

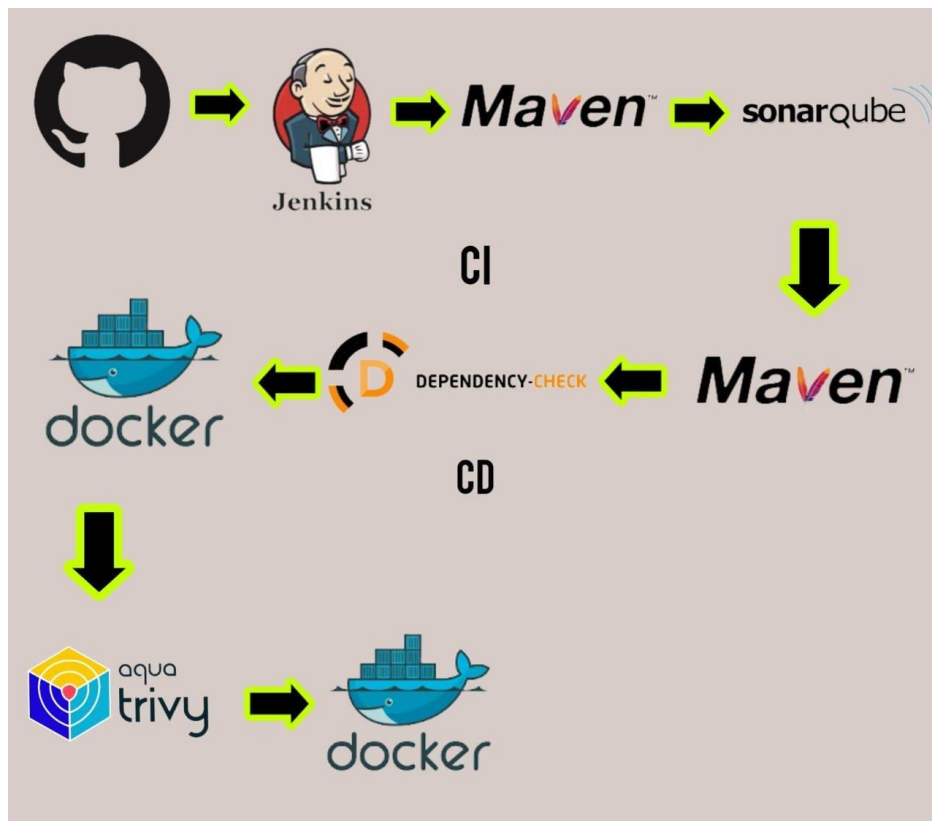


PROJECT – 3

COURSE: DEVOPS

MODULE: Docker Automation

TRAINER: Mr. MADHUKAR



Name: Thella Naveen

Mobile: 7095396259

Email: navee9404@gmail.com

Project Repo: <https://github.com/Venn1991/jpetstore-6.git>

Steps:-

Step 1 — Create an Ubuntu(22.04) T2 Large Instance

Step 2 — Install Jenkins, Docker and Trivy. Create a SonarQube Container using Docker.

Step 3 — Install Plugins like JDK, SonarQube Scanner, Maven, and OWASP Dependency Check.

Step 4 — Create a Pipeline Project in Jenkins using a Declarative Pipeline

Step 5 — Install OWASP Dependency Check Plugins

Step 6 — Docker Image Build and Push

Step 7 — Scan image using Trivy

Step 8 — Deploy the image using Docker Step

9 — Access the Real-World Application

Step 10 — Terminate the AWS EC2 Instances.

STEP1: Create an Ubuntu (22.04) T2 Large Instance

Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group and open all ports (not best case to open all ports but just for learning purposes it's okay).

Create AWS instance and connect to web server:

- First enter into AWS console login page. ● Login into AWS account.
- Select Ec2 service in AWS account.
- Create new instance by click on launch instance.
- Give name, select ubuntu, select T2 large and keypair name click on launch instance.

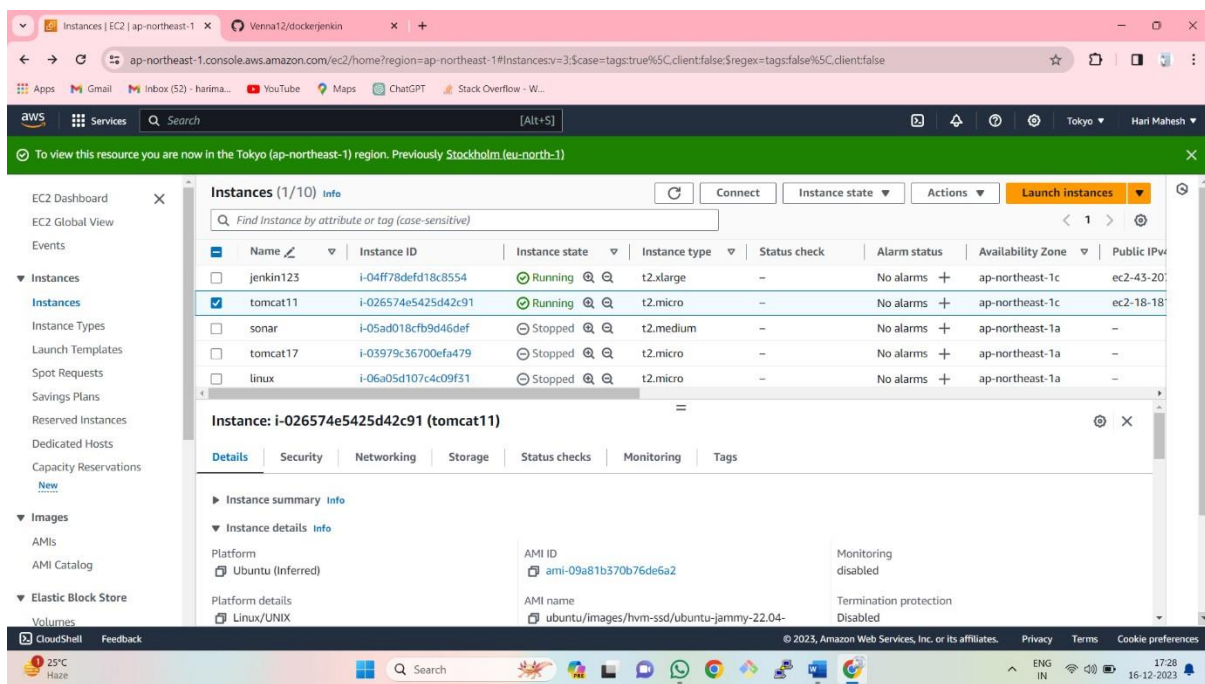
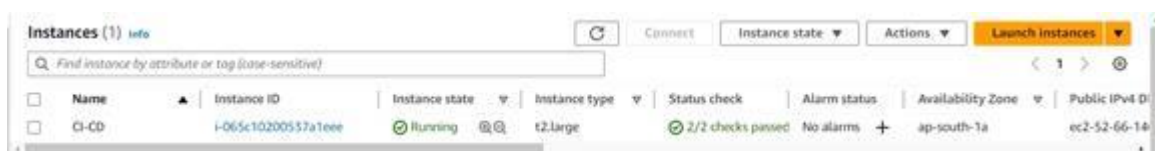


Fig.1 After completing launch instance.



