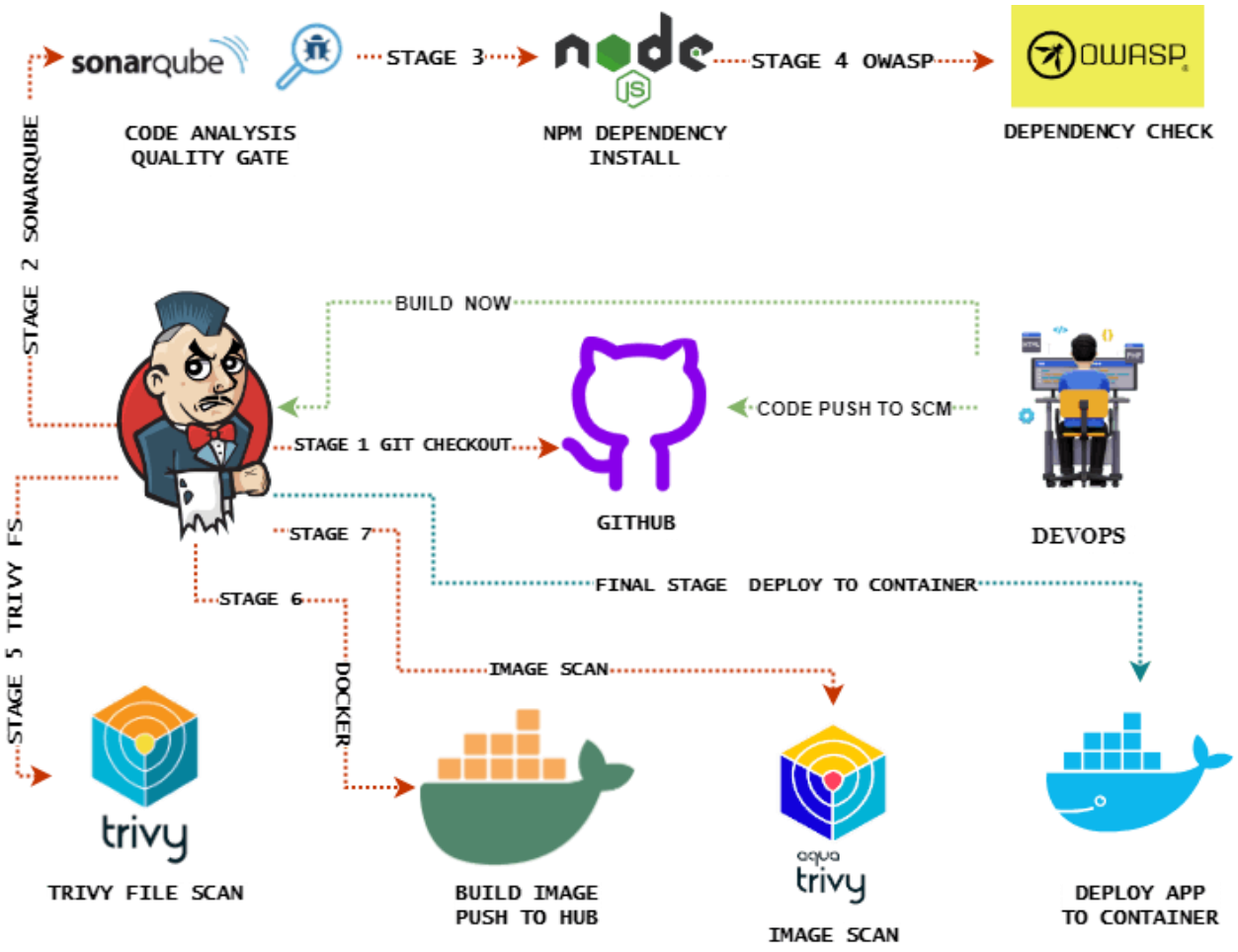
DEPLOYING ZOMATO APP WITH DEVSECOPS CI/CD



Step 1 — Launch 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, Nodejs, and OWASP Dependency Check.

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

Step 5 — Install OWASP Dependency Check Plugins

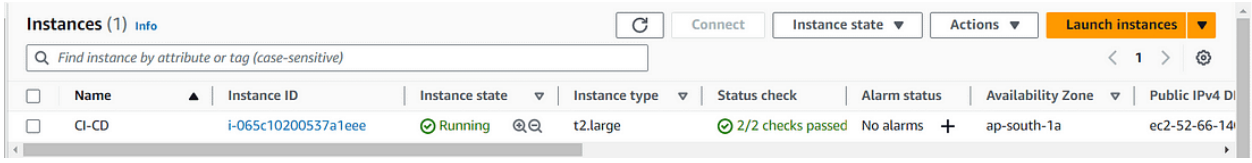
Step 6 — Docker Image Build and Push

Step 7 — Deploy the image using Docker

Step 8 — Terminate the AWS EC2 Instances.

