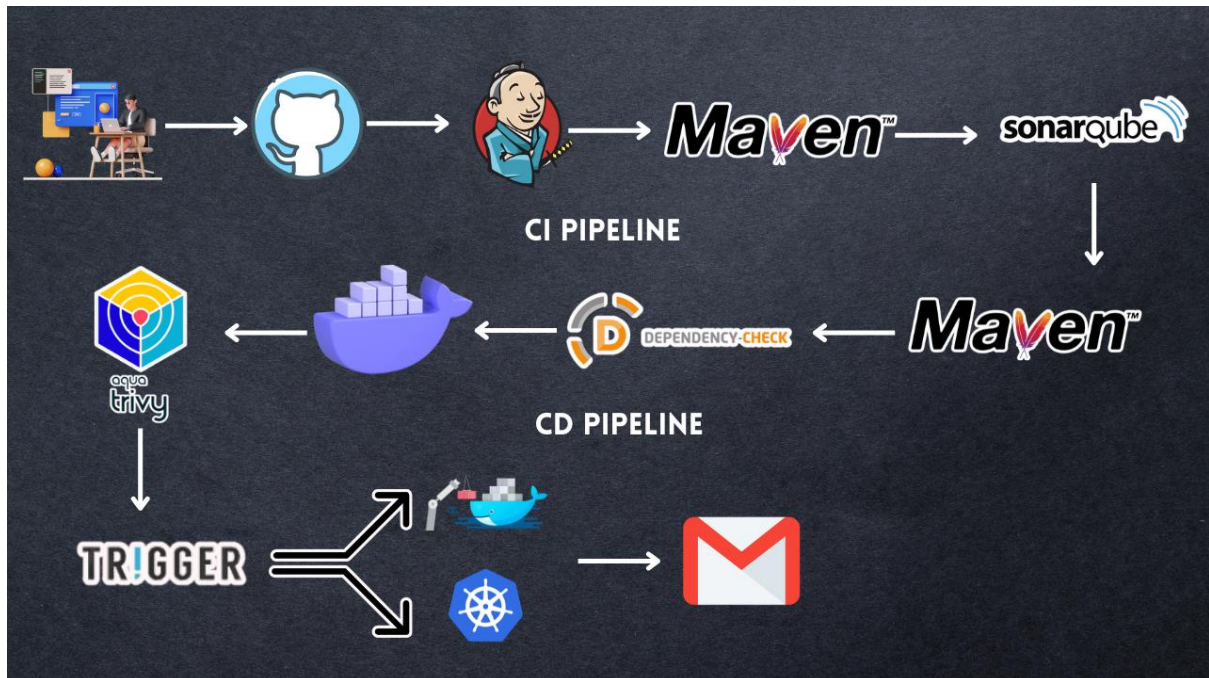


DEVSECOPS Project : Complete CI-CD (3 tier app)-Petshop



Hello friends, we will be deploying a Petshop Java Based Application. This is an everyday use case scenario used by several organizations. We will be using Jenkins as a CICD tool and deploying our application on a Docker container and Kubernetes cluster. Hope this detailed blog is useful.

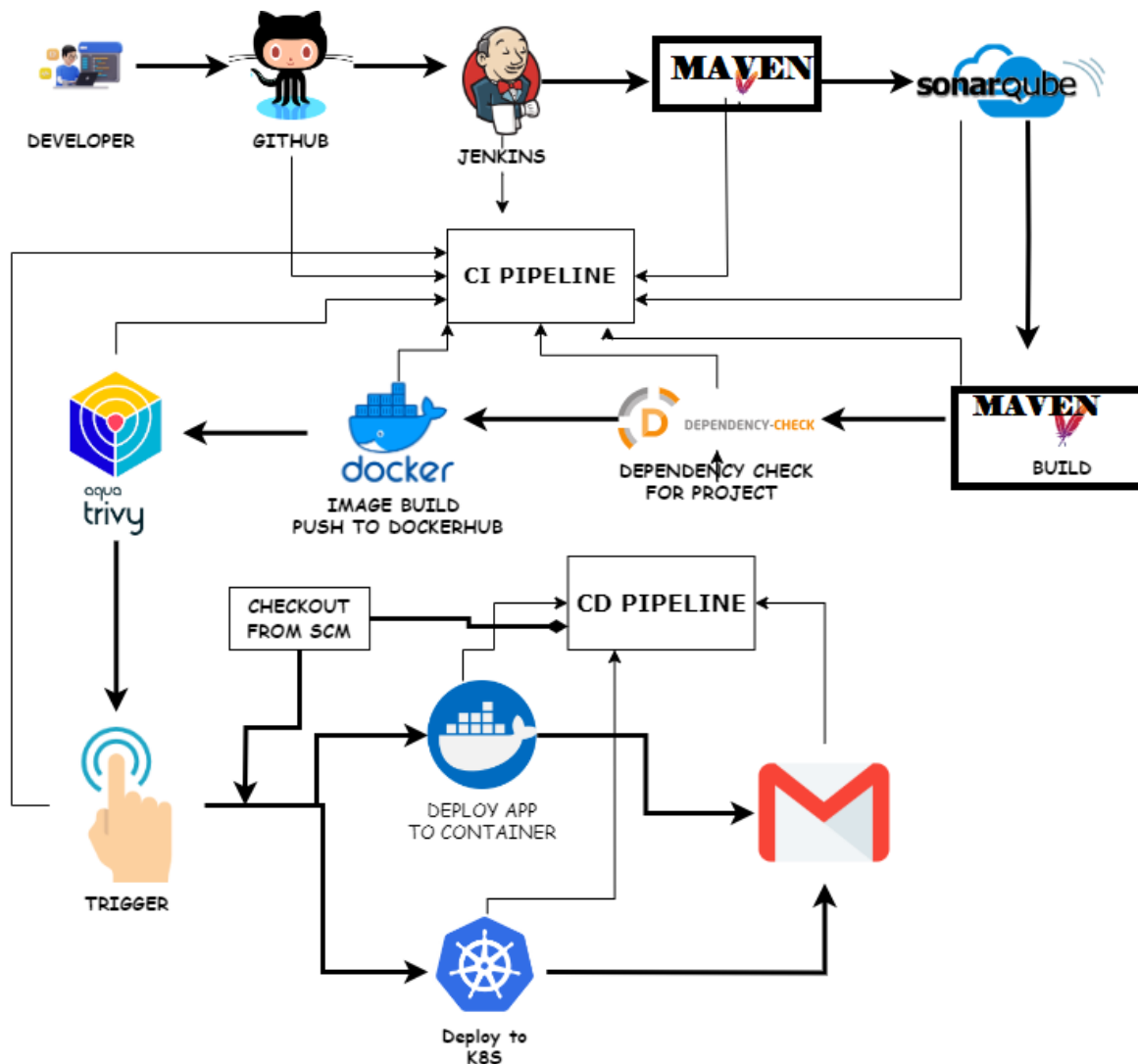
We will be deploying our application in two ways, one using Docker Container and the other using K8S cluster.

Project Repo: <https://github.com/Ai7Ay/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 — Deploy the image using Docker
- Step 8 — Kubernetes master and slave setup on Ubuntu (20.04)
- Step 9 — Access the Real World Application
- Step 10 — Terminate the AWS EC2 Instances.

Now, let's get started and dig deeper into each of these steps:-



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).

Instances (1) Info								
Find instance by attribute or tag (case-sensitive)								
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 Di
<input type="checkbox"/>	CI-CD	i-065c10200537a1eee	Running	t2.large	2/2 checks passed	No alarms	ap-south-1a	ec2-52-66-14

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

/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 Environment="Jenkins_port=8090" save and exit

