

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



Ajay Kumar Yegireddi

Sep 10, 2023 · 12 min read

TABLE OF CONTENTS

STEP1:Create an Ubuntu(22.04) T2 Large Instance

Step 2 — Install Jenkins,



2A — To Install Jenkins

2B — Install Docker

2C — Install Trivy

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

3A — Install Plugin

Show more ▾

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/Aj7Ay/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

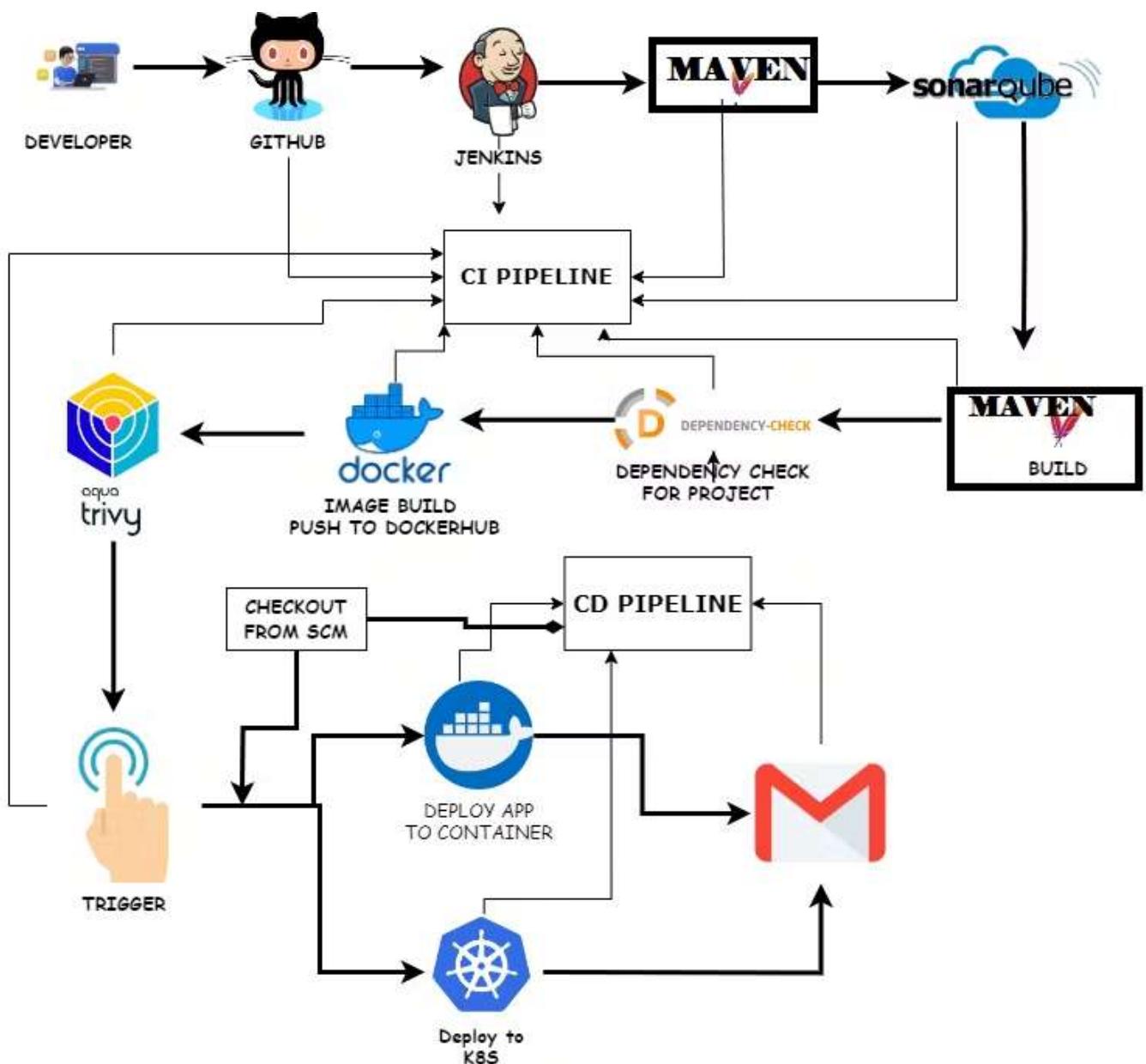
Step 7 — Deploy the image using Docker

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 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									
C Connect Instance state Actions Launch instances									
<input type="checkbox"/> Find instance by attribute or tag (case-sensitive)									
	Name	▲	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
<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
```

COPY

```
#!/bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/trusted.gpg.d/adoptium.asc
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/api/debian stable main" | sudo 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 /etc/apt/trusted.gpg.d/jenkins.asc
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

COPY

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \n\nhttp://jenkins. | tee binary/ | sudo tee \n/etc/apt/sources.list.d/jenkins.list > /dev/n\nsudo apt-get update -y\nsudo apt-get install jenkins -y\nsudo systemctl start jenkins\nsudo systemctl status jenkins
```

COPY

```
sudo chmod 777 jenkins.sh\n./jenkins.sh # this will installl 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 command.