STEP1: Launch 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



Install Jenkins, Docker and Trivy

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
/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
```

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.

Now, grab your Public IP Address

```
<EC2 Public IP Address:8080>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

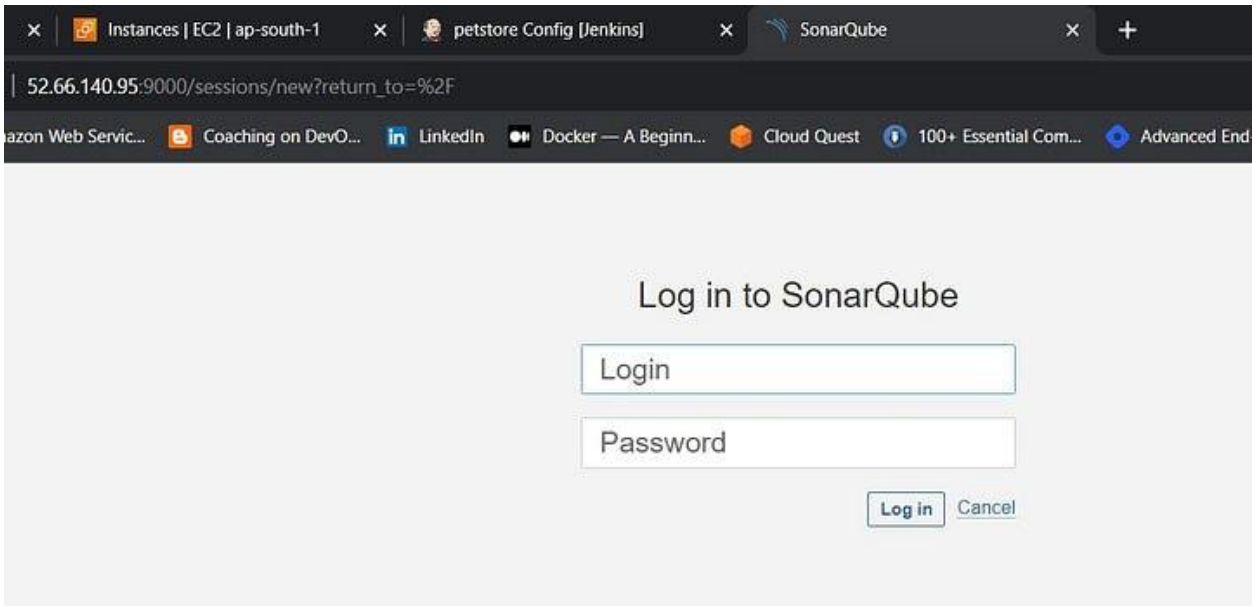
Install Docker

```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

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

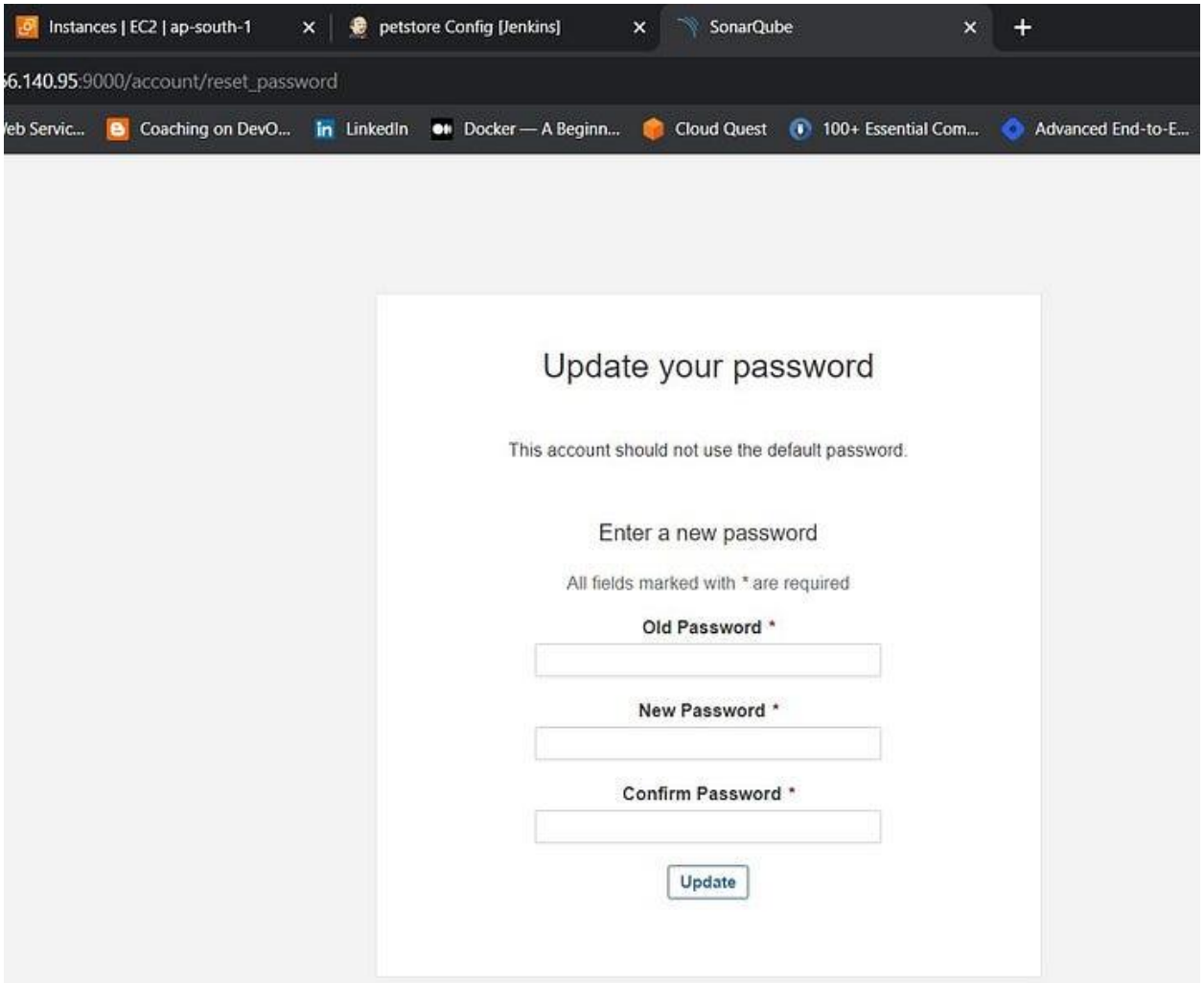
```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

