

Course: DevOps
Module: DOCKER
Topic: DOCKER IMAGE
Trainer Name: Mr. Madhukar sir

Name: Billipati Sai Teja
Mail-ID: billipatisaiteja@gmail.com
Batch no: 115
Project No: 02
Date of submission: 2nd – JAN – 2024

.....

Project Title: DEVSCOPS PROJECT – PET PROJECT

- In this project will deploy the Project using the Jenkins 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.

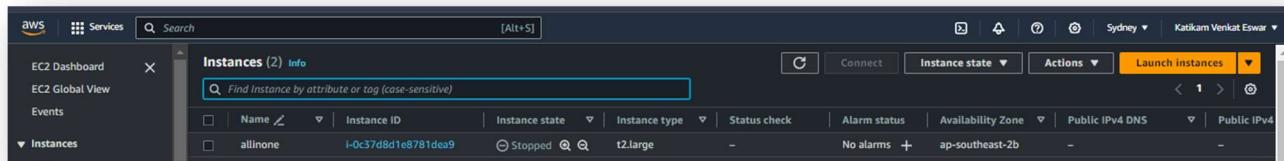
Step 6 — Install tomcat and deploy the project in it.

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 JENKINS AND DOCKER & REQUIRED PLUGINS IN JEKNKINS

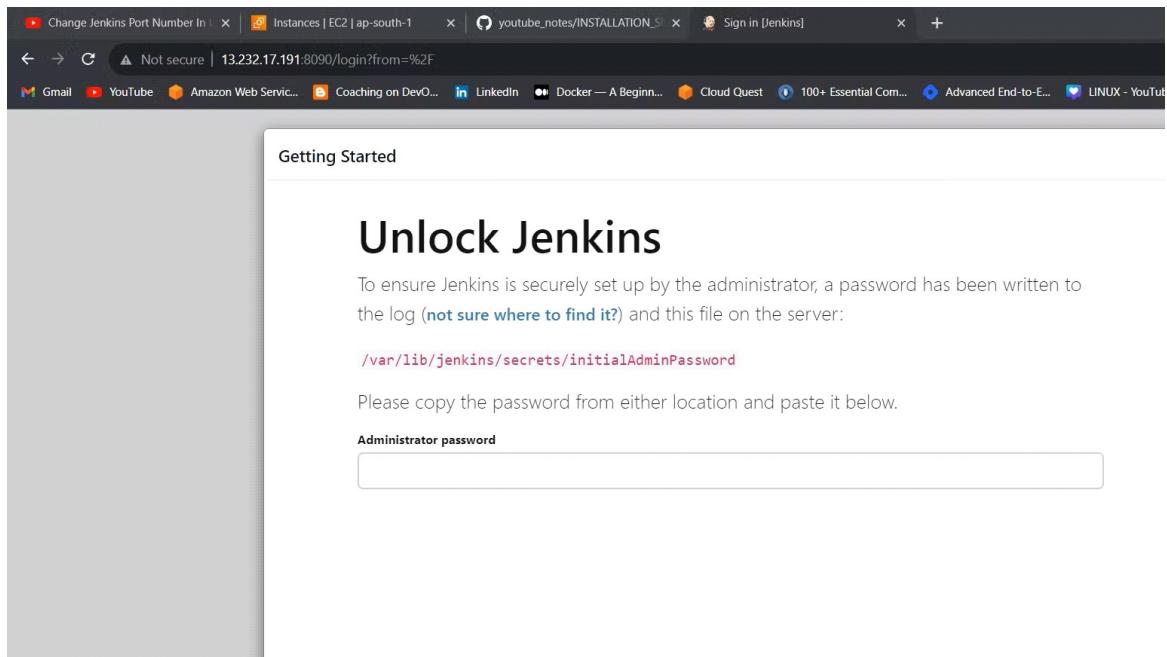
- ❖ For the installation of Jenkins, we can install Jenkins through the terminal by implementing the following steps.
- ❖ We implement these commands by connecting through the console.
- ❖ Execute the below command
vi jenkins.sh
- ❖ Write the code in it.

```
#!/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 install Jenkins
```

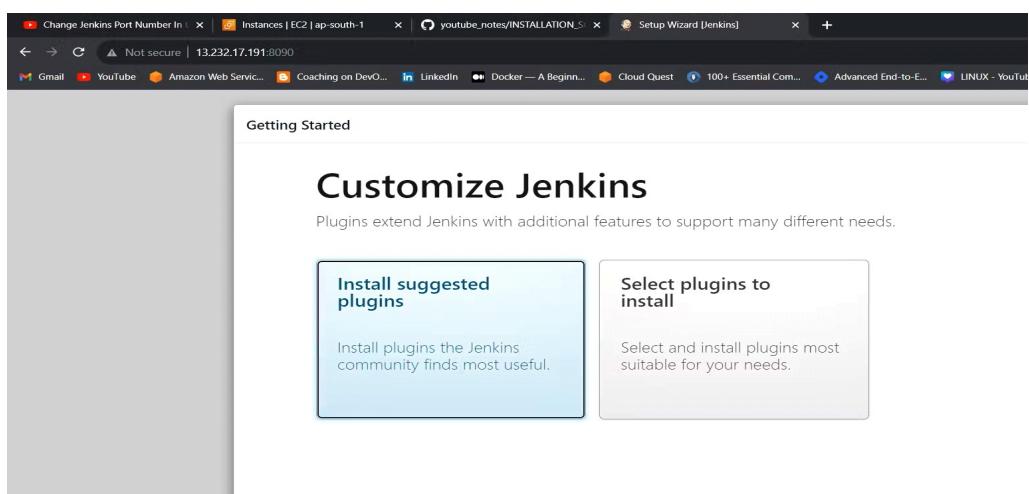
- ❖ Now copy the the Ip address and go through the Jenkins portal the interface will like shown below.



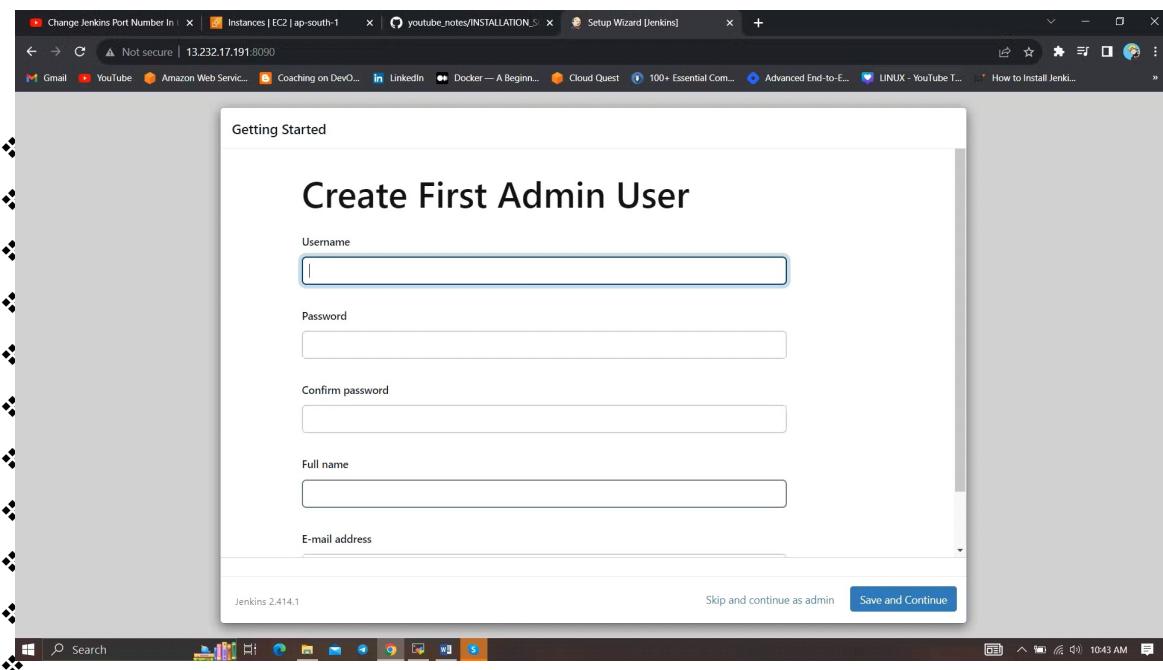
- ❖ Unlock the Jenkins by using command through the given path then we will get the passwd.

