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

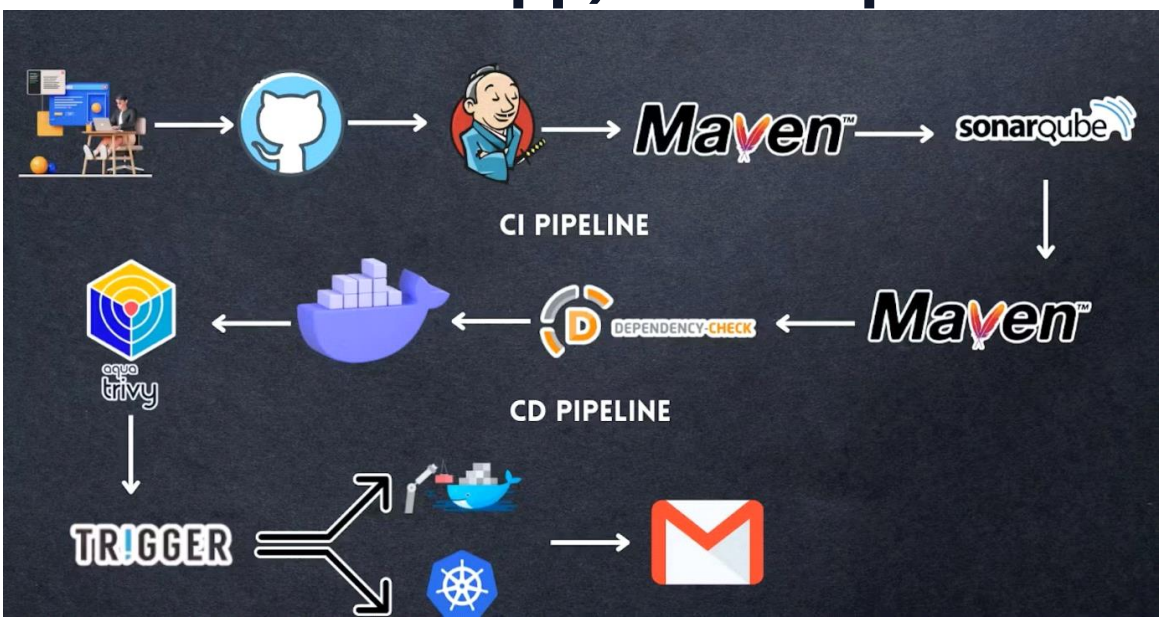


TABLE OF CONTENTS

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

- 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

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 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/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