COPY

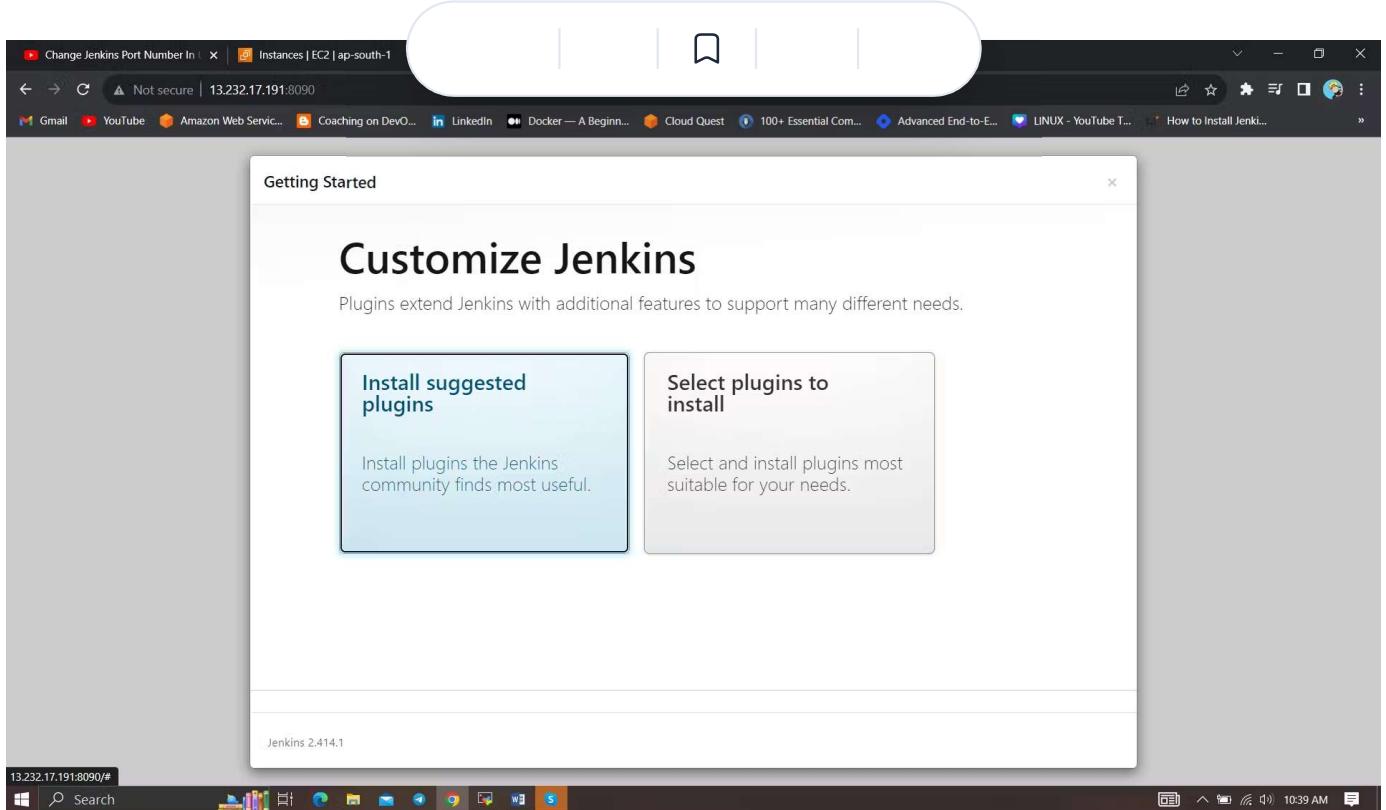
```
sudo systemctl stop jenkins\nsudo systemctl status jenkins\ncd /etc/default\nsudo vi jenkins #chnage port HTTP_PORT=8090 and save and exit\ncd /lib/systemd/system\nsudo vi jenkins.service #change Environments="Jenkins_port=8090" save and\ncsudo systemctl daemon-reload\ncsudo systemctl restart jenkins\ncsudo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

Now, grab your Public IP Address

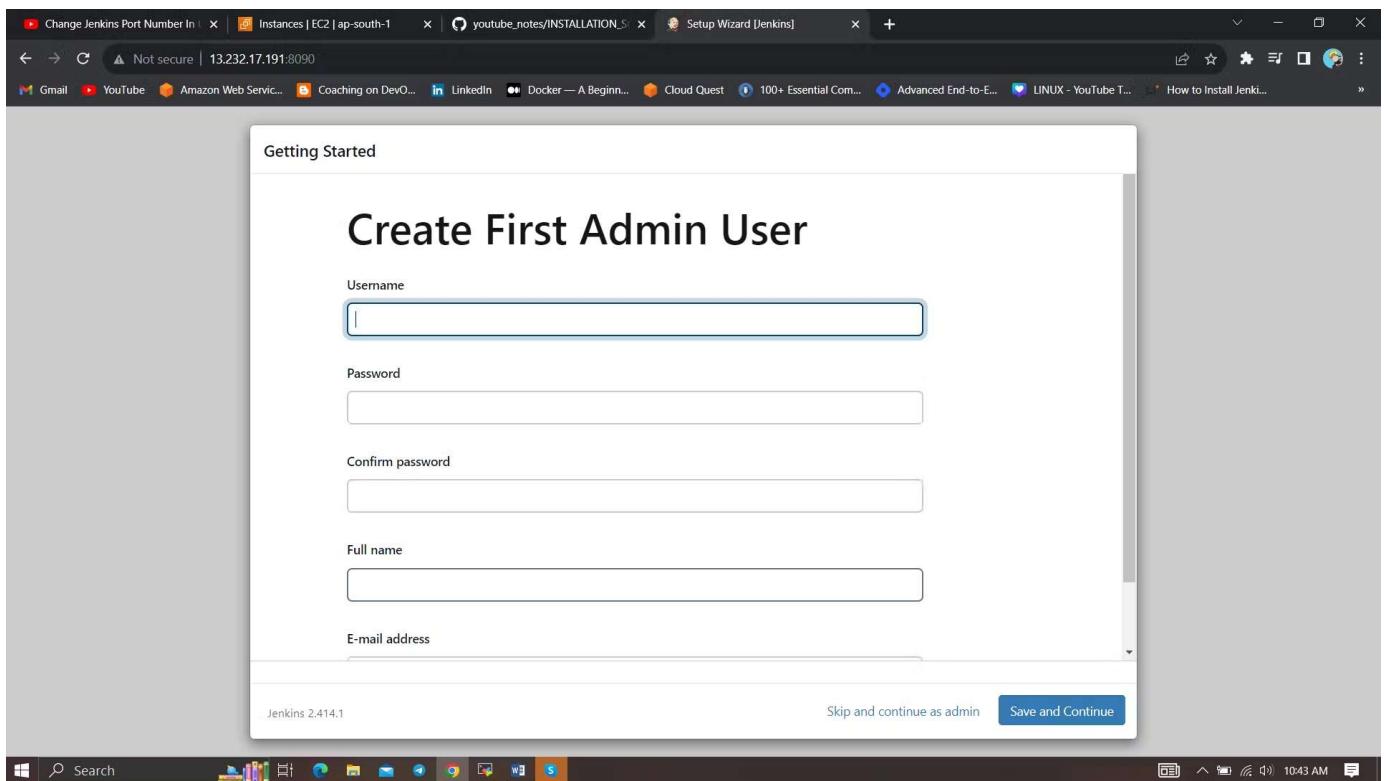
```
<EC2 Public IP Address:8090>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

The screenshot shows a web browser window with the URL `13.232.17.191:8090/login?from=%2F`. The page title is "Getting Started" and the main heading is "Unlock Jenkins". A note states: "To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server: `/var/lib/jenkins/secrets/initialAdminPassword`". Below this, a placeholder text area is labeled "Administrator password". At the bottom right is a blue "Continue" button.

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.

Dashboard >

+ New Item Add description

People Welcome to Jenkins!

Build History This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Manage Jenkins Start building your software project

Build Queue Create a job

No builds in the queue.

