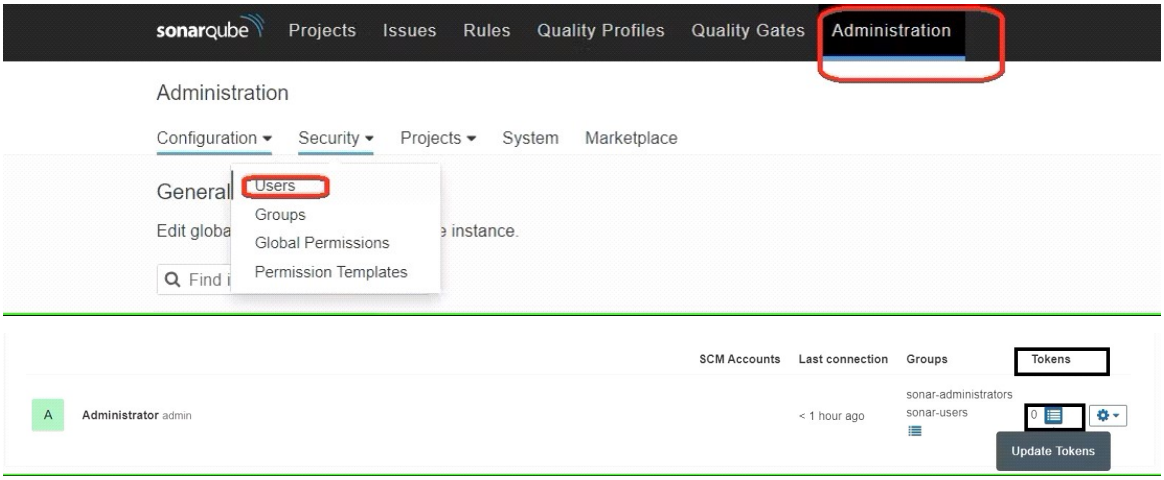
```
pipeline{
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    stages{
        stage ('clean Workspace'){
            steps{
                cleanWs()
            }
        }
        stage ('checkout scm') {
            steps {
                git 'https://github.com/Aj7Ay/jpetstore-6.git'
            }
        }
        stage ('maven compile') {
            steps {
                sh 'mvn clean compile'
            }
        }
        stage ('maven Test') {
            steps {
                sh 'mvn test'
            }
        }
    }
}
```

❖ The build stage view would look like this,

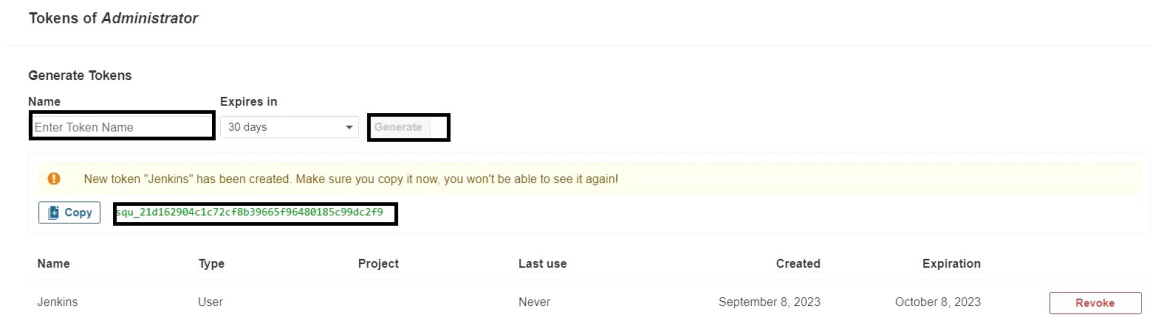


Step 3 - Configure Sonar Server in Manage Jenkins :

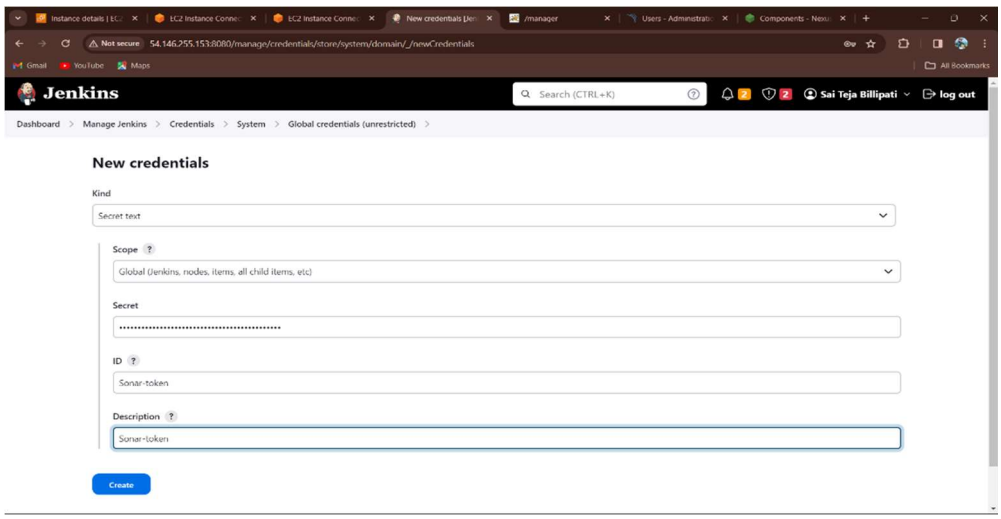
- Create a token with a name and generate.



- copy the Token.



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

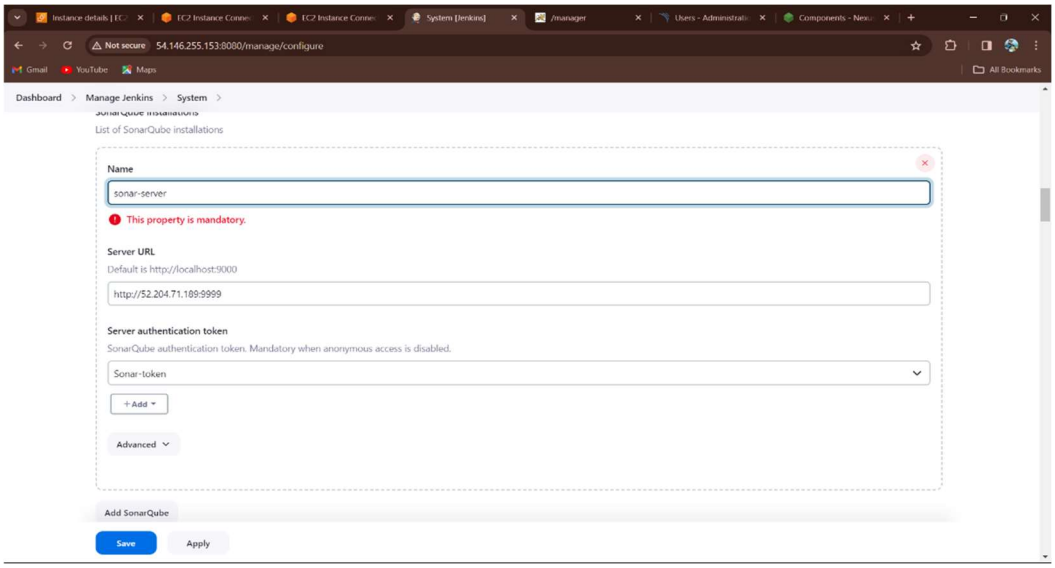


❖ After the creation of credentials, the interface will be like this.

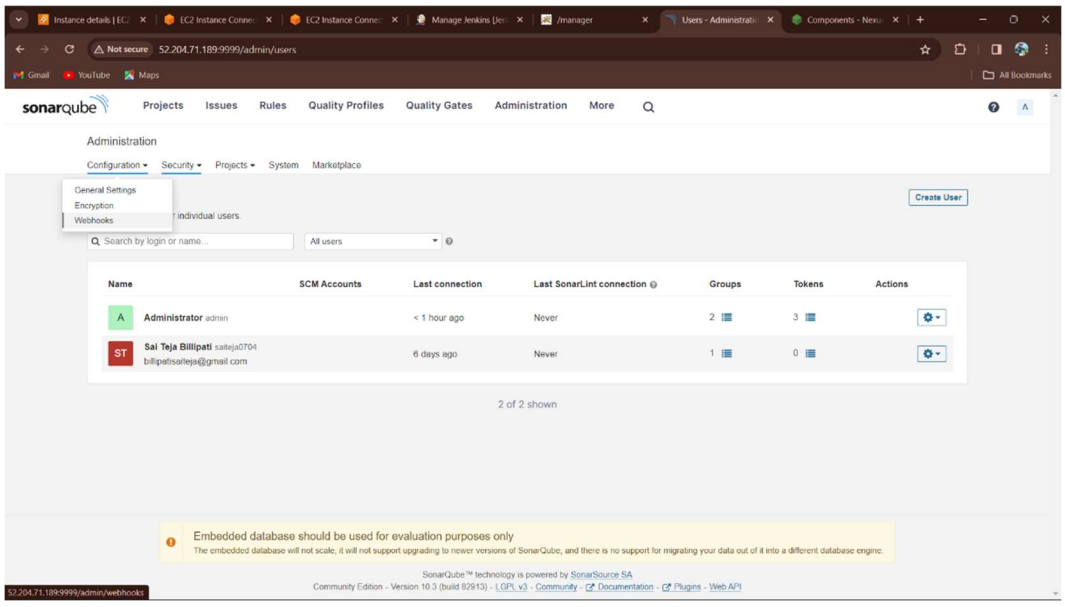
Credentials that should be available irrespective of domain specification to requirements matching.

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.

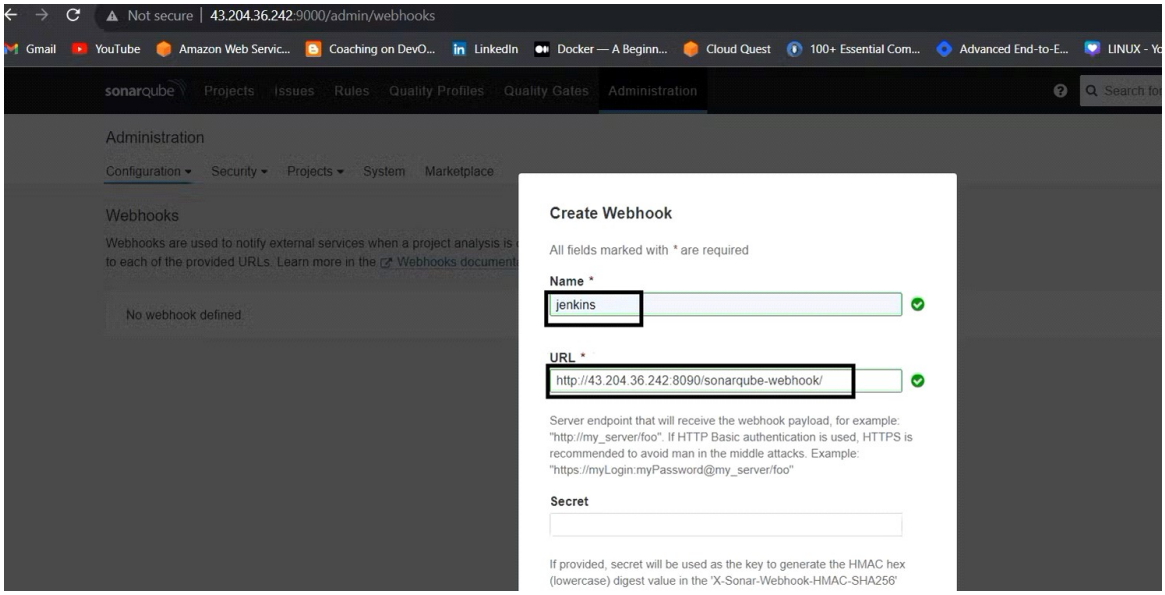


❖ In the SonarQube Dashboard add a quality gate also Administration--> Configuration---->Webhooks.