sudo systemctl daemon-reload

sudo systemctl restart jenkins

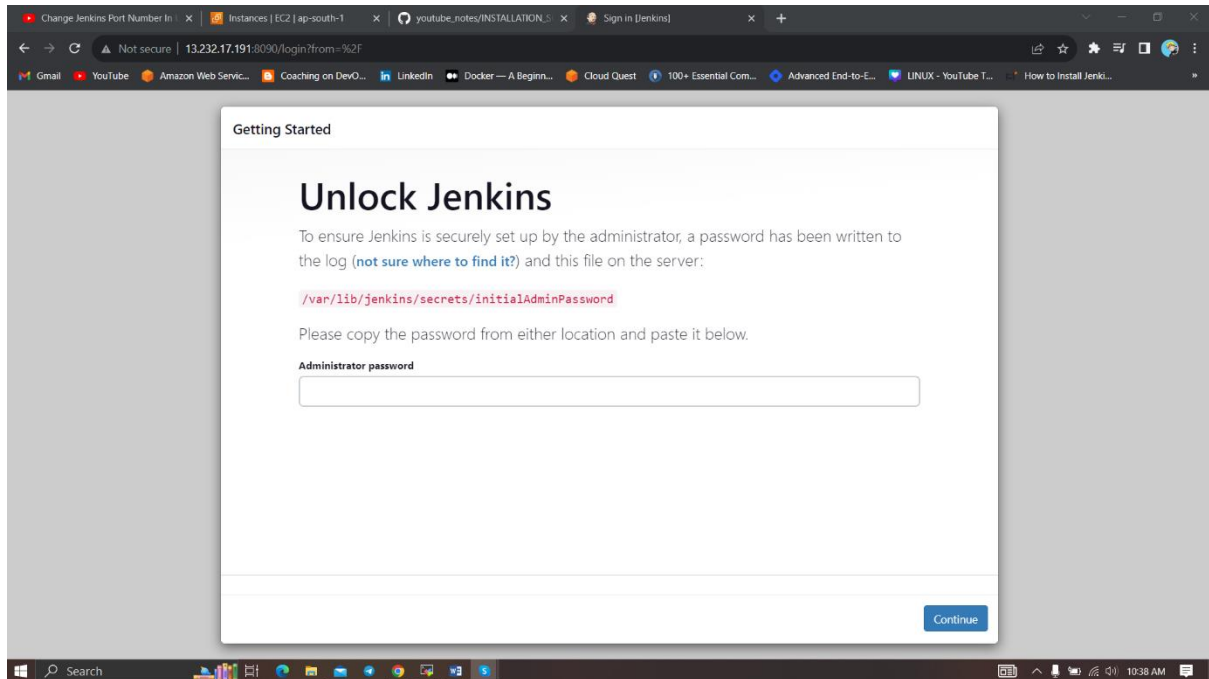
sudo systemctl status jenkins

```

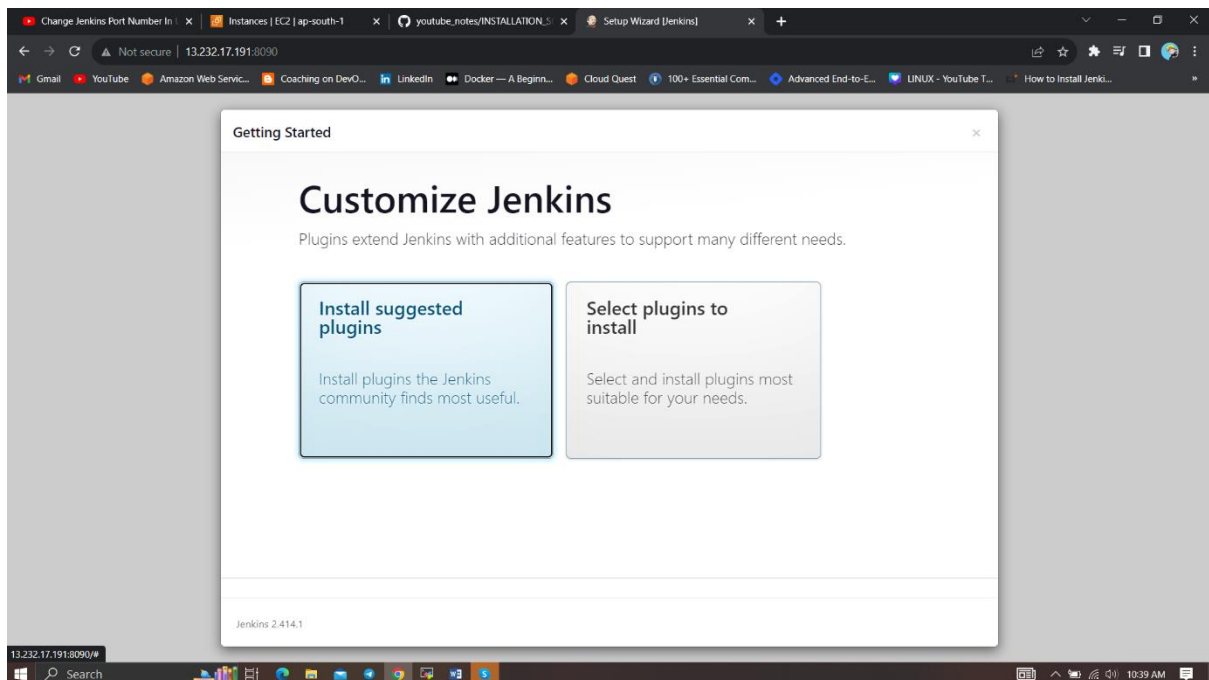
Now, grab your Public IP Address

EC2 Public IP Address:8090

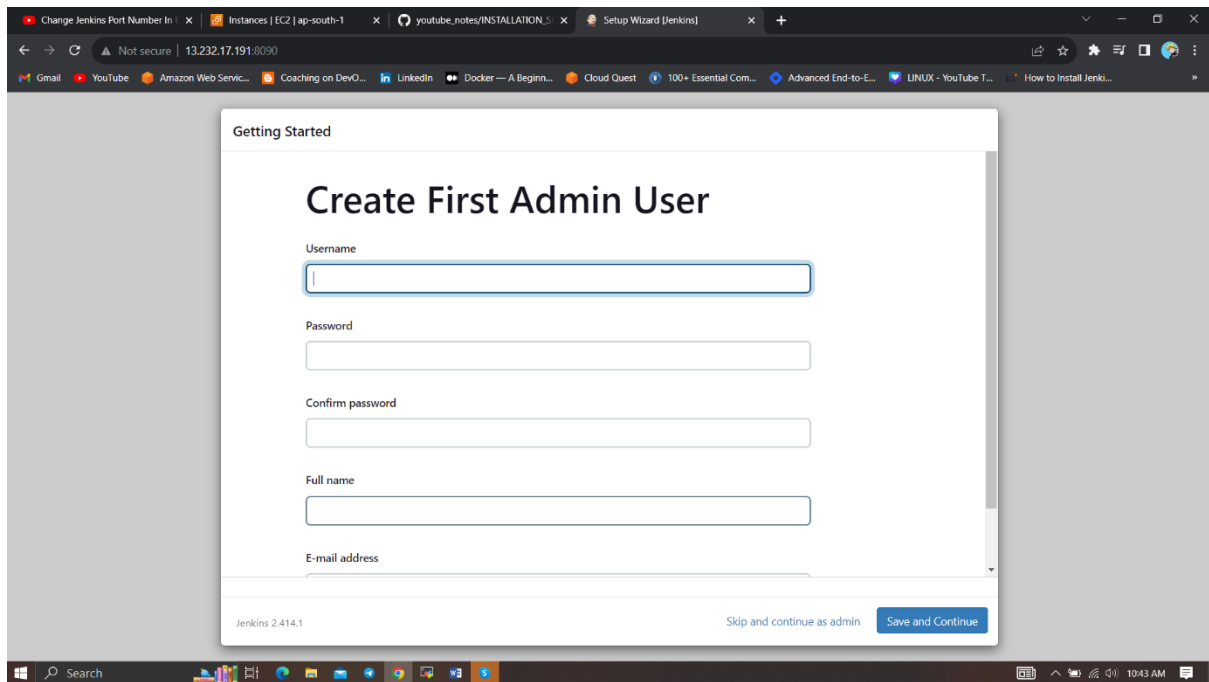
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



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

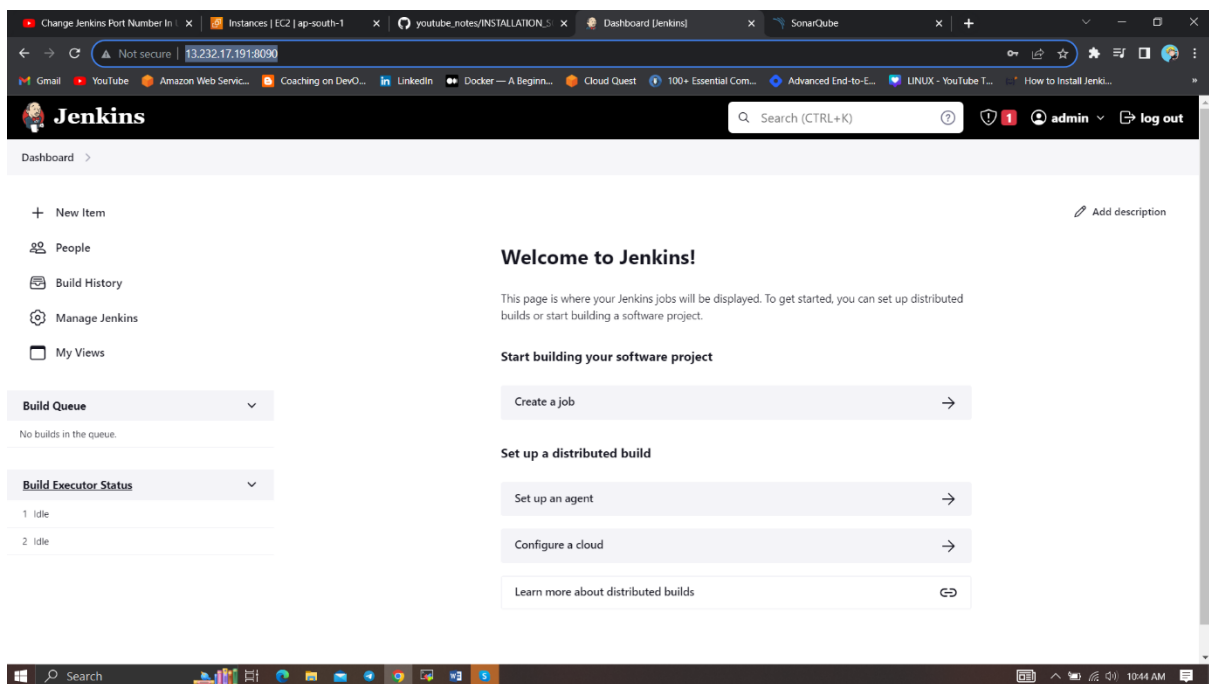


Jenkins will now get installed and install all the libraries.



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
```

```
newgrp docker
```