- After launch instance click on connect then it enters into web server.
- After enter into server the window looks like above.
- To convert root user, enter `sudo -i`.

Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins

```
vi jenkins.sh
```

```
#!/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 sudo  
chmod 777 jenkins.sh  
./jenkins.sh # this will install Jenkins
```

Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

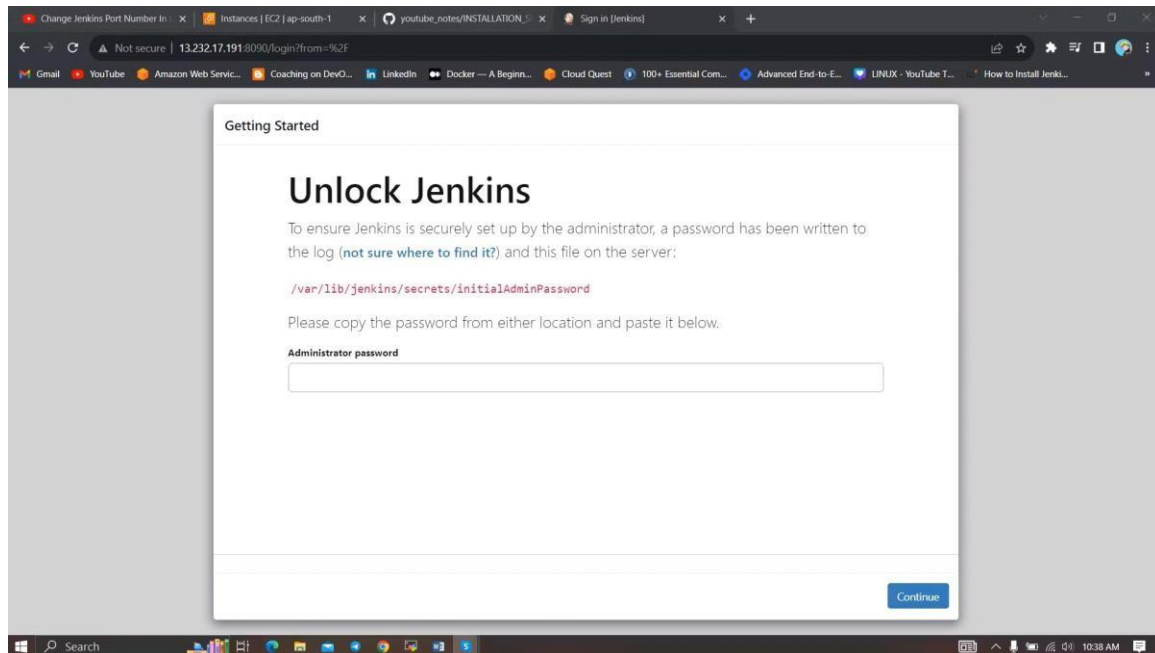
But for this Application case, we are running Jenkins on another port. so change the port to 8090 using the below commands.

```
sudo systemctl stop jenkins  
sudo systemctl status jenkins cd  
/etc/default  
sudo vi jenkins #change port HTTP_PORT=8090 and save and exit cd  
/lib/systemd/system  
sudo vi jenkins.service #change Environments="Jenkins_port=8090" save and exit  
sudo systemctl daemon-reload sudo systemctl restart jenkins sudo systemctl status  
Jenkins
```