- Add details #in URL section of quality gate.

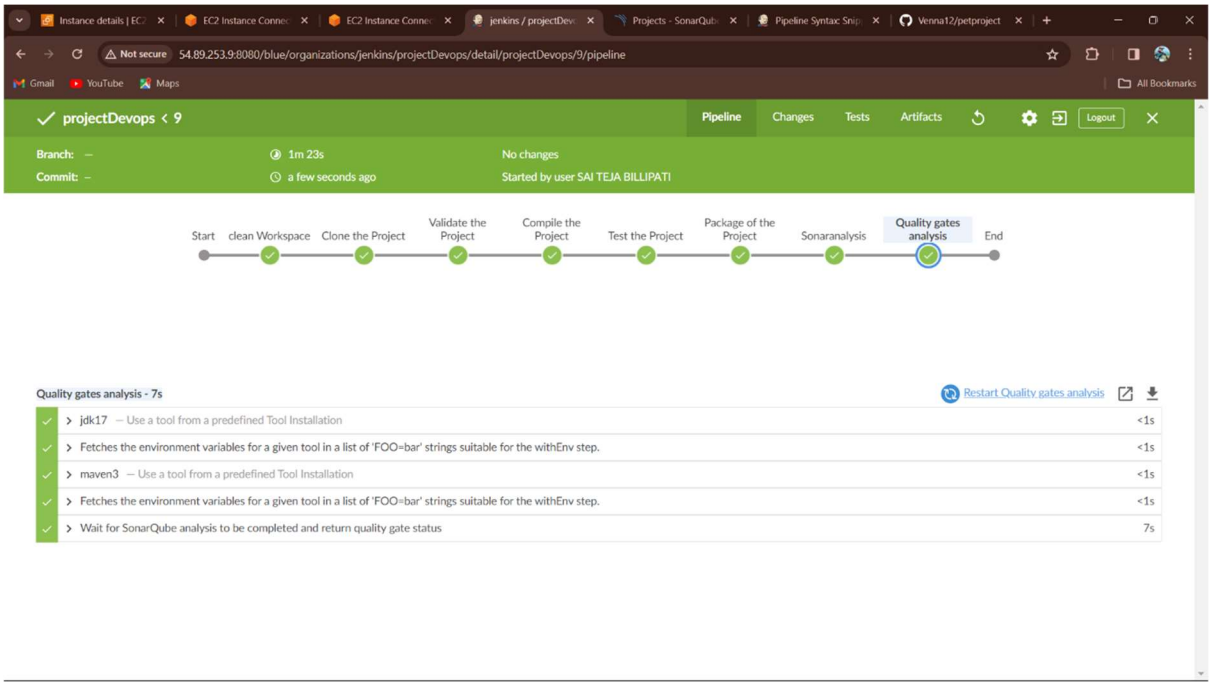
  <http://jenkins-public-ip:8090>/sonarqube-webhook/



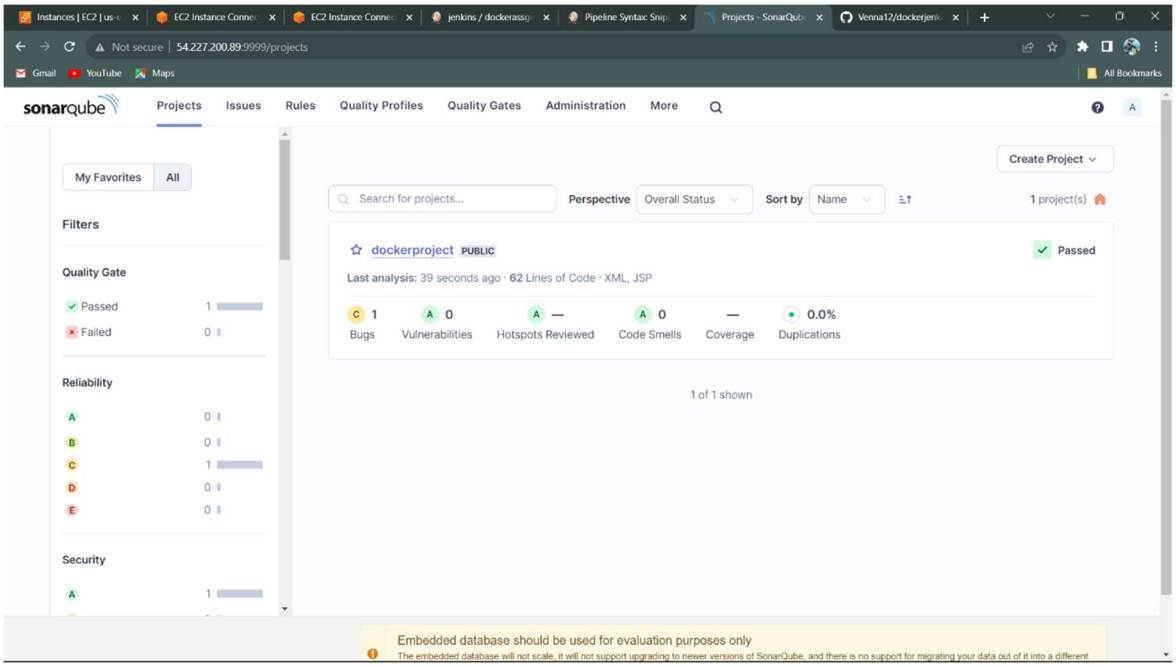
- Let's go to our Pipeline and add SonarQube Stage in our Pipeline Script.

```
#under tools section add this environment
environment {
      SCANNER_HOME=tool 'sonar-scanner'
   }
# in stages add this
stage("SonarQube Analysis "){
         steps{
            withSonarQubeEnv('sonar-server') {
               sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=dockerproject\
               -Dsonar.java.binaries=. \
               -Dsonar.projectKey=dockerproject'''
            }
         }
      }
      stage("quality gate"){
         steps {
            script {
              waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
            }
          }
      }
```

❖ Click on Build now, you will see the stage view like this.

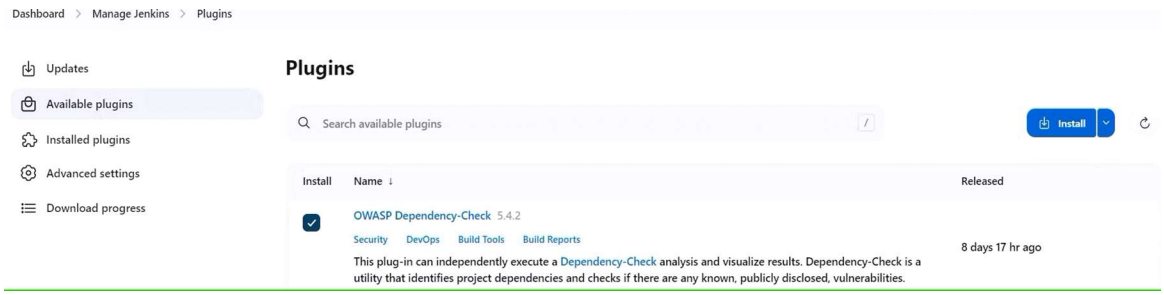


❖ To see the report, you can go to SonarQube Server and go to Projects.