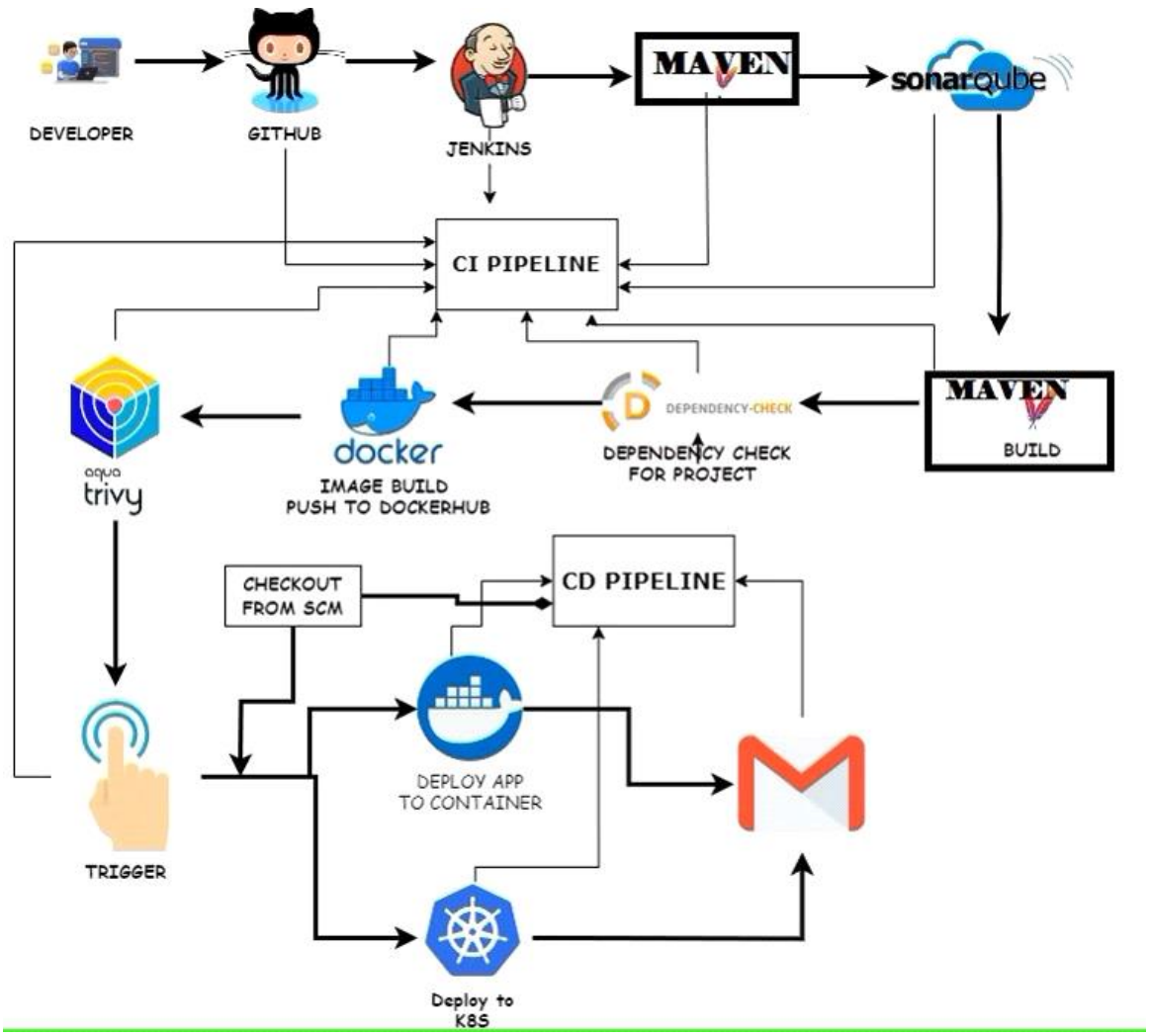
Step 6 — Docker Image Build and Push

Step 7 — Deploy the image using Docker

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

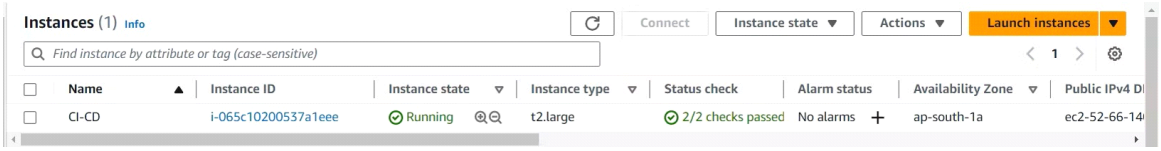
Step 9 — Access the Real-World Application

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 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.

