

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

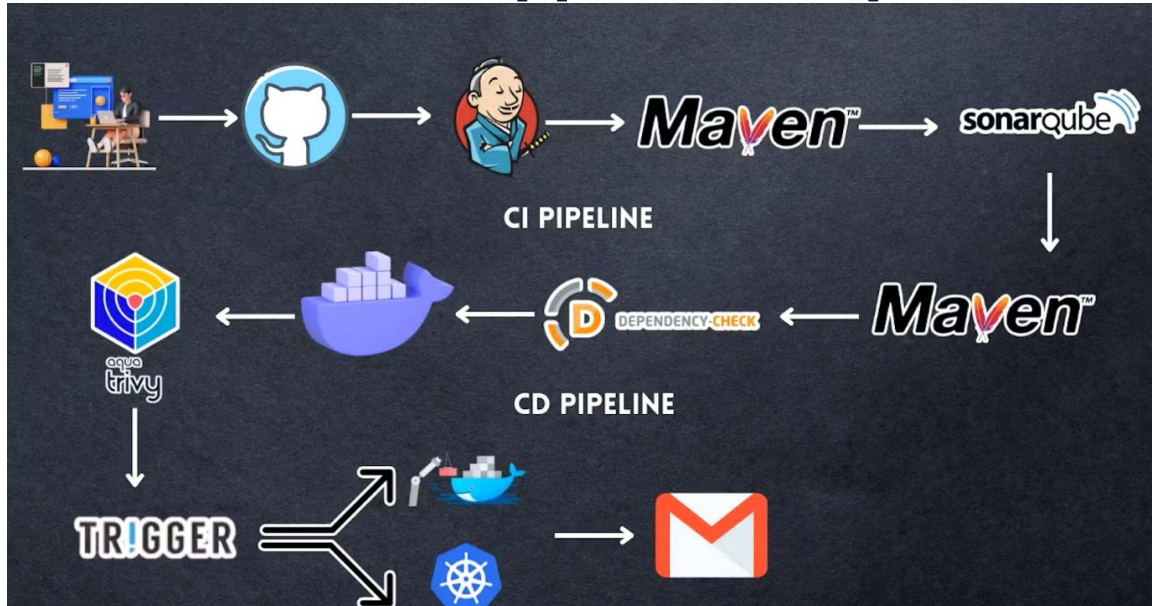


TABLE OF CONTENTS

- [STEP1: HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop) [HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop) Create an [HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop) Ubuntu ([HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop) 22.04) T2 Large Instance
- [Step 2 — Install Jenkins, Docker and Trivy](#)
 - [2A — To Install Jenkins](#)
 - [2B — Install Docker](#)
 - [2C — Install Trivy](#)
- [Step 3 — Install Plugins like JDK, \[HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"\]\(https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop\) SonarQube \[HYPERLINK\]\(#\)](#)

["https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop) Scanner, Maven, OWASP Dependency Check

- [3A — Install Plugin](#)
- [3B — Configure Java and Maven in Global Tool Configuration](#)
- [3C — Create a Job](#)
- [Step 4 — Configure Sonar Server in Manage Jenkins](#)
 - [Step 5 — Install OWASP Dependency Check Plugins](#)
- [Step 6 — Docker Image Build and Push](#)
- [Step 8 — HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"Kubernetes HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop" Setup](#)
 - [Kubectl on Jenkins to be installed](#)
 - [Part 1 -----Master Node-----](#)
 - [-----Worker Node-----](#)
 - [Part 2 -----Both Master HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"& HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop" Node -----](#)
 - [Part 3 ----- Master -----](#)
 - [-----Worker Node-----](#)
- [Configuring mail server in Jenkins \(Gmail \)](#)
- [STEP9:Access from a Web browser with](#)
- [Step 10: Terminate instances.](#)
- [Complete Pipeline](#)
- [Trigger code](#)
- [CI- HYPERLINK "https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"pet shop HYPERLINK](#)

["https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"-pipeline](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop)

- CD- HYPERLINK "<https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop>"pet shop HYPERLINK "[https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop"-pipeline](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop)

Hello friends, we will be deploying a Pet shop Java Based Application. This is an everyday use case scenario used by several organizations. We will be using Jenkins as a CI/CD 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/Venn1991/jpetstore-6.git>