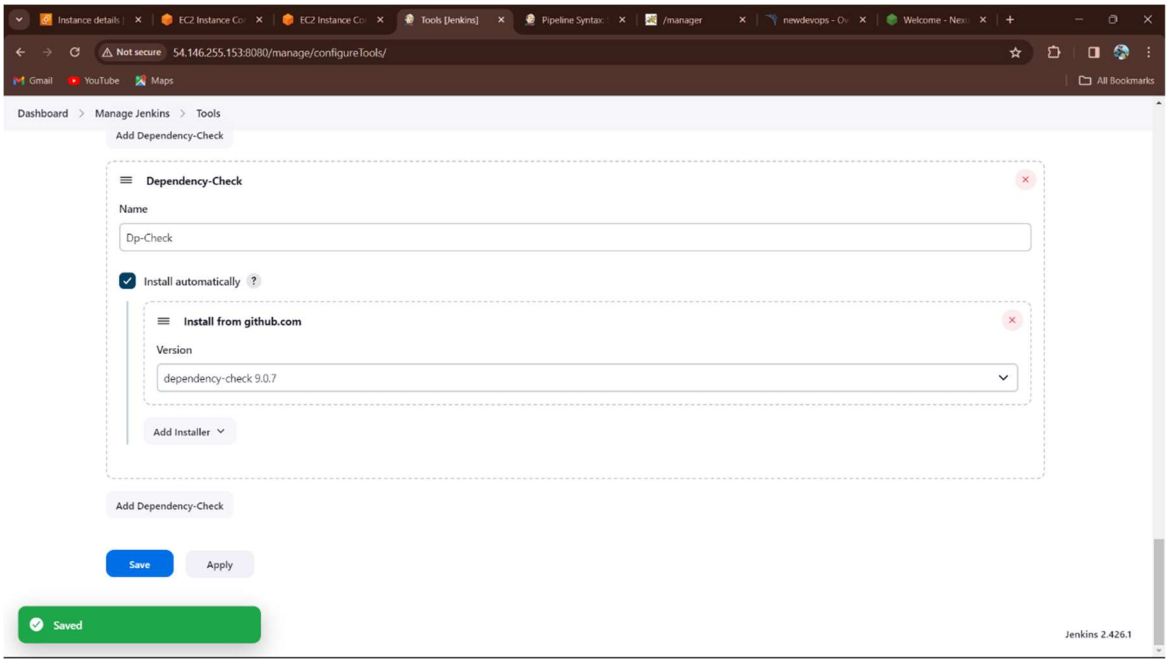


Step 4 — Install OWASP Dependency Check Plugins

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



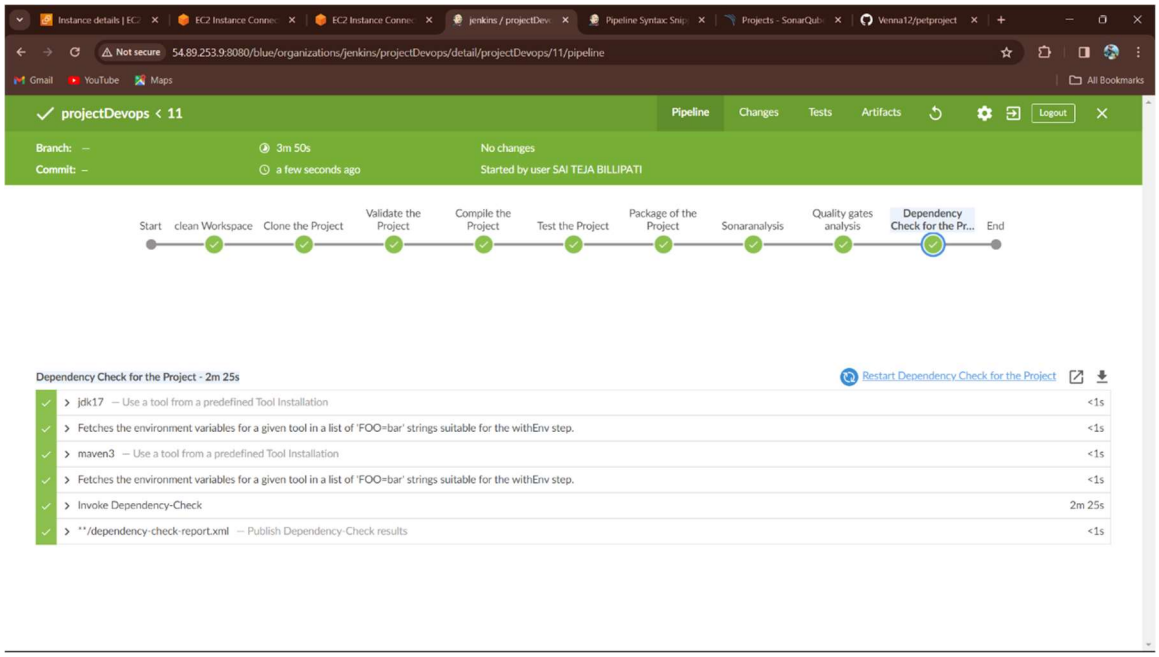
- Go to Dashboard → Manage Jenkins → Tools → Click on Apply and Save here.



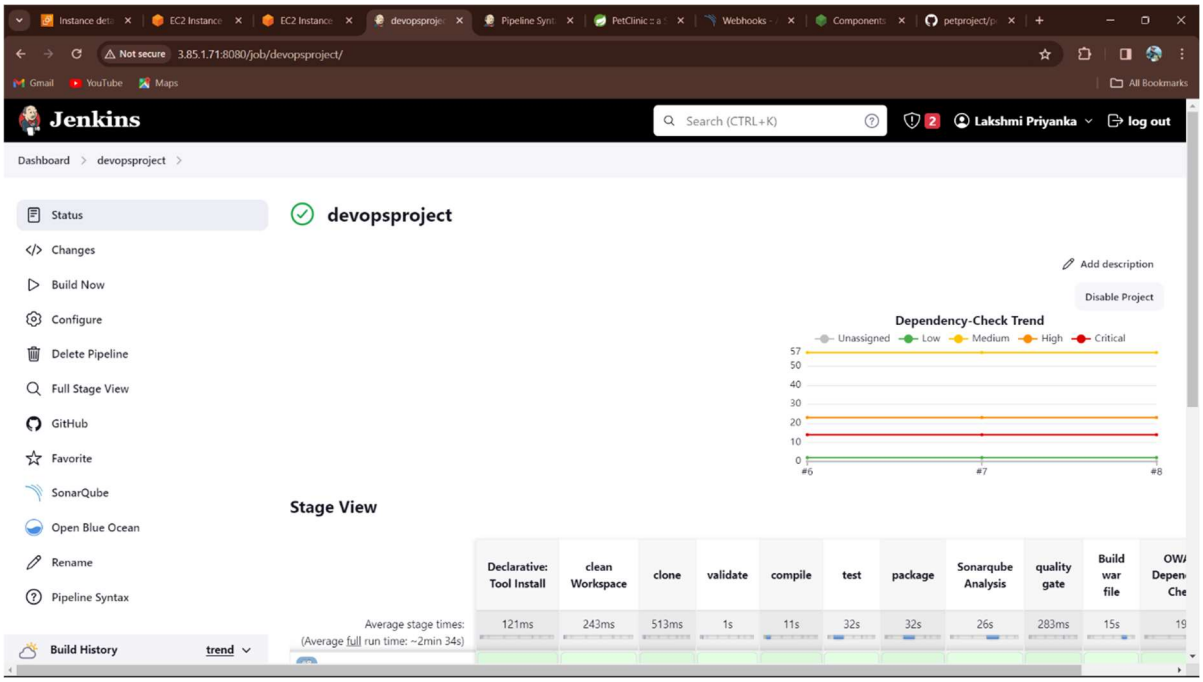
- Now go configure → Pipeline and add this stage to your pipeline and build.

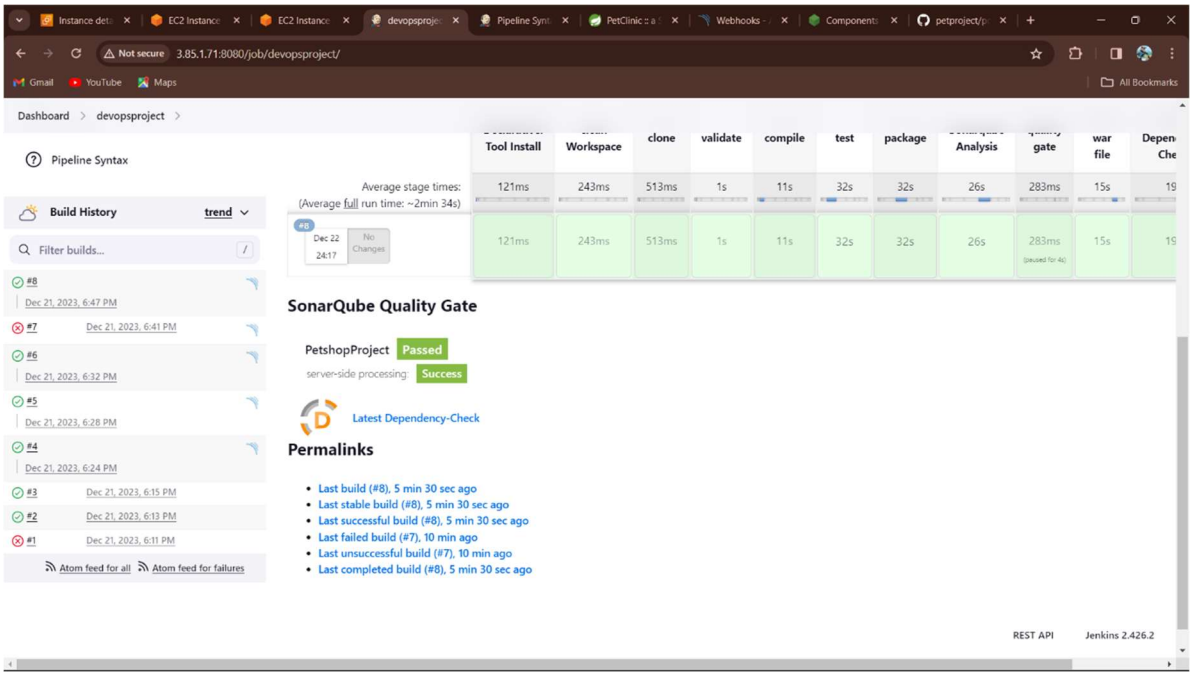
```
stage ('Build war file'){
        steps{
           sh 'mvn package'
        }
    }
    stage("OWASP Dependency Check"){
        steps{
           dependencyCheck additionalArguments: '--scan ./ --format XML ',
odcInstallation: 'DP-Check'
           dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
        }
    }
```

- The stage view would look like this,



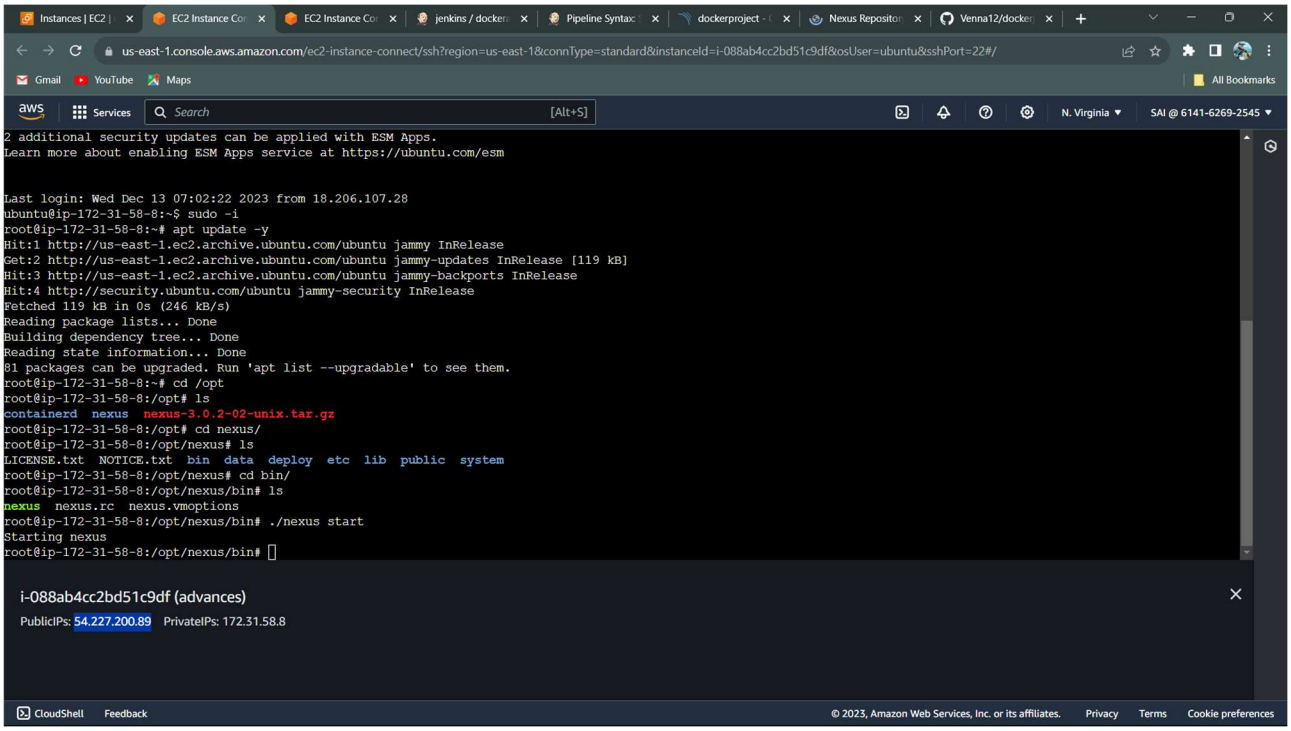
❖ You will see that in status, a graph will also be generated and Vulnerabilities.