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

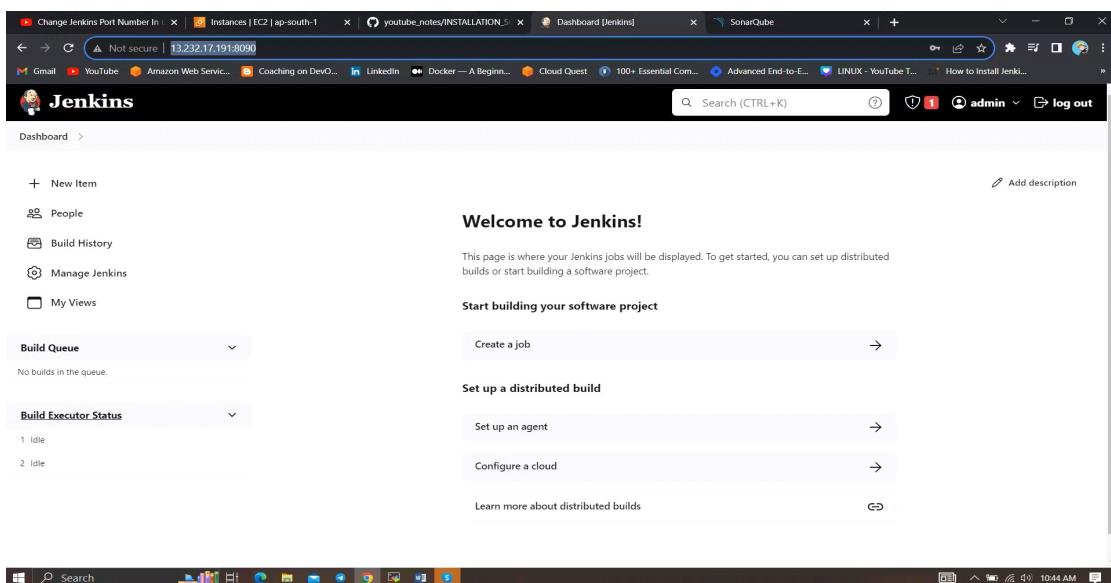
- ❖ Install the suggested Plugins.



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



- ❖ After the creation of credentials, 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:its-community.

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

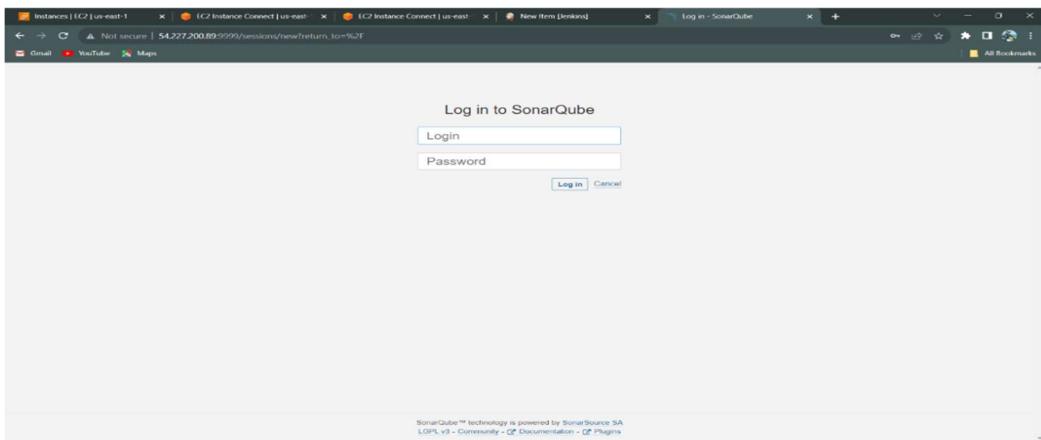
docker PS

docker PS -l

```
Last login: Tue Dec 12 07:36:39 2023 from 18.206.107.28
ubuntu@ip-172-31-58-8:~$ sudo apt update
[...]
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1256 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [258 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1242 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [202 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1018 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1037 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [197 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [815 kB]
Fetched 6255 kB in 1s (4784 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:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
root@ip-172-31-58-8:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
8cc1d92ee76e        sonarqube:latest   "/opt/sonarqube/doc..."   23 hours ago      Exited (130)   23 hours ago          sonarproject1
sonarproject1
root@ip-172-31-58-8:~# ^[[2;1i
i-088ab4cc2bd51c9df (advances)
PublicIPs: 54.227.200.89 PrivateIPs: 172.31.58.8

```

Now the SonarQube will be running.



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

The screenshot shows the SonarQube interface for creating a new project. At the top, there are several tabs: Instances | EC2 | us-east-1, EC2 Instance Connect | us-east-1, EC2 Instance Connect | us-east-1, New Item [Jenkins], and How do you want to create your project?. Below these are links to Gmail, YouTube, and Maps, and a 'All Bookmarks' section. The main content area is titled 'How do you want to create your project?'. It contains a note: 'Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.' Below this, it says 'First, you need to set up a DevOps platform configuration.' and lists five import options with 'Setup' buttons: Import from Azure DevOps, Import from Bitbucket Cloud, Import from Bitbucket Server, Import from GitHub, and Import from GitLab. Underneath these, a note says 'Are you just testing or have an advanced use-case? Create a local project.' followed by a 'Create a local project' button. At the bottom, there is a warning about using an embedded database for evaluation purposes, a note about SonarQube's technology stack, and a 'Get the most out of SonarQube!' sidebar with a 'Learn More' link and a 'Dismiss' button.

REQUIRED PLUGINS IN THE JENKINS:

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

The screenshot shows the Jenkins 'Manage Jenkins' → 'Plugins' page. The sidebar on the left includes 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area is titled 'Plugins' and features a search bar with 'Search available plugins'. Below the search bar, there are two listed plugins: 'Eclipse Temurin installer' (version 1.5) and 'SonarQube Scanner' (version 2.15). Both plugins have a checked 'Install' button next to them. The 'Eclipse Temurin installer' plugin has a note: 'This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt-a-Plugin initiative for more information.' and was released 11 months ago. The 'SonarQube Scanner' plugin has a note: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.' and was released 9 months and 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 | EC2 Instance Connect | us-east-1 | EC2 Instance Connect | us-east-1 | New Item [jenkins] | Log in - SonarQube | + | +

Gmail YouTube Maps

Jenkins

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 builds.

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.