Steps:-

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

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

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

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

Step 5 — Install OWASP Dependency Check Plugins

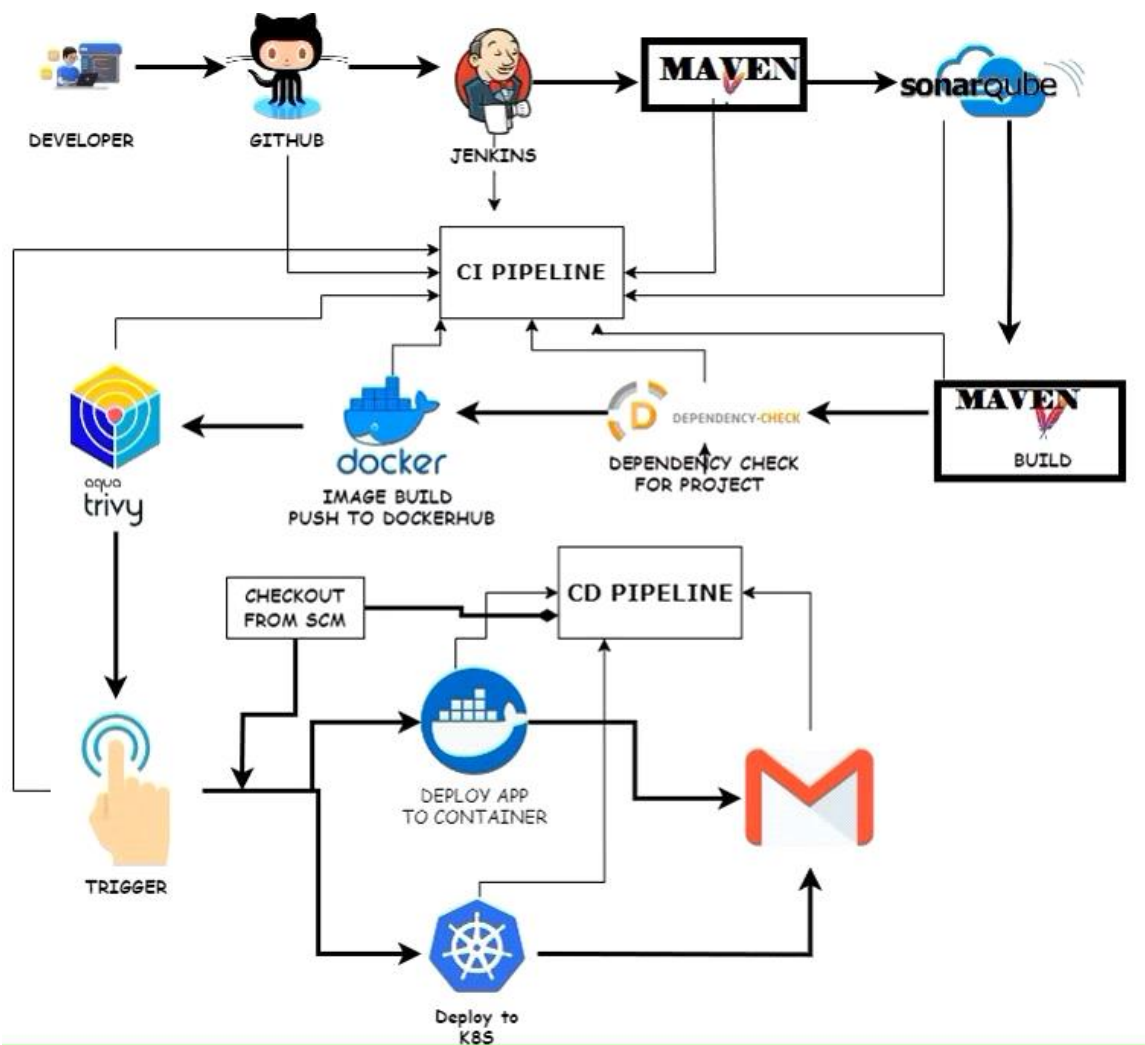
Step 6 — Docker Image Build and Push

Step 7 — Deploy the image using Docker

Step 8 — Kubernetes master and slave setup on Ubuntu (20.04)

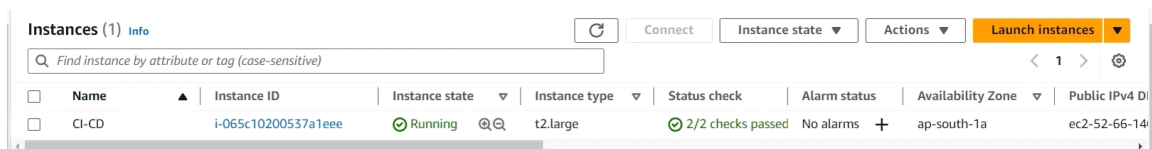
Step 9 — Access the Real-World Application

Step 10 — Terminate the AWS EC2 Instances.



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).



Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

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

```
vi jenkins.sh
```

```
#!/bin/bash
```

```
sudo apt update -y
```

```
#sudo apt upgrade -y
```

```
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee  
/etc/apt/keyrings/adoptium.asc
```

```
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc]  
https://packages.adoptium.net/artifactory/deb ${awk -F= '/^VERSION_CODENAME/{print$2}'  
/etc/os-release} main" | tee /etc/apt/sources.list.d/adoptium.list
```

```
sudo apt update -y
```

```
sudo apt install temurin-17-jdk -y
```

```
sudo apt install maven -y
```

```

/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
    /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
    https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
sudo systemctl status Jenkins
sudo chmod 777 jenkins.sh
./jenkins.sh # this will install Jenkins

```

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

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

```

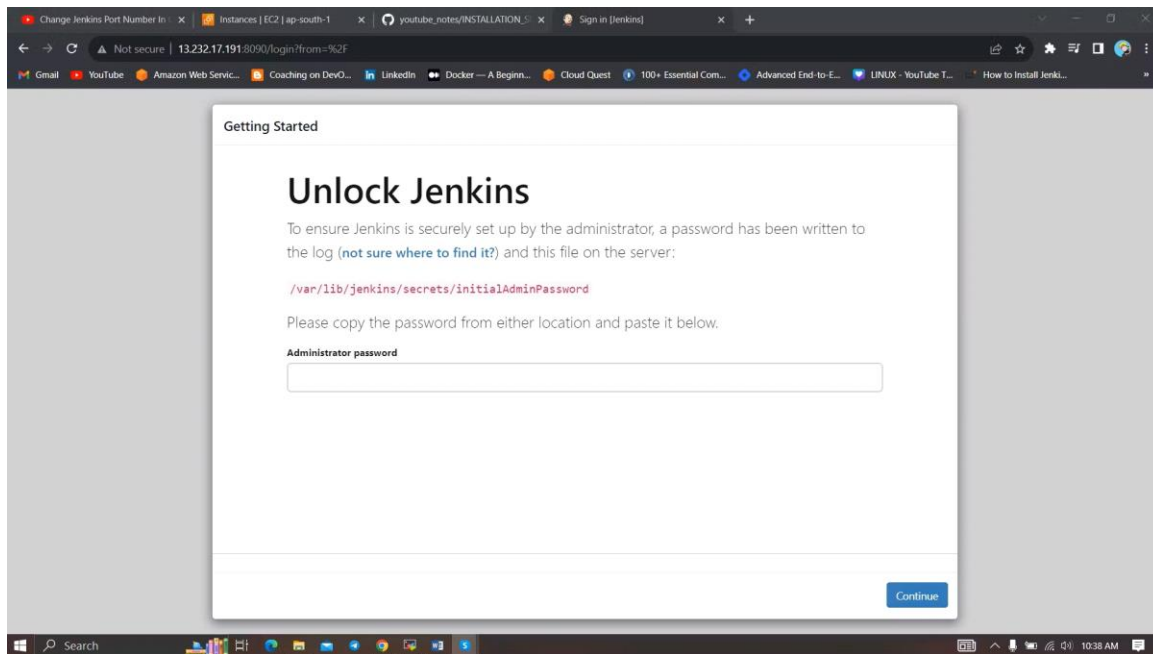
sudo systemctl stop jenkins
sudo systemctl status jenkins
cd /etc/default
sudo vi jenkins #change port HTTP_PORT=8090 and save and exit
cd /lib/systemd/system
sudo vi jenkins.service #change Environments="Jenkins_port=8090" save and exit
sudo systemctl daemon-reload
sudo systemctl restart jenkins
sudo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