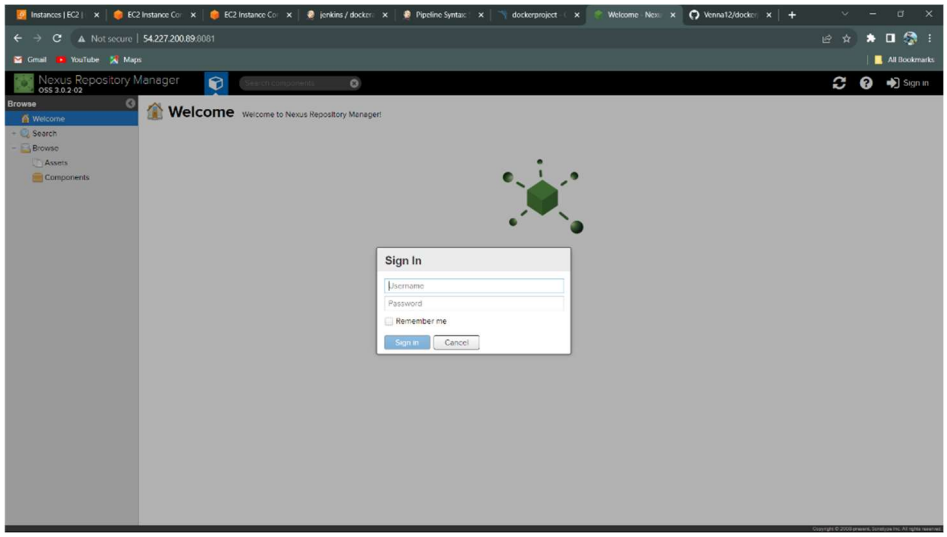


Step 5 — Install NEXUS ARTIFACT

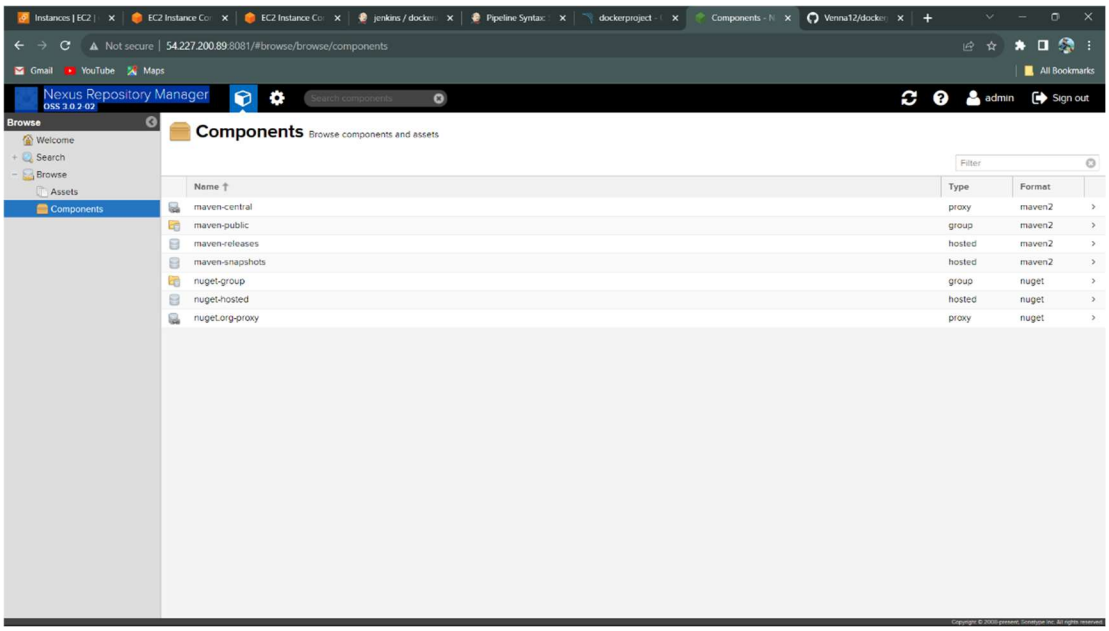
❖ Install the nexus and untar it as below process.



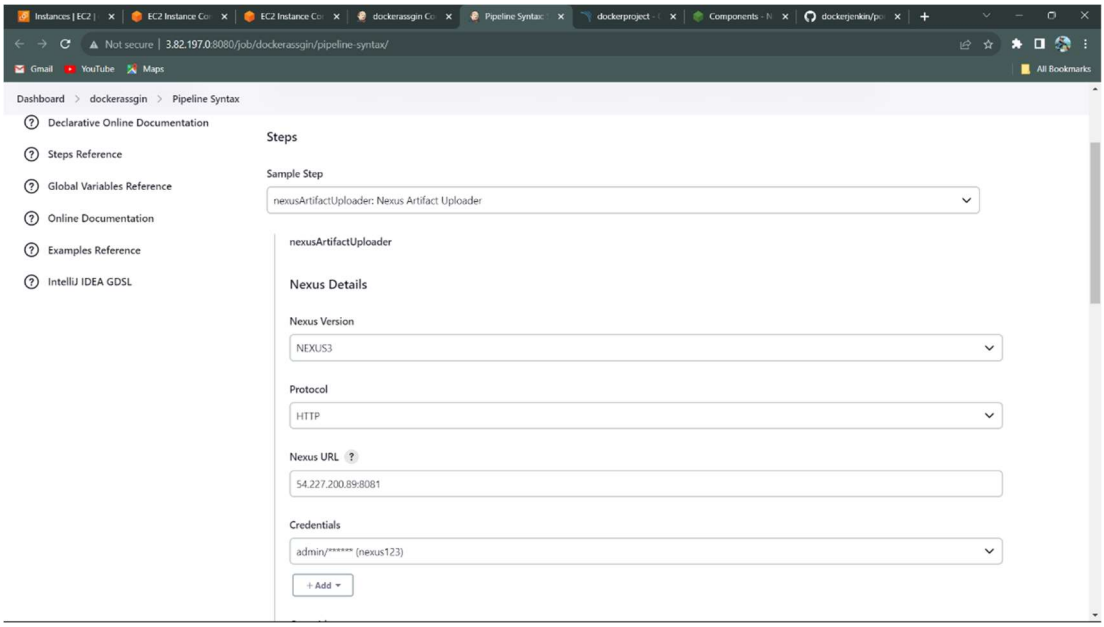
- And login through the nexus **USER:admin & PASSWD:admin**.



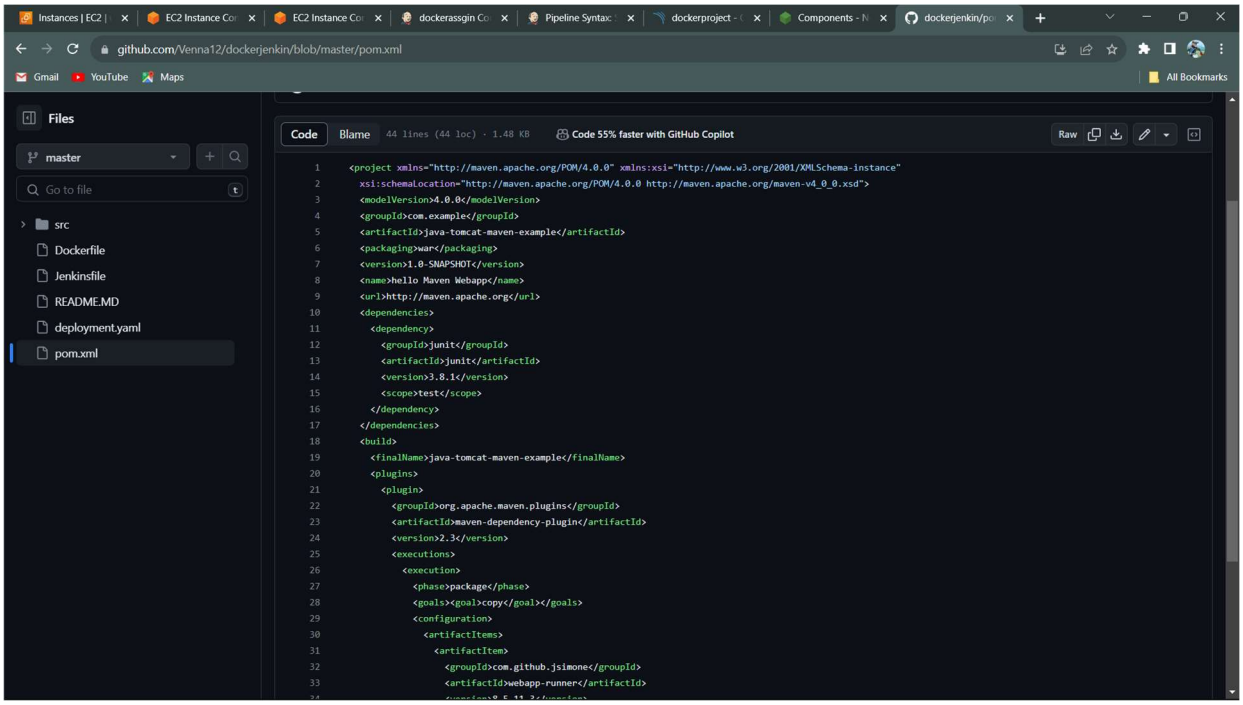
- Go to > Components > maven-Snapshots!