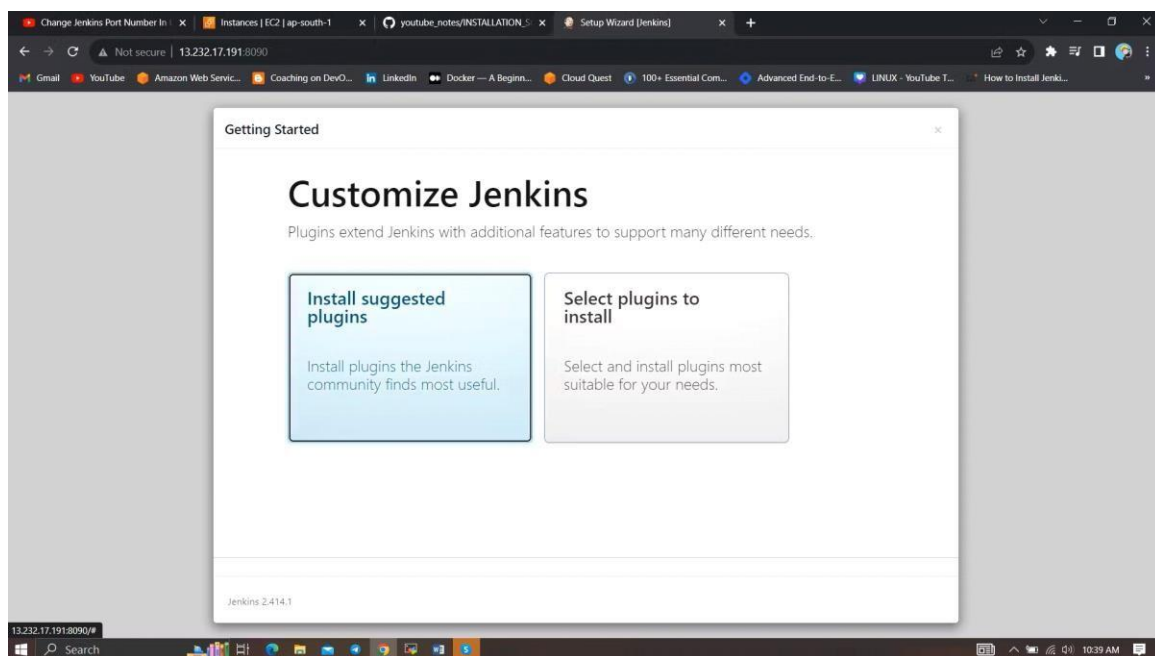
Now, grab your Public IP Address

<EC2 Public IP Address:8090>

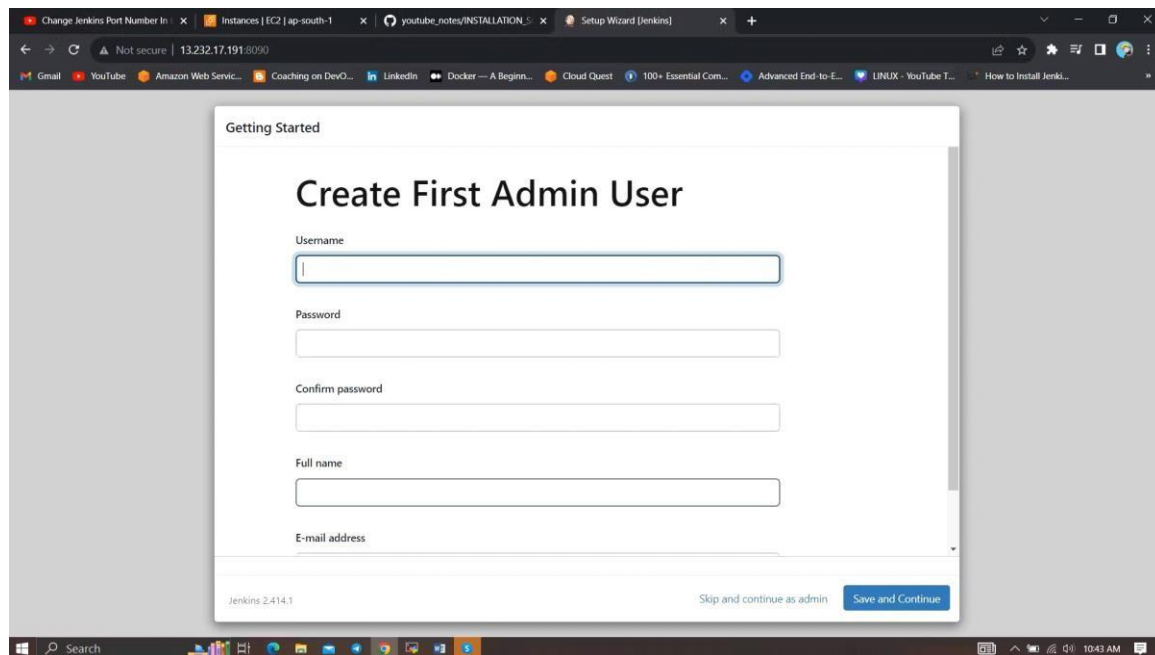
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



Unlock Jenkins using an administrative password and install the suggested plugins.

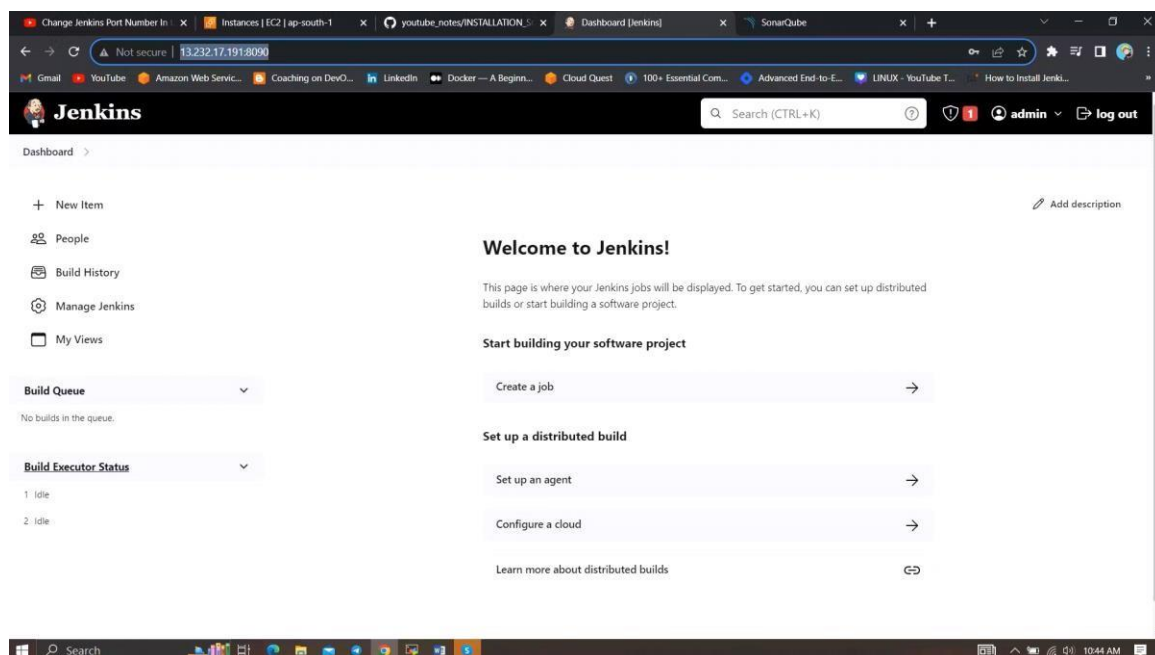


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



2B — Install Docker

```
sudo apt-get update sudo apt-
```

```
get install docker.io -y
```

```
sudo usermod -aG docker $USER #my case is ubuntu newgrp
```

```
docker
```