```

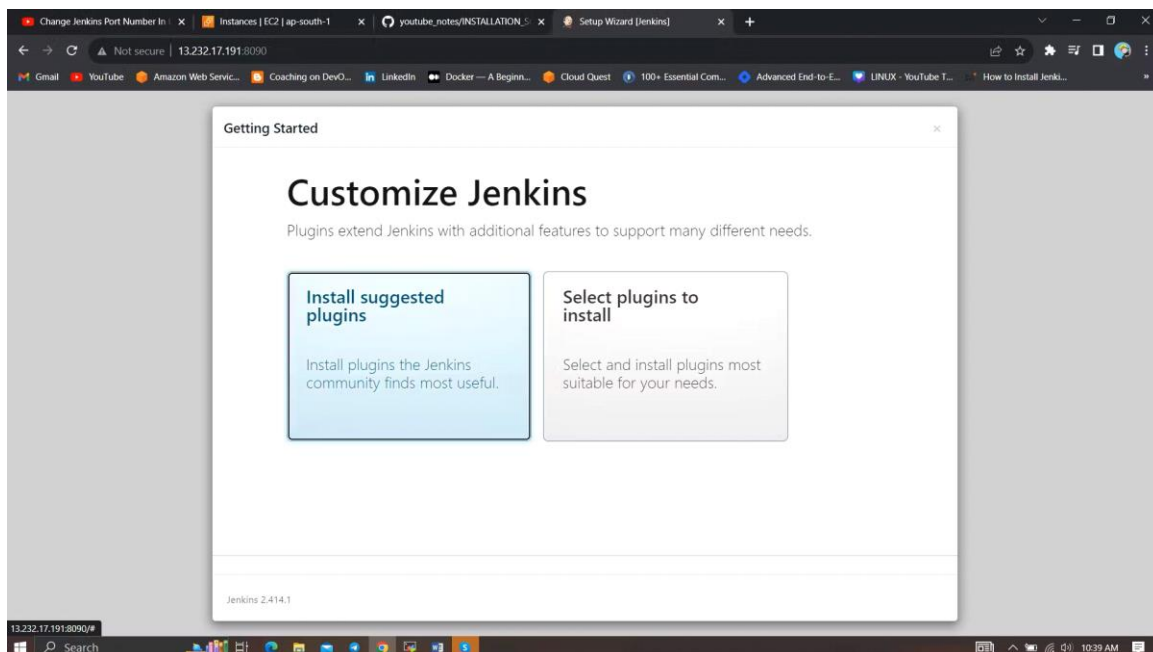
Now, grab your Public IP Address

<EC2 Public IP Address:8090>

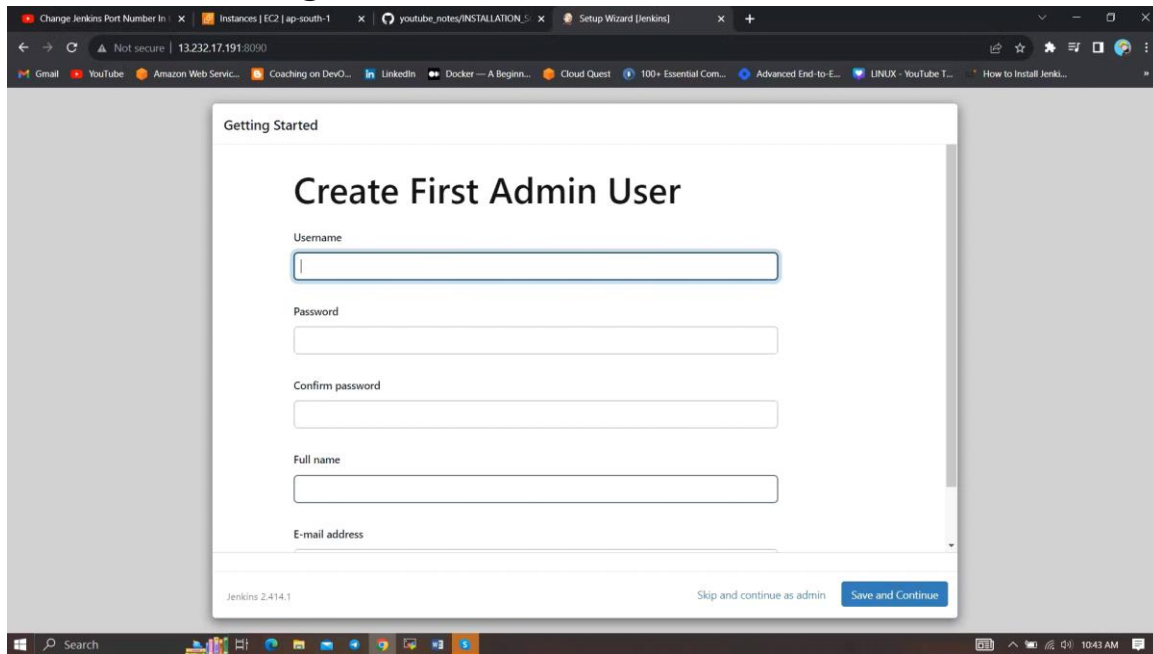
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