Build Executor Status Set up a distributed build

1 Idle Set up an agent

2 Idle Configure a cloud

Learn more about distributed builds

2B – Install Docker

COPY

```
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
```

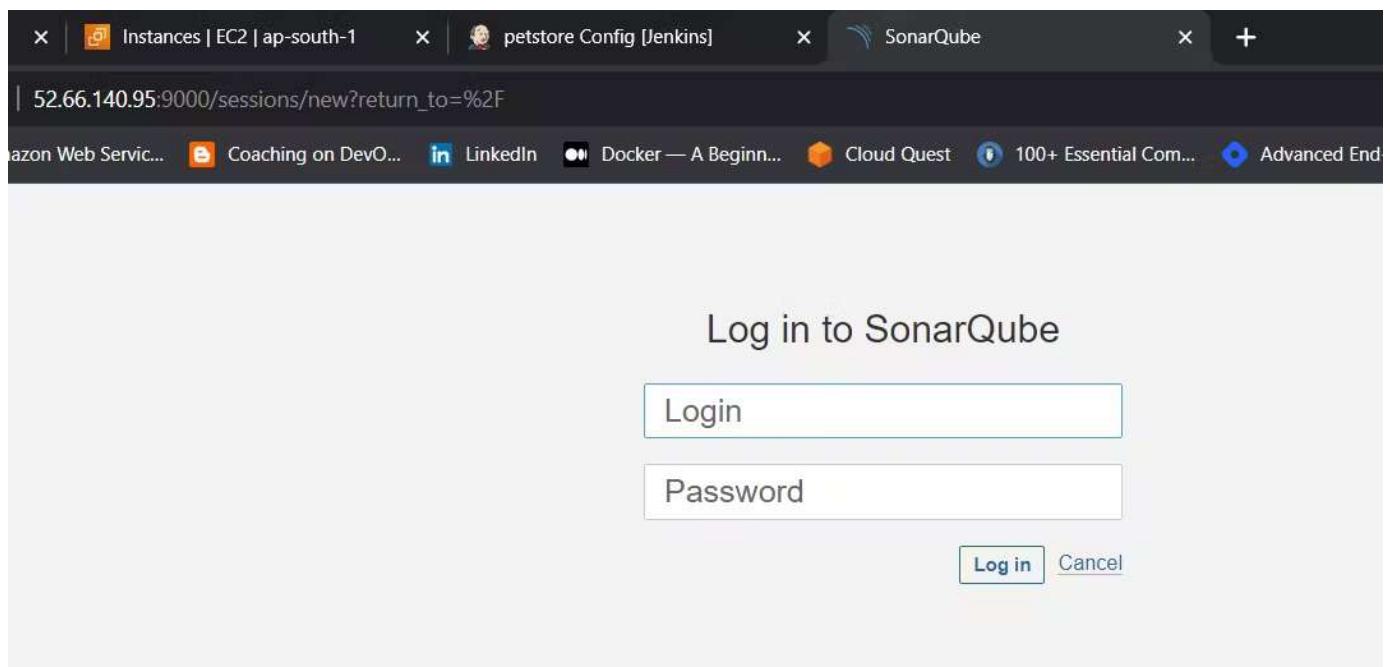
After the docker installation, we create a sonarqube container (Remember added 9000 ports in the security group).

COPY

```
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 sonarqube
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a18f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436aff7f752fdb04993092d0ca3065e2f2e32301b50139
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
```

COPY

Instances | EC2 | ap-south-1

6.140.95:9000/account/reset_password

Web Servic... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E...

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

Update New password, This is Sonar Dashboard.

Not secure | 52.66.140.95:9000/projects/create

Gmail YouTube Amazon Web Servic... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E... LINUX - YouTube T... How to Install Jenki...

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects... A

How do you want to create your project?

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. First, you need to set up a DevOps platform configuration.

From Azure DevOps **From Bitbucket Server** **From Bitbucket Cloud** **From GitHub** **From GitLab**

Set up global configuration Set up global configuration Set up global configuration Set up global configuration Set up global configuration