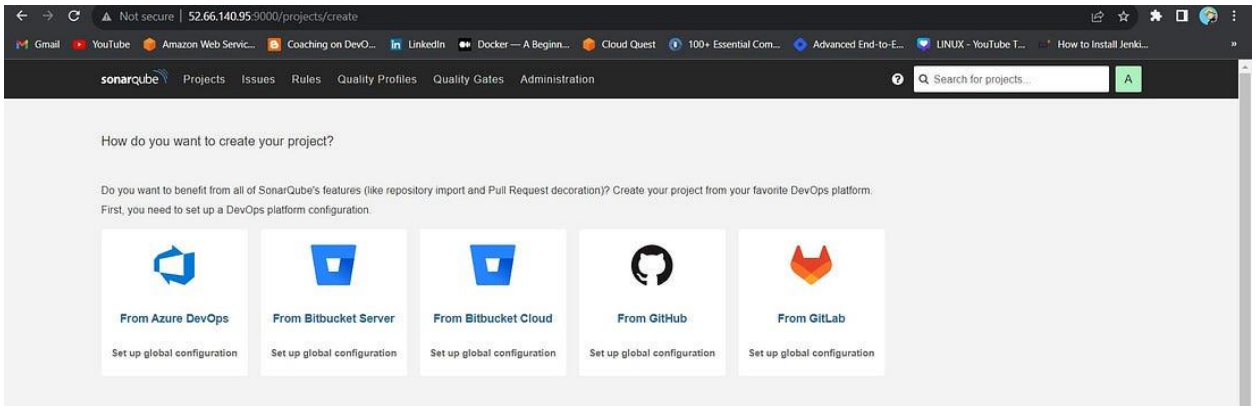
Now our sonarqube is up and running



Enter username and password, click on login and change password

```
username admin
password admin
```