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



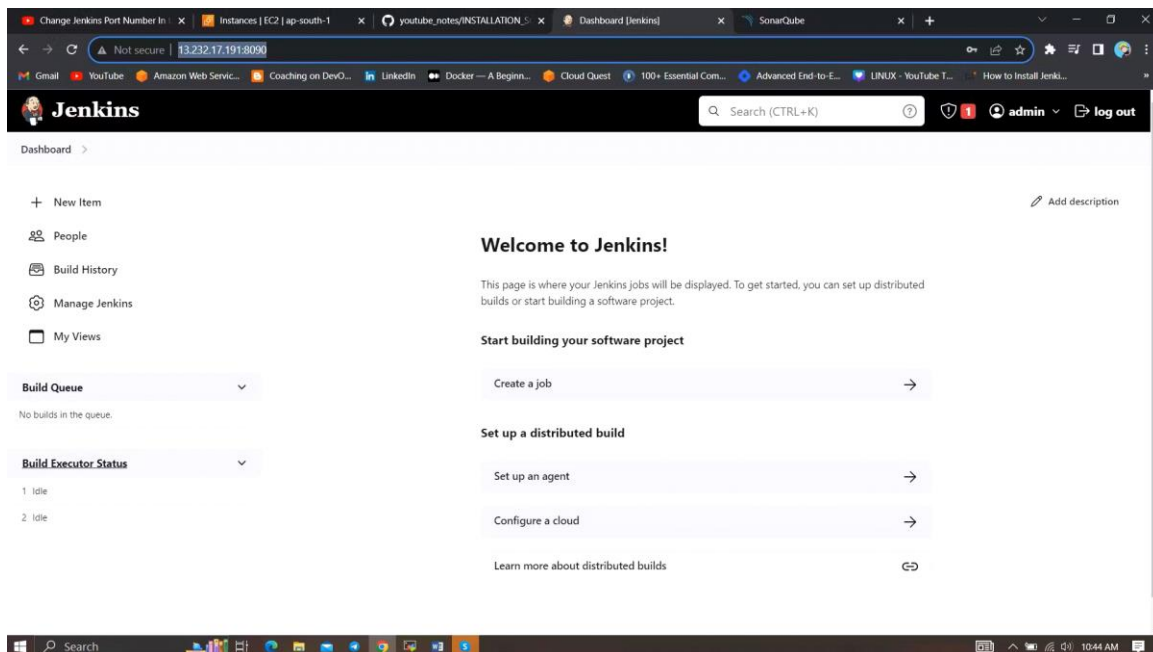
Jenkins will now get installed and install all the libraries.



The screenshot shows a web browser window with the Jenkins 'Getting Started' page. The main heading is 'Create First Admin User'. Below this, there are five input fields: 'Username', 'Password', 'Confirm password', 'Full name', and 'E-mail address'. At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The browser's address bar shows the URL '13.232.17.191:8090'. The Windows taskbar is visible at the bottom of the screen.

Create a user click on save and continue.

Jenkins Getting Started Screen.



The screenshot shows the Jenkins Dashboard. The top navigation bar includes the Jenkins logo, a search bar, and a user profile dropdown for 'admin' with a 'log out' button. The main content area is titled 'Welcome to Jenkins!' and includes a brief introduction. Below this, there are two main sections: 'Start building your software project' and 'Set up a distributed build'. The 'Start building your software project' section has a 'Create a job' button. The 'Set up a distributed build' section has buttons for 'Set up an agent', 'Configure a cloud', and a link to 'Learn more about distributed builds'. On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The 'Build Queue' section shows 'No builds in the queue.' and the 'Build Executor Status' section shows two idle executors.

2B — Install Docker

```
sudo apt-get update
```

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

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

```
newgrp docker
```

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

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

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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c2b4312b4b6a: Pull complete
b7fb17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d9c3d4ea8b4455a5a6560654511c88a6816f1603f764d5d5cc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d8ca3665e2f2e3230b50139
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