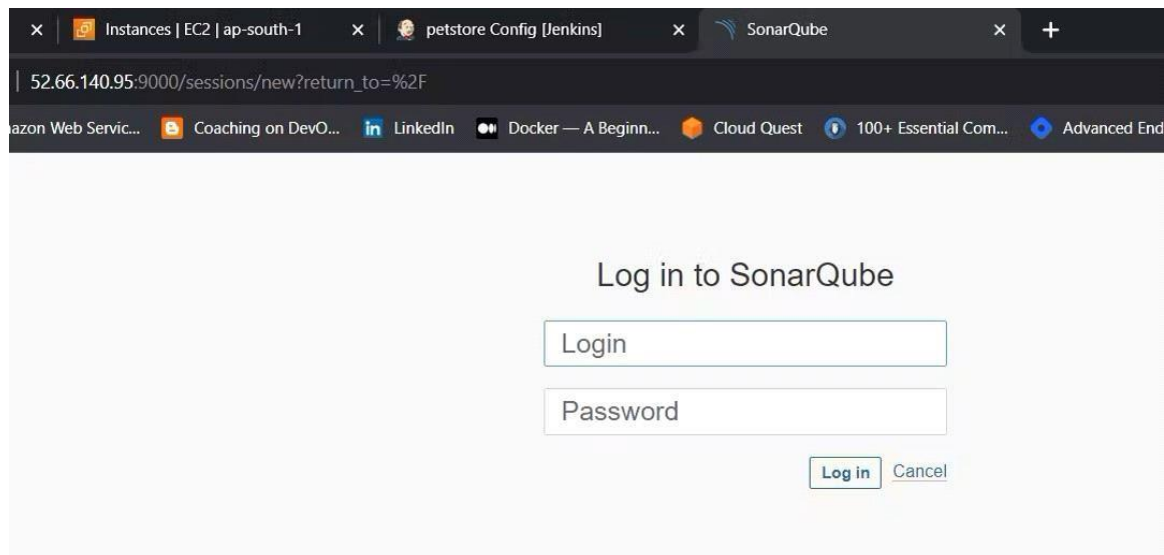
```
sudo chmod 777 /var/run/docker.sock
```

After the docker installation, we create a sonarqube container (Remember added 9000 ports in the security group)

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b0a: Pull complete
b17b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f50f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dccc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752f8b04993092d8ca3865c2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS        PORTS                    NAMES
4b60c96bf9ad   sonarqube:lts-community   "/opt/sonarqube/dock..."   9 seconds ago   Up 5 seconds   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-172-31-42-253:~$
```

Now our SonarQube is up and running



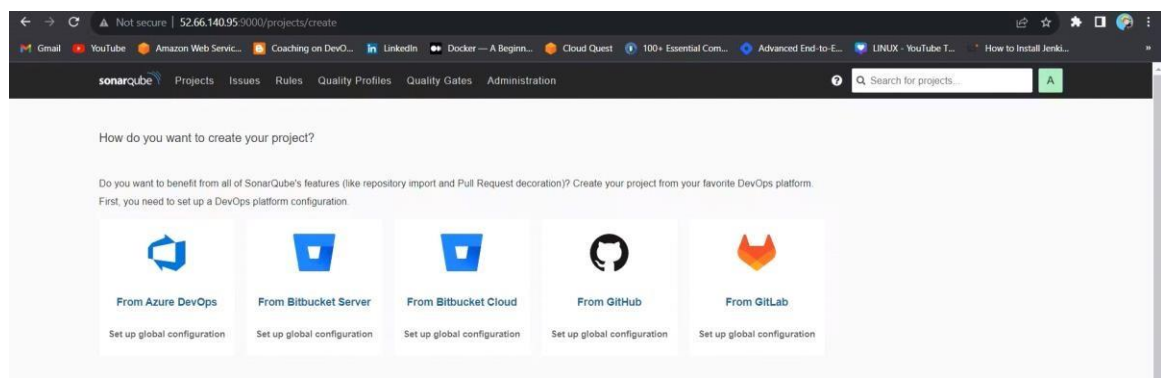
Enter username and password, click on login and change password

username admin password

admin

The screenshot shows a web browser window with the URL `52.66.140.95:9000/account/reset_password`. The browser tabs include 'Instances | EC2 | ap-south-1', 'petstore Config [Jenkins]', and 'SonarQube'. The page title is 'Update your password'. Below the title, it says 'This account should not use the default password.' and 'Enter a new password'. A note states 'All fields marked with * are required'. There are three input fields: 'Old Password *', 'New Password *', and 'Confirm Password *'. An 'Update' button is at the bottom.

Update New password, This is Sonar Dashboard.



2C — Install Trivy

vi trivy.sh

```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
```

```
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
```

```
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
```

```
sudo apt-get update
```

```
sudo apt-get install trivy -y
```

Next, we will log in to Jenkins and start to configure our Pipeline in Jenkins

Step 3 — Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check

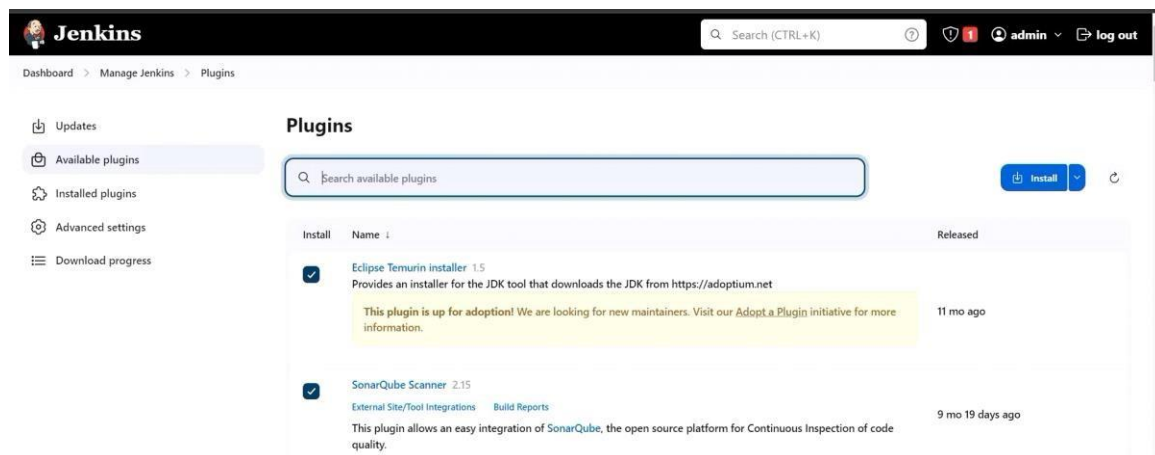
3A — Install Plugin

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)



3B — Configure Java and Maven in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and Maven3(3.6.0) → Click on Apply and Save

The first screenshot shows the 'JDK' configuration in Jenkins. The 'Name' field is 'jdk17'. The 'Install automatically' checkbox is checked. Under 'Install from adoptium.net', the 'Version' dropdown is set to 'jdk-17.0.8.1+1'. The 'Add Installer' button is visible at the bottom.

The second screenshot shows the 'Maven' configuration in Jenkins. The 'Name' field is 'maven3'. The 'Install automatically' checkbox is checked. Under 'Install from Apache', the 'Version' dropdown is set to '3.6.0'. The 'Add Installer' button is visible at the bottom.

3C — Create a Job

The screenshot shows the 'Create a new job' dialog in Jenkins. The 'Enter an item name' field contains 'petstore'. Below the field are four options: 'Freestyle project', 'Maven project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is selected. An 'OK' button is at the bottom.

Enter this in Pipeline Script,