Install Trivy

```
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, NodeJs, OWASP Dependency Check

Install Plugin

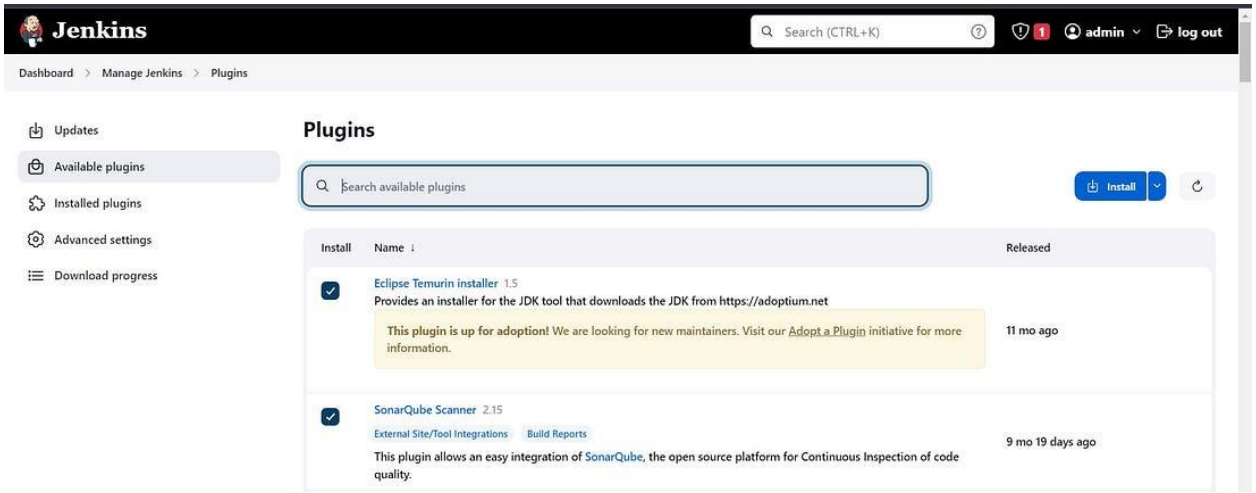
Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

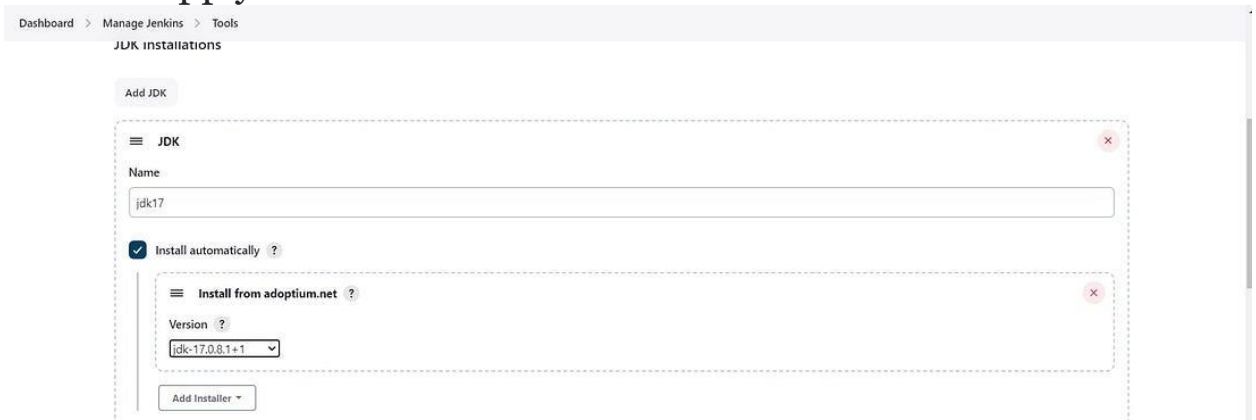
2 → SonarQube Scanner (Install without restart)

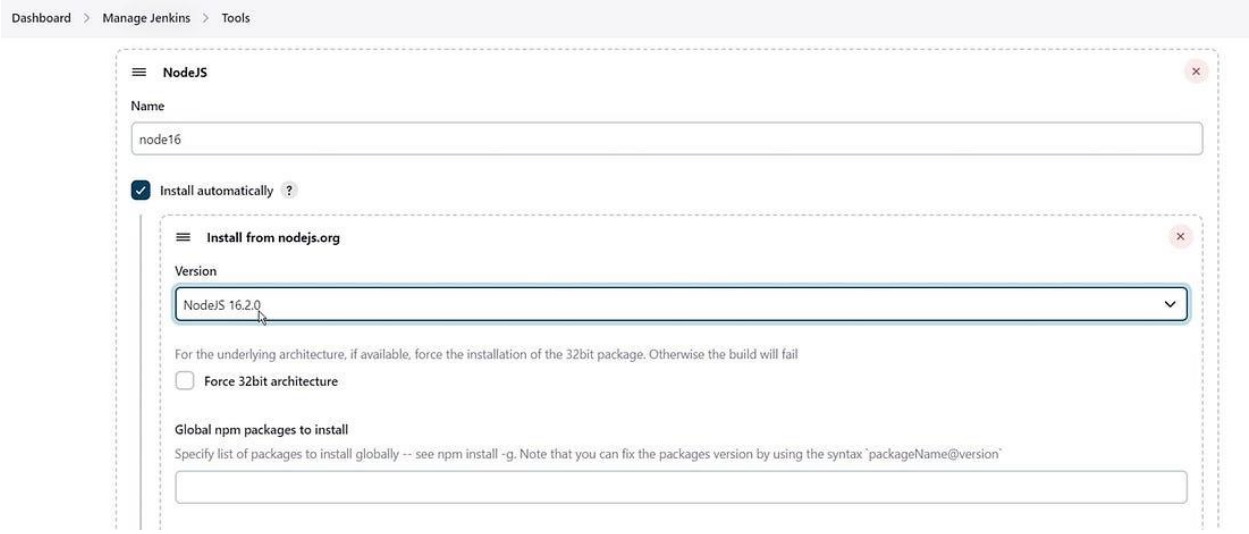
3 → NodeJs Plugin (Install Without restart)



Configure Java and Nodejs in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save