Log in to SonarQube

Login

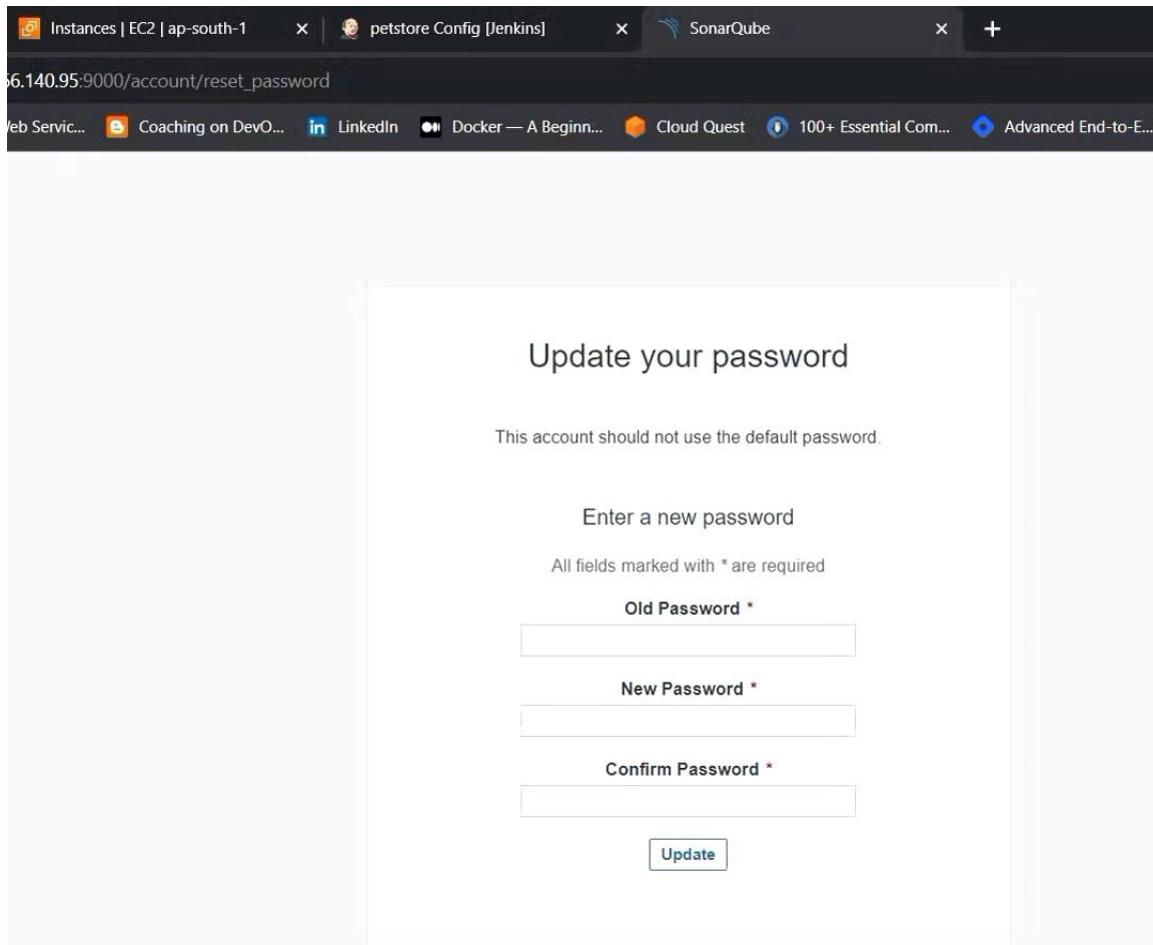
Password

Log in Cancel

Enter username and password, click on login and change password

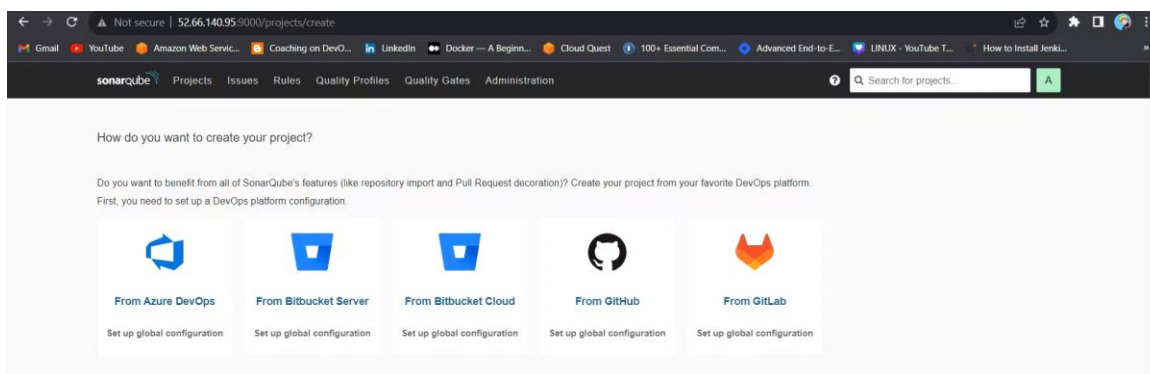
username admin

password admin



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

Update New password, This is Sonar Dashboard.



The screenshot shows the SonarQube 'Create Project' page. The browser address bar shows `52.66.140.95:9000/projects/create`. The page has a navigation bar with 'sonarqube' logo and links for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is on the right. The main content area asks 'How do you want to create your project?' and provides instructions: '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.' Below this are five buttons: 'From Azure DevOps', 'From Bitbucket Server', 'From Bitbucket Cloud', 'From GitHub', and 'From GitLab'. Each button has a corresponding icon and the text 'Set up global configuration' below it.

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

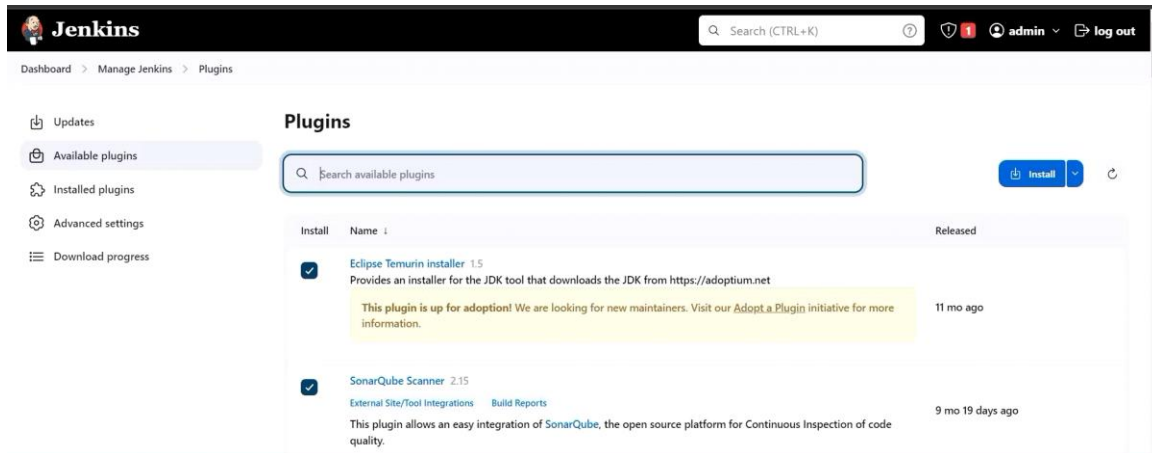
3A — Install Plugin

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

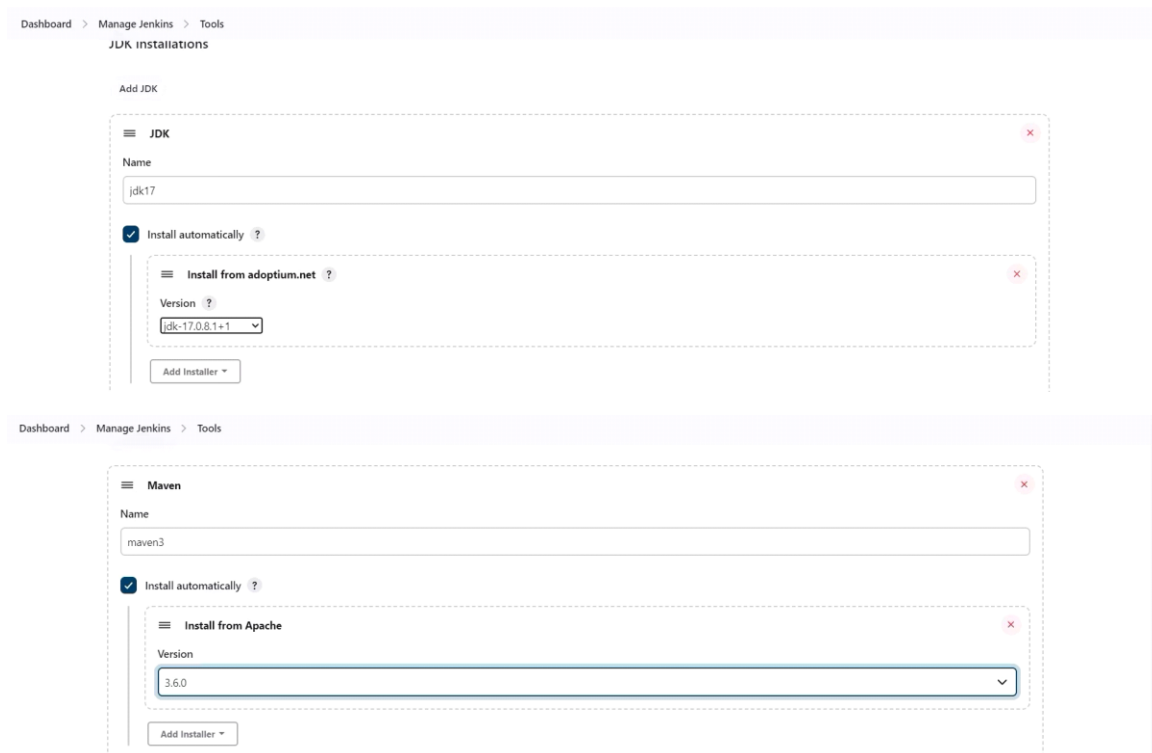
1 → Eclipse Temurin Installer (Install without restart)