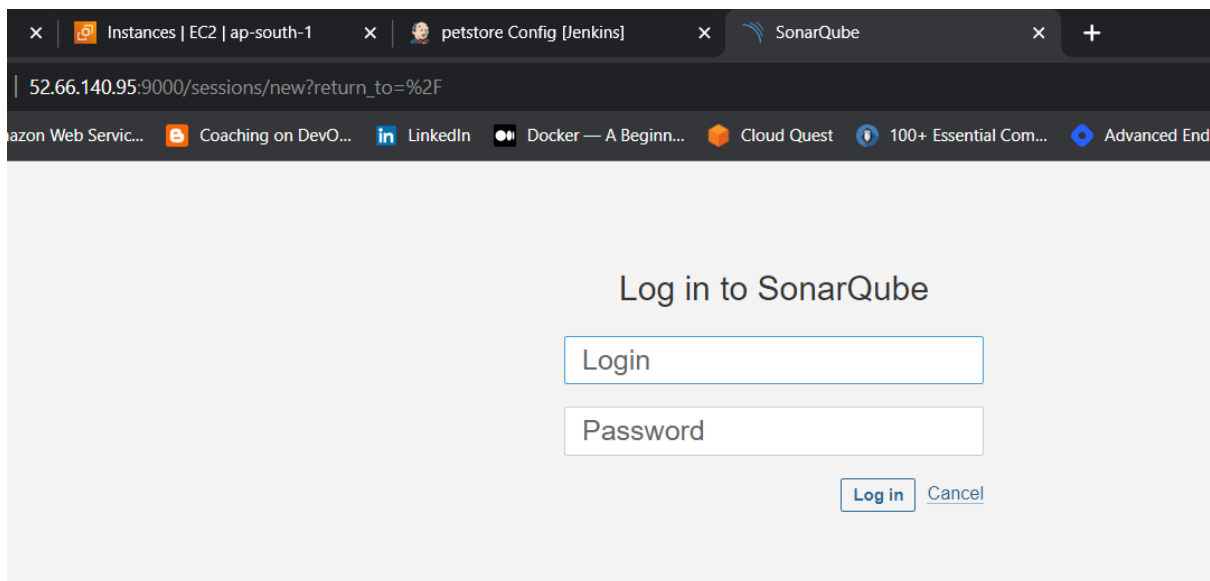
```
sudo chmod 777 /var/run/docker.sock
```

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
c2048138466a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f98f5e: Pull complete
68a29588c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                               NAMES
4b60c96bf9ad   sonarqube:lts-commu... "/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:~$
```

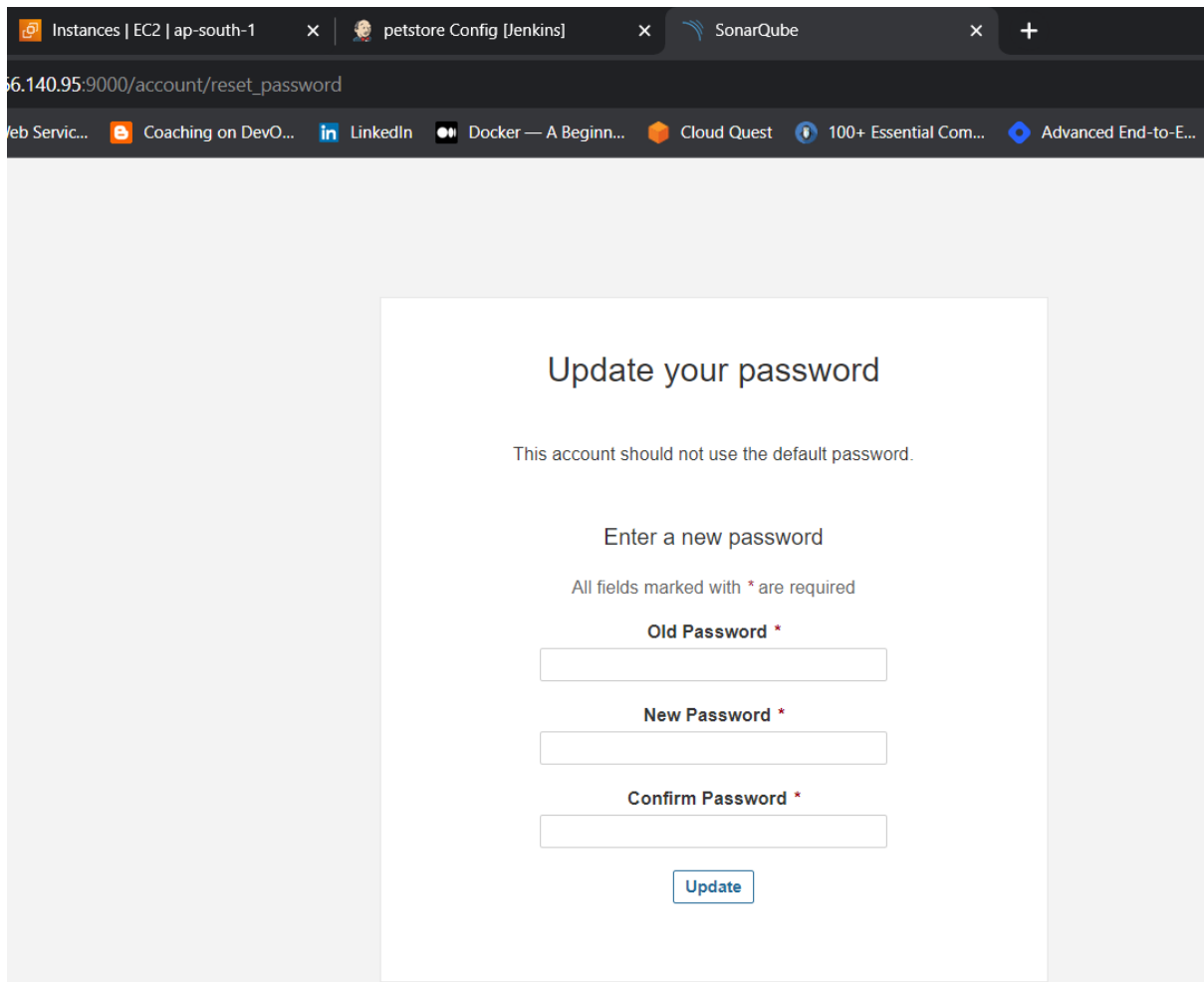
Now our sonarqube is up and running



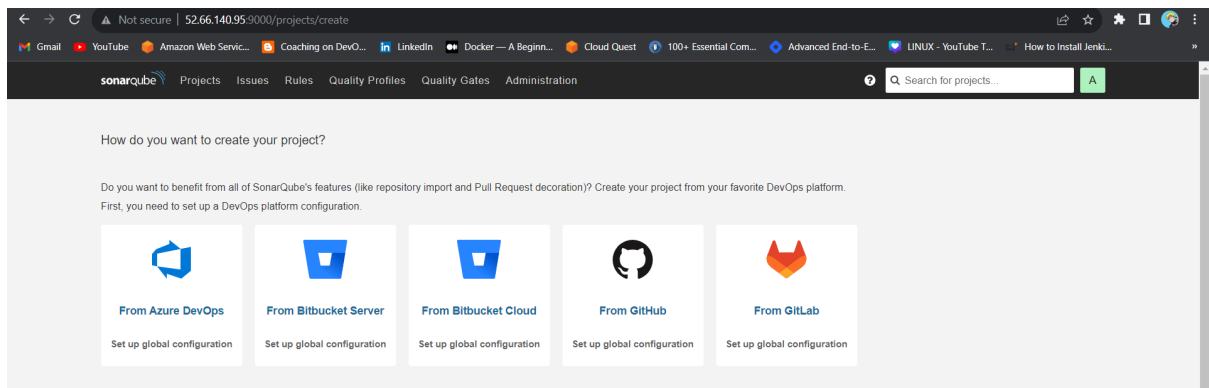
Enter username and password, click on login and change password

username admin

password admin



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

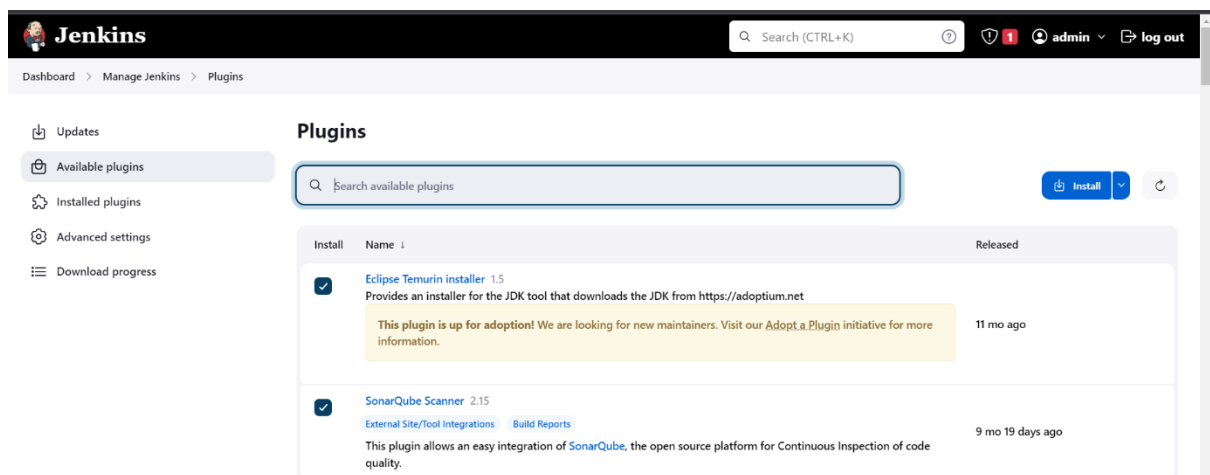
3A — Install Plugin

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

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

3C — Create a Job

Label it as PETSHOP, click on Pipeline and OK.

Jenkins Search (CTRL+K) admin log out

Dashboard > All >

Enter an item name

petstore

» 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 build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

OK

Enter this in Pipeline Script,