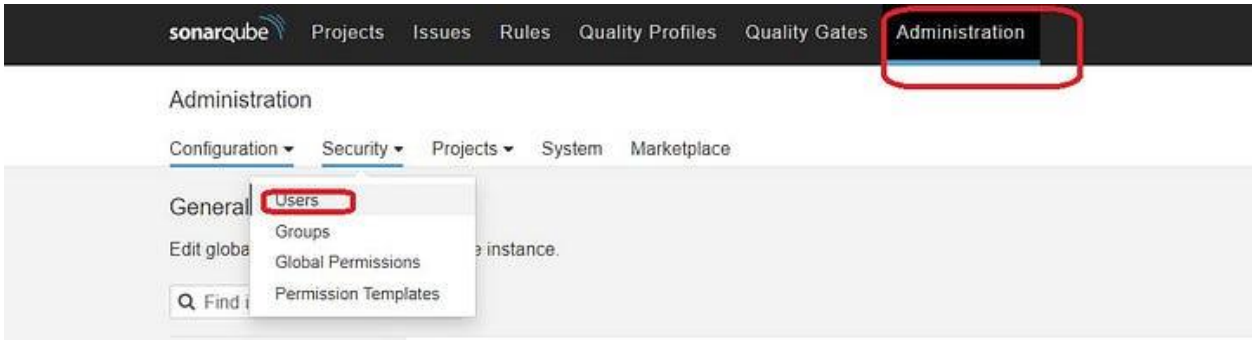


Create a Job

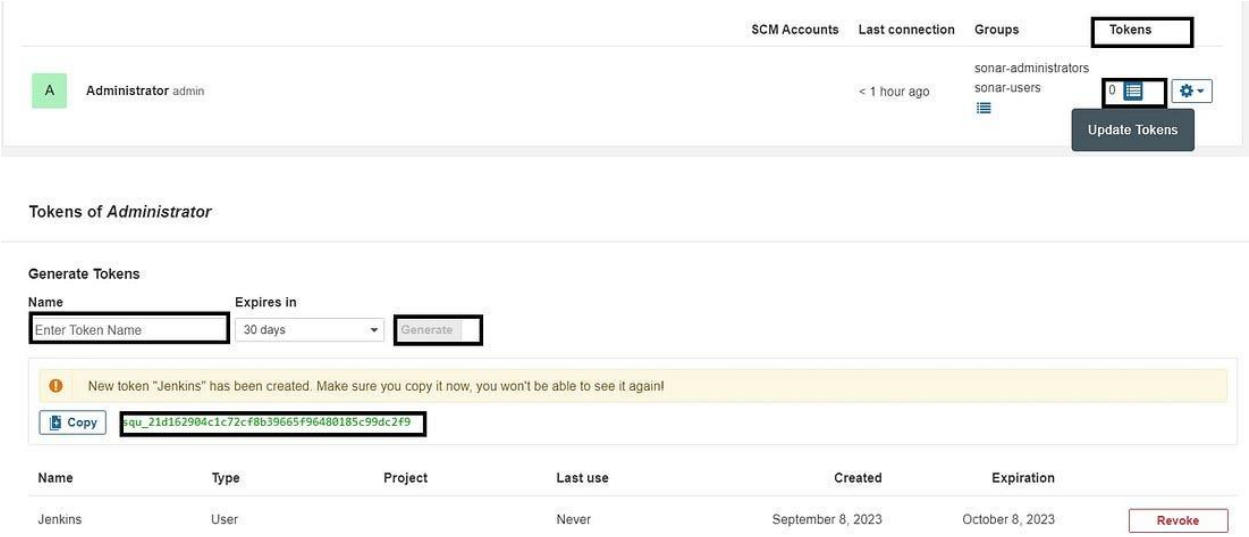
create a job as Zomato Name, select pipeline and click on ok.

Step 4 — Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

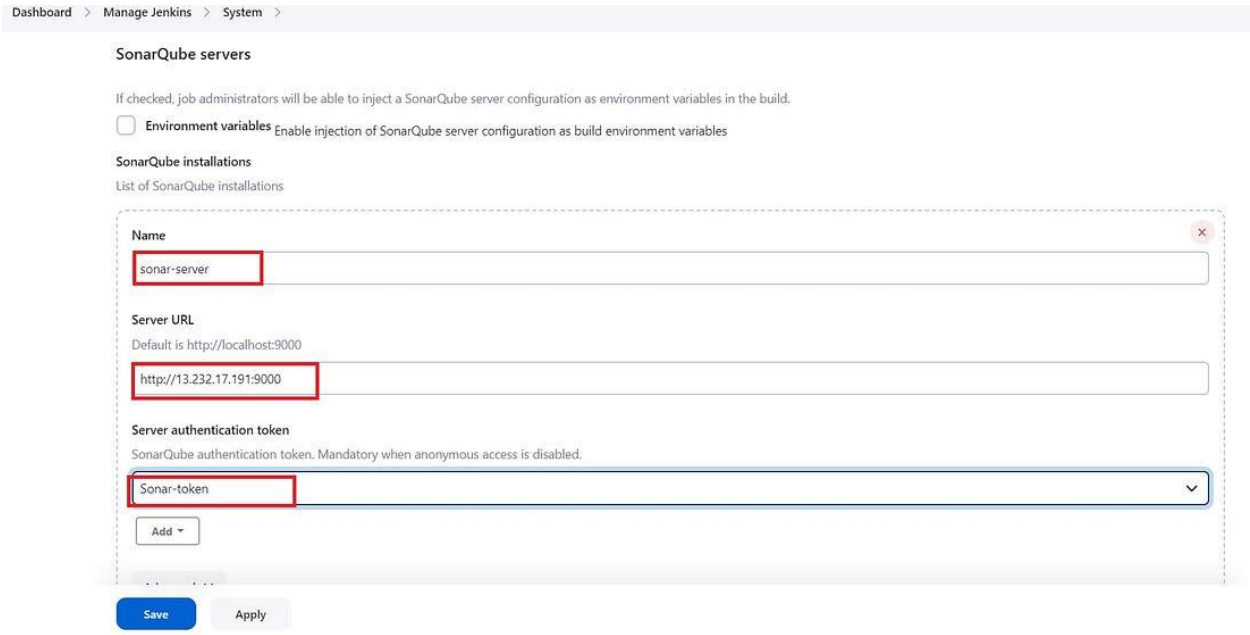


copy Token

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

You will this page once you click on create

Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