```

pipeline{
    agent
    any
    tools
    {
        jdk 'jdk17'
        maven 'maven3'
    }
    stages{
        stage ('clean
Workspace'){
            steps{
                cleanWs()
            }
        }
        stage ('checkout') {
            steps {
                checkout scmGit(branches: [[name: '*/master']],
extensions: [], userRemoteConfigs: [[url: 'https://github.com/harimahesh2344/jpetstore-6.git']])
            }
        }
        stage ('mvn compile')
        {
            steps {
                sh 'mvn compile'
            }
        }
        stage ('mvn test')
        {
            steps
        {
            sh 'mvn
test'
        }
        }
    }
}

```

The stage view would look like this,

Pipeline petstore

[Add description](#)

[Disable Project](#)

Stage View



Step 4 — Configure Sonar Server in Manage Jenkins

sonarqube Projects Issues Rules Quality Profiles Quality Gates **Administration**

Administration

Configuration Security Projects System Marketplace

General **Users** Groups Global Permissions Permission Templates

SCM Accounts Last connection Groups **Tokens**

A Administrator admin < 1 hour ago sonar-administrators sonar-users 0 Update Tokens

Generate Tokens

Name Expires in

Enter Token Name 30 days Generate

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `sq_u_21d162984c1c72cf8b39665f96480185c99dc2f9`

Name	Type	Project	Last use	Created	Expiration	Revoke
Jenkins	User		Never	September 8, 2023	October 8, 2023	Revoke

Create a token with a name and generate

Tokens of Administrator

Generate Tokens

Name Expires in

Enter Token Name 30 days Generate

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `sq_u_21d162984c1c72cf8b39665f96480185c99dc2f9`

Name	Type	Project	Last use	Created	Expiration	Revoke
Jenkins	User		Never	September 8, 2023	October 8, 2023	Revoke

copy Token

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

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

New credentials

Kind
Secret text

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

Secret
POST THE TOKEN HERE


ID ?
Sonar-token

Description ?
Sonar-token

Create

You will this page once you click on create

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 like the below image.

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name

sonar-server

Server URL

Default is http://localhost:9000

http://13.232.17.191:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-token

Add

Save Apply

Click on Apply and Save

The Configure System option is used in Jenkins to configure different server

Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

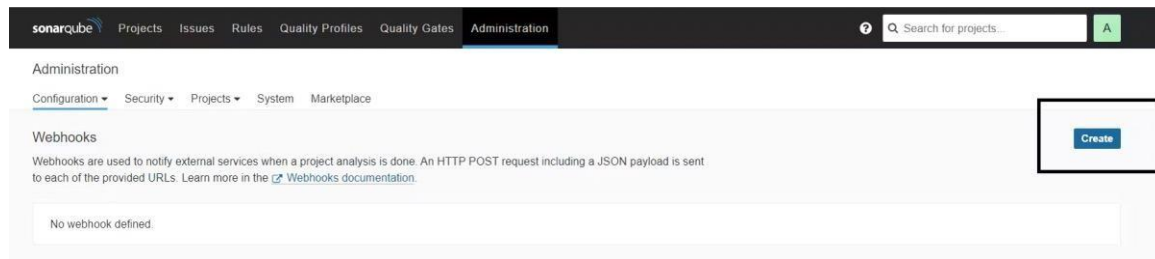
The screenshot shows the 'SonarQube Scanner installations' page in the SonarQube interface. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Tools'. The page title is 'SonarQube Scanner installations'. Below the title is a section 'Add SonarQube Scanner'. Inside this section, there is a form for configuring a new scanner. The form has a 'Name' field with the value 'sonar-scanner'. Below the name field is a checkbox labeled 'Install automatically' which is checked. Underneath the checkbox is a sub-section titled 'Install from Maven Central' which contains a 'Version' dropdown menu set to 'SonarQube Scanner 5.0.1.3006'. At the bottom of this sub-section is a button labeled 'Add Installer'. Below the main configuration form is another 'Add SonarQube Scanner' button. At the very bottom of the page are two buttons: 'Save' and 'Apply'.

In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks

The screenshot shows the SonarQube Administration page. The breadcrumb trail at the top is 'Administration > Configuration > Security > Projects > System > Marketplace'. The 'Administration' tab is selected and highlighted with a red box. Below the breadcrumb trail is a dropdown menu with options: 'General Settings', 'Encryption', and 'Webhooks'. The 'Webhooks' option is selected and highlighted with a red box. Below the dropdown menu is a search bar labeled 'Search by login or name'. Below the search bar is a table with columns: 'SCM Accounts', 'Last connection', 'Groups', and 'Tokens'. The table contains one row for the 'Administrator admin' user. The 'Groups' column for this user lists 'sonar-administrators' and 'sonar-users'. The 'Tokens' column shows a count of '1' and a gear icon for configuration. Below the table is a pagination indicator '1 of 1 shown'. At the top right of the page is a 'Create User' button.

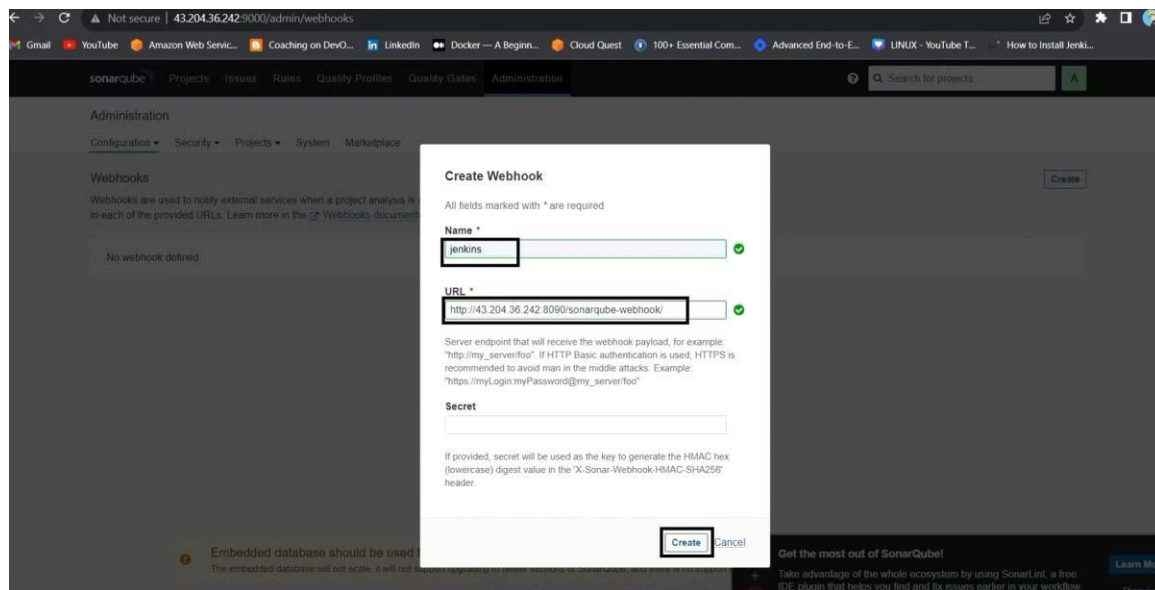
Click on Create



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("Sonar

Analysis ") {

steps {

withSonarQubeEnv('sonar-server') {

sh "' \$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Petshop \

Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop "'