2C – Install Trivy

```
vi trivy.sh
```

COPY 

```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y  
  
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg -  
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.gi  
  
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)

Plugins

Available plugins

Install	Name	Released
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net	11 mo ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.15 <small>External Site/Tool Integrations Build Reports</small> This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	9 mo 19 days ago

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

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.

Dashboard > All >

Enter an item name

petstore » Required field

Pipeline 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.

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.

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 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

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

The screenshot shows the SonarQube Administration interface. At the top, there's a navigation bar with links for Configuration, Security, Projects, System, and Marketplace. Below this, a dropdown menu is open under the 'General' section, with 'Users' highlighted by a red box. Other options in the dropdown include Groups, Global Permissions, and Permission Templates. A search bar labeled 'Find' is also visible.

click on update Token

The screenshot shows the SonarQube Tokens page. At the top, there are tabs for SCM Accounts, Last connection, Groups, and Tokens, with 'Tokens' highlighted by a red box. Below this, a table lists tokens for the user 'Administrator'. The table includes columns for Name, Type, Project, Last use, Created, and Expiration. A 'Tokens' icon is shown next to the user name. At the bottom right of the table, there's a 'Update Tokens' button highlighted with a red box.

Create a token with a name and generate

The screenshot shows the 'Tokens of Administrator' page. At the top, there's a 'Generate Tokens' section with fields for 'Name' (containing 'Enter Token Name') and 'Expires in' (set to '30 days'). Below this, a message says 'New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!' with a 'Copy' button and the token value 'squ_21d162904c1c72cf8b39665f96480185c99dc2f9'. A table below lists the token details:

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

copy Token

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

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 see this page once you click on create

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

ID	Name	Kind	Description	Action
Sonar-token	sonar	Secret text	sonar	

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

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	
<input type="button" value="Add ▾"/>		
<input type="button" value="Save"/> <input type="button" value="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 Jenkins 'Manage Jenkins' interface under the 'Tools' section. It's titled 'SonarQube Scanner installations'. A sub-section titled 'SonarQube Scanner' is active. The configuration form includes:

- Name:** sonar-scanner
- Install automatically:** checked
- Install from Maven Central:** sub-section with:
 - Version:** SonarQube Scanner 5.0.1.3006
 - Add Installer:** button

At the bottom are 'Save' and 'Apply' buttons.

In the Sonarqube Dashboard add a quality gate also

Administration → Configuration → Webhooks

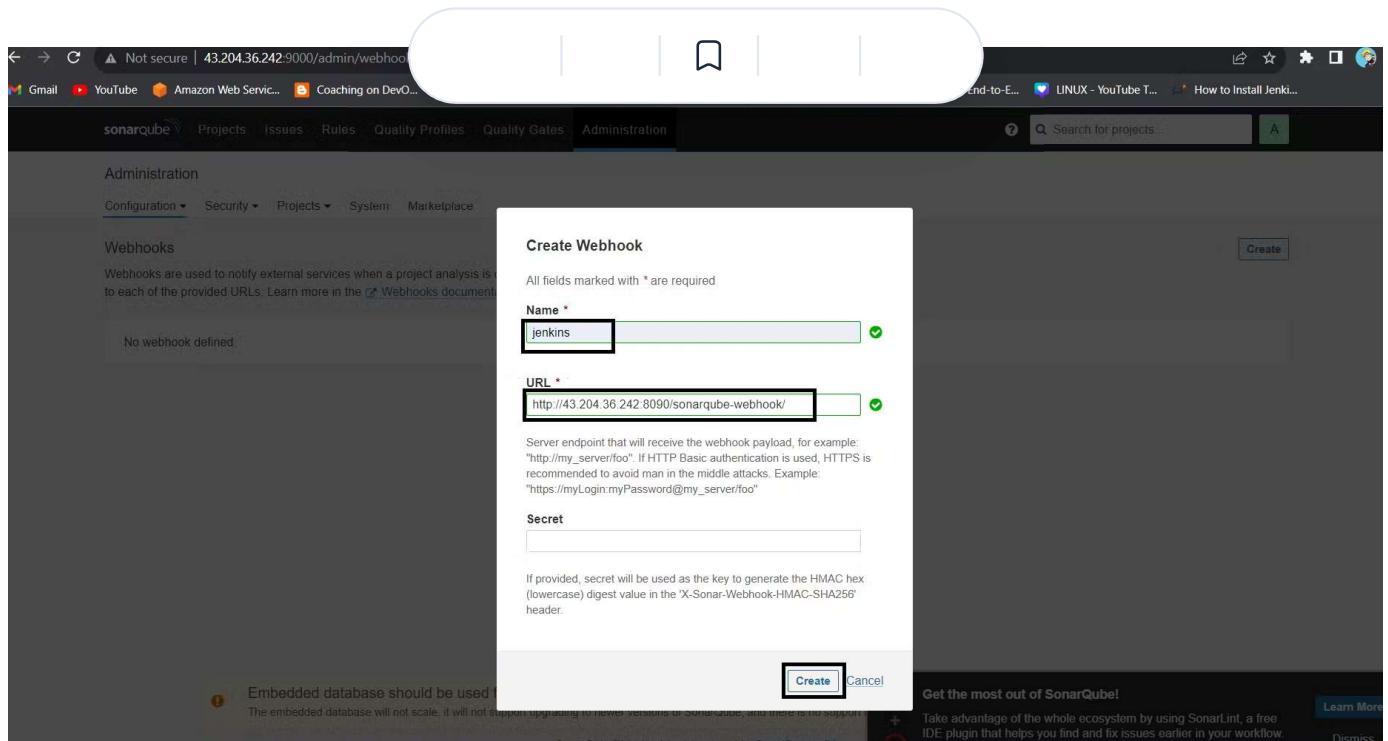
The screenshot shows the SonarQube Administration interface. The top navigation bar has tabs for Configuration, Security, Projects, System, and Marketplace, with 'Administration' selected. Below this, there are sections for General Settings, Encryption, and Webhooks. The 'Webhooks' section is highlighted with a red box. A search bar labeled 'Search by login or name...' is present. On the right, there's a 'Create User' button. The main content area displays user information: 'Administrator admin' (with a green 'A' icon), last connection time ('< 1 hour ago'), groups ('sonar-administrators', 'sonar-users'), and tokens ('1'). A note at the bottom says '1 of 1 shown'.

Click on Create

The screenshot shows the SonarQube Webhooks creation page. The top navigation bar has tabs for Configuration, Security, Projects, System, and Marketplace, with 'Administration' selected. Below this, there's a 'Webhooks' section. A note says '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](#)'. A 'Create' button is highlighted with a black box. Below it, a message says 'No webhook defined.'

Add details

The screenshot shows a code block containing a webhook URL. The URL is: #in url section of quality gate <http://jenkins-public-ip:8090>/sonarqube-webhook/. A 'COPY' button with a clipboard icon is highlighted with a black box.

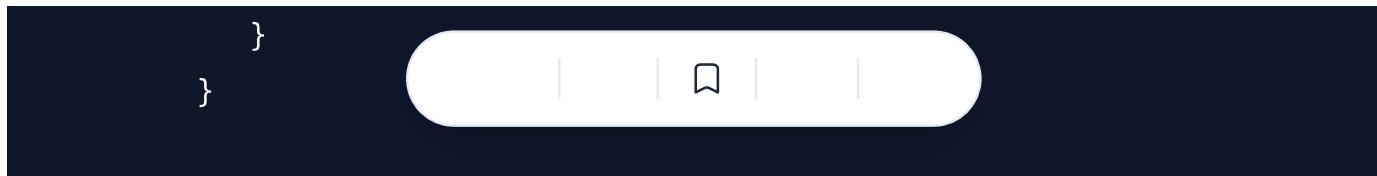


Let's go to our Pipeline and add Sonarqube Stage in our Pipeline Script.

```
#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'''
        }
    }
}

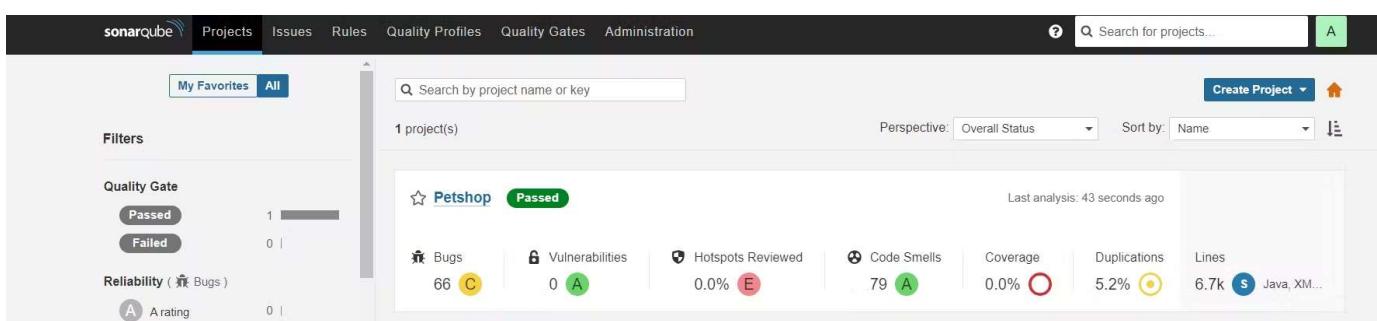
stage("quality gate"){
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: ''
        }
    }
}
```



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.

The screenshot shows the Jenkins 'Plugins' page. On the left, there are navigation links: 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. A search bar at the top right contains the placeholder 'Search available plugins'. Below the search bar is a button labeled 'Install'. The main area displays a list of available plugins. One plugin is highlighted: 'OWASP Dependency-Check 5.4.2'. The details for this plugin include its category ('Security'), release date ('8 days 17 hr ago'), and a brief description: 'This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.'.

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

Goto Dashboard → Manage Jenkins → Tools →

The screenshot shows the Jenkins 'Tools' configuration page. At the top, it says 'Dependency-Check installations'. Below that is a 'Add Dependency-Check' button. The main section is titled 'Dependency-Check' and contains a 'Name' field with the value 'DP-Check'. There is also a checked checkbox for 'Install automatically'. A dashed box highlights the 'Install from github.com' section, which includes a 'Version' field with the value 'dependency-check 6.5.1' and a 'Add Installer' button.

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'
        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

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

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

The screenshot shows the Jenkins Plugins page. The URL is [Dashboard > Manage Jenkins > Plugins](#). The search bar contains "docker". The "Installed plugins" section is visible. The "Docker" plugin is listed with the version 1.5, released 3 days 15 hr ago. It has a checked checkbox and the status "Installed". Below it, the "Docker Commons" and "Docker Pipeline" plugins are also listed with their respective versions and release dates. The "Docker API" plugin is listed at the bottom.

Plugin	Version	Released
Docker	1.5	3 days 15 hr ago
Docker Commons	439.va_3cb_0a_6a_fb_29	1 mo 29 days ago
Docker Pipeline	572.v950f58993843	27 days ago
Docker API	3.3.1-79.v20b_53427e041	3 mo 4 days ago

Now, goto Dashboard → Manage Jenkins → Tools →

Add Docker

Docker

Name: docker

Install automatically

Download from docker.com

Docker version: latest

Add Installer

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