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



3C — Create a Job

The screenshot shows the Jenkins 'Create a new job' dialog. At the top, the Jenkins logo and a search bar are visible. Below the header, the breadcrumb 'Dashboard > All >' is shown. The main area is titled 'Enter an item name'. A text input field contains 'petstore', with a small red asterisk and the text 'Required field' below it. Below the input field are four selectable options, each with an icon and a description:

- 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. (This option is highlighted with a blue border)
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

At the bottom left is a blue 'OK' button. At the bottom right, there is a small text note: 'Create a folder: 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/Venn1991/jpetstore-6.git'
      }
    }
  }
}
```

```

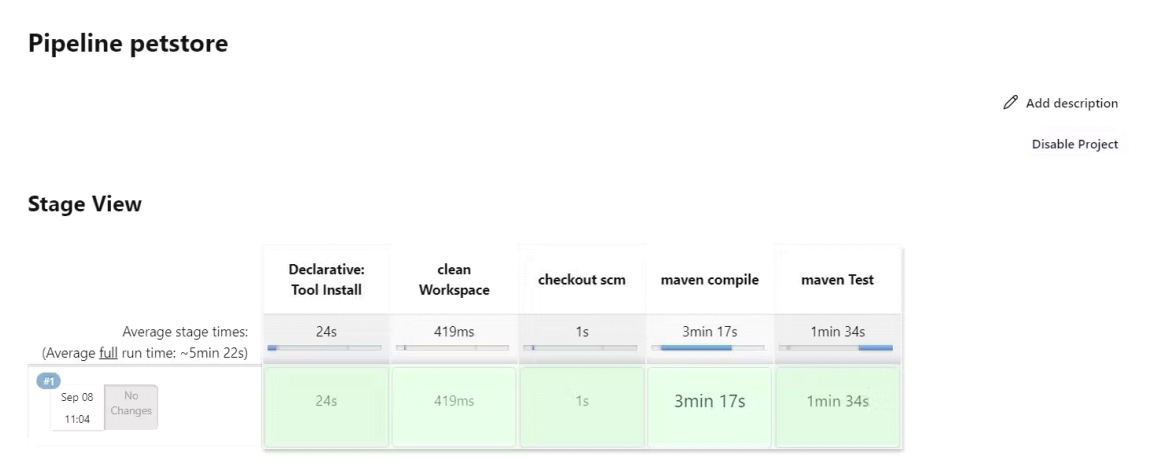
}

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

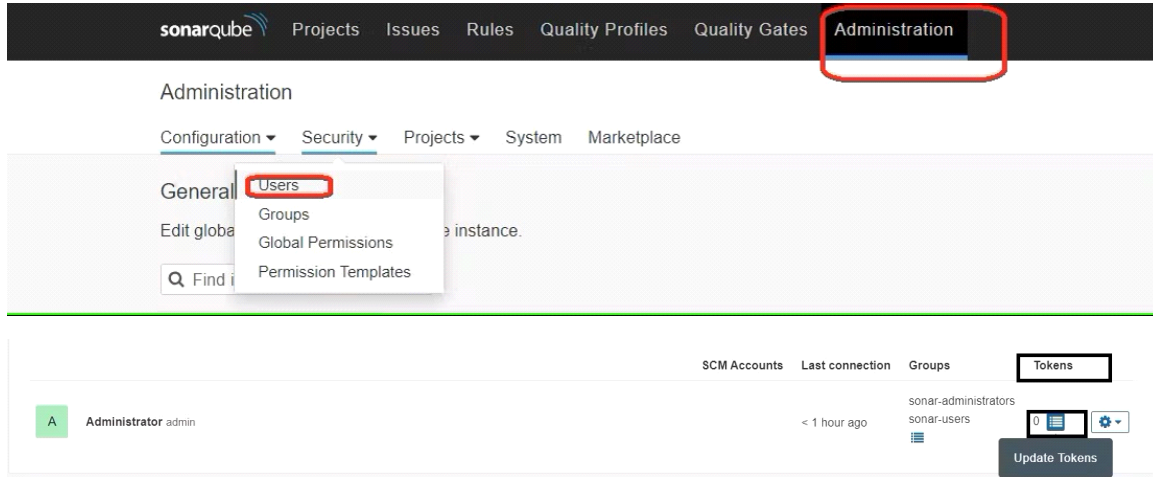
stage ('maven Test') {
    steps {
        sh 'mvn test'
    }
}
}

```

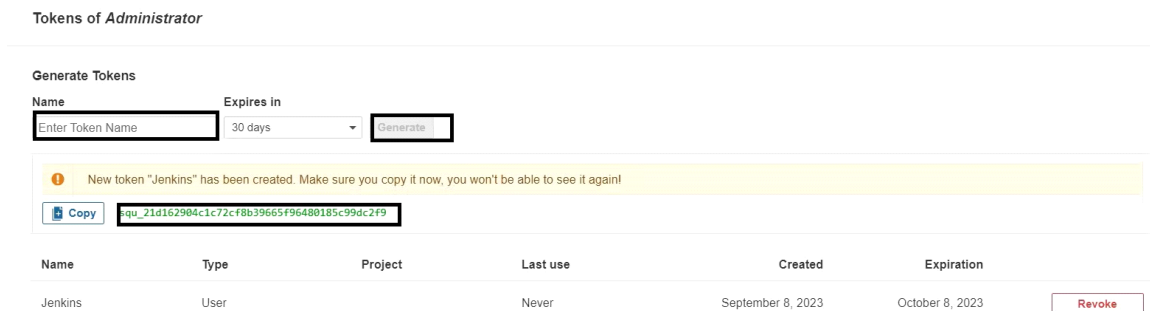
The stage view would look like this,



Step 4 — Configure Sonar Server in Manage Jenkins