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

OK

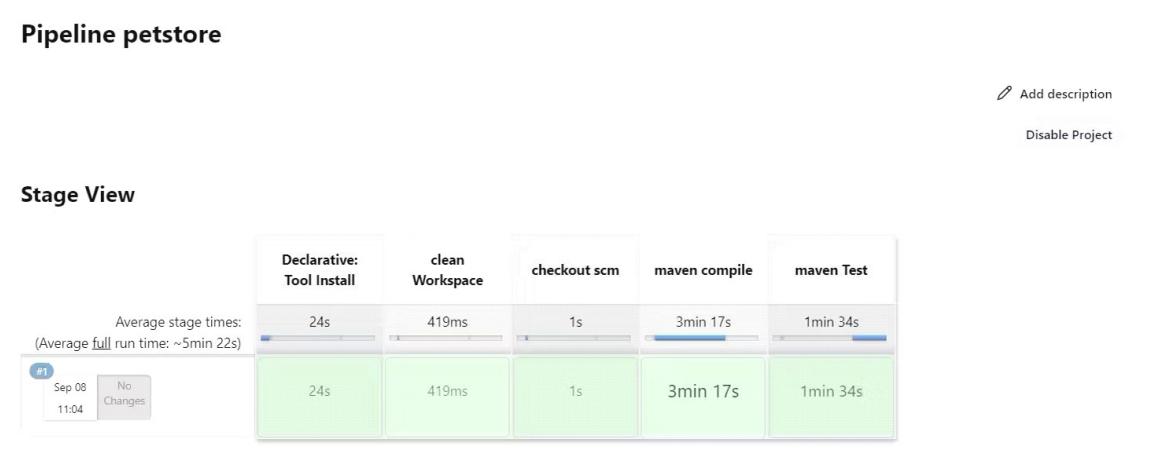
- ❖ Enter the pipeline script as below.

```

pipeline{
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    stages{
        stage ('clean Workspace'){
            steps{
                cleanWs()
            }
        }
        stage ('checkout scm') {
            steps {
                git 'https://github.com/Aj7Av/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.

The screenshot shows the SonarQube Administration interface. The top navigation bar includes links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration link is highlighted with a red box. Below the navigation, there's a sub-navigation bar with Configuration, Security, Projects, System, and Marketplace. A dropdown menu for 'Users' is open, also highlighted with a red box. The main content area displays user information for 'Administrator admin'. It includes columns for SCM Accounts, Last connection, Groups, and Tokens. A 'Tokens' button is highlighted with a red box. At the bottom right, there's a 'Update Tokens' button.

- ❖ copy the Token.

The screenshot shows the 'Tokens of Administrator' page. It has a 'Generate Tokens' section where a new token named 'Jenkins' is being created for 30 days. A success message indicates the token has been created. Below this, a table lists the generated token. The token value is highlighted with a red box. The table columns are Name, Type, Project, Last use, Created, and Expiration. A 'Revoke' button is visible at the end of the row.

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

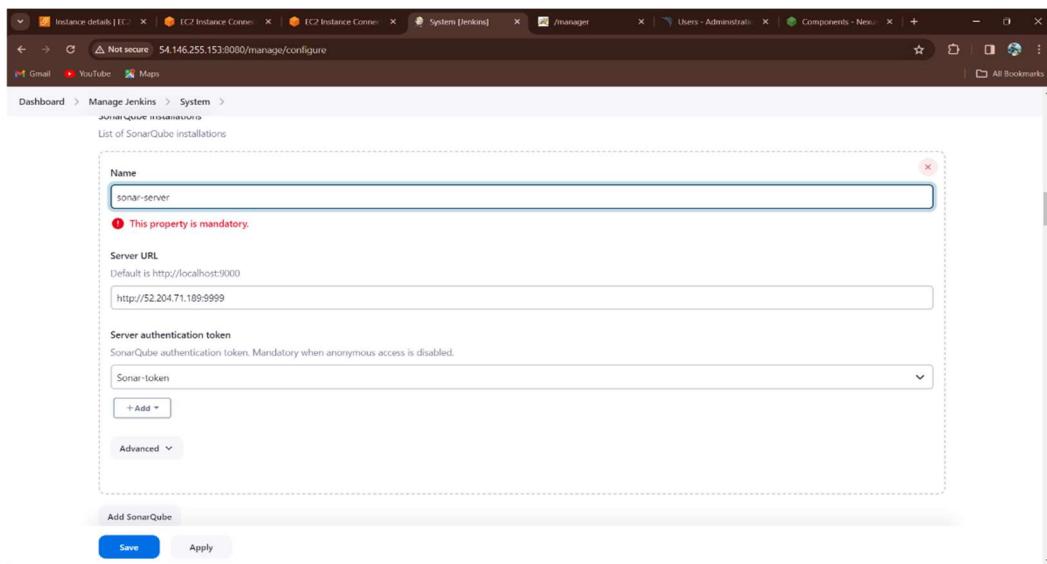
The screenshot shows the Jenkins 'New credentials' configuration page. The 'Kind' field is set to 'Secret text'. The 'Scope' field is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains the copied Sonar token. The 'ID' field is set to 'Sonar-token'. The 'Description' field also contains 'Sonar-token'. A 'Create' button is at the bottom left.

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

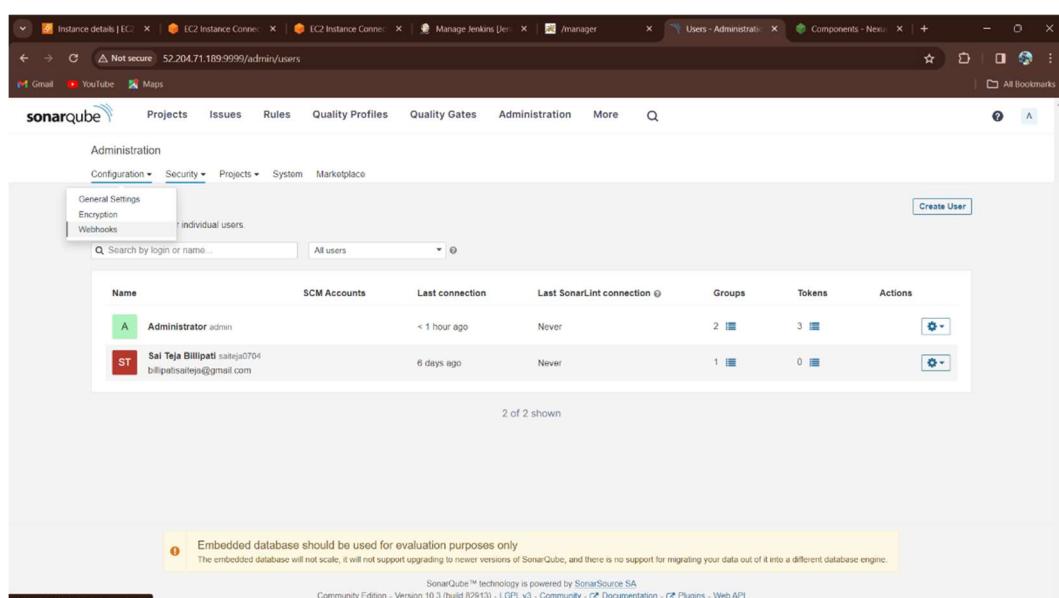
Credentials that should be available irrespective of domain specification to requirements matching.

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.

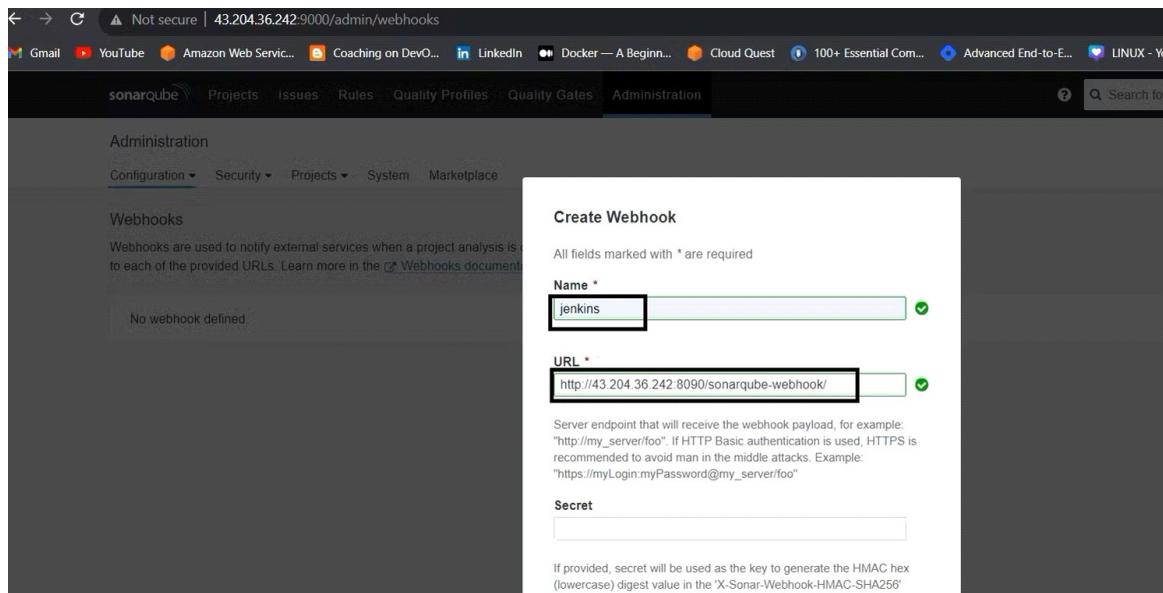


- ❖ In the SonarQube Dashboard add a quality gate also Administration--> Configuration--->Webhooks.



- ❖ 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.

The screenshot shows a Jenkins pipeline stage view. At the top, there is a green header bar with the text "projectDevOps < 9". Below it, a status summary shows "Branch: -" and "Commit: -" with a timestamp of "1m 23s" and "a few seconds ago". It also indicates "No changes" and was "Started by user SAI TEJA BILLIPATI". Below this, a horizontal progress bar represents the pipeline stages: Start, clean Workspace, Clone the Project, Validate the Project, Compile the Project, Test the Project, Package of the Project, Sonaranalysis, Quality gates analysis, and End. Each stage has a green circle with a checkmark, except for "Quality gates analysis" which has a blue circle with a checkmark. The "Quality gates analysis" section is expanded, showing a list of steps with their descriptions and execution times: "jdk17 – Use a tool from a predefined Tool Installation" (<1s), "Fetches the environment variables for a given tool in a list of 'FOO-bar' strings suitable for the withEnv step." (<1s), "maven3 – Use a tool from a predefined Tool Installation" (<1s), "Fetches the environment variables for a given tool in a list of 'FOO-bar' strings suitable for the withEnv step." (<1s), and "Wait for SonarQube analysis to be completed and return quality gate status" (7s). There are buttons for "Restart Quality gates analysis" and a download icon.

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

The screenshot shows the SonarQube interface. The top navigation bar includes links for Instances, EC2, sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search bar. A "Create Project" button is also present. The main content area displays a project named "dockerproject PUBLIC". It shows the last analysis was 39 seconds ago, involving 62 Lines of Code - XML, JSP. The analysis results are summarized as follows:

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

On the left, there are filters for Quality Gate (Passed 1, Failed 0), Reliability (A 0, B 0, C 1, D 0, E 0), and Security (A 1). A message at the bottom states: "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 migrating your data out of it into a different database." A yellow warning icon is shown next to the message.

Step 4 — Install OWASP Dependency Check Plugins

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

The screenshot shows the Jenkins 'Plugins' page. On the left, there's a sidebar with links: 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area is titled 'Plugins' and has a search bar. A table lists the 'OWASP Dependency-Check' plugin, version 5.4.2. The table columns are 'Name', 'Released', and a 'Description' section. The 'Install' button is visible at the top right of the table row.

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

The screenshot shows the Jenkins 'Tools' configuration page. The URL in the browser is 'http://54.146.255.153:8080/manage/configureTools/'. The page title is 'Manage Jenkins > Tools'. A sub-section titled 'Dependency-Check' is open, showing a form to add a new tool. The 'Name' field is set to 'Dp-Check'. The 'Install automatically?' checkbox is checked. Below this, there's a dropdown menu for 'install from github.com' with 'dependency-check 9.0.7' selected. At the bottom of the form are 'Save' and 'Apply' buttons. A green success message 'Saved' is displayed at the bottom left. The Jenkins version 'Jenkins 2.426.1' is shown at the bottom right.

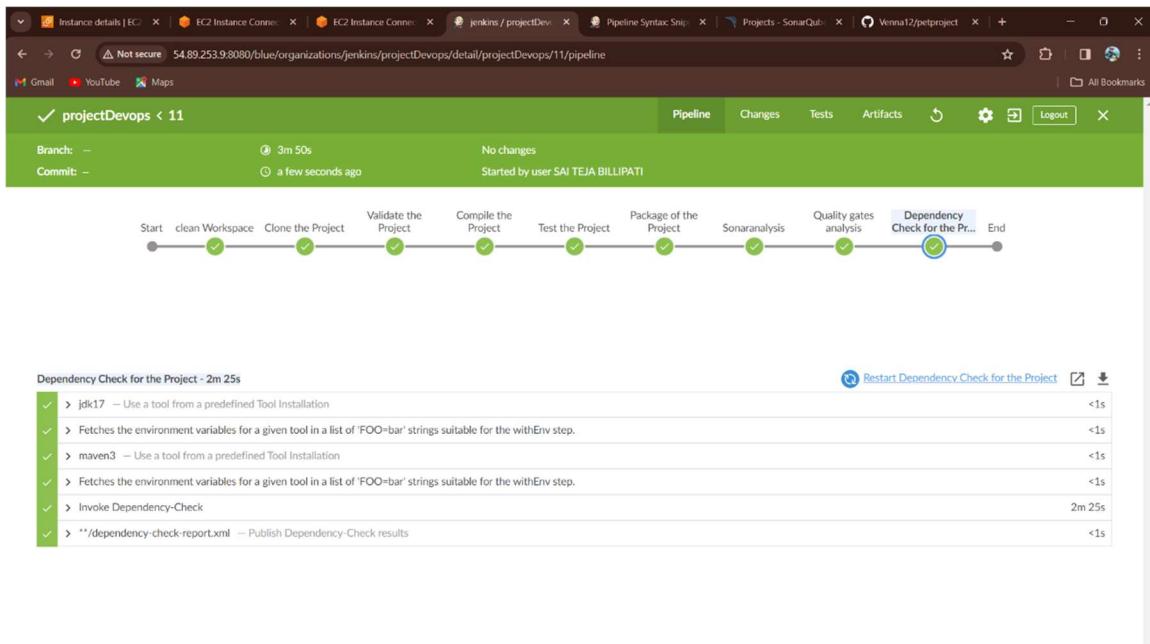
- ❖ Now go configure → Pipeline and add this stage to your pipeline and build.