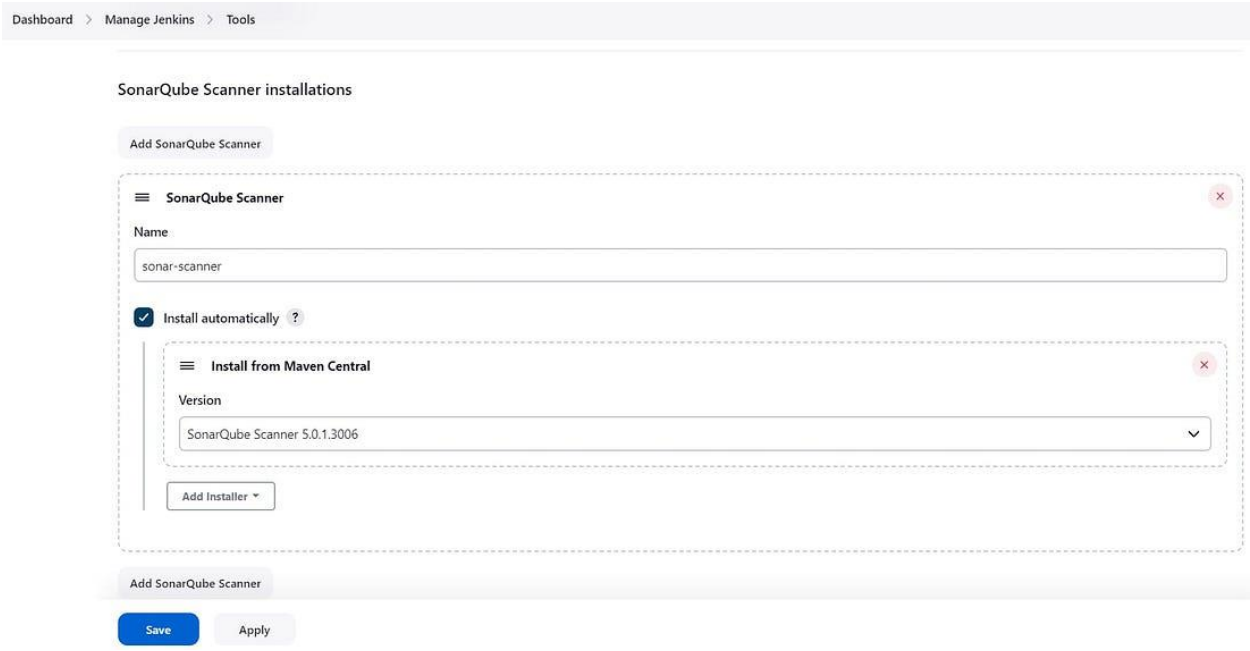


Click on Apply and Save

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

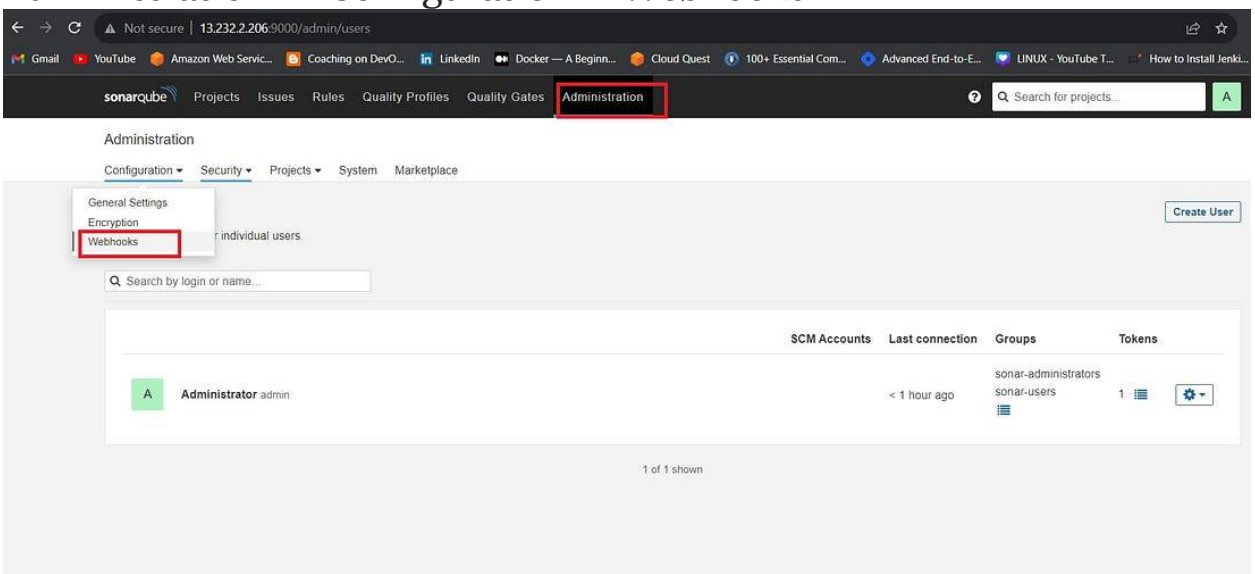
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

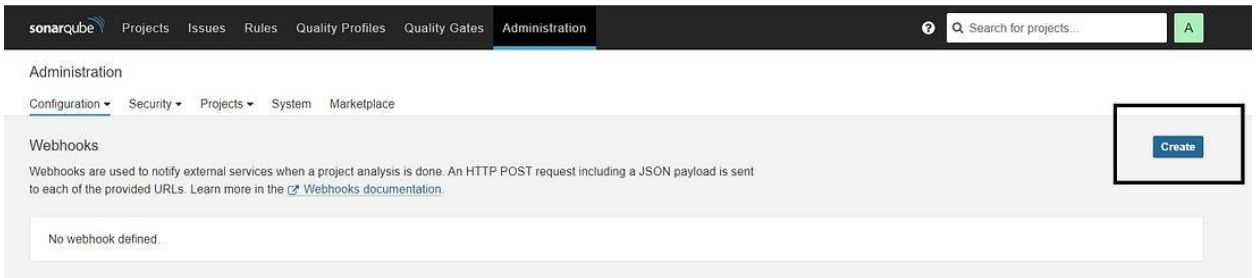


In the Sonarqube Dashboard add a quality gate also

Administration → Configuration →Webhooks

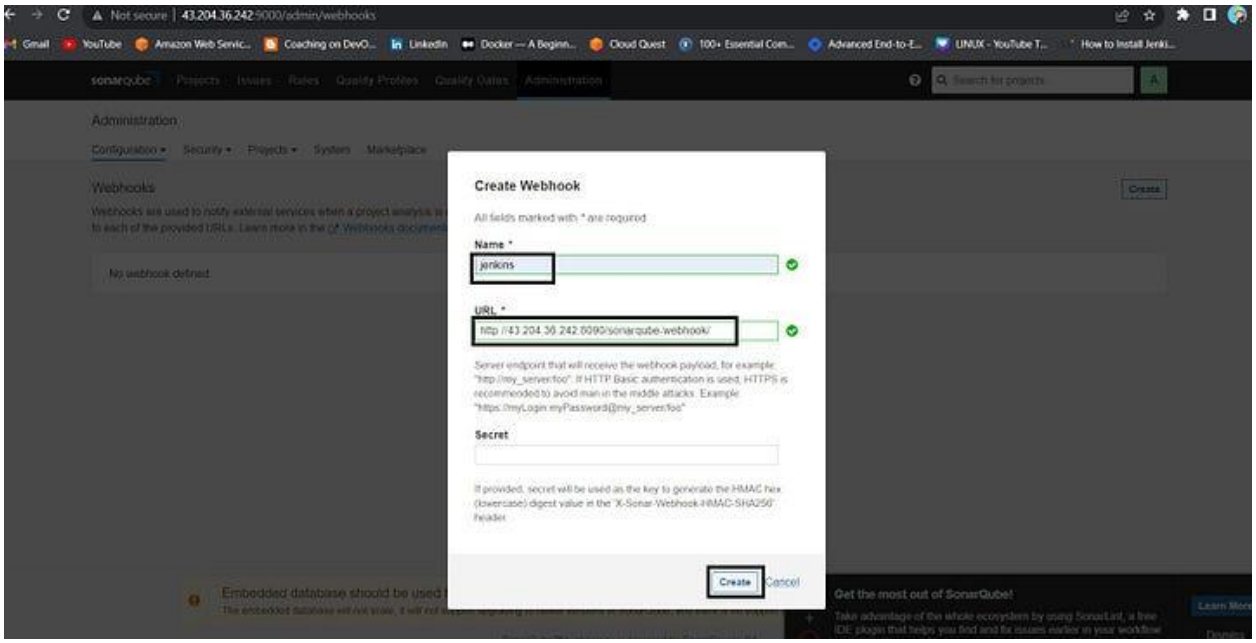


Click on Create



Add details