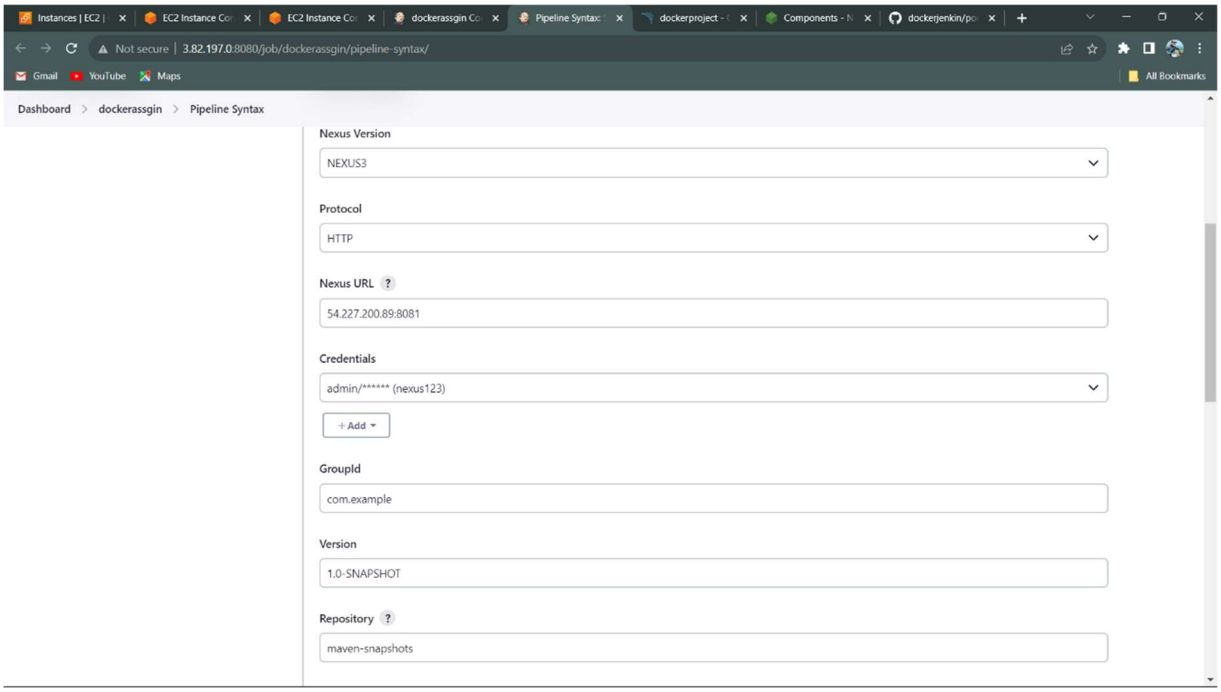
❖ Now go through the Jenkins and in the pipeline, syntax select the **Nexus artifact Uploder.** Fill it in accordance through that required one with the current Ip address of the server.



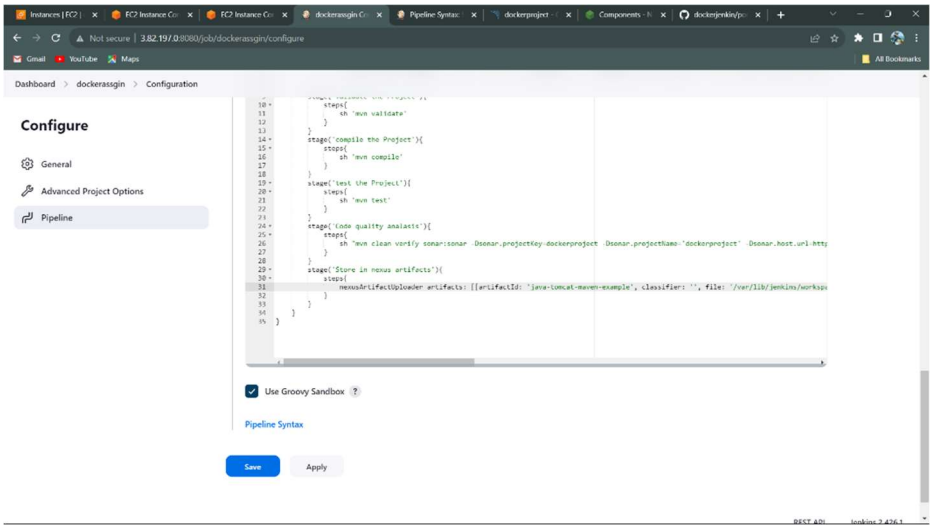
❖ For the Group-id, versions we can see that in the project of the **port.xml.**



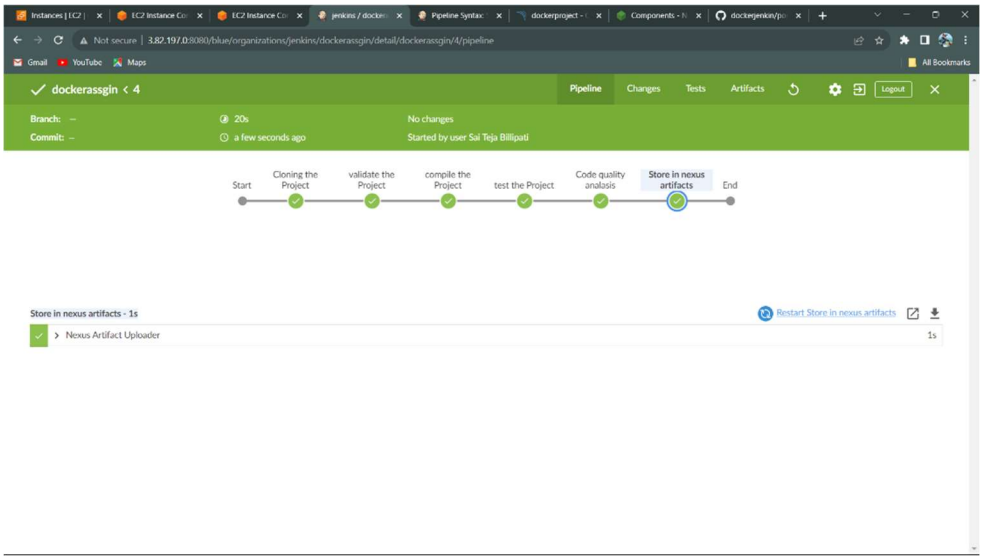
❖ Now we have a below Artifact we have group-id ,version, and repostarty all we can by the port.xml



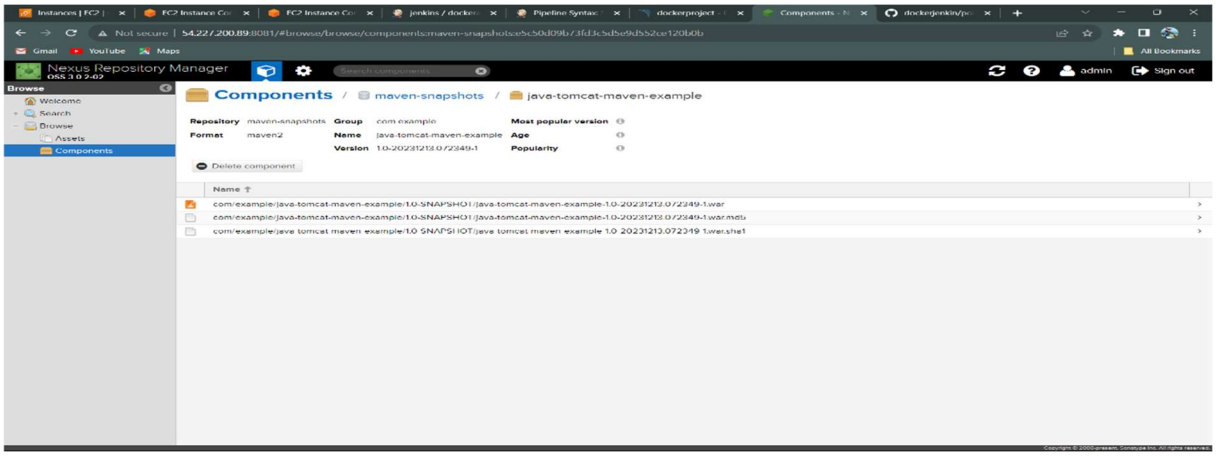
❖ Write the pipeline syntax for the nexus artifact.



- And build it.

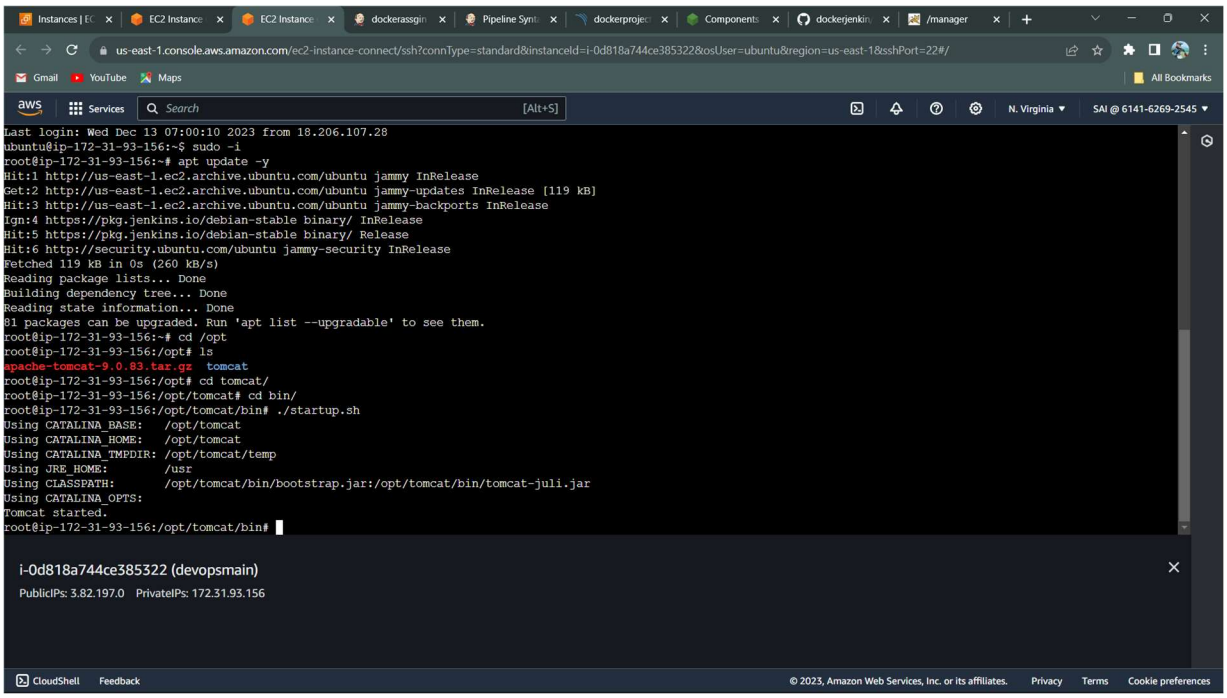


- The required artifact will be generated.