```
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"
        }
    }
}
```

COPY

```

        sh "docker push sevenajay/petshop:latest"
    }
}

}

stage("TRIVY"){

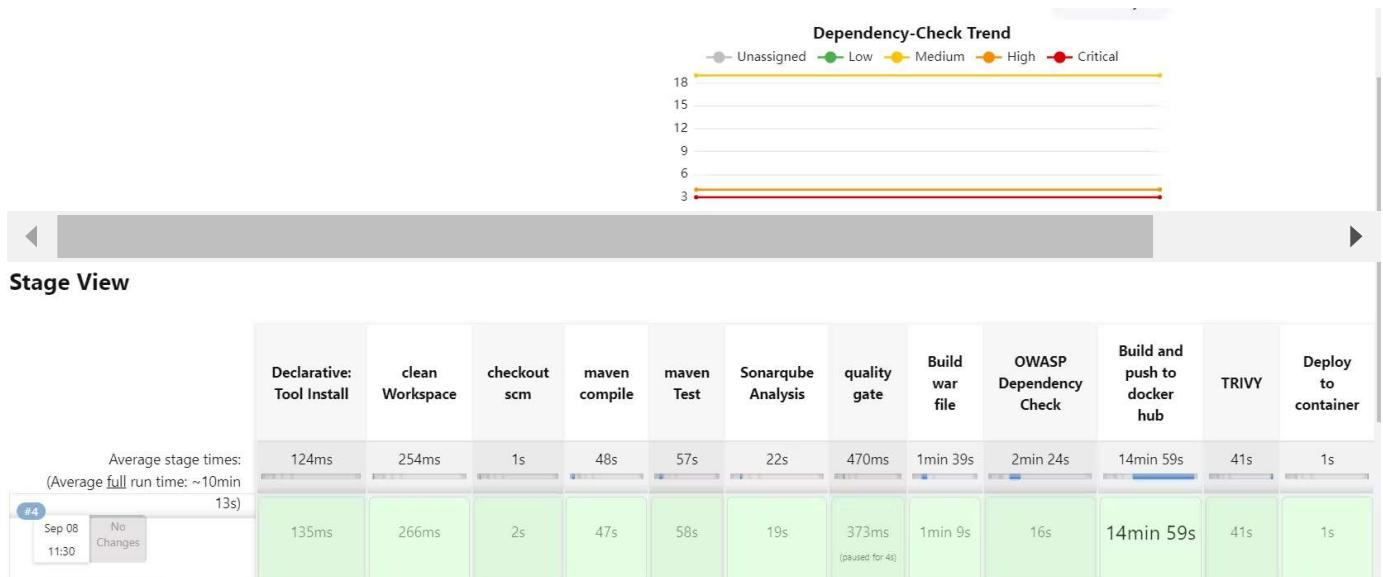
    steps{
        sh "trivy image sevenajay/petshop:latest > trivy.txt"
    }
}

stage ('Deploy to container'){

    steps{
        sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop'
    }
}

```

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

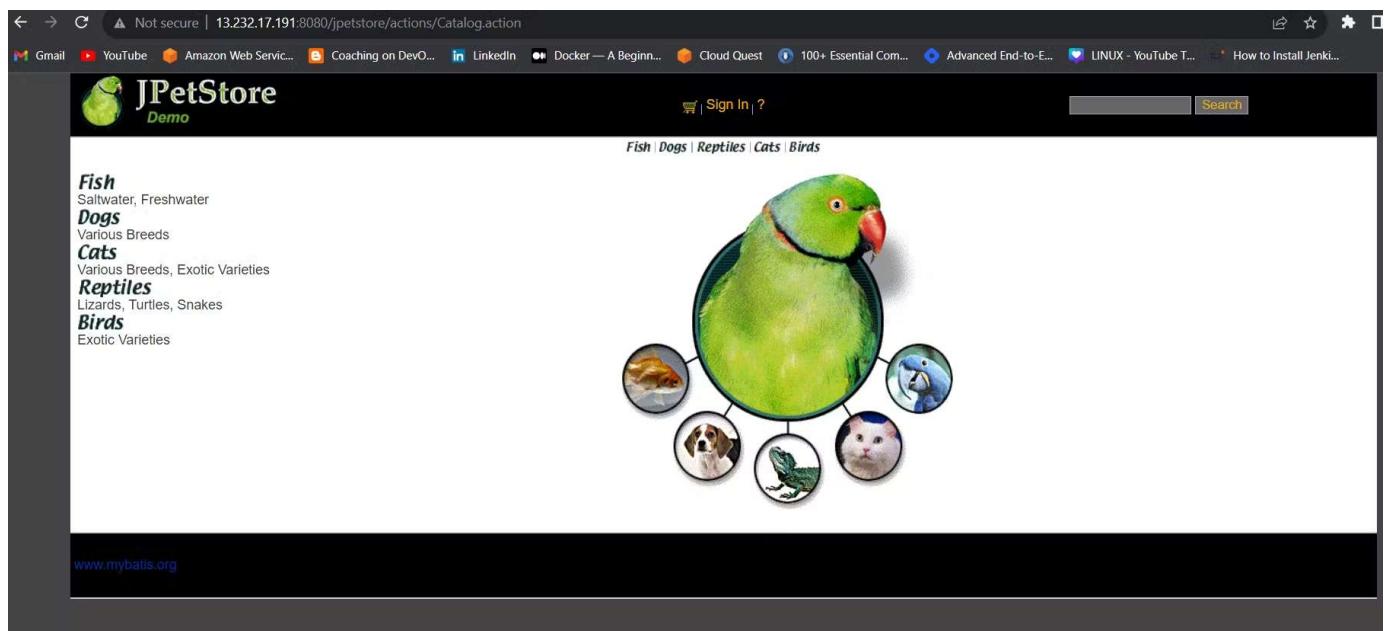
Description
This repository does not have a description

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 – Kuberentes 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-version)
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

Part 1 -----Master Node-----