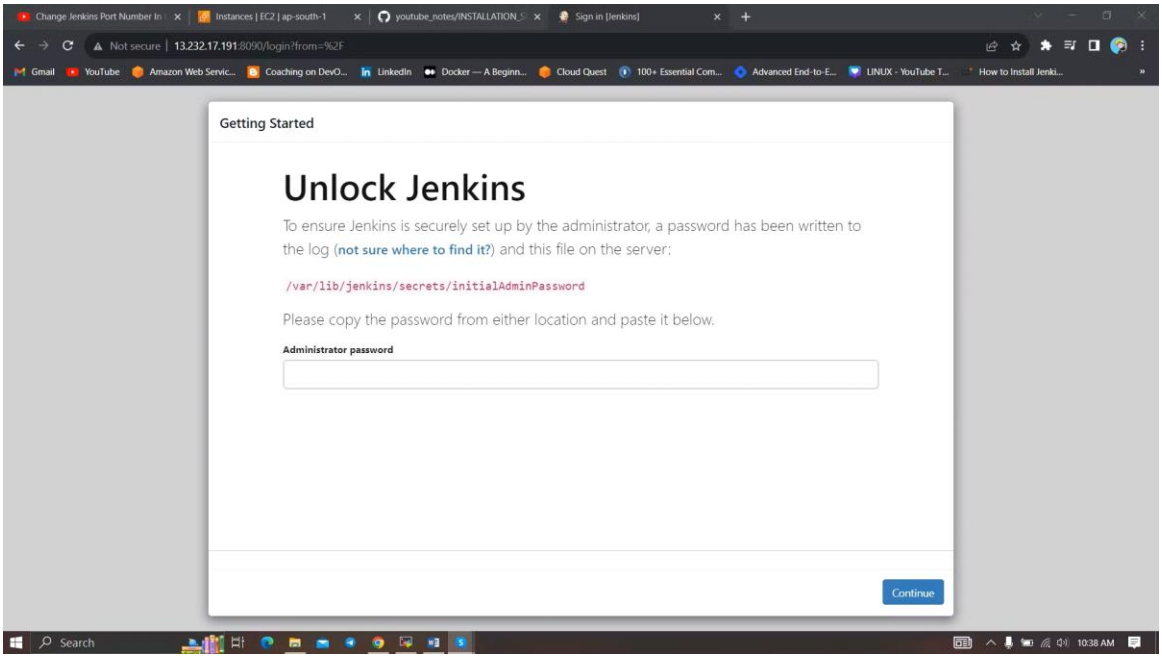
But for this Application case, we are running Jenkins on another port. so change the port to 8090 using the below commands.