```

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.

The screenshot shows the Jenkins Pipeline Status page for the 'devopsproject'. On the left, there's a sidebar with various project management links like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, GitHub, Favorite, SonarQube, Open Blue Ocean, Rename, and Pipeline Syntax. The main area is titled 'Stage View' and displays a horizontal timeline of pipeline stages: Declarative: Tool Install, clean Workspace, clone, validate, compile, test, package, Sonarqube Analysis, quality gate, Build war file, and OWI/Depen Che. Each stage has a corresponding duration: 121ms, 243ms, 513ms, 1s, 11s, 32s, 32s, 26s, 283ms, 15s, and 19. Above the timeline is a 'Dependency-Check Trend' chart showing a downward trend from approximately 57 to 19 over 8 stages. There are also buttons for 'Add description' and 'Disable Project'.

This screenshot shows a more detailed view of the Jenkins Pipeline Status for the 'devopsproject'. It includes a 'Build History' section on the left listing builds #8, #7, #6, #5, #4, #3, #2, and #1 with their respective run times. Below the build history is a 'SonarQube Quality Gate' section indicating a 'Passed' status for 'PetshopProject' with a 'Success' message. The main timeline at the top shows average stage times of 121ms, 243ms, 513ms, 1s, 11s, 32s, 32s, 26s, 283ms, 15s, and 19. The pipeline stages themselves are shown in green boxes. At the bottom right, there are links for 'REST API' and 'Jenkins 2.426.2'.