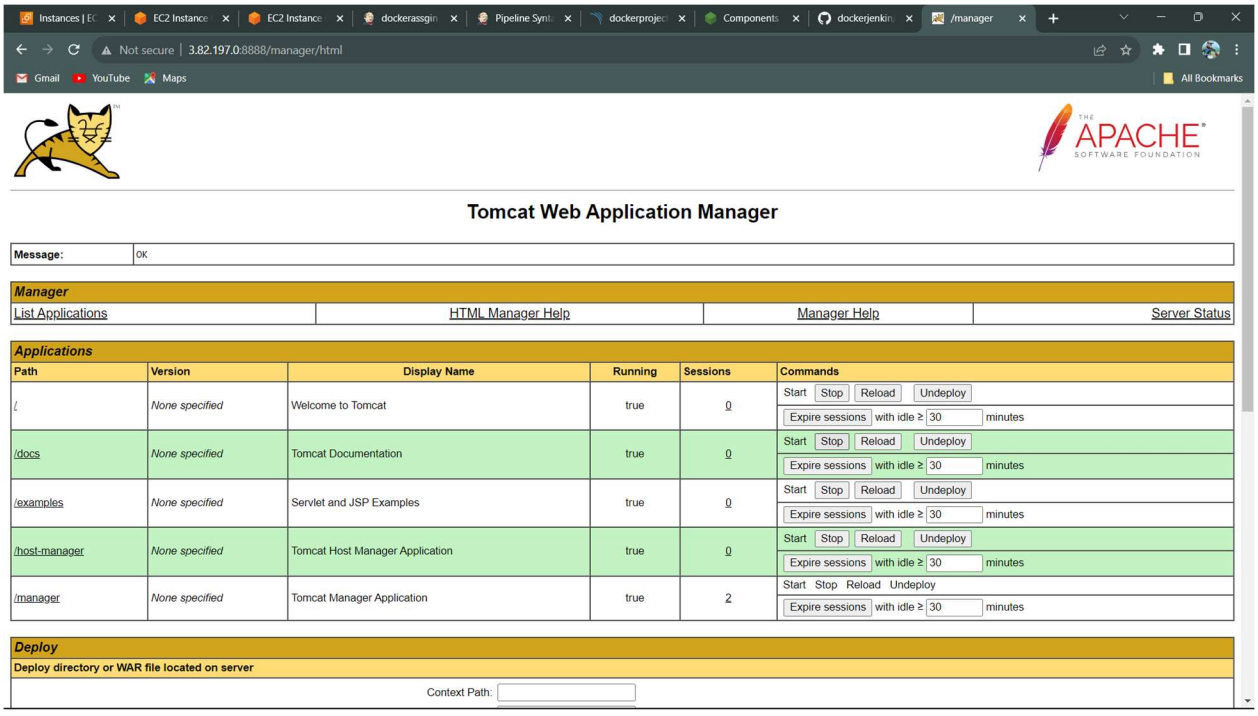


Step 6 — Install TOMCAT and Deploy the project in it.

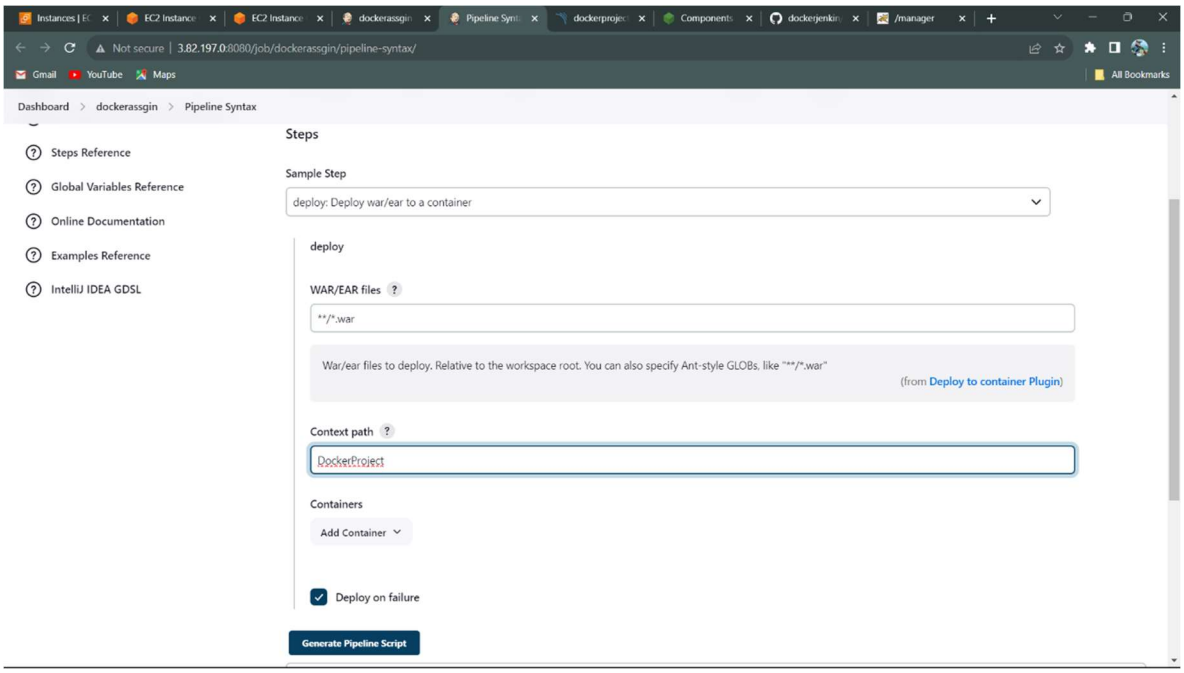
- Installing the tomcat in CD/OPT. Untaring the tomcat and moving the file to the tomcat file and run the tomcat.



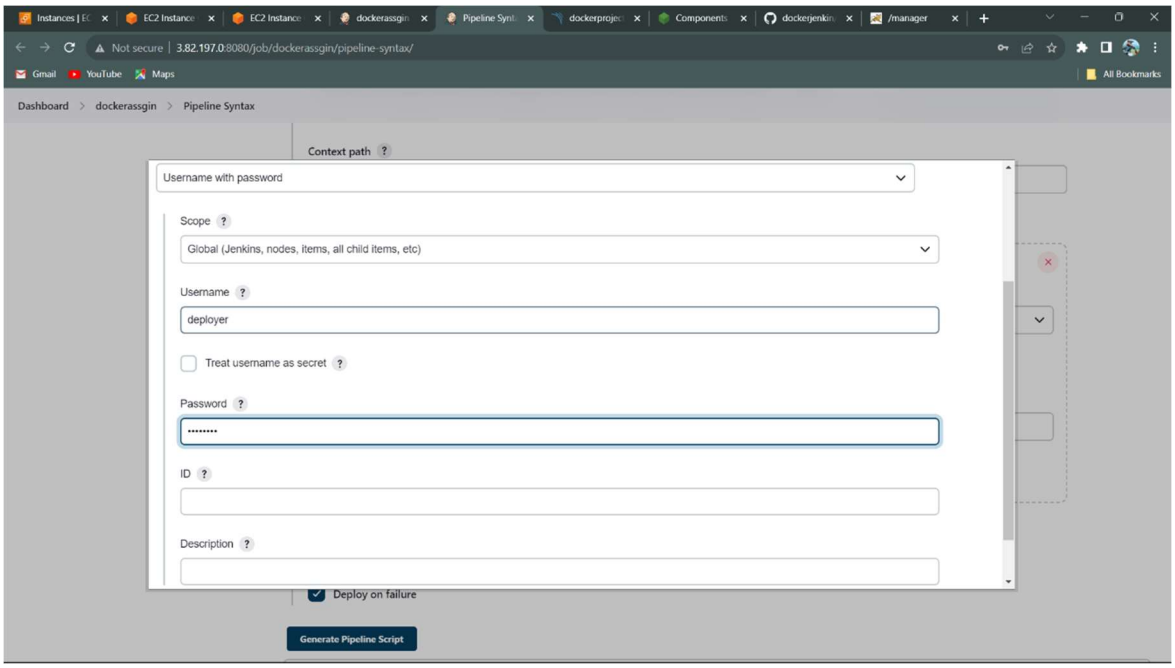
- The tomcat interface will be like this when we run it.



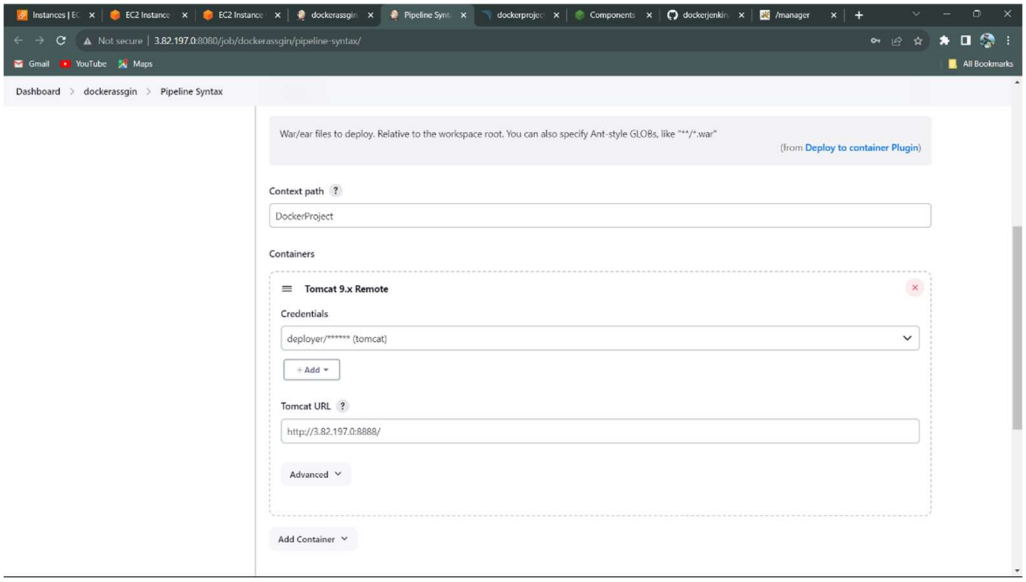
❖ Now go to Jenkins pipeline syntax and select the deploy the container and the context path name it as Docker project.



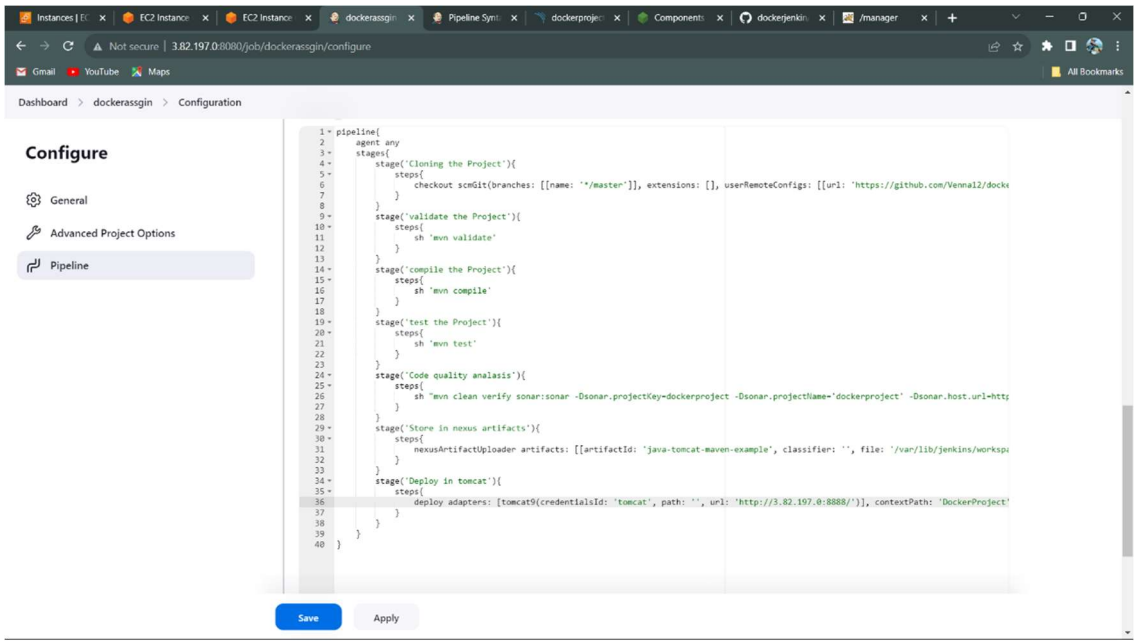
❖ Now go through the add container select the USERNAME AND PASSWORDS and give the tomcat credentials add and save.