```

    }
  }
}

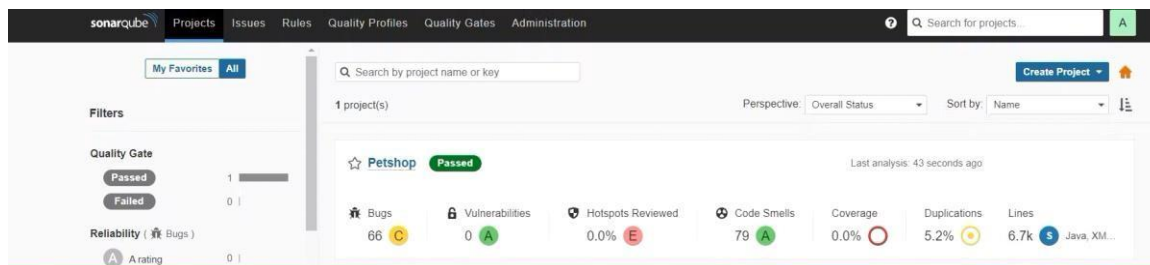
stage("quality
gate"){
    steps
    {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
        }
    }
}
}

```

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.



You can see the report has been generated and the status shows as passed. You can see that there are 6.7k lines. To see a detailed report, you can go to issues.

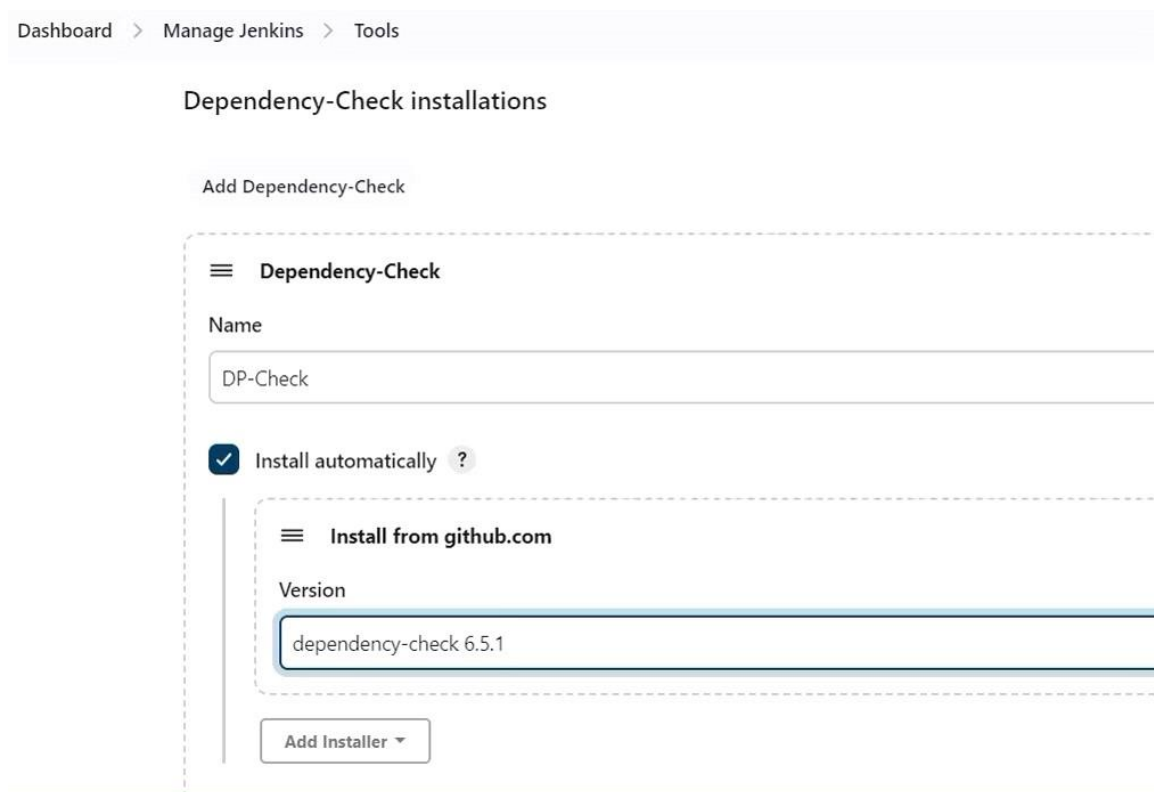
Step 5 — Install OWASP Dependency Check Plugins

GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.



First, we configured the Plugin and next, we had to configure the Tool

Goto 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 clean package'

}

```

    }

    stage("Dp Check"){

        steps{

            dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'