```
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'
        }
    }
}
}

```

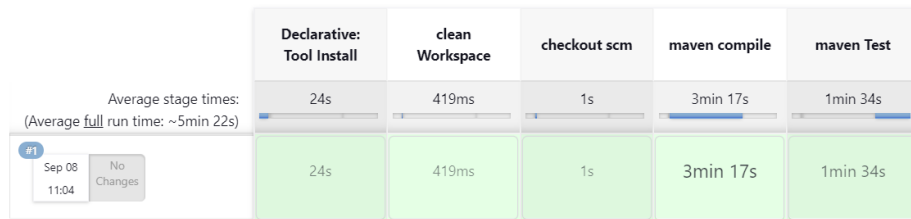
The stage view would look like this,

Pipeline petstore

Add description

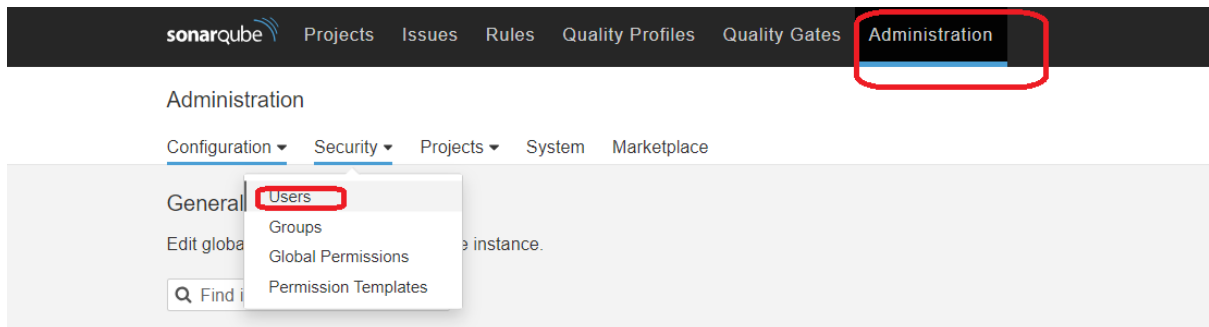
Disable Project

Stage View

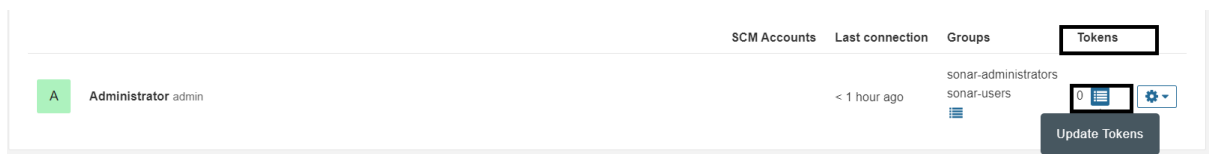


Step 4 — Configure Sonar Server in Manage Jenkins

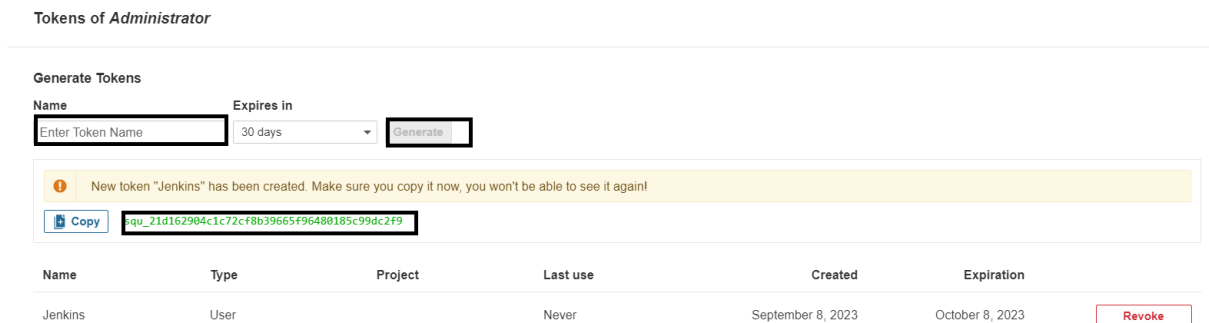
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token



Create a token with a name and generate



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.

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name

sonar-scanner

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 5.0.1.3006

Add Installer

Add SonarQube Scanner

Save Apply

In the Sonarqube Dashboard add a quality gate also

Administration→ Configuration→Webhooks

sonarqube Projects Issues Rules Quality Profiles Quality Gates **Administration** ? Search for projects... A

Administration

Configuration Security Projects System Marketplace

General Settings Encryption **Webhooks** individual users

Search by login or name...

	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	1

1 of 1 shown

Click on Create

sonarqube Projects Issues Rules Quality Profiles Quality Gates **Administration** ? Search for projects... A

Administration

Configuration Security Projects System Marketplace

Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).

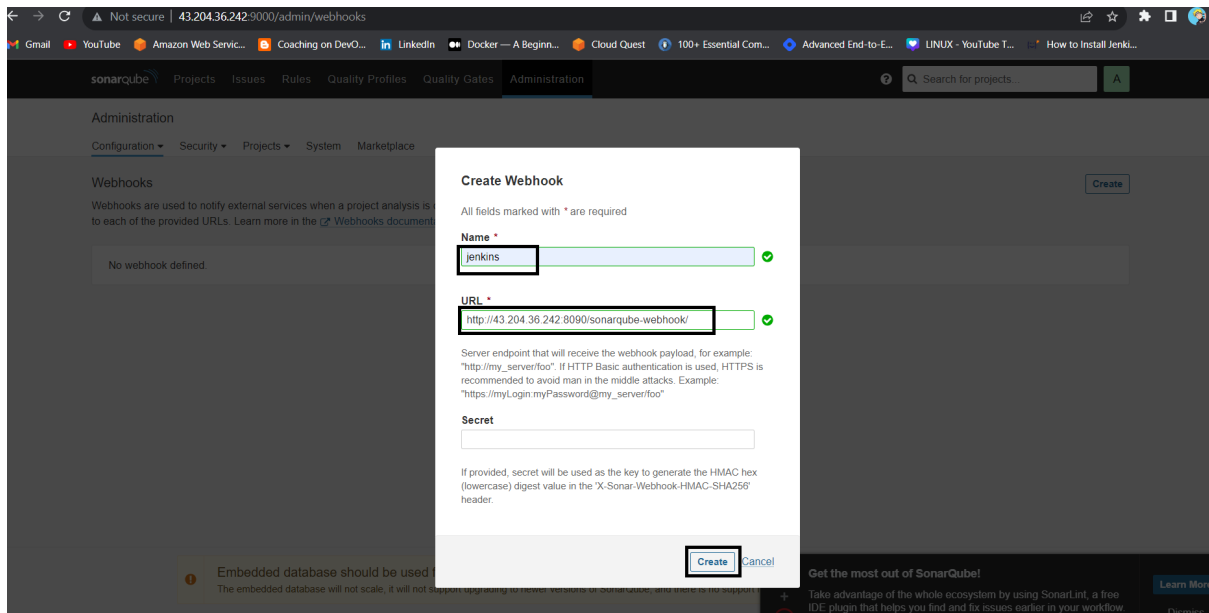
No webhook defined.

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("Sonarqube 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 {
```