```
COPY ▾
sudo su
hostname master
bash
clear
```

-----Worker Node-----

```
COPY ▾
sudo su
hostname worker
bash
clear
```

Part 2 -----Both Master & Node -----

```
COPY ▾
sudo apt-get update

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

```
sudo usermod -aG docker $USER
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

sudo apt-get install -y kubelet kubeadm kubectl

sudo snap install kube-apiserver
```

Part 3 ----- Master -----

COPY 

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
# in case you're in root exit from it and run below commands
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Do
```

----- Worker Node -----

```
sudo kubeadm join <master-ip> --token <token> --dis
```

Copy the config file to Jenkins master or the local file manager and save it

copy it and save it in documents or another folder save it as secret-file.txt

Install Kubernetes Plugin, Once it's installed successfully

Dashboard > Manage Jenkins > Plugins

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Kubernetes Client API 6.8.1-224.vd388fca_4db_3b_ Released 9 days 16 hr ago

Kubernetes Credentials 0.11 Released 9 days 16 hr ago

Kubernetes Client API plugin for use by other Jenkins plugins.

Kubernetes 4029.v5712230ccb_f8 Released 9 days 15 hr ago

Cloud Providers Cluster Management Kubernetes Agent Management

This plugin integrates Jenkins with Kubernetes

Kubernetes CLI 1.12.1 Released 8 days 22 hr ago

kubernetes

Configure kubectl for Kubernetes

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



New credentials

Kind

Secret file

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

File

Choose File Secret File.txt

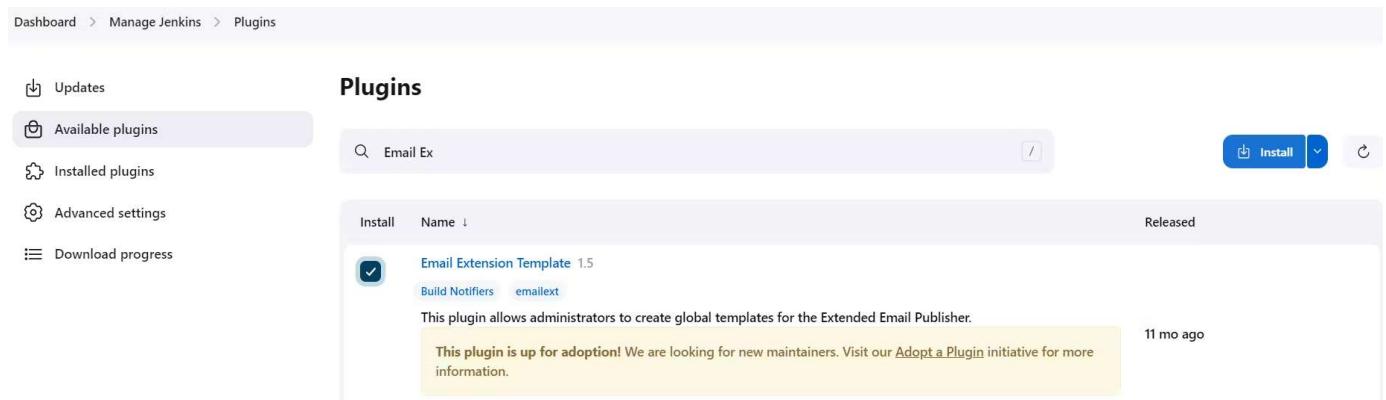
ID ? k8s

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

Search: Email Ex

Install Name Released

Email Extension Template 1.5 Build Notifiers emailext 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

The screenshot shows the Google Account security page. On the left, a sidebar lists options: Home, Personal info, Data and privacy, Security (which is selected and highlighted in blue), People and sharing, Payments and subscriptions, and About. The main content area has a header "Recent security activity". It lists three events: "App password removed" (1 Sept · Telangana, India), "App password created" (31 Aug · Telangana, India), and another "App password removed" (31 Aug · Telangana, India). Below this is a section titled "How you sign in to Google" with a sub-section "2-Step Verification" which is turned "On since 8 Oct 2022".

2-step verification should be enabled.

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

The screenshot shows the Google Account search results for "app". The search bar contains "app" and there are 3 results. The results are: "Your connections to third-party apps and services" (Security), "App passwords" (Security, highlighted with a red box), and "Web & App Activity" (Data and privacy). The sidebar on the left is identical to the one in the previous screenshot.

[Google Account](#)

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device for which you want to generate the app password.

Jenkins

X

[GENERATE](#)

Click on other and provide the password

The screenshot shows the 'Generated app password' screen in the Google Account settings. At the top, there's a back arrow and the title 'App passwords'. Below that, it says 'Generated app password'. A yellow box contains the generated password 'bkec mhur oddp hppw', which is highlighted with a red border. To the left, there are fields for 'Email' (set to 'securesally@gmail.com') and 'Password' (showing a masked password). To the right, instructions say 'Your app password for your device' and 'How to use it'. The 'How to use it' section contains text about replacing the normal password with this 16-character one for app access. At the bottom right is a blue 'DONE' button.

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

The screenshot shows the Jenkins System Configuration page under the 'E-mail Notification' section. The 'SMTP server' field contains 'smtp.gmail.com'. The 'Default user e-mail suffix' field is empty. Below these fields, there is an 'Advanced' dropdown set to 'Edited'. Under 'Advanced', the 'Use SMTP Authentication' checkbox is checked, and the 'User Name' field contains 'postbox.aj99@gmail.com'. The 'Password' field is filled with a series of dots. Under 'Advanced' settings, the 'Use SSL' checkbox is checked, while 'Use TLS' is unchecked. The 'SMTP Port' field is set to '465'. Other fields shown include 'Reply-To Address' (empty), 'Charset' (set to 'UTF-8'), and a 'Test configuration by sending test e-mail' checkbox (unchecked). At the bottom, there are 'Save' and 'Apply' buttons.

Click on Apply and save.

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

Dashboard > Manage Jenkins > Credentials > System >

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 > Extended E-mail Notification

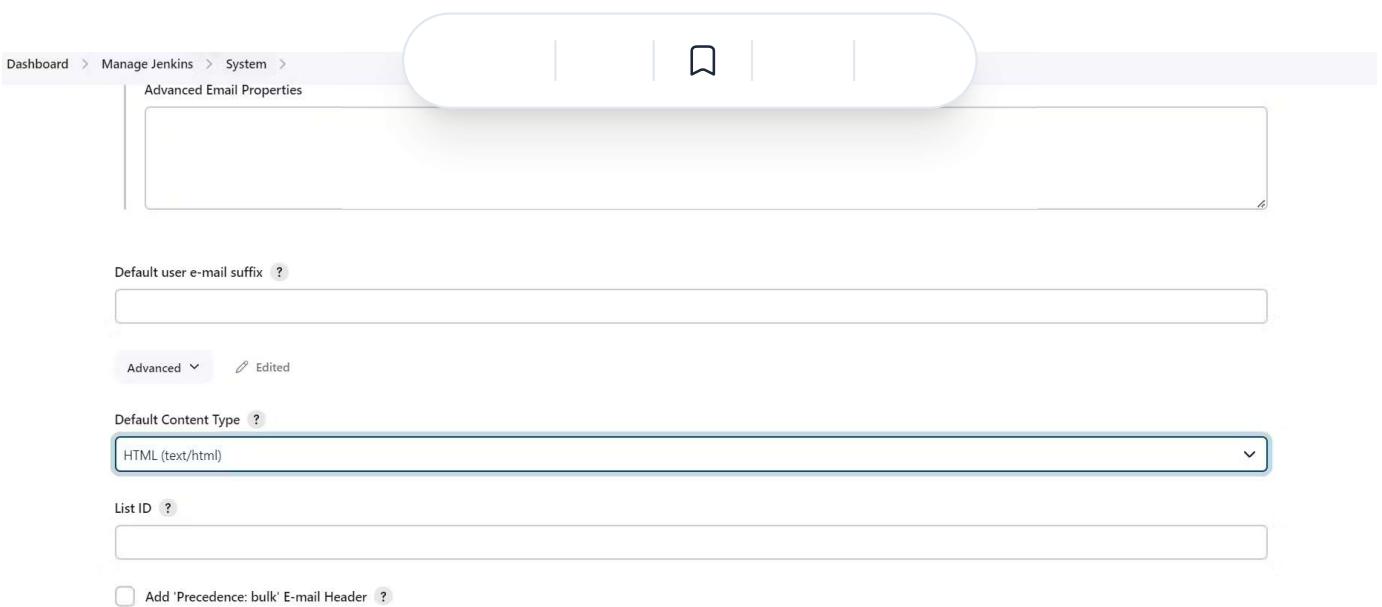
SMTP server
smtp.gmail.com

SMTP Port
465

Advanced ^ Edited

Credentials
postbox.aj99@gmail.com/***** (mail)

Use SSL
 Use TLS
 Use OAuth 2.0



The screenshot shows the 'Advanced Email Properties' configuration page. It includes fields for 'Default user e-mail suffix', 'Default Content Type' (set to 'HTML (text/html)'), 'List ID', and an unchecked checkbox for 'Add 'Precedence: bulk' E-mail Header'. Below this is a section for 'Additional groovy classpath' with an 'Add' button and a list of checkboxes for various Jenkins features.

Advanced Email Properties

Default user e-mail suffix ?

Default Content Type ?

HTML (text/html)

List ID ?

Add 'Precedence: bulk' E-mail Header ?

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: '')
            sh 'kubectl apply -f deployment.yaml'
        }
    }
}
```