```
<http://jenkins-public-ip:8080>/sonarqube-webhook/
```



Let's go to our Pipeline and add the script in our Pipeline Script.

```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
```

```
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Milky19/Zomato-
Clone.git'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=zomato \
                    -Dsonar.projectKey=zomato '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-token'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
    }
}
```

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

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies
5s	379ms	1s	16s	520ms	1min 12s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s

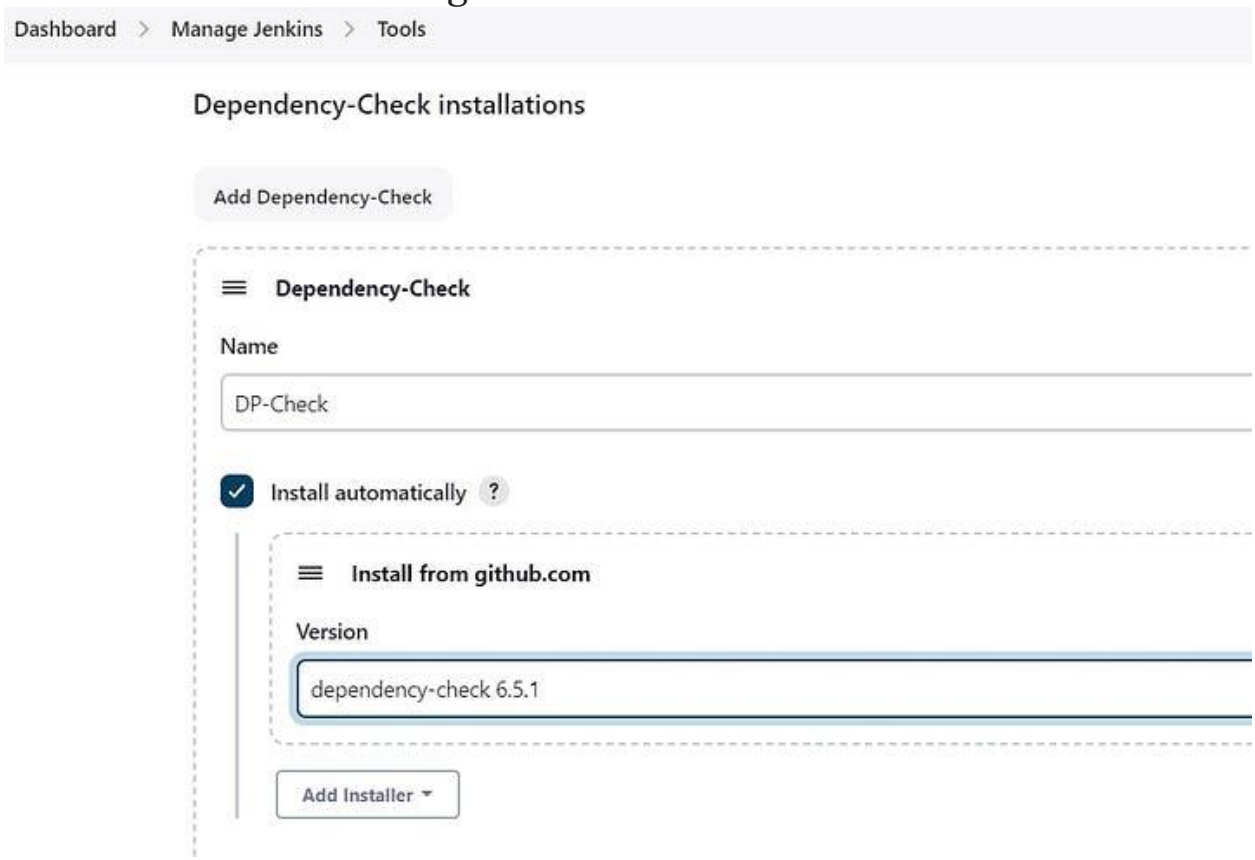
To see the report, you can go to Sonarqube Server and go to Projects.

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('OWASP FS SCAN') {
            steps {
                dependencyCheck additionalArguments: '--scan ./ --
disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
```

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN
5s	379ms	1s	16s	520ms	1min 12s	1min 45s	13s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s	3min 31s	27s