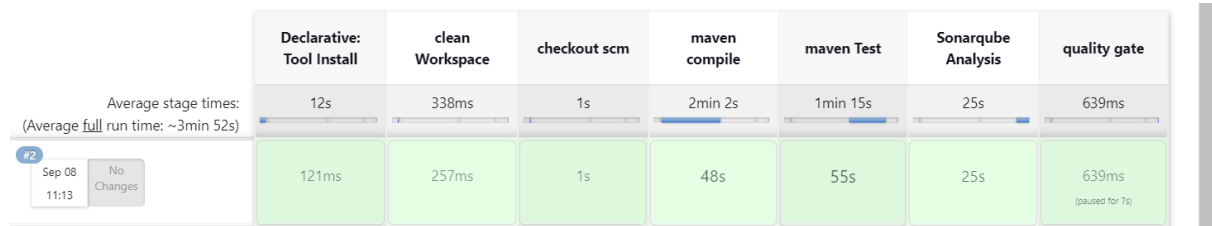
```
            waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
```

```
        }
```

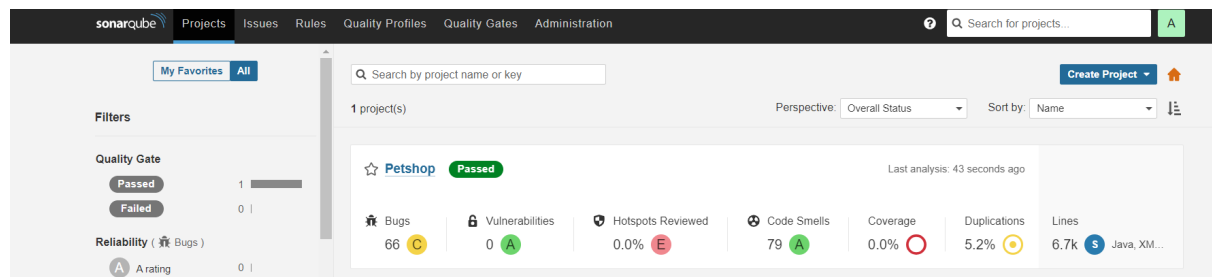
```
}
```

```
}
```

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 →

Dependency-Check installations

[Add Dependency-Check](#)**Dependency-Check**

Name

DP-Check

☒ Install automatically ?**Install from github.com**

Version

dependency-check 6.5.1

[Add Installer](#) ▼

Click on Apply and Save here.

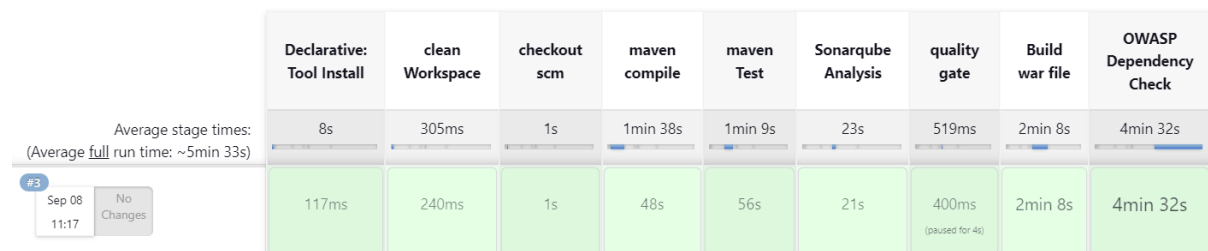
Now go configure → Pipeline and add this stage to your pipeline and build.

```
stage('Build war file'){
    steps{
        sh 'mvn clean install -DskipTests=true'
    }
}

stage("OWASP Dependency Check"){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format XML ', odciInstallation: 'DP-
Check'
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
    }
}
```