Step 5 — Install NEXUS ARTIFACT

- ❖ Install the nexus and untar it as below process.

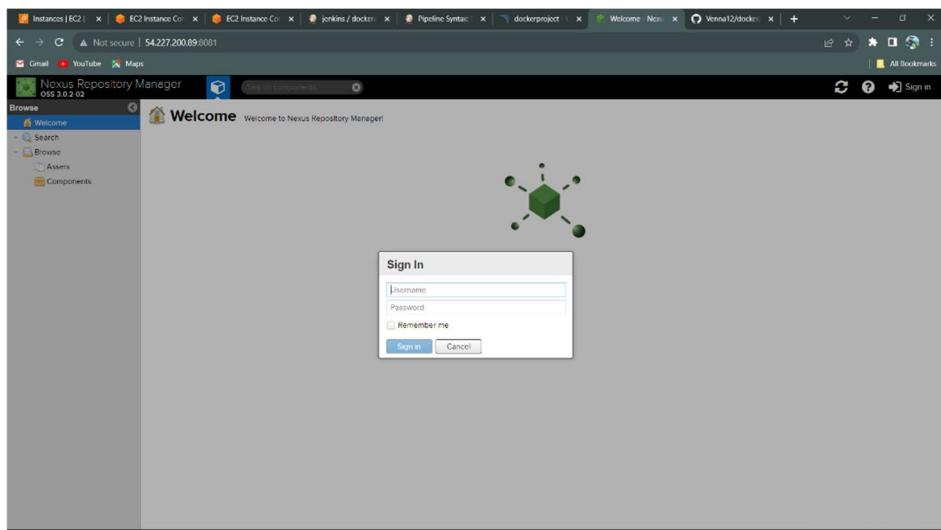
```

Last login: Wed Dec 13 07:02:22 2023 from 18.206.107.28
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
Get: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
Reading 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/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#

```

i-088ab4cc2bd51c9df (advances)
Public IPs: 54.227.200.89 Private IPs: 172.31.58.8

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



❖ Go to > Components > maven-Snapshots!

The screenshot shows the Nexus Repository Manager interface. On the left, there's a sidebar with 'Browse' and 'Components' selected. The main area is titled 'Components' with the subtitle 'Browse components and assets'. It displays a table with columns 'Name', 'Type', and 'Format'. The components listed are:

Name	Type	Format
maven-central	proxy	maven2
maven-public	group	maven2
maven-releases	hosted	maven2
maven-snapshots	hosted	maven2
nuget-group	group	nuget
nuget-hosted	hosted	nuget
nuget.org-proxy	proxy	nuget

- ❖ 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.

The screenshot shows the Jenkins Pipeline Syntax configuration page. On the left, there's a sidebar with links like 'Declarative Online Documentation', 'Steps Reference', 'Global Variables Reference', etc. The main area is titled 'Pipeline Syntax' and shows a 'Steps' section with a dropdown menu containing 'nexusArtifactUploader: Nexus Artifact Uploader'. Below this, under 'Nexus Details', there are fields for 'Nexus Version' (set to 'NEXUS3'), 'Protocol' (set to 'HTTP'), 'Nexus URL' (set to '54.227.200.89:8081'), and 'Credentials' (set to 'admin/******** (nexus123)').

- ❖ For the Group-id, versions we can see that in the project of the **port.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 <groupId>com.example</groupId>
4 <artifactId>java-tomcat-maven-example</artifactId>
5 <packaging>war</packaging>
6 <version>1.0-SNAPSHOT</version>
7 <name>Hello Maven Webapp</name>
8 <url>http://maven.apache.org</url>
9 <dependencies>
10 <dependency>
11   <groupId>junit</groupId>
12   <artifactId>junit</artifactId>
13   <version>3.8.1</version>
14   <scope>test</scope>
15 </dependency>
16 </dependencies>
17 <build>
18   <finalName>java-tomcat-maven-example</finalName>
19   <plugins>
20     <plugin>
21       <groupId>org.apache.maven.plugins</groupId>
22       <artifactId>maven-dependency-plugin</artifactId>
23       <version>2.3</version>
24       <execution>
25         <execution>
26           <phase>package</phase>
27           <goals><goal>copy</goal></goals>
28           <configuration>
29             <artifactItems>
30               <artifactItem>
31                 <groupId>com.github.jsimone</groupId>
32                 <artifactId>webapp-runner</artifactId>
33                 <optional>true</optional>
34             </artifactItems>
35           </execution>
36         </execution>
37       </plugin>
38     </plugins>
39   </build>
40 </project>

```

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

Nexus Version: NEXUS3

Protocol: HTTP

Nexus URL: 54.227.200.89:8081

Credentials: admin/***** (nexus123)

GroupId: com.example

Version: 1.0-SNAPSHOT

Repository: maven-snapshots

- ❖ Write the pipeline syntax for the nexus artifact.

```

10+           steps{
11+             sh 'mvn validate'
12+         }
13+       }
14+     }
15+   }
16+   }
17+ }
18+
19+ stages{
20+   stage('compile the Project'){
21+     steps{
22+       sh 'mvn compile'
23+     }
24+   }
25+   stage('test the Project'){
26+     steps{
27+       sh 'mvn test'
28+     }
29+   }
30+   stage('Code quality analysis'){
31+     steps{
32+       sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=dockerproject -Dsonar.projectName=dockerproject -Dsonar.host.url=http://sonarqube:9000'
33+     }
34+   }
35+ }
36+
37+ }

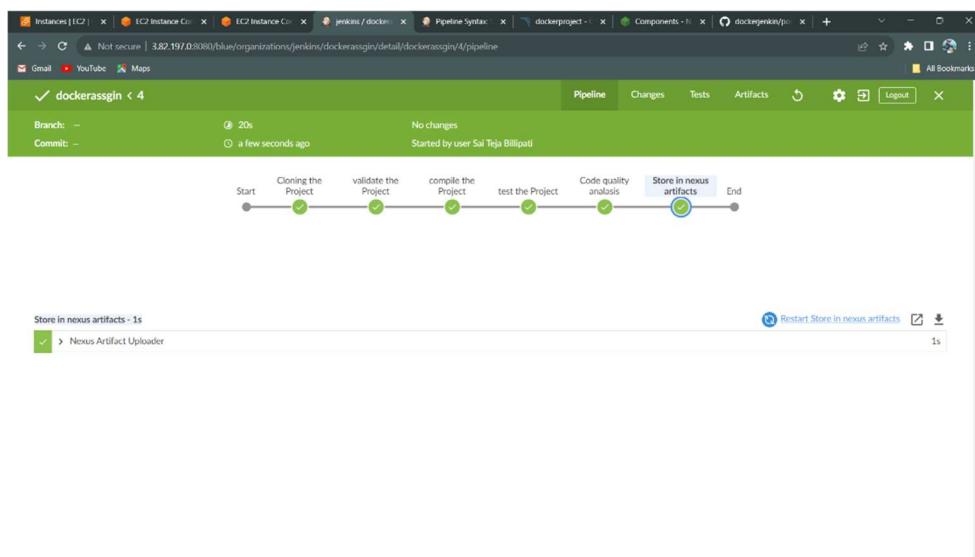
```

Use Groovy Sandbox ?

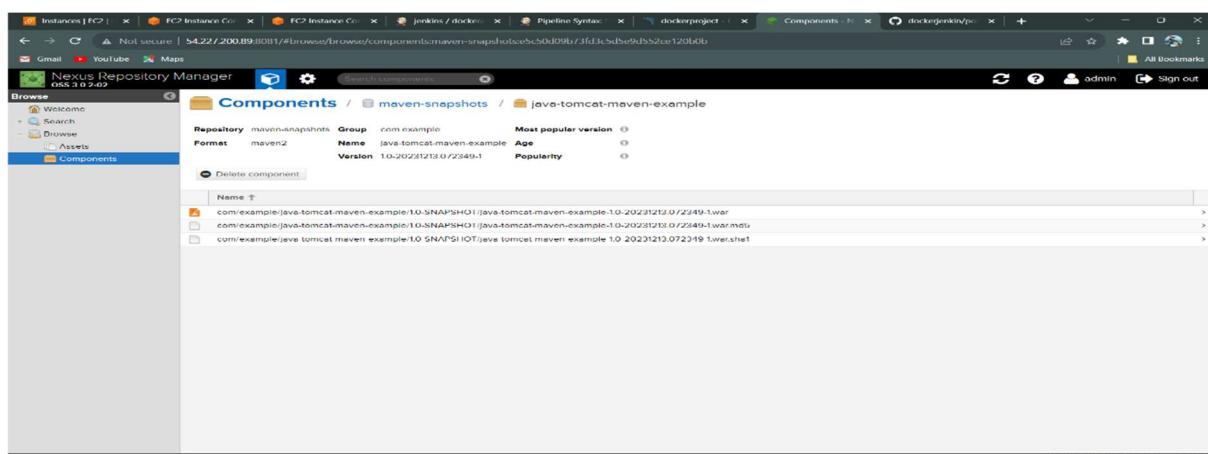
Pipeline Syntax

Save Apply

❖ 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 screenshot shows a terminal window within an AWS CloudShell interface. The user has run several commands to install and start Apache Tomcat 9.0.83. The output of the commands is as follows:

```
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
Get: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
Ign:4 https://pkg.jenkins.io/debian-stable binary/
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:6 https://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
0 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#
```

i-0d818a744ce385322 (devopsmain)
Public IPs: 3.82.197.0 Private IPs: 172.31.93.156

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

The screenshot shows the Tomcat Web Application Manager interface. At the top, there's a logo for "THE APACHE SOFTWARE FOUNDATION" and a cartoon cat. Below the header, there's a message box and tabs for "Manager", "List Applications", "HTML Manager Help", "Manager Help", and "Server Status".

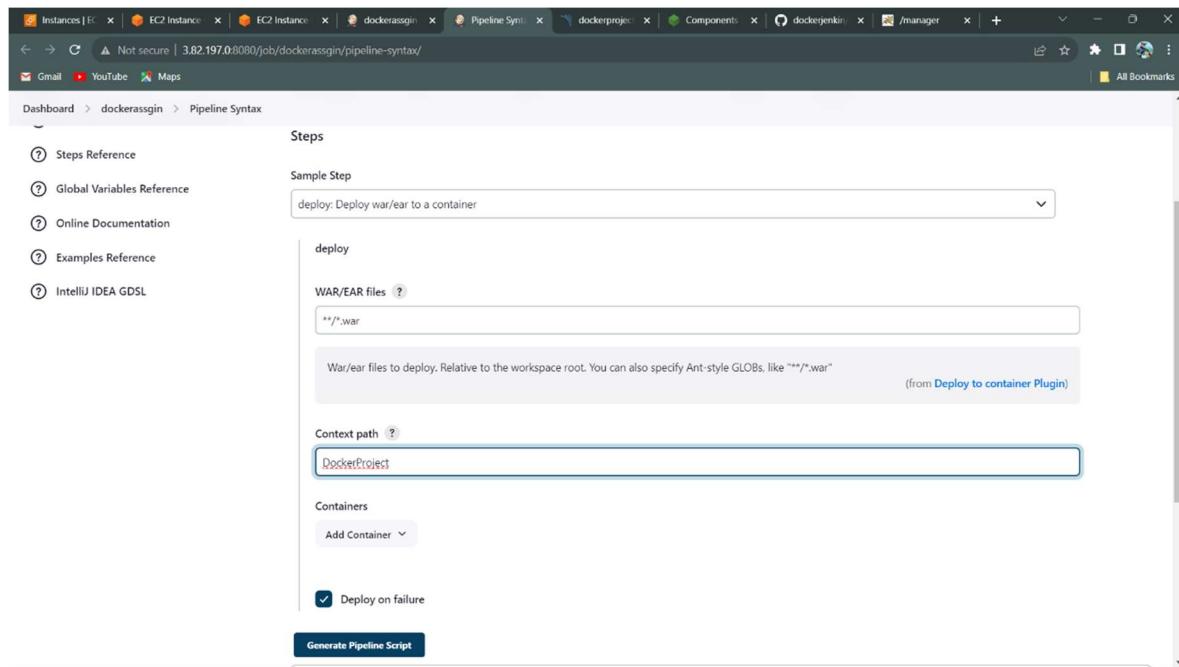
Applications

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	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

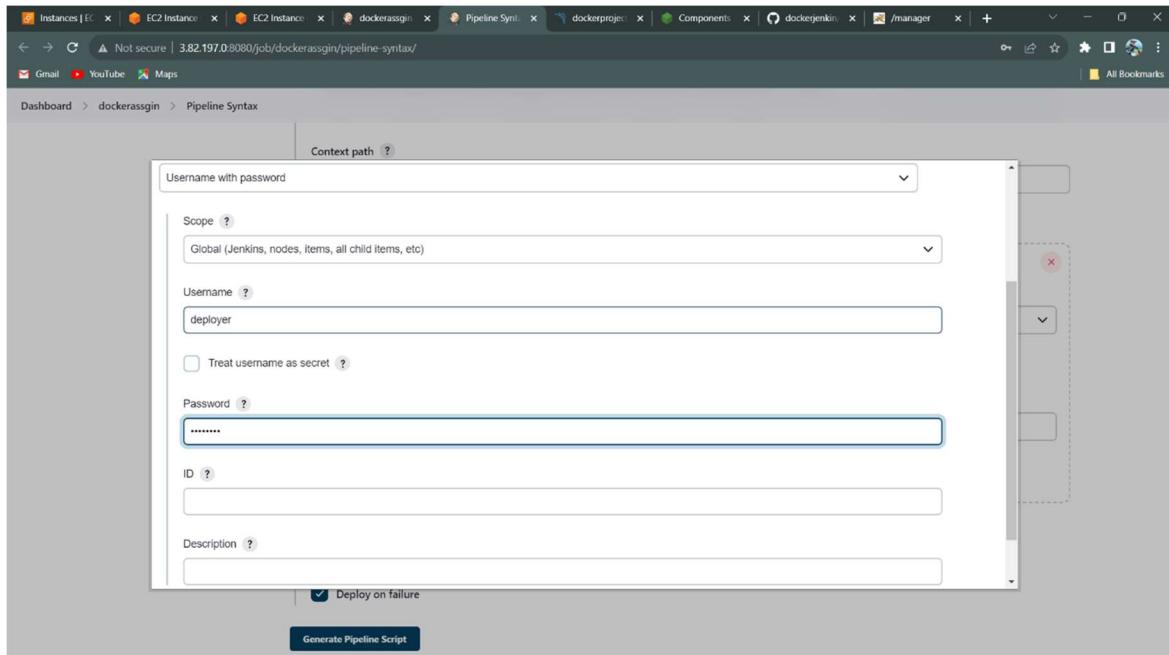
Deploy

Deploy directory or WAR file located on server
Context Path:

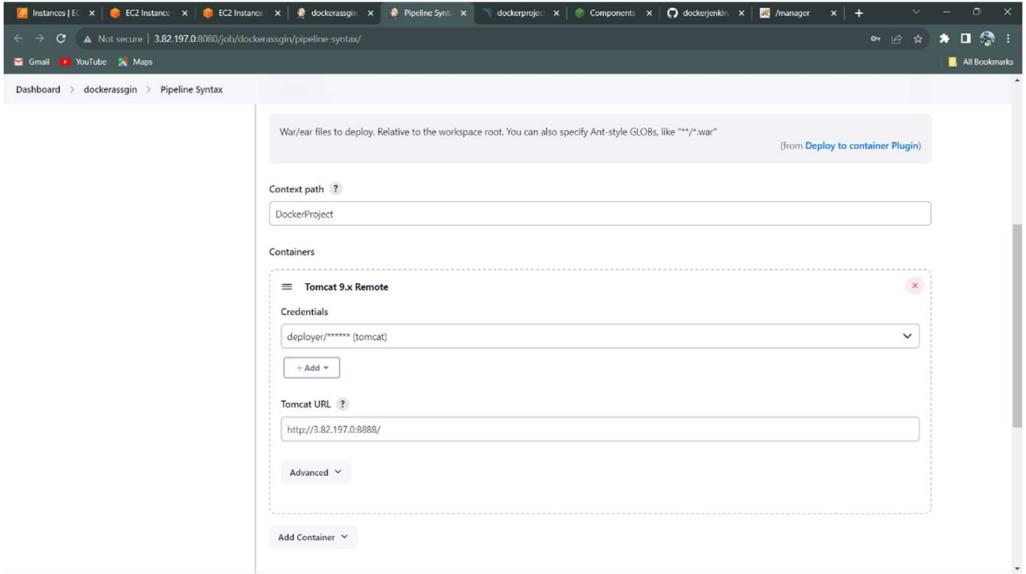
- ❖ Now go to Jenkins pipeline syntax and select the deploy the container and the context path name it as Docker project.



- ❖ 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.