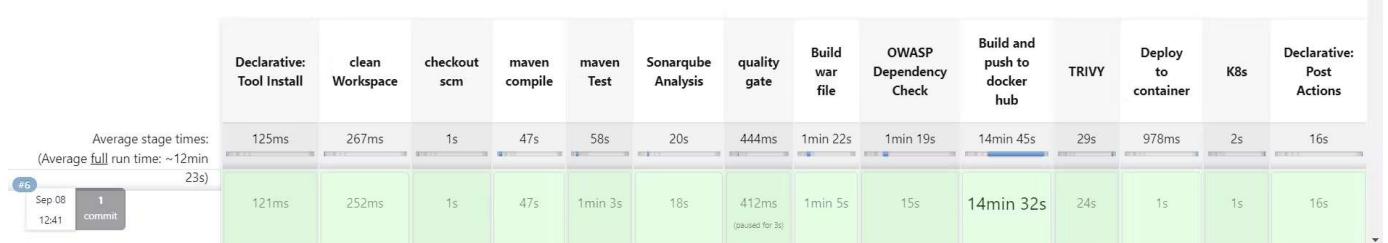
```
}
```

```
COPY 🗂️

#post block after stages
post {
    always {
        emailext 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



In the Kubernetes cluster give this command

```
COPY 🗂️

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

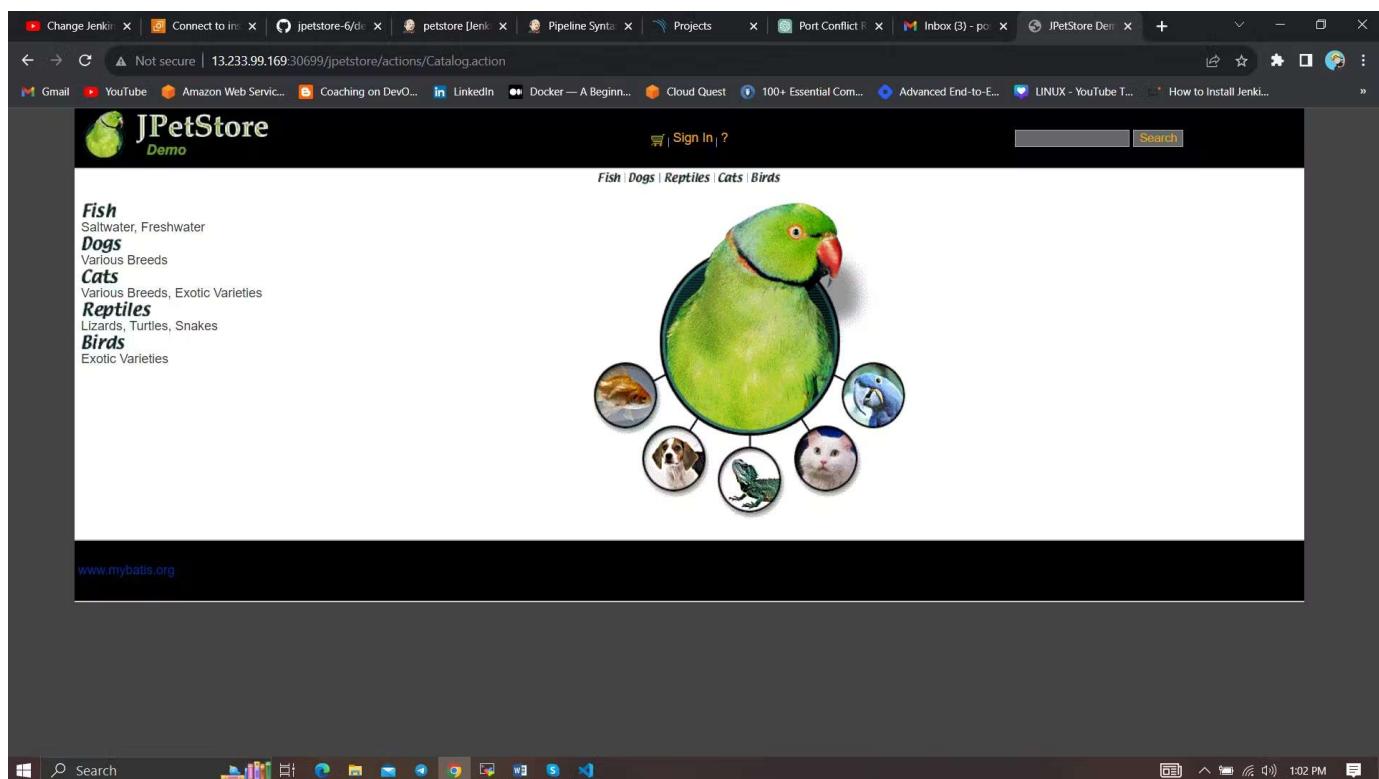
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/petshop  1/1      1           1          43s