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

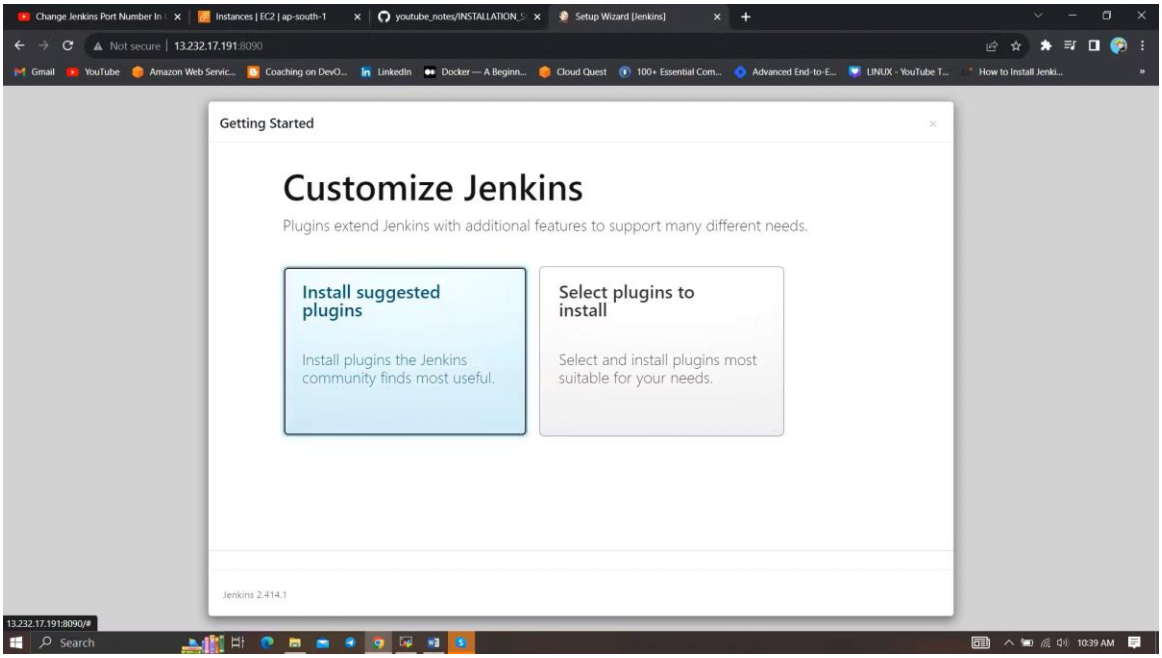
Now, grab your Public IP Address

<EC2 Public IP Address:8090>

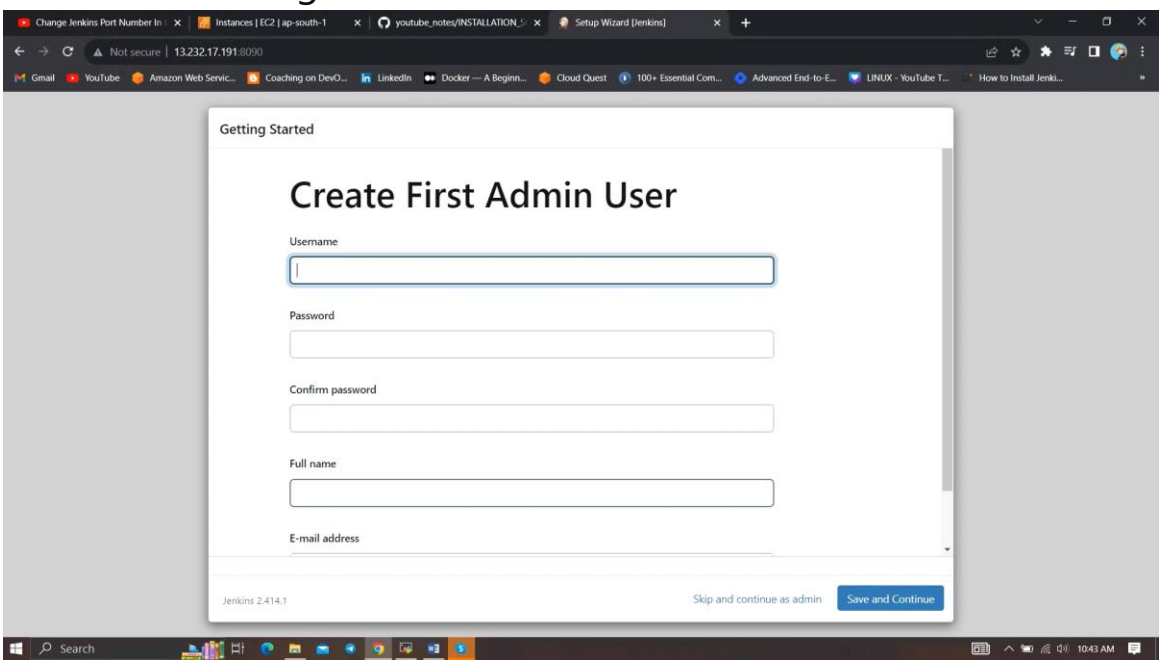
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