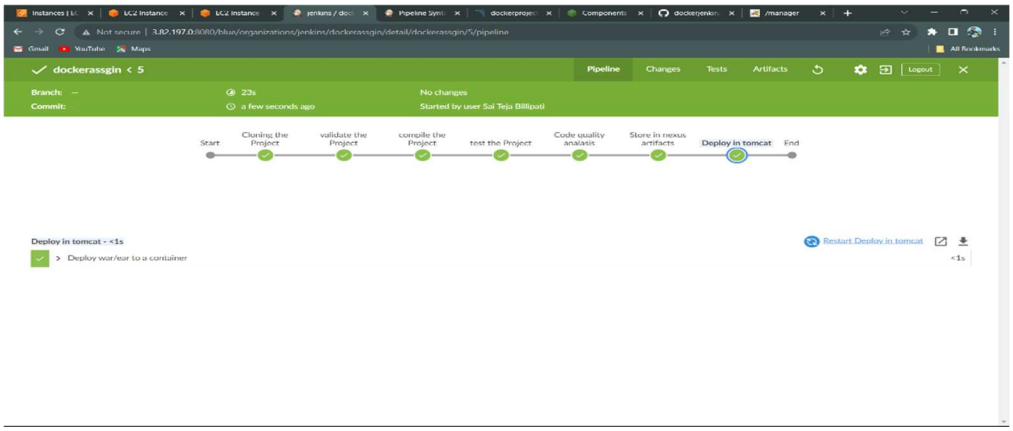
❖ Select the TOMCAT9 container. add the credential's and the URL of the tomcat.



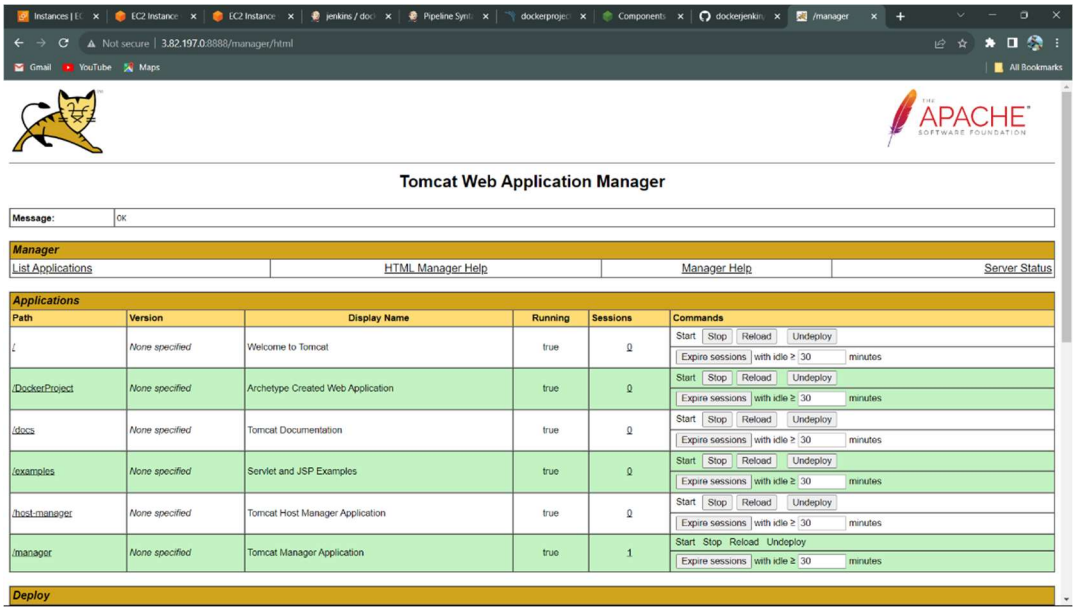
❖ Generate the pipeline for the deployment of tomcat.



❖ Build the tomcat server.



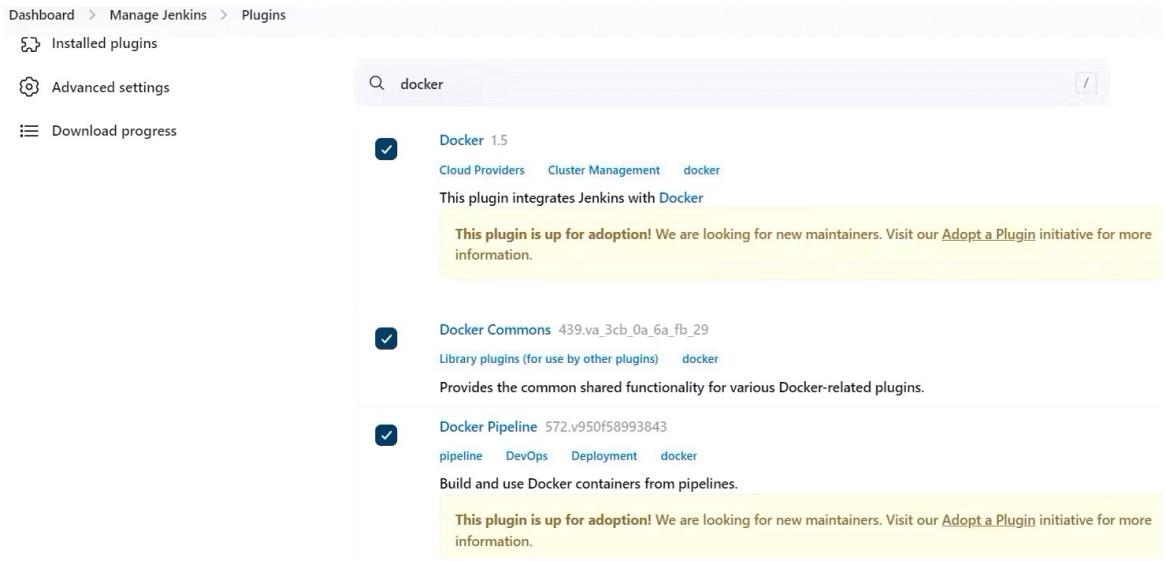
❖ After the deployment we can see the Docker Project in the Tomcat.



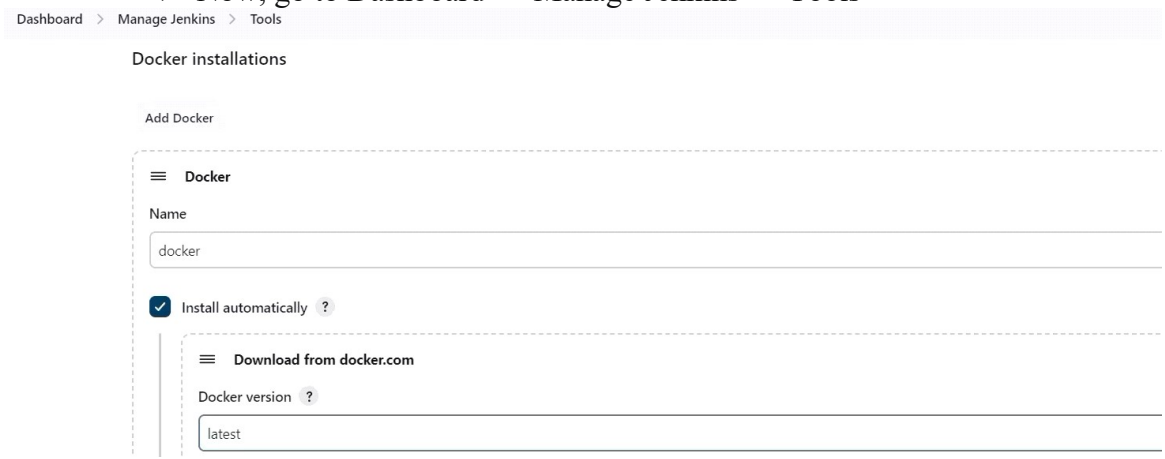
Step 7 — Install DOCKER & Build a Docker Image and push:

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

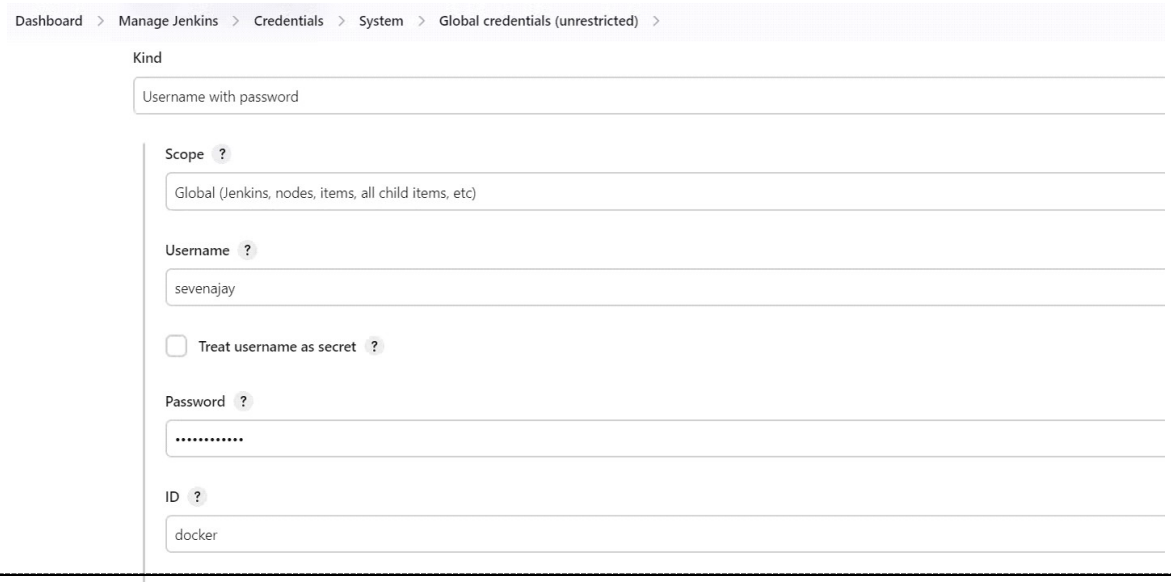
- Docker, Docker commons,Docker pipeline,Docker API,Docker-Build-Step.