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

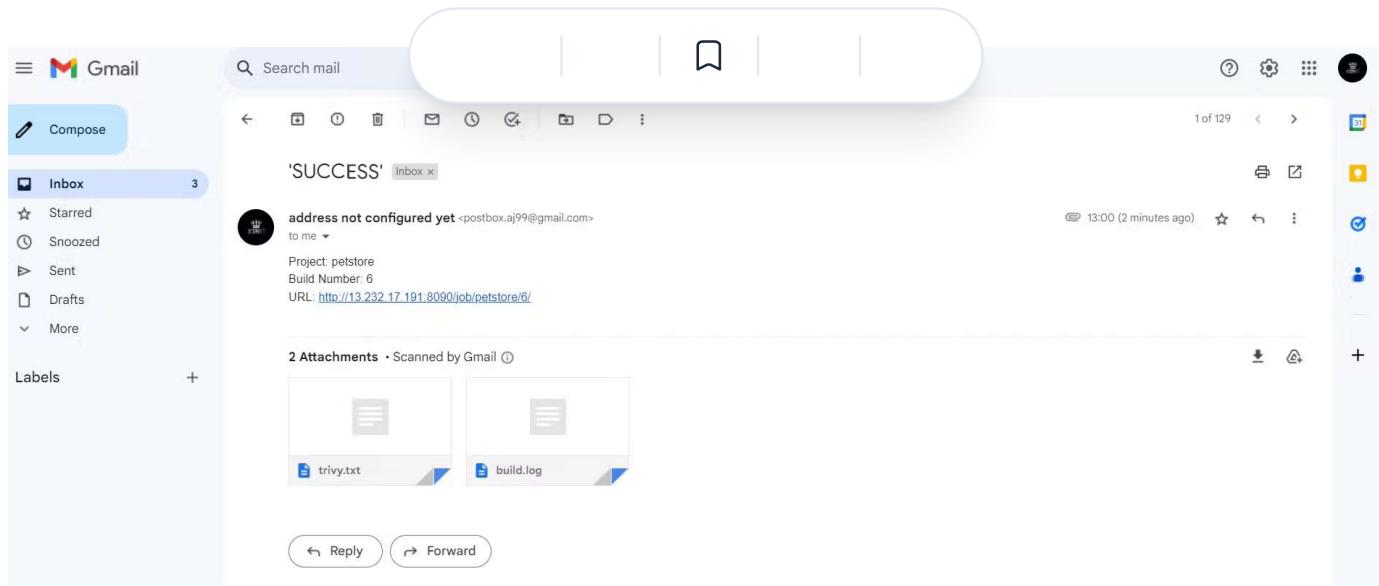
STEP9:Access from a Web browser with

<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'
            }
        }
    }
}
```

COPY

```
        }
    }

    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: ''
            }
        }
    }

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

    stage("OWASP Dependency Check"){
        steps{
            dependencyCheck additionalArguments: '--scan ./ --format XML'
            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'
        }
    }

    stage('K8s'){
        steps{
            script{
                withKubeConfig(caCertificate: '', clusterName: '', contextName: '')
                    sh 'kubectl apply -f deployment.yaml'
                }
            }
        }
    }

    post {
        always {
            emailext attachLog: true,
            subject: "${currentBuild.result}",
        }
    }
}
```

```
body: "Project: ${env.TOB_NAME}  
" +  
    "Build Number: ${env.BUILD_NUMBER}<br/>" +  
    "URL: ${env.BUILD_URL}<br/>","  
to: 'postbox.aj99@gmail.com',  
attachmentsPattern: 'trivy.txt'  
}  
}  
}
```

Trigger code

CI-petshop-pipeline

COPY 

```
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') {
    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: ''
        }
    }
}

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

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

```
}

stage ('Build') {
    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 {
        emailext 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{
    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'
            }
        }
        stage('K8s'){
            steps{
                script{
                    withKubeConfig(caCertificate: '', clusterName: '', contextName: '')
                    sh 'kubectl apply -f deployment.yaml'
                }
            }
        }
    }
    post {
        always {
            emailext attachLog: true,
            subject: "${currentBuild.result}",
            body: "Project: ${env.JOB_NAME}<br/> +"
        }
    }
}
```

```
"Build Number: ${env.BUILD_NUMBER}<br/>" +  
"URL: ${env.BUILD_URL}" +  
to: 'postbox.aj99@gmail.com',  
attachmentsPattern: 'trivy.txt'  
}  
}  
}
```

Subscribe to my newsletter

Read articles from **Ajay Kumar Yegireddi's blog** directly inside your inbox.

Subscribe to the newsletter, and don't miss out.

SUBSCRIBE

Did you find this article valuable?

Support **Ajay Kumar Yegireddi** by becoming a sponsor. Any amount is appreciated!

**Sponsor**

[Learn more about Hashnode Sponsors](#)

[AWS](#)[ci-cd](#)[Jenkins](#)[projects](#)[Cloud](#)

MORE ARTICLES**Ajay Kumar Yegireddi**

Complete Ci/Cd Gradle Project Devsecops

Hello friends, we will be deploying a Gradle Java Based Application. This is an everyday use case sc...

Ajay Kumar Yegireddi

Complete Ci/cd Petclinic Project Devsecops

Hello friends, we will be deploying a Pet Clinic Java Based Application. This is an everyday use cas...

©2023 Ajay Kumar Yegireddi's blog

[Archive](#) • [Privacy policy](#) • [Terms](#)



[Publish with Hashnode](#)

Powered by [Hashnode](#) - Home for tech writers and readers