        }

    }
}

```

The stage view would look like this,

Stage View

	Declarative: Tool Install	clean Workspace	checkout scm	maven compile	maven Test	Sonarqube Analysis	quality gate	Build war file	OWASP Dependency Check
Average stage times: (Average full run time: ~5min 33s)	8s	305ms	1s	1min 38s	1min 9s	23s	519ms	2min 8s	4min 32s
#3 Sep 08 11:17 No Changes	117ms	240ms	1s	48s	56s	21s	400ms (paused for 4s)	2min 8s	4min 32s

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

Dashboard

>

petstore

>

#3

>

Dependency-Check

Status

</> Changes

Console Output

View as plain text

Edit Build Information

Delete build '#3'

Git Build Data

Dependency-Check

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Dependency-Check Results

SEVERITY DISTRIBUTION

3

4

19

Search

Q

File Name	Vulnerability	Severity	Weakness
+ bootstrap.jar	<div>NVD</div> CVE-2023-28708	<div>Medium</div>	CWE-523
+ bootstrap.jar	<div>NVD</div> CVE-2023-41080	<div>Medium</div>	CWE-601
+ catalina-ant.jar	<div>NVD</div> CVE-2023-28708	<div>Medium</div>	CWE-523
+ catalina-ant.jar	<div>NVD</div> CVE-2023-41080	<div>Medium</div>	CWE-601
+ catalina.jar	<div>NVD</div> CVE-2023-28708	<div>Medium</div>	CWE-523
+ catalina.jar	<div>NVD</div> CVE-2023-41080	<div>Medium</div>	CWE-601
+ commons-daemon.jar	<div>NVD</div> CVE-2021-37533	<div>Medium</div>	CWE-20
+ jasper.jar	<div>NVD</div> CVE-2023-28708	<div>Medium</div>	CWE-523
+ jasper.jar	<div>NVD</div> CVE-2023-41080	<div>Medium</div>	CWE-601
+ jspic-api.jar	<div>NVD</div> CVE-2023-28708	<div>Medium</div>	CWE-523

Step 6 — Docker Image Build 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

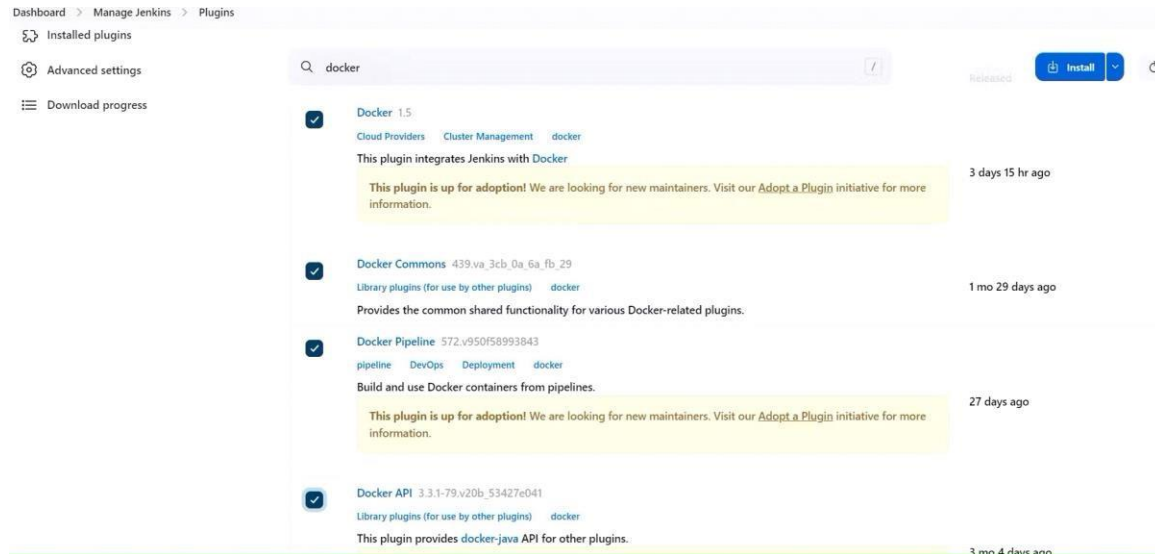
Docker

Docker Commons

Docker Pipeline Docker

API docker-build-step

and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



Add DockerHub 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

Description ?
docker

Create

Add this stage to Pipeline Script

For automation add under environment variables

Environment {

APP_NAME="hari"

VERSION="v1.0.0"

COUNTAINER_NAME="nani12"

DOCKER_USER="harimahesh2344"

DOCKER_PASS="Mahesh@2344"

IMAGE_NAME="\${DOCKER_USER}"/"\${APP_NAME}"
IMAGE_TAG="\${VERSION}-\${BUILD_NUMBER}"

}

Add below stages under the stage: stage

('Build and push to docker hub'){

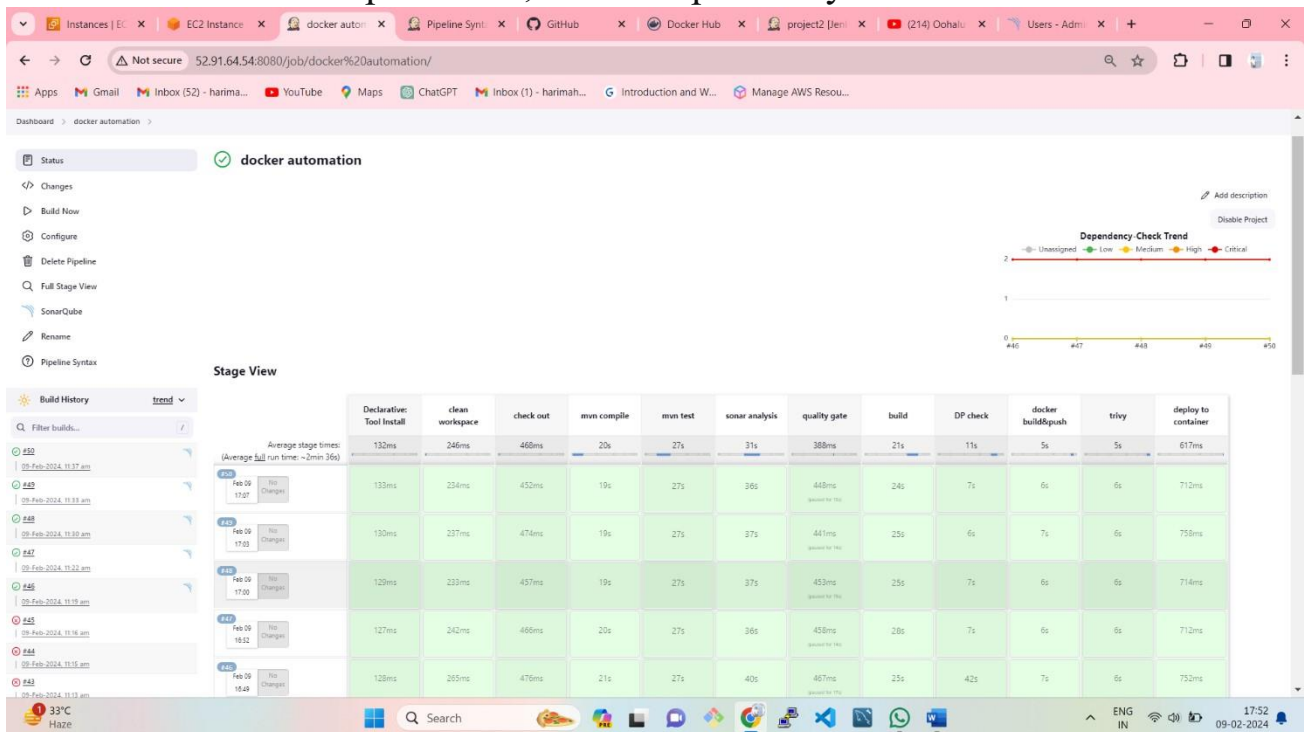
```
steps{
    script {
        withDockerRegistry(credentialsId: 'docker')
        {
            sh 'docker rm nani12 -f'
        }
        docker_image=docker.build "${IMAGE_NAME}"
    }
}
```