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

Dependency-Check Results

SEVERITY DISTRIBUTION

5 | 8 | 3

Search

File Name	Vulnerability	Severity	Weakness
+ css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
+ ejs:3.1.8	OSSINDEX CVE-2023-29827	High	CWE-74
+ json5:1.0.1	NVD CVE-2022-46175	High	CWE-1321
+ jsonpointer:5.0.1	NVD CVE-2022-4742	Critical	CWE-1321
+ nth-check:1.0.2	NVD CVE-2021-3803	High	CWE-1333
+ parseurl:1.3.3	NVD CVE-2022-0722	High	CWE-200
+ parseurl:1.3.3	NVD CVE-2022-2216	Critical	CWE-918
+ parseurl:1.3.3	NVD CVE-2022-2217	Medium	CWE-79
+ parseurl:1.3.3	NVD CVE-2022-2218	Medium	CWE-79
+ parseurl:1.3.3	NVD CVE-2022-2900	Critical	CWE-918

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`

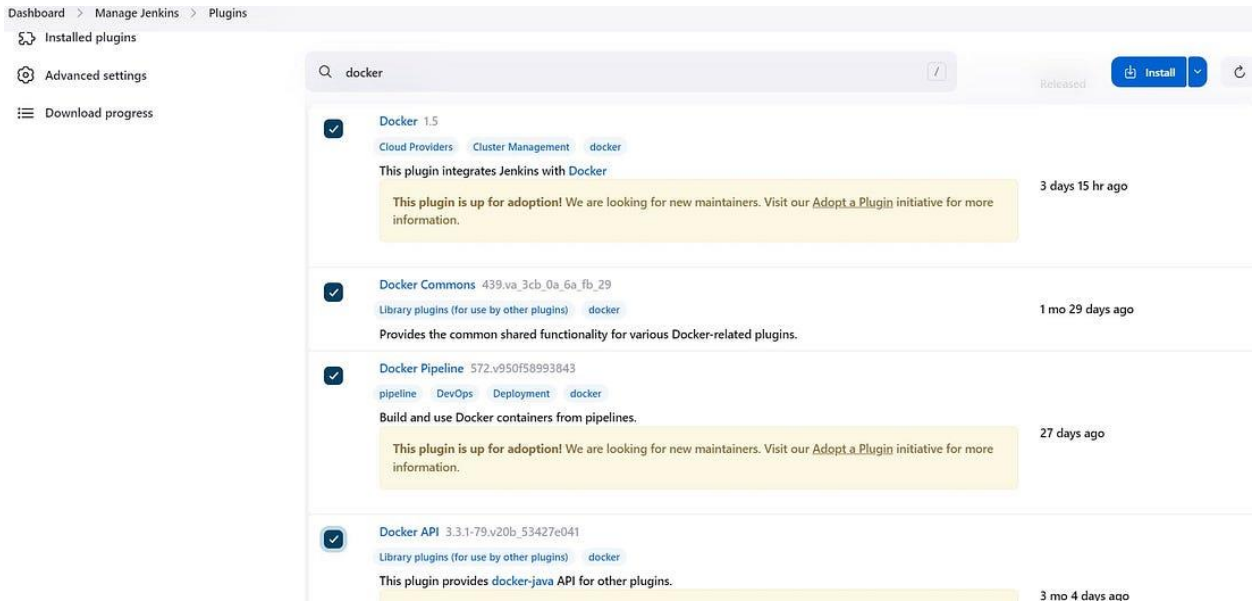
`Docker Commons`

`Docker Pipeline`

`Docker API`

`docker-build-step`

and click on install without restart



Add DockerHub Username and Password under Global Credentials

Add this stage to Pipeline Script

```
stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName:
'docker'){
                        sh "docker build -t zomato ."
                        sh "docker tag zomato hanvitha/zomato:latest "
                        sh "docker push hanvitha/zomato:latest "
                    }
                }
            }
```

```
        }
        stage("TRIVY"){
            steps{
                sh "trivy image hanvitha/zomato:latest > trivy.txt"
            }
        }
```

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

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY
3s	366ms	1s	19s	451ms	1min 20s	2min 1s	16s	3min 9s	4s
154ms	341ms	1s	25s	315ms	1min 36s	2min 31s	23s	3min 9s	4s

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

Now Run the container to see if the app coming up or not by adding the below stage

```
stage('Deploy to container'){
    steps{
            sh 'docker run -d --name zomato -p 3000:3000
hanvitha/zomato:latest'
        }
    }
```

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
144ms	284ms	1s	25s	410ms	1min 47s	2min 43s	23s	2min 7s	36s	789ms
146ms	251ms	1s	26s	305ms	1min 36s	2min 35s	23s	1min 50s	2min 8s	1s

You will get this output

<Jenkins-public-ip:3000>

COMPLETE PIPELINE SCRIPT:

```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
```

```
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Milky19/Zomato-
Clone.git'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=zomato \
                    -Dsonar.projectKey=zomato '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-token'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
        stage('OWASP FS SCAN') {
            steps {
                dependencyCheck additionalArguments: '--scan ./ --
disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
        stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName:
```

```
'docker'){
                        sh "docker build -t zomato ."
                        sh "docker tag zomato hanvitha/zomato:latest "
                        sh "docker push hanvitha/zomato:latest "
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image hanvitha/zomato:latest > trivy.txt"
            }
        }
        stage('Deploy to container'){
            steps{
                sh 'docker run -d --name zomato -p 3000:3000
hanvitha/zomato:latest'
            }
        }
    }
}
```