The stage view would look like this,

Stage View



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

Dashboard > petstore > #3 > Dependency-Check

Dependency-Check Results

SEVERITY DISTRIBUTION

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

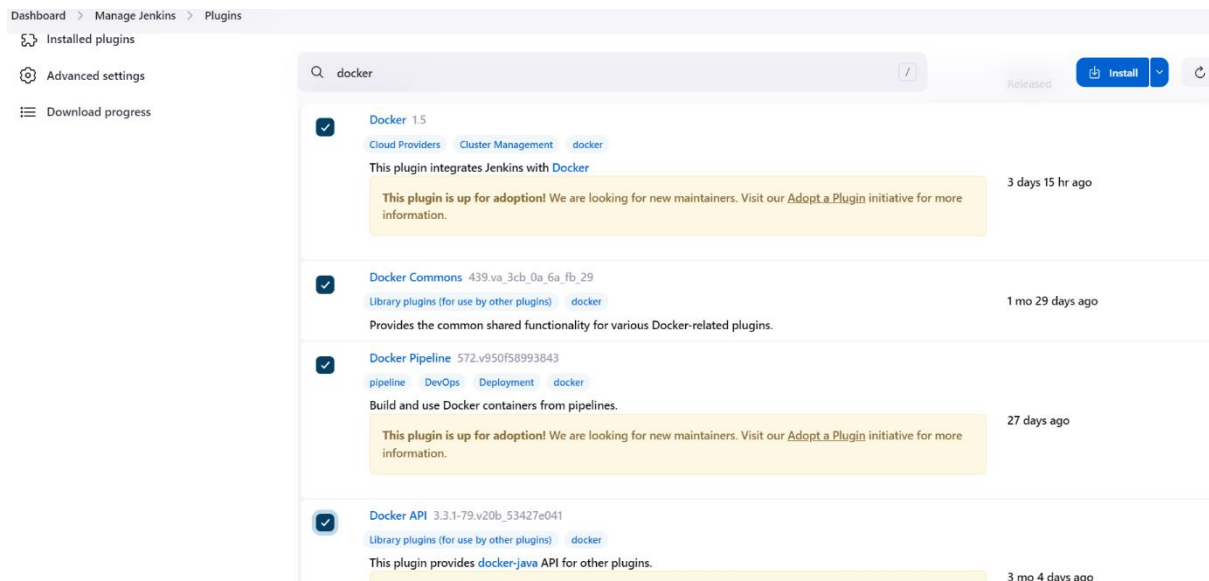
Docker Commons

Docker Pipeline

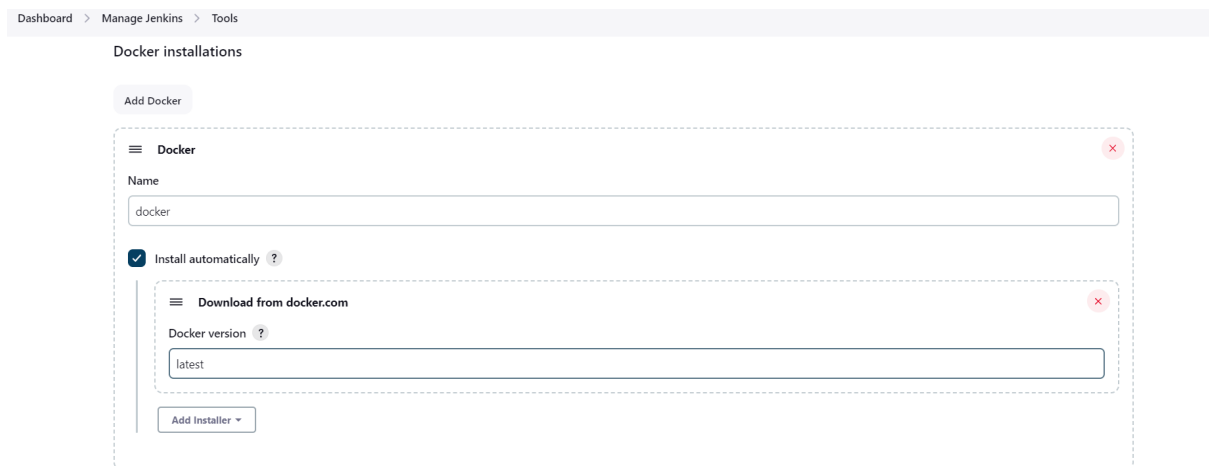
Docker API

docker-build-step

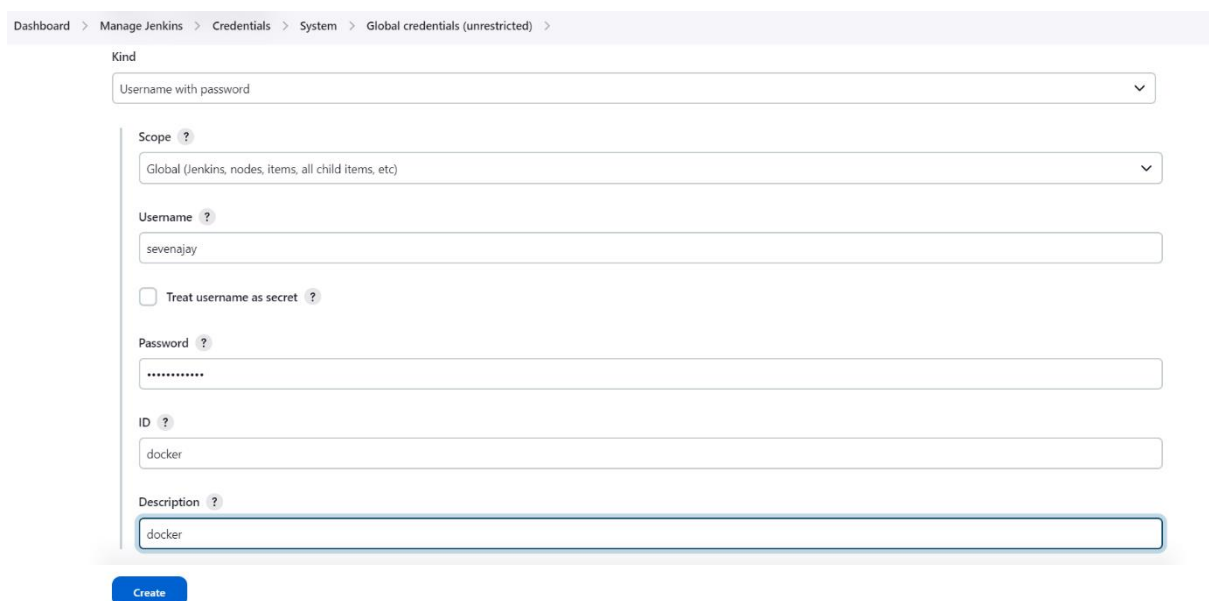
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



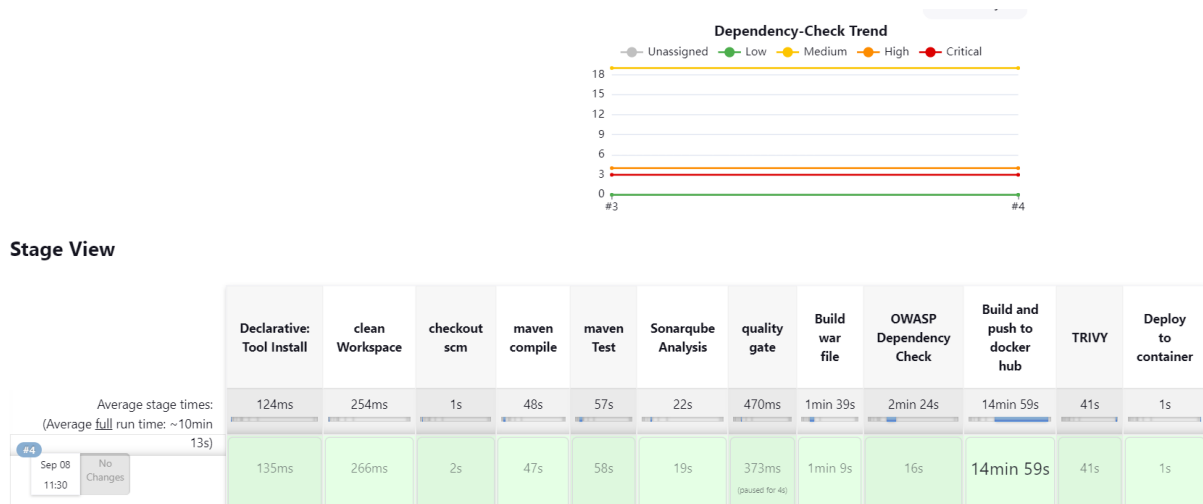
Add DockerHub Username and Password under Global Credentials



Add this stage to Pipeline Script

```
stage ('Build and push to docker hub'){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
                sh "docker build -t petshop ."
                sh "docker tag petshop sevenajay/petshop:latest"
                sh "docker push sevenajay/petshop:latest"
            }
        }
    }
}
stage("TRIVY"){
    steps{
        sh "trivy image sevenajay/petshop:latest > trivy.txt"
    }
}
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.



Now, when you do

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

sevenajay / petshop

Docker commands
[Public View](#)

Description

This repository does not have a description

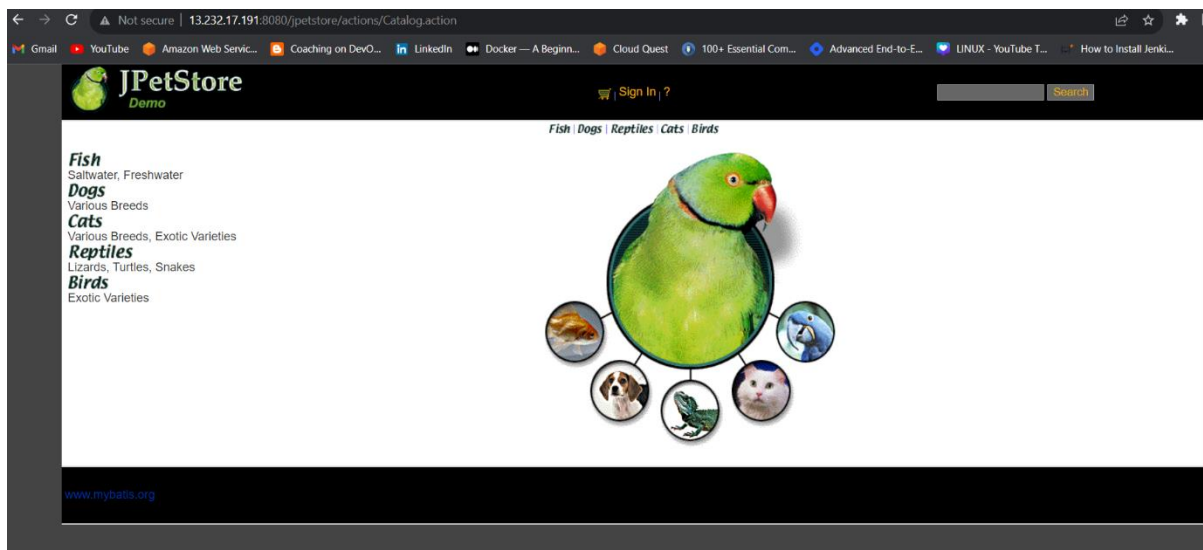
Last pushed: an hour ago

To push a new tag to this repository:

```
docker push sevenajay/petshop:tagname
```

<Ec2-public-ip:8080/jpetstore>

You will get this output



Step 8 — Kubernetes Setup

Connect your machines to Putty or Mobaxtreme

Take-Two Ubuntu 20.04 instances one for k8s master and the other one for worker.

Install Kubectl on Jenkins machine also.

Kubectl on Jenkins to be installed

```
sudo apt update
```

```
sudo apt install curl
```

```
curl -LO https://dl.k8s.io/release/$(curl -L -s  
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
kubectl version --client
```

Part 1 ———Master Node———

```
sudo su
```

```
hostname master
```

```
bash
```

```
clear
```

———Worker Node———

```
sudo su
```

```
hostname worker
```

```
bash
```

```
clear
```

Part 2 ———Both Master & Node ———

```
sudo apt-get update
```

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

```
sudo usermod -aG docker Ubuntu
```

```
newgrp docker
```

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

```
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

```
sudo tee /etc/apt/sources.list.d/kubernetes.list &&&EOF
```

```
deb https://apt.kubernetes.io/ kubernetes-xenial main
```

```
EOF
```

```
sudo apt-get update
```

Install Kubernetes Plugin, Once it's installed successfully

Dashboard > Manage Jenkins > Plugins

Updates
Available plugins
Installed plugins
Advanced settings
Download progress

Plugins

Q Kuber

Install

Install	Name	Released
✓	Kubernetes Credentials 0.11 kubernetes credentials Common classes for Kubernetes credentials	9 days 16 hr ago
✓	Kubernetes Client API 6.8.1-224.vd388fca_4db_3b_ kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	9 days 17 hr ago
✓	Kubernetes 4029.v5712230ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	9 days 15 hr ago
✓	Kubernetes CLI 1.12.1 kubernetes Configure kubectl for Kubernetes	8 days 22 hr ago

goto manage Jenkins → manage credentials → Click on Jenkins global → add credentials

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

New credentials

Kind
Secret file

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

File
Choose File Secret File.txt

ID ?
k8s

Description ?
k8s

Create

Configuring mail server in Jenkins (Gmail)