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

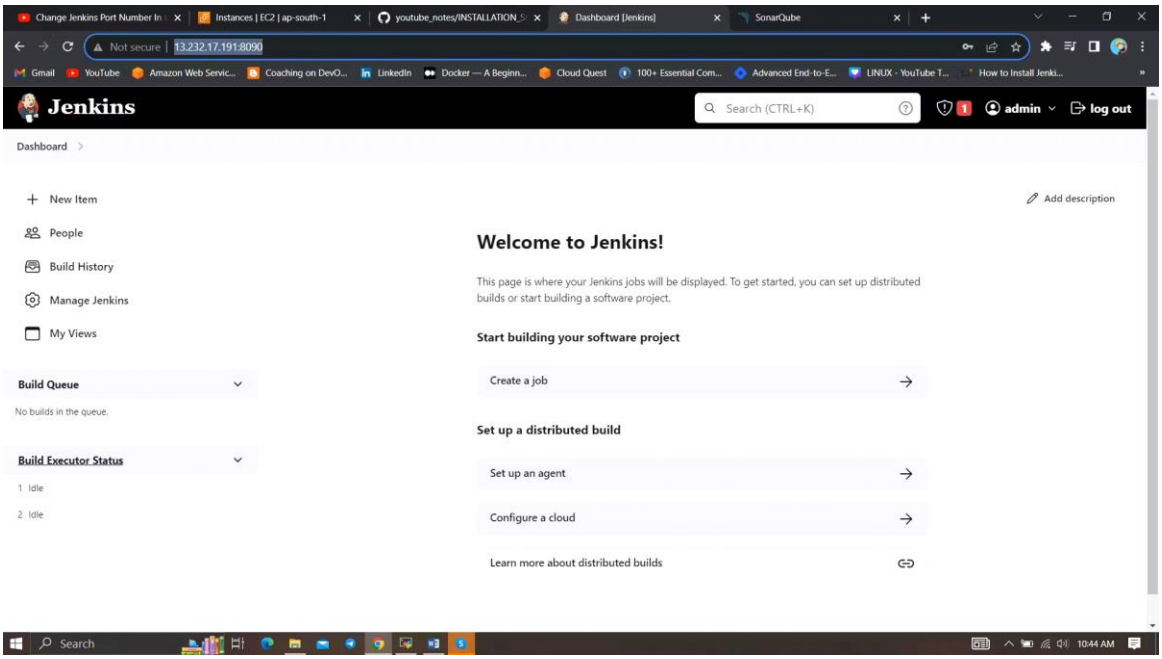


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



2B — Install Docker

sudo apt-get update

sudo apt-get install docker.io -y

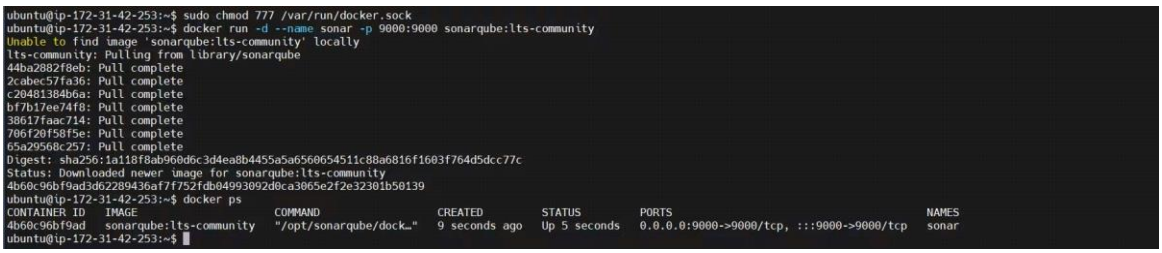
sudo usermod -aG docker $USER #my case is ubuntu