```

1 > pipeline{
2   agent any
3   stages{
4     stage('Cloning the Project'){
5       steps{
6         checkout scm
7       }
8     }
9     stage('Validate the Project'){
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://3.82.197.0:8888'
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/tomcat-1.0-SNAPSHOT.war']]
32    }
33  }
34  stage('Deploy in tomcat'){
35    steps{
36      deploy adapters: [tomcat9(credentialsId: 'tomcat', path: '', url: 'http://3.82.197.0:8888')], contextPath: 'DockerProject'
37    }
38  }
39 }
40

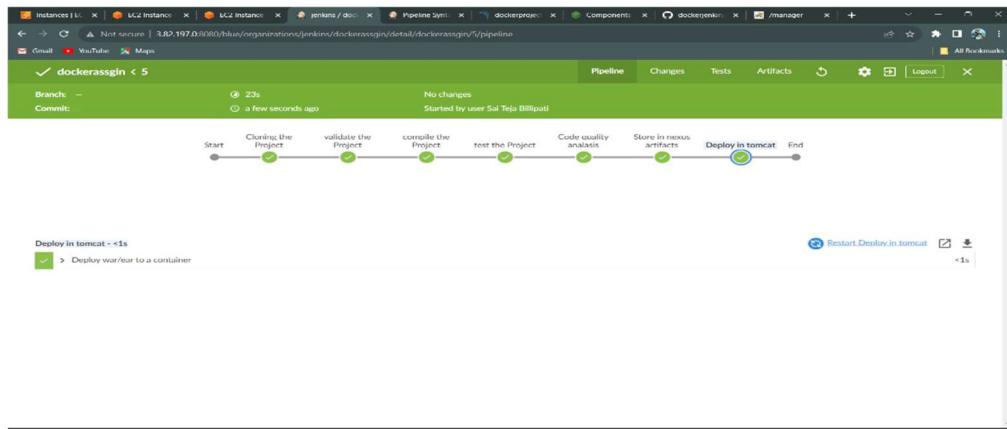
```

Configure

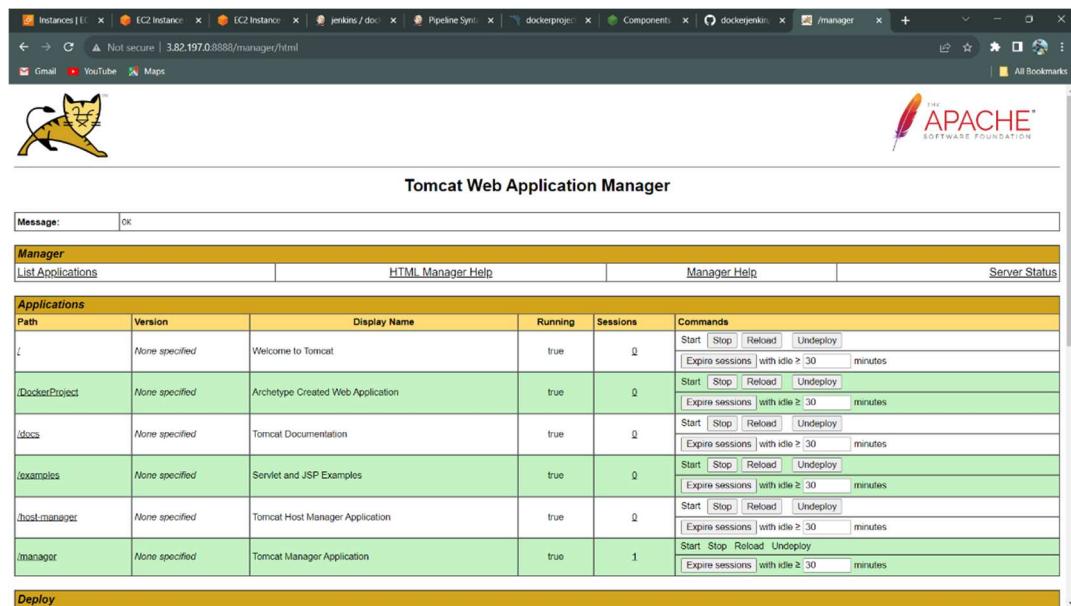
- General
- Advanced Project Options
- Pipeline

Save **Apply**

- ❖ 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.

Dashboard > Manage Jenkins > Plugins

Installed plugins

Advanced settings

Download progress

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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind: Username with password

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

Username: sevenajay

Treat username as secret ?

Password:

ID: docker

- ❖ Add this stage to Pipeline Script

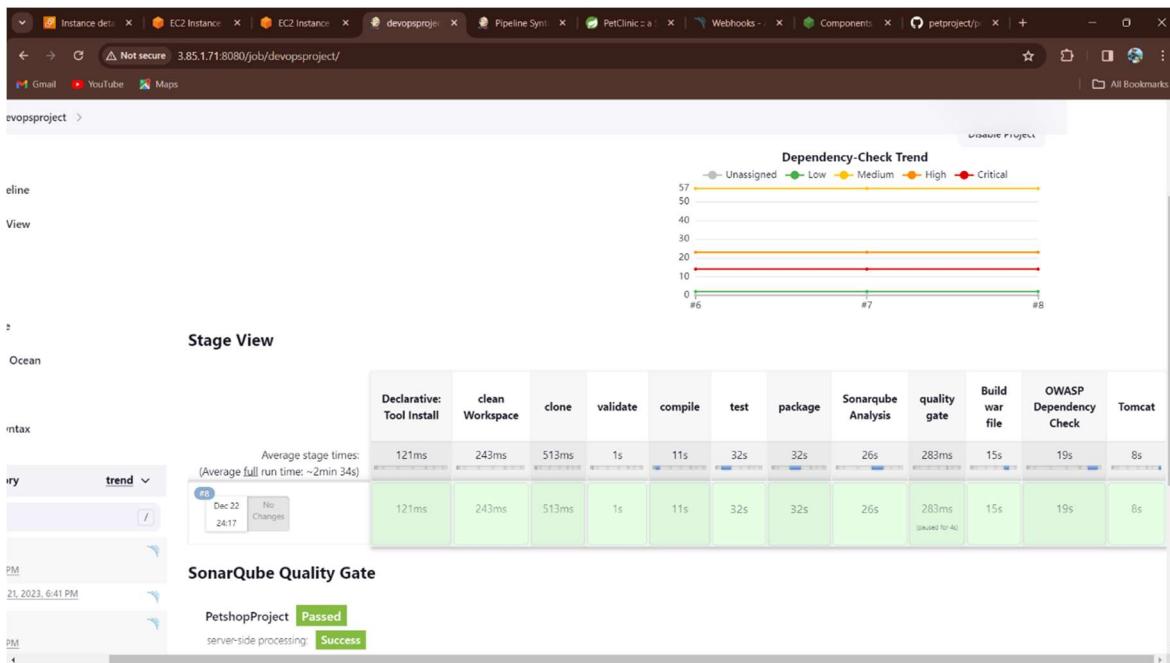
```