Install Email Extension Plugin in Jenkins

Dashboard > Manage Jenkins > Plugins

Updates
Available plugins
Installed plugins
Advanced settings
Download progress

Plugins

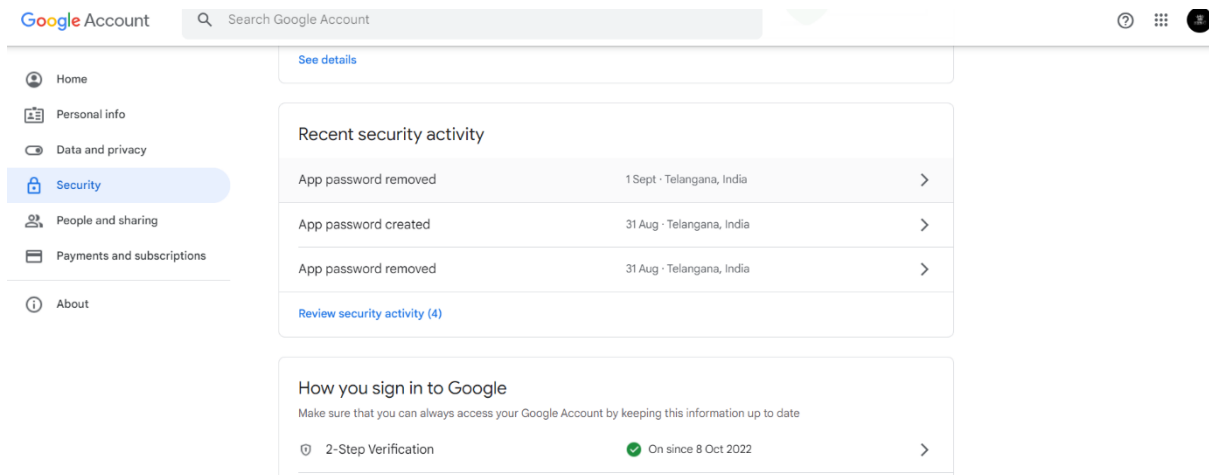
Q Email Ex

Install

Install	Name	Released
✓	Email Extension Template 1.5 Build Notifiers emailtext This plugin allows administrators to create global templates for the Extended Email Publisher. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	11 mo ago

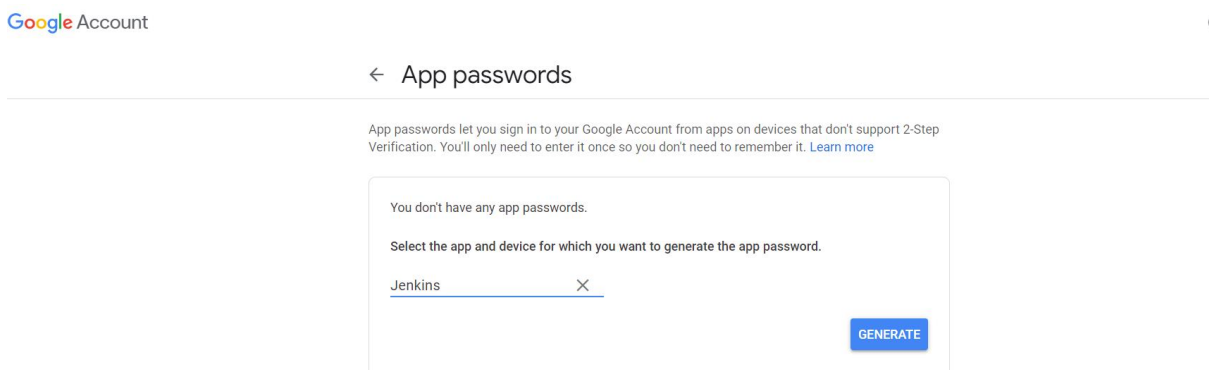
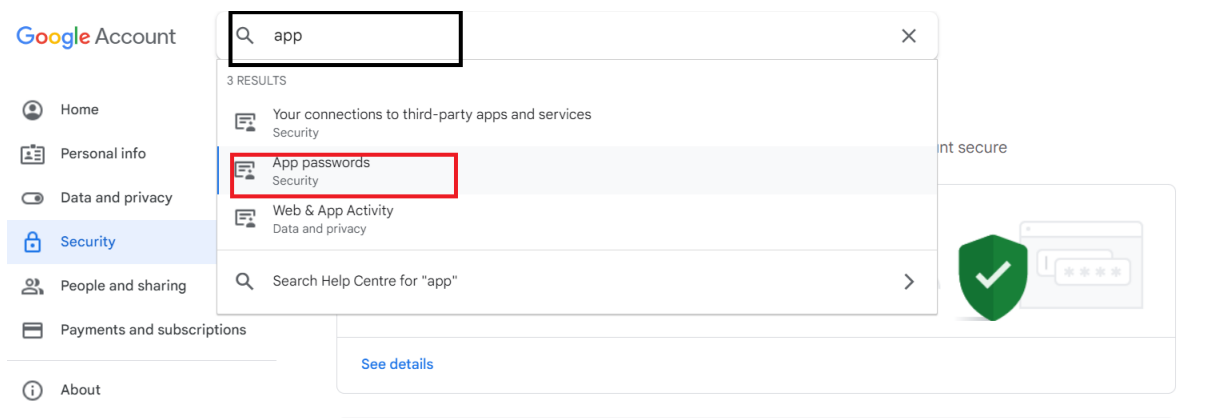
Go to your Gmail and click on your profile

Then click on Manage Your Google Account → click on the security tab on the left side panel you will get this page



2-step verification should be enabled.

Search for the app in the search bar you will get app passwords like the below image



Click on other and provide your name and click on Generate and copy the password

← App passwords

Generated app password

Your app password for your device

bkec mhur oddp hppw

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

[DONE](#)

Once the plugin is installed in Jenkins, click on manage Jenkins → configure system there under the E-mail Notification section configure the details as shown in the below image

Dashboard > Manage Jenkins > System >

E-mail Notification

SMTP server
smtp.gmail.com

Default user e-mail suffix ?

Advanced ^ Edited

☒ Use SMTP Authentication ?

User Name
postbox.aj99@gmail.com

Password
.....

☒ Use SSL ?
☐ Use TLS

SMTP Port ?
465

Reply-To Address

Charset
UTF-8

☐ Test configuration by sending test e-mail

Dependency-Track

Save Apply

Click on Apply and save.

Click on Manage Jenkins→ credentials and add your mail username and generated password

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

Username with password

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

Username ?
postbox.aj99@gmail.com

☐ Treat username as secret ?

Password ?

ID ?
mail

Description ?
mail

Create

This is to just verify the mail configuration

Now under the Extended E-mail Notification section configure the details as shown in the below images

Dashboard > Manage Jenkins > System >

SMTP server

smtp.gmail.com

SMTP Port

465

Advanced ^ Edited

Credentials

postbox.aj99@gmail.com/***** (mail) ▼

Add ▼

☒ Use SSL

☐ Use TLS

☐ Use OAuth 2.0

Dashboard > Manage Jenkins > System >

Advanced Email Properties

Default user e-mail suffix ?

Advanced ▼ Edited

Default Content Type ?

HTML (text/html) ▼

List ID ?

☐ Add 'Precedence: bulk' E-mail Header ?

Dashboard > Manage Jenkins > System >

Additional groovy classpath ?

Add

☐ Enable Debug Mode ?

☐ Require Administrator for Template Testing ?

☐ Enable watching for jobs ?

☐ Allow sending to unregistered users ?

Default Triggers ^

Default Triggers ?

☐ Aborted

☒ Always

☐ Before Build

☐ Failure - 1st

☐ Failure - 2nd

☒ Failure - Any

☐ Failure - Still

☐ Failure - X

☐ Failure -> Unstable (Test Failures)

☐ Fixed

☐ Not Built

Save Apply

Click on Apply and save.

final step to deploy on the Kubernetes cluster and email pipeline.

```
stage('K8s'){
    steps{
```

```

script{
    withKubeConfig(caCertificate: "", clusterName: "", contextName: "", credentialsId: 'k8s',
namespace: "", restrictKubeConfigAccess: false, serverUrl: "") {
        sh 'kubectl apply -f deployment.yaml'
    }
}
}
}
}

```

#post block after stages

```

post {
    always {
        emailx attachLog: true,
        subject: "${currentBuild.result}",
        body: "Project: ${env.JOB_NAME}<br/>" +
            "Build Number: ${env.BUILD_NUMBER}<br/>" +
            "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',
        attachmentsPattern: 'trivy.txt'
    }
}

```

stage view

Stage View

	Declarative: Tool Install	clean Workspace	checkout scm	maven compile	maven Test	Sonarqube Analysis	quality gate	Build war file	OWASP Dependency Check	Build and push to docker hub	TRIVY	Deploy to container	K8s	Declarative: Post Actions
Average stage times: (Average full run time: ~12min 23s)	125ms	267ms	1s	47s	58s	20s	444ms	1min 22s	1min 19s	14min 45s	29s	978ms	2s	16s
#10 Sep 08 12:41 1 commit	121ms	252ms	1s	47s	1min 3s	18s	412ms (skipped for 3s)	1min 5s	15s	14min 32s	24s	1s	1s	16s