newgrp docker

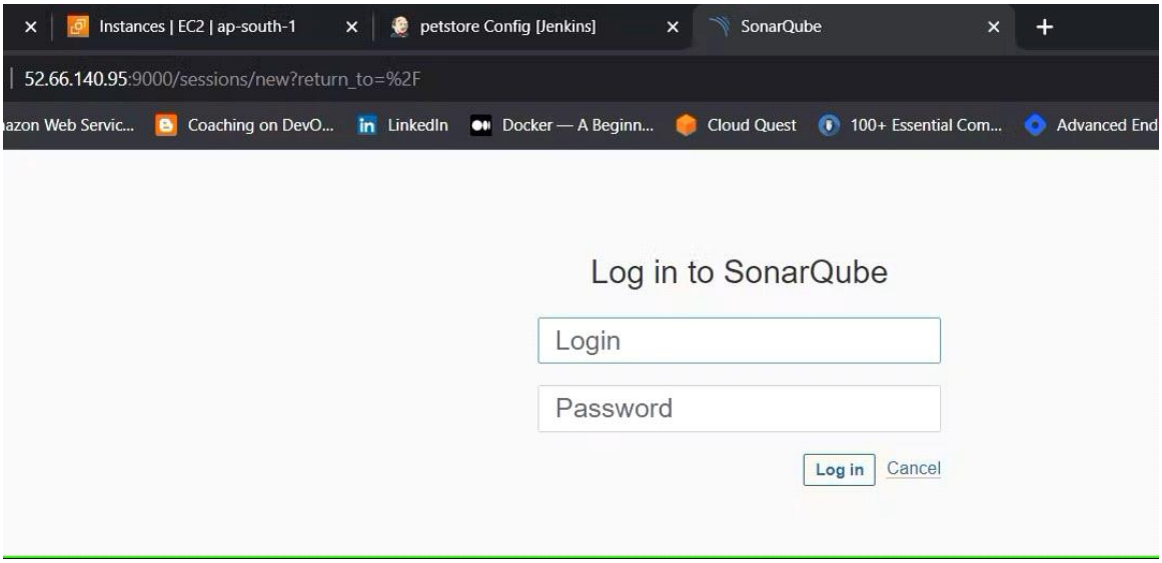
sudo chmod 777 /var/run/docker.sock

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

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



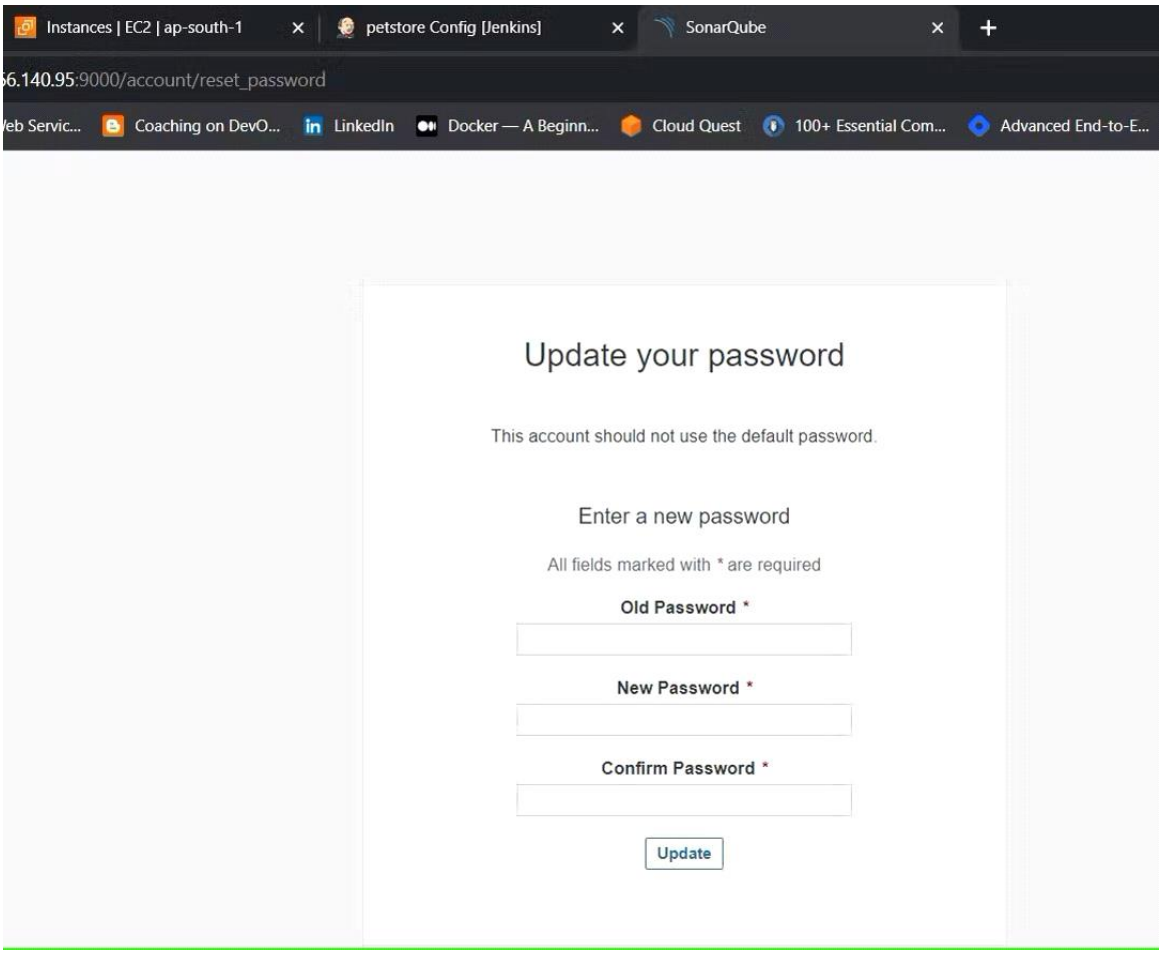
Now our SonarQube is up and running