```

        docker_image.push("${IMAGE_TAG}")
    }
}
}
}
stage("TRIVY"){
    steps{
        sh "trivy image ${DOCKER_USER}/${APP_NAME}:latest > trivy.txt"
    }
}
stage('Deploy to
container'){
    steps{
        sh "docker run --name ${CONTAINER_NAME} -d -p 8085:8080
${IMAGE_NAME}:${IMAGE_TAG}"
    }
}
}

```

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



Now, when you do **Entire**

pipeline:

```
pipeline{
    agent
    any

    tools{
        jdk
        'jdk17'
        maven
        'maven3'

    }

    environment {

        SCANNER_HOME=tool 'sonar-scanner'

        APP_NAME="hari"

        VERSION="v1.0.0"

        CONTAINER_NAME="nani12"

        DOCKER_USER="harimahesh2344"

        DOCKER_PASS="Mahesh@2344"

        IMAGE_NAME="${DOCKER_USER}"/"${APP_NAME}"

        IMAGE_TAG="${VERSION}-${BUILD_NUMBER}"

    }

    stages{
        stage('clean
workspace'){
```

```

steps{          cleanW
s()
    }
}

stage('check
out'){          steps{
                checkout scmGit(branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/harimahesh2344/jpetstore-6.git']])
    }
}

stage('mvn compile'){
    steps{          sh
'mvn compile'
    }
}

stage('mvn
test'){          steps{
    sh 'mvn test'
    }
}

```



```

        stage('sonar analysis'){
            steps{
                withSonarQubeEnv('sonar-token')
            {
                sh "$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Petshop1 \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=Petshop1"
            }
        }
    }
    stage('quality
gate'){
        steps{
            script {
                waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
            }
        }
    }
    stage('build'){
        steps{
            sh
'mvn package'
        }
    }
    stage('DP
check'){
        steps{

```

```

        dependencyCheck additionalArguments: '--scan ./ --format XML ',
odcInstallation: 'DP-Check'        dependencyCheckPublisher pattern:
'*/dependency-check-report.xml'

    }

}

stage("docker build&push"){

steps{
    script{
        withDockerRegistry(
credentialsId: 'docker') {
            sh 'docker rm
nani12 -f

            docker_image=docker.build "${IMAGE_NAME}"
docker_image.push("${IMAGE_TAG}")

        }

    }

}

stage('trivy'){
steps{

    sh "trivy image ${DOCKER_USER}"/"/"${APP_NAME}:latest >
trivy.txt"

}

}

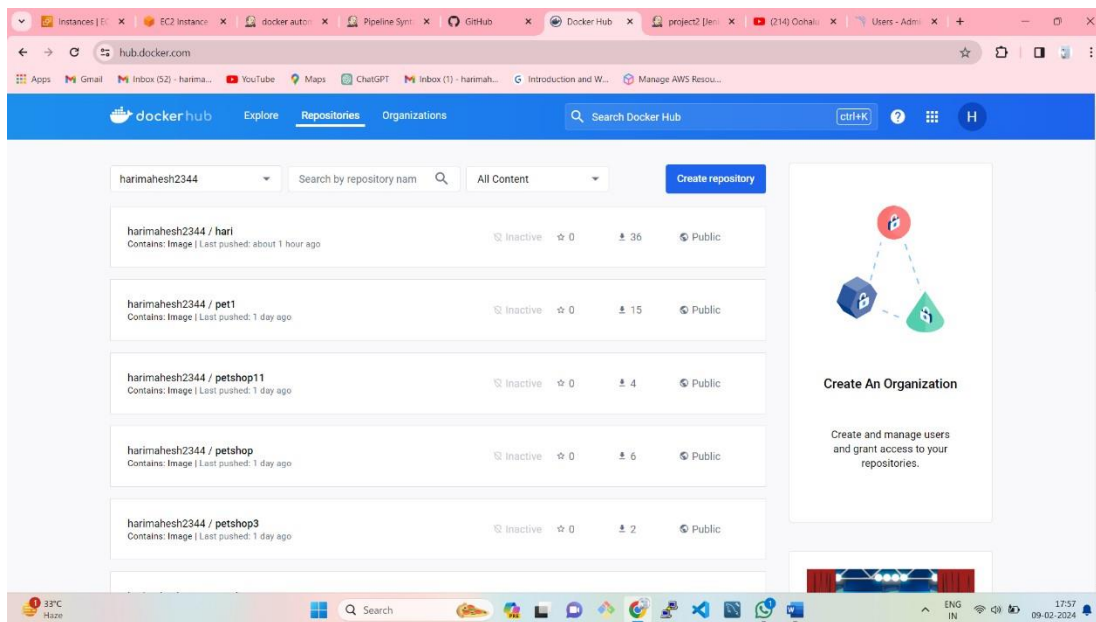
```

```

stage('deploy to
container'){
    steps{
        sh "docker run --name ${CONTAINER_NAME} -d -p 8085:8080
${IMAGE_NAME}:${IMAGE_TAG}"
    }
}
}
}
}
}

```

When you log in to Dockerhub, you will see a new image is created



<Ec2-public-ip:8080/jpetstore>

You will get this output