In the Kubernetes cluster give this command

kubectl get all

kubectl get svc

```

ubuntu@ip-172-31-40-131:~$ kubectl get all
NAME                                READY    STATUS    RESTARTS   AGE
pod/petshop-768578655f-kzcd9       1/1      Running   0           43s

NAME                                TYPE                      CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/kubernetes                 ClusterIP             10.96.0.1       <none>            443/TCP          58m
service/petshop                    LoadBalancer         10.104.122.152  <pending>        80:30699/TCP     21m

NAME                                READY    UP-TO-DATE   AVAILABLE   AGE
deployment.apps/petshop            1/1      1             1           43s

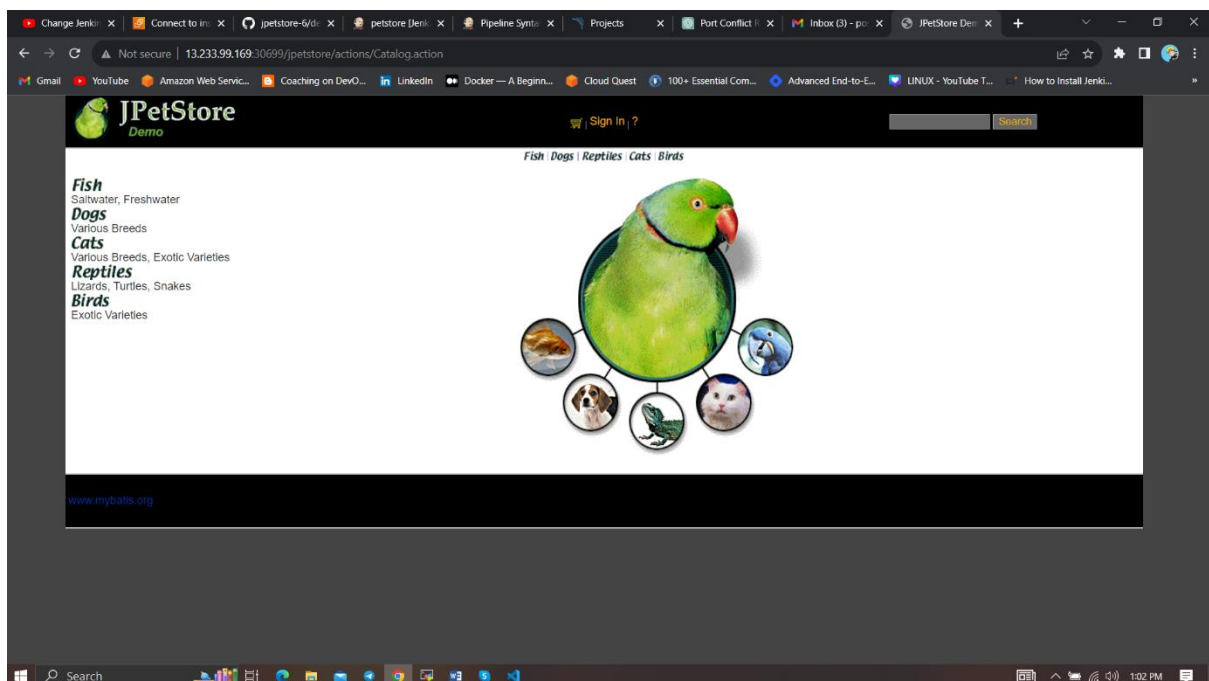
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/petshop-768578655f 1         1         1       43s
ubuntu@ip-172-31-40-131:~$

```

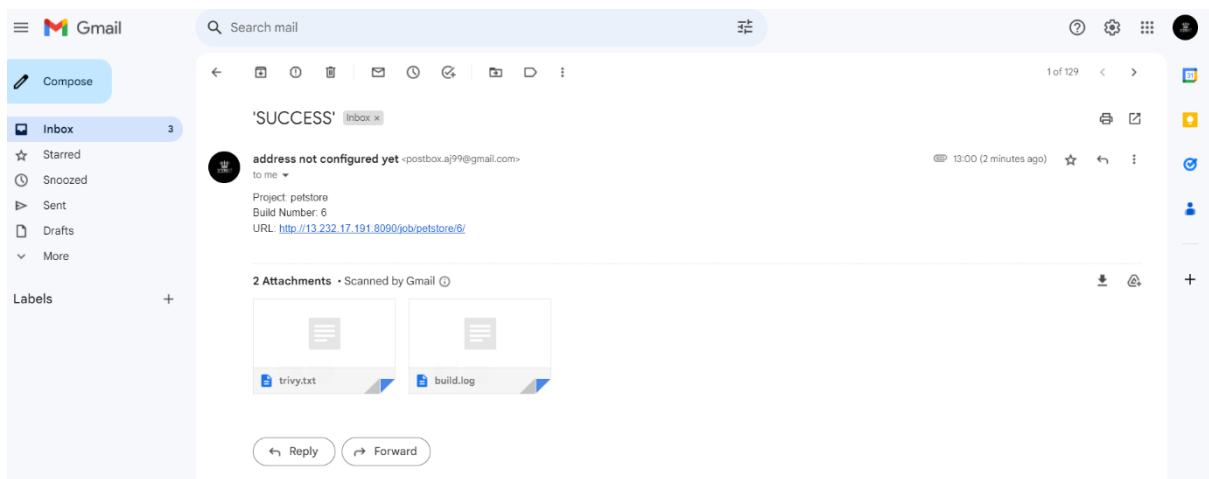
STEP9: Access from a Web browser with

<public-ip-of-slave:30699>

output:



Mail



Step 10: Terminate instances.

Complete Pipeline

```
pipeline{
    agent any

    tools {
        jdk 'jdk17'
        maven 'maven3'
    }

    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

    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'
            }
        }
    }
}
```

```

}
stage("Sonarqube 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'
        }
    }
}
stage ('Build war file'){
    steps{
        sh 'mvn clean install -DskipTests=true'
    }
}
stage("OWASP Dependency Check"){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-
Check'
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
    }
}
stage ('Build and push to docker hub'){
    steps{

```



```

    script{
        withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
            sh "docker build -t petshop ."
            sh "docker tag petshop sevenajay/petshop:latest"
            sh "docker push sevenajay/petshop:latest"
        }
    }
}

stage("TRIVY"){
    steps{
        sh "trivy image sevenajay/petshop:latest > trivy.txt"
    }
}

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

stage('K8s'){
    steps{
        script{
            withKubeConfig(caCertificate: "", clusterName: "", contextName: "", credentialsId: 'k8s',
namespace: "", restrictKubeConfigAccess: false, serverUrl: "") {
                sh 'kubectl apply -f deployment.yaml'
            }
        }
    }
}

post {

```

```

always {
    emailx attachLog: true,
    subject: "${currentBuild.result}",
    body: "Project: ${env.JOB_NAME}<br/>" +
        "Build Number: ${env.BUILD_NUMBER}<br/>" +
        "URL: ${env.BUILD_URL}<br/>",
    to: 'postbox.aj99@gmail.com',
    attachmentsPattern: 'trivy.txt'
}
}
}

```

Trigger code

CI-petshop-pipeline

```

pipeline{
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    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'
    }
}
stage("Sonarqube 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'
        }
    }
}
stage ('Build war file'){
    steps{
        sh 'mvn clean install -DskipTests=true'
    }
}

```

```

    }
}
stage("OWASP Dependency Check"){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-
Check'
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
    }
}
stage ('Build and push to docker hub'){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
                sh "docker build -t petshop ."
                sh "docker tag petshop sevenajay/petshop:latest"
                sh "docker push sevenajay/petshop:latest"
            }
        }
    }
}
stage("TRIVY"){
    steps{
        sh "trivy image sevenajay/petshop:latest > trivy.txt"
    }
}
stage("Trigger deployment"){
    steps{
        // Trigger the deployment pipeline and wait for it to complete
        build job: 'CD-petshop', wait: true
    }
}
}

```

```

}
post {
  always {
    emailx attachLog: true,
    subject: "${currentBuild.result}",
    body: "Project: ${env.JOB_NAME}<br/>" +
      "Build Number: ${env.BUILD_NUMBER}<br/>" +
      "URL: ${env.BUILD_URL}<br/>",
    to: 'postbox.aj99@gmail.com',
    attachmentsPattern: 'trivy.txt'
  }
}
}

```

CD-petshop-pipeline

```

pipeline{
  agent any
  stages{
    stage ('clean Workspace'){
      steps{
        cleanWs()
      }
    }
    stage ('checkout scm') {
      steps {
        git 'https://github.com/Aj7Ay/jpetstore-6.git'
      }
    }
    stage ('Deploy to container'){
      steps{
        sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop:latest'
      }
    }
  }
}

```

```

    }

    stage('K8s'){
        steps{
            script{
                withKubeConfig(caCertificate: "", clusterName: "", contextName: "", credentialsId: 'k8s',
namespace: "", restrictKubeConfigAccess: false, serverUrl: "") {
                    sh 'kubectl apply -f deployment.yaml'
                }
            }
        }
    }
}

post {
    always {
        emailx attachLog: true,
        subject: "${currentBuild.result}",
        body: "Project: ${env.JOB_NAME}<br/>" +
            "Build Number: ${env.BUILD_NUMBER}<br/>" +
            "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',
        attachmentsPattern: 'trivy.txt'
    }
}
}

```