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

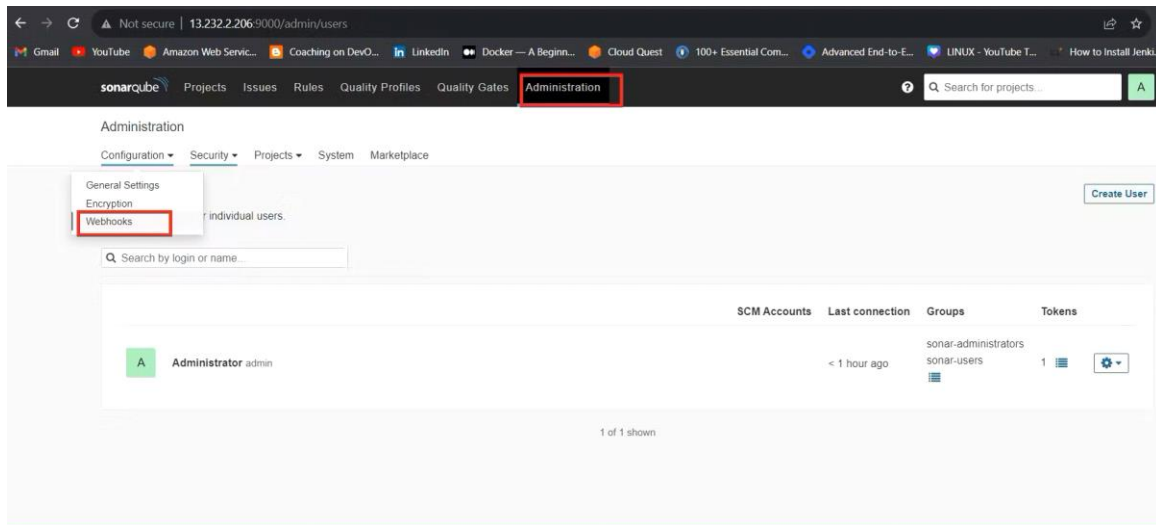
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

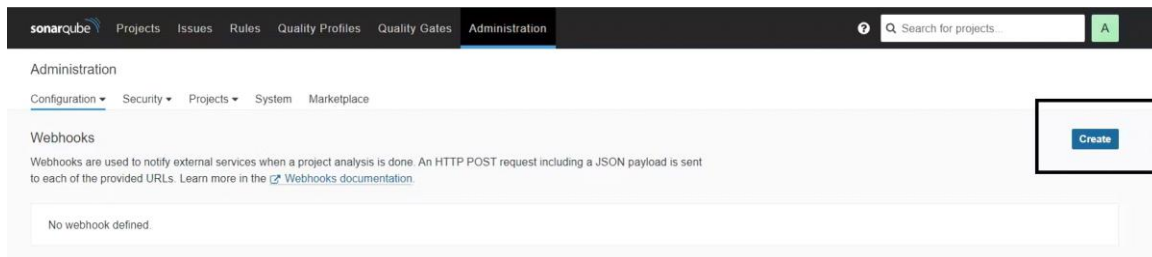
The screenshot shows the Jenkins 'Tools' configuration page for 'SonarQube Scanner installations'. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Tools'. The page title is 'SonarQube Scanner installations'. Below the title is a section 'Add SonarQube Scanner'. Inside this section is a dashed box containing the configuration for a 'SonarQube Scanner'. The 'Name' field is set to 'sonar-scanner'. The 'Install automatically' checkbox is checked. Below this is a sub-section 'Install from Maven Central' with a 'Version' dropdown menu set to 'SonarQube Scanner 5.0.1.3006'. An 'Add Installer' button is at the bottom of the dashed box. Below the dashed box is another 'Add SonarQube Scanner' section. At the bottom of the page are 'Save' and 'Apply' buttons.

In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



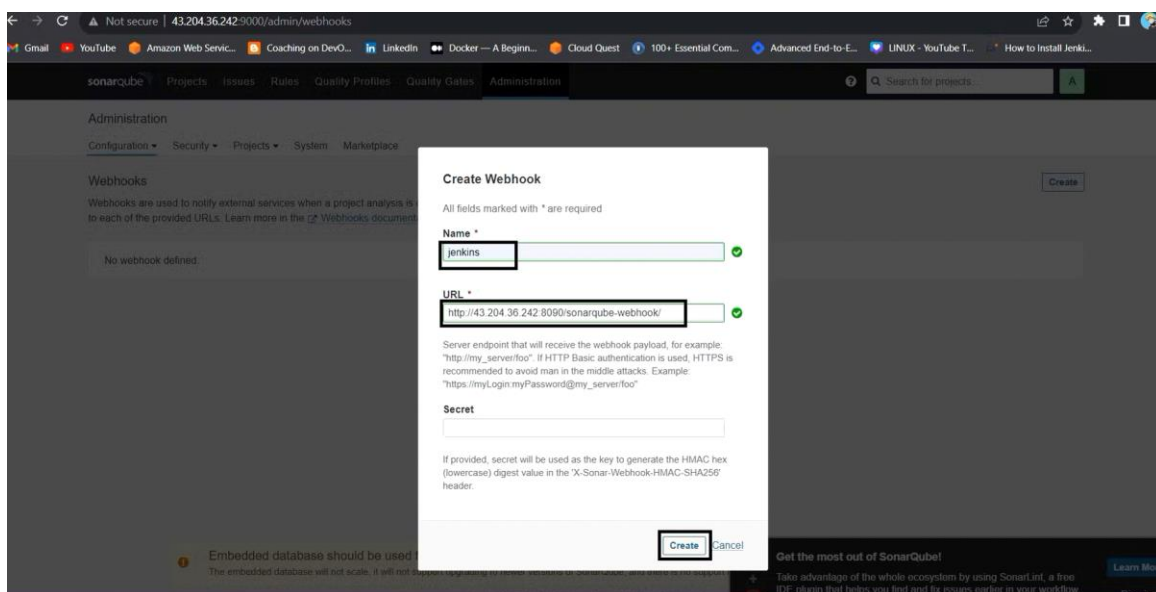
Click on Create



Add details

#in url section of quality gate

<<http://jenkins-public-ip:8090>>/sonarqube-webhook/



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

#under tools section add this environment