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



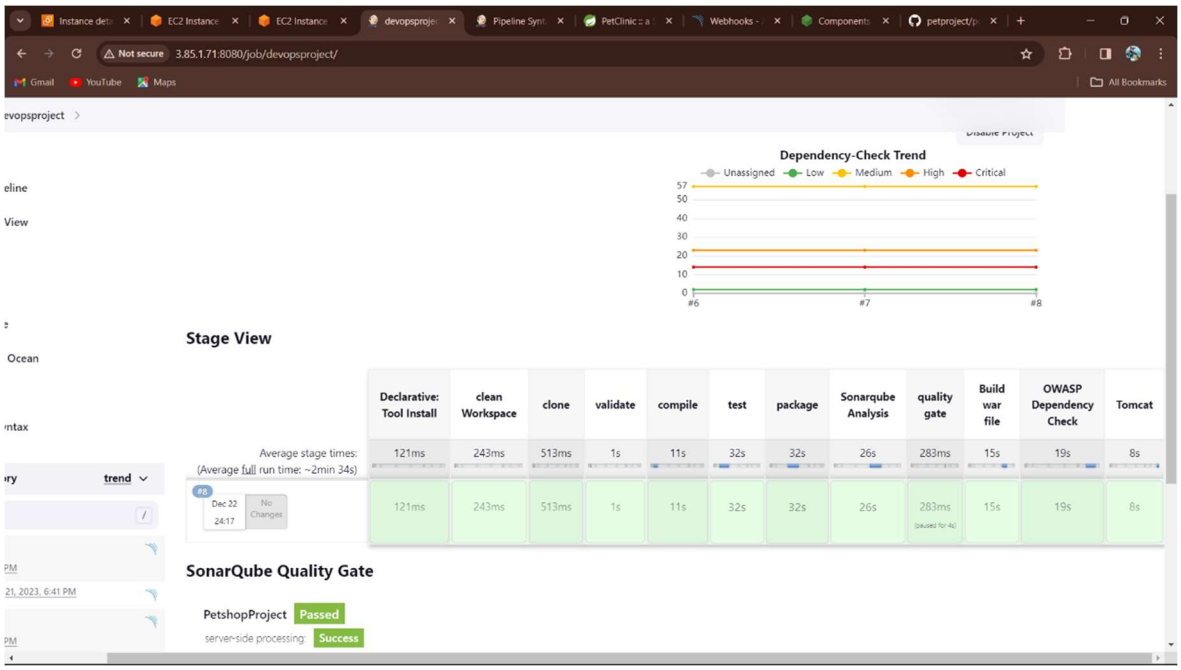
❖ Add Docker Hub Username and Password under Global Credentials



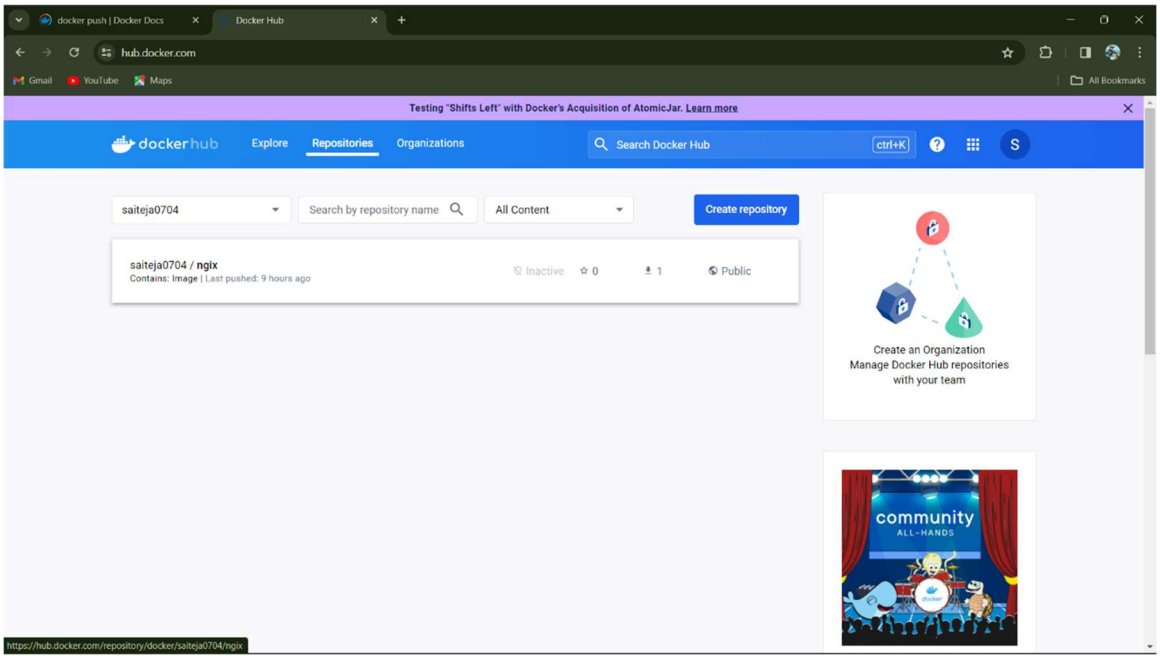
❖ Add this stage to Pipeline Script

```
stage ('Build and push to docker hub'){
        steps{
           script{
              withDockerRegistry(credentials: 'docker', toolName: 'docker') {
                 sh "docker build -t petshop ."
                 sh "docker tag petshop sevenajay/petshop:latest"
                 sh "docker push sevenajay/petshop:latest"
              }
           }
        }
     }
     stage ('Deploy to container'){
        steps{
           sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'
        }
     }
```

❖ You will see the output below, with a dependency trend.



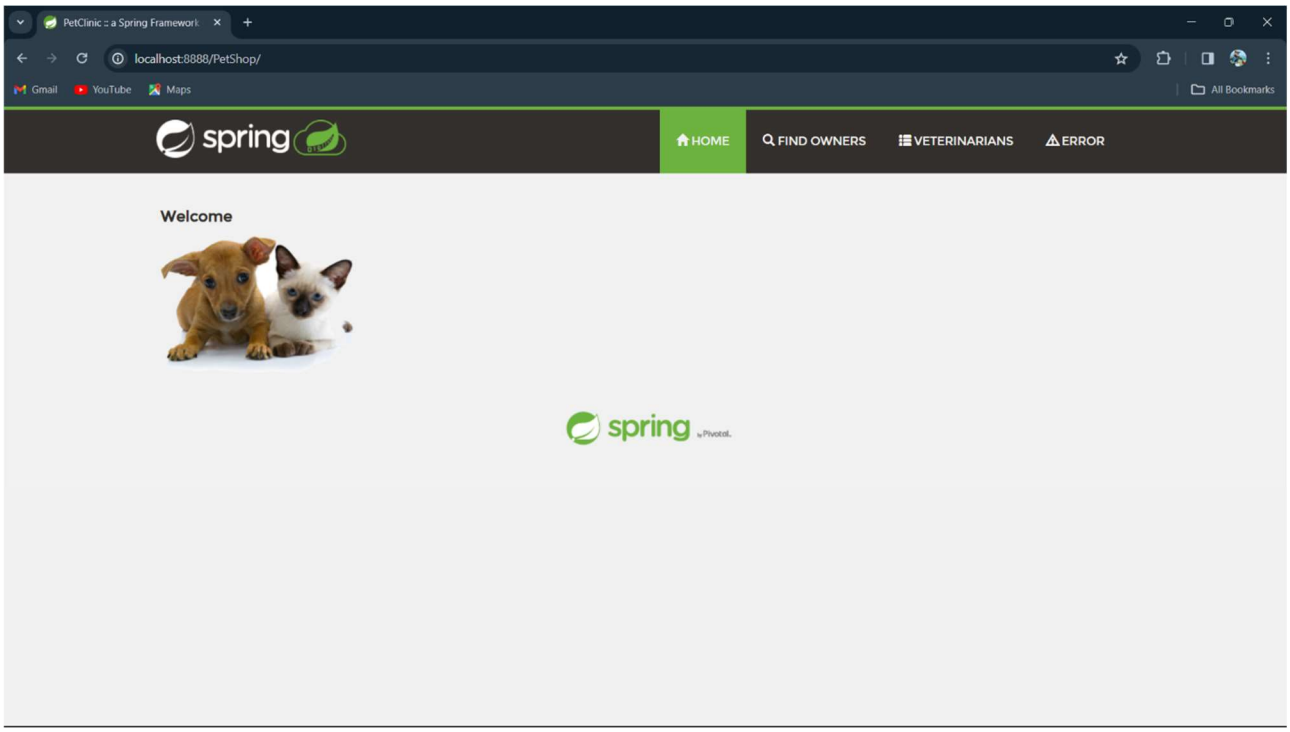
❖ When we login in the Docker we can see a new image has been added.

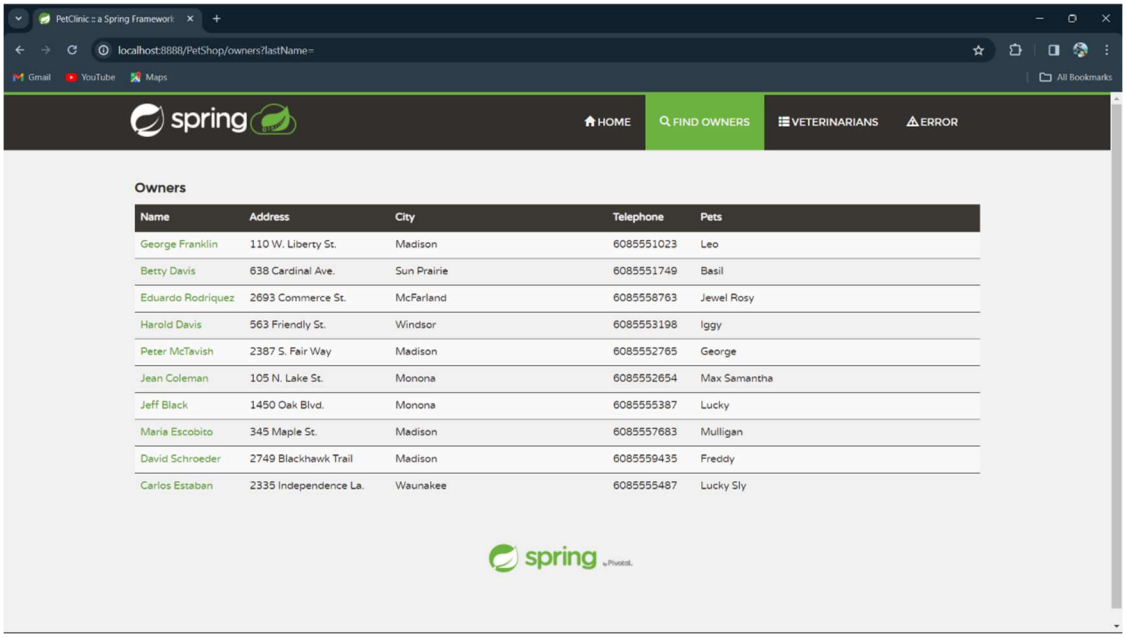


- The Final output can be check through this

  <Ec2-public-ip:8080/Petproject>

❖ FINAL OUTPUT…….





……<<<< END>>>>……..