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"
            }
        }
    }
}

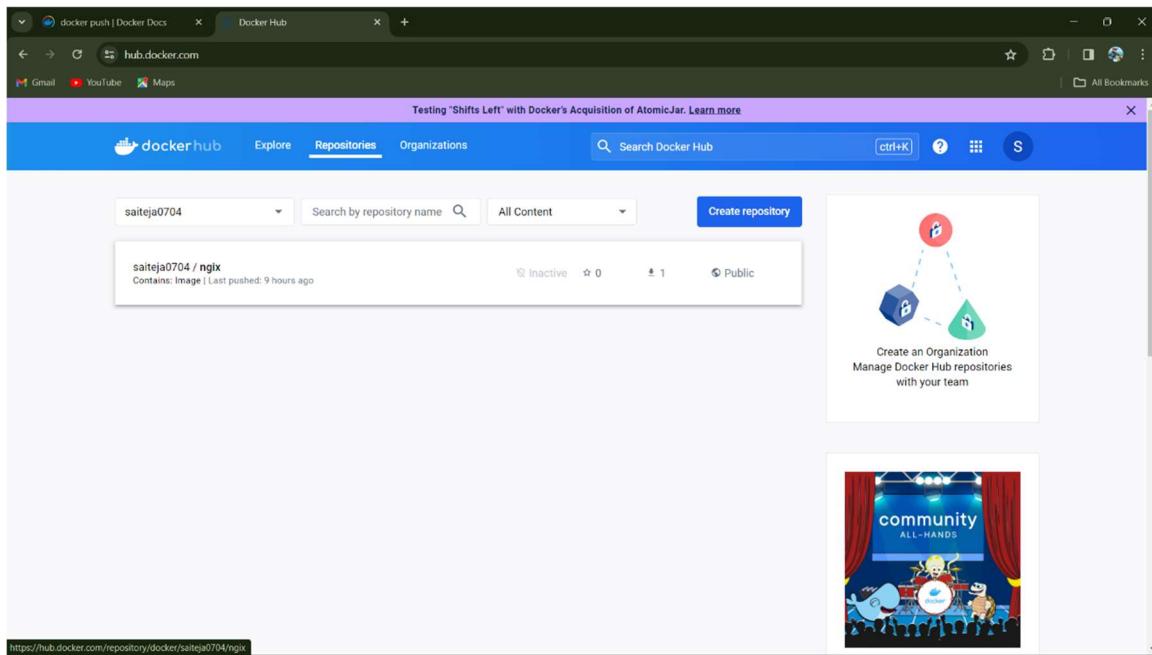
stage ('Deploy to container'){
    steps{
        sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'
    }
}

```

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



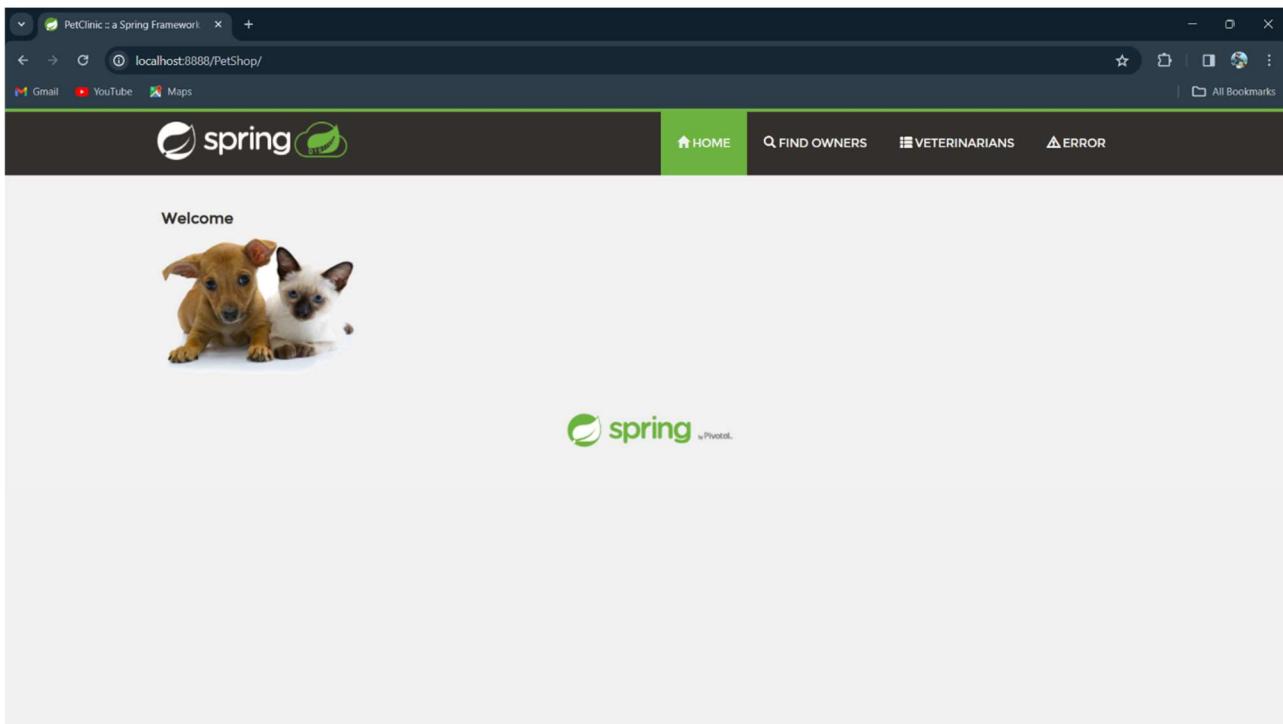
- ❖ When we login in the Docker we can see a new image has been added.



- ❖ The Final output can be check through this

<Ec2-public-ip:8080/Petproject>

❖ FINAL OUTPUT.....

A screenshot of a web browser showing the PetClinic application. The title bar says "PetClinic : a Spring Framework" and the address bar shows "localhost:8888/PetShop/owners?lastName=". The page features a dark header with the "spring" logo and navigation links for "HOME", "FIND OWNERS", "VETERINARIANS", and "ERROR". Below the header is a section titled "Owners" with a table listing ten entries. The table has columns for Name, Address, City, Telephone, and Pets. The data is as follows:

Name	Address	City	Telephone	Pets
George Franklin	110 W. Liberty St.	Madison	6085551023	Leo
Betty Davis	638 Cardinal Ave.	Sun Prairie	6085551749	Basil
Eduardo Rodriguez	2693 Commerce St.	McFarland	6085558763	Jewel Rosy
Harold Davis	563 Friendly St.	Windsor	6085553198	Iggy
Peter McTavish	2387 S. Fair Way	Madison	6085552765	George
Jean Coleman	105 N. Lake St.	Monona	6085552654	Max Samantha
Jeff Black	1450 Oak Blvd.	Monona	6085555387	Lucky
Maria Escobito	345 Maple St.	Madison	6085557683	Mulligan
David Schroeder	2749 Blackhawk Trail	Madison	6085559435	Freddy
Carlos Estaban	2335 Independence La.	Waunakee	6085555487	Lucky Sly

A large green "spring" logo is centered at the bottom of the page.

.....<<< END>>>.....