```
environment {  
    SCANNER_HOME=tool 'sonar-scanner'  
}
```

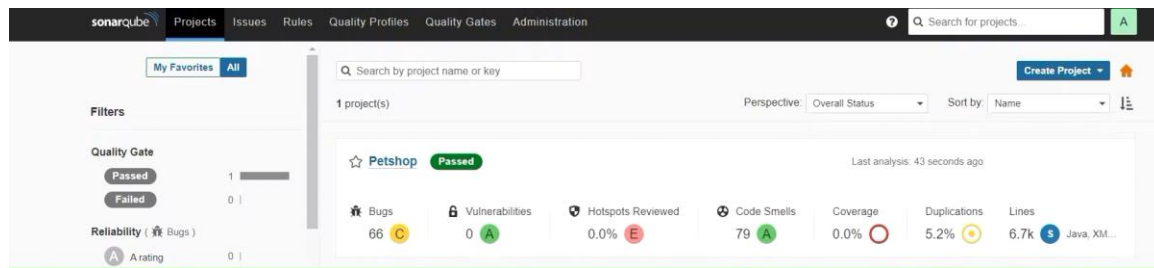
in stages add this

```
stage("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

	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	Dependency-Check Results
Status	SEVERITY DISTRIBUTION
</> Changes	3 4 19
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	

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

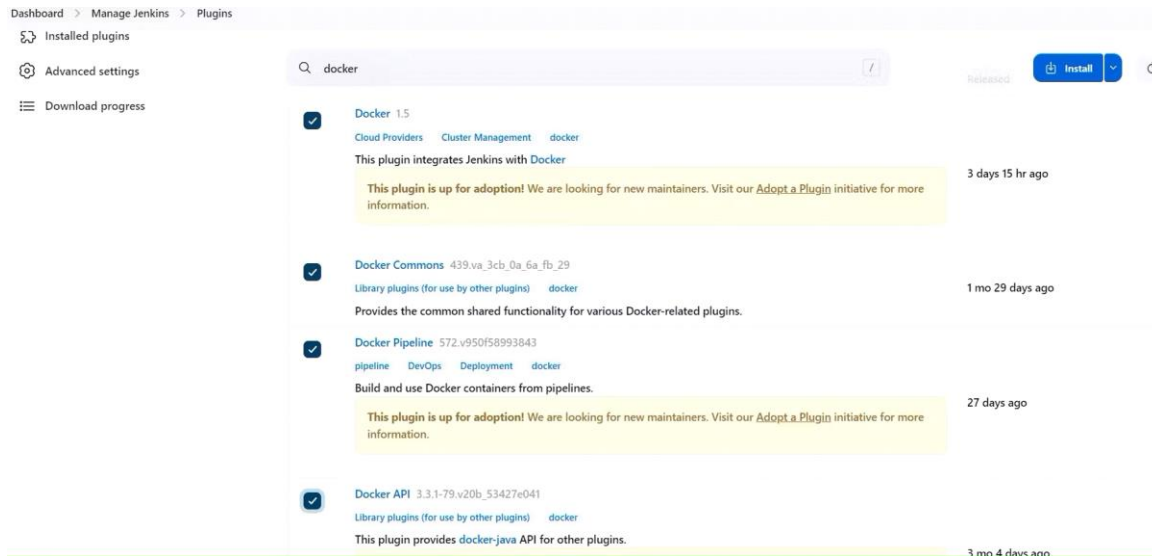
Docker Commons

Docker Pipeline

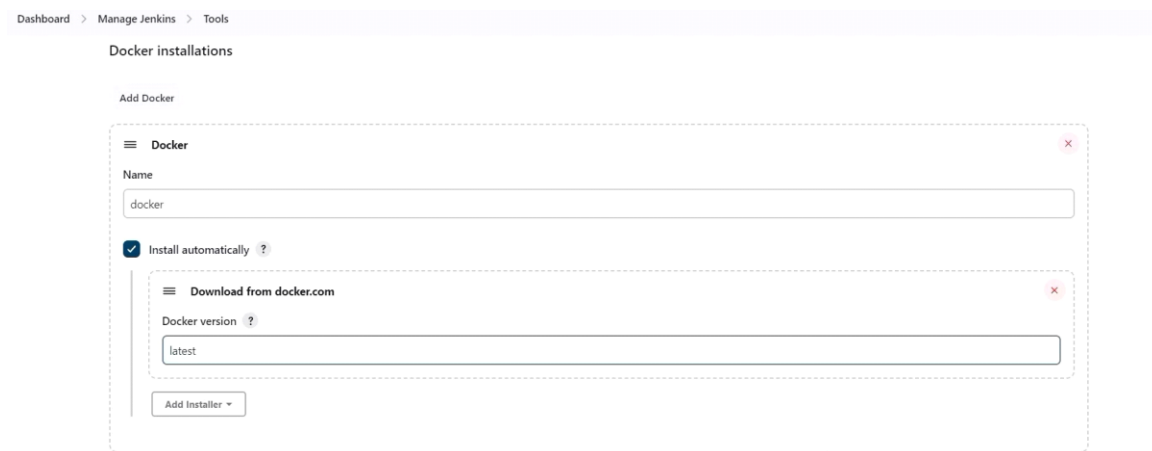
Docker API

docker-build-step

and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



Add DockerHub Username and Password under Global Credentials

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 petshopimaged devopsvmr/petshop:latest"
                sh "docker push devopsvmr/petshop: latest"
            }
        }
    }
}

stage("TRIVY"){
    steps{
        sh "trivy image devopsvmr/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

Description

This repository does not have a description

🕒 Last pushed: an hour ago

Docker commands

To push a new tag to this repository:

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

[Public View](#)

```
<Ec2-public-ip:8080/jpetstore>
```

You will get this output