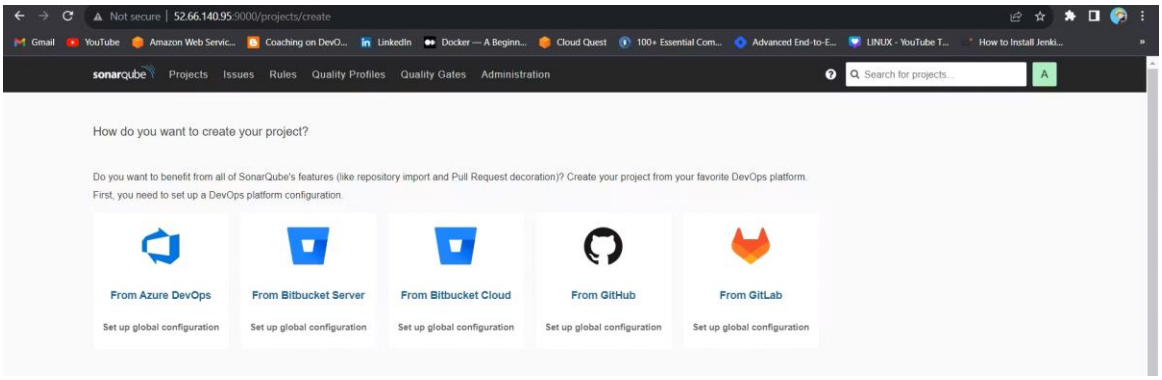
Enter username and password, click on login and change password

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

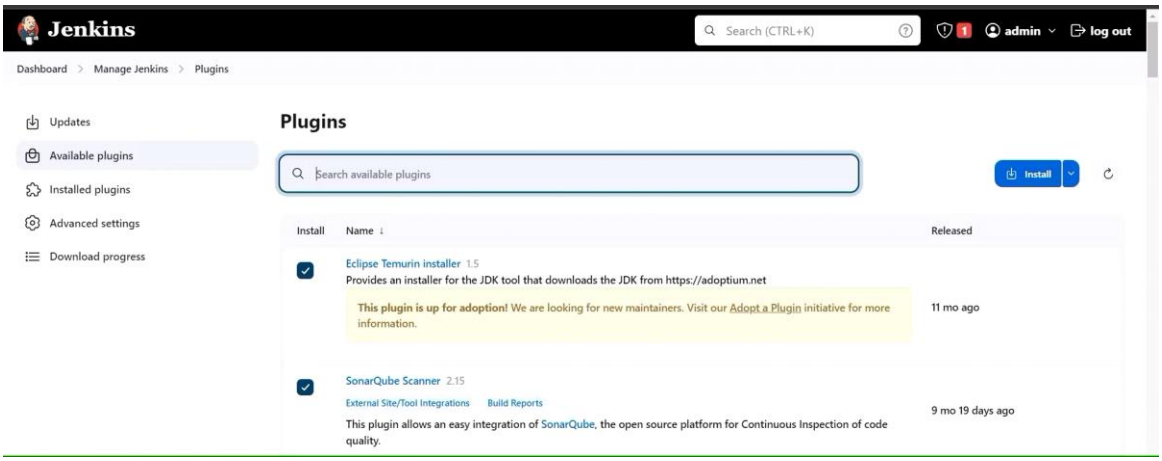
3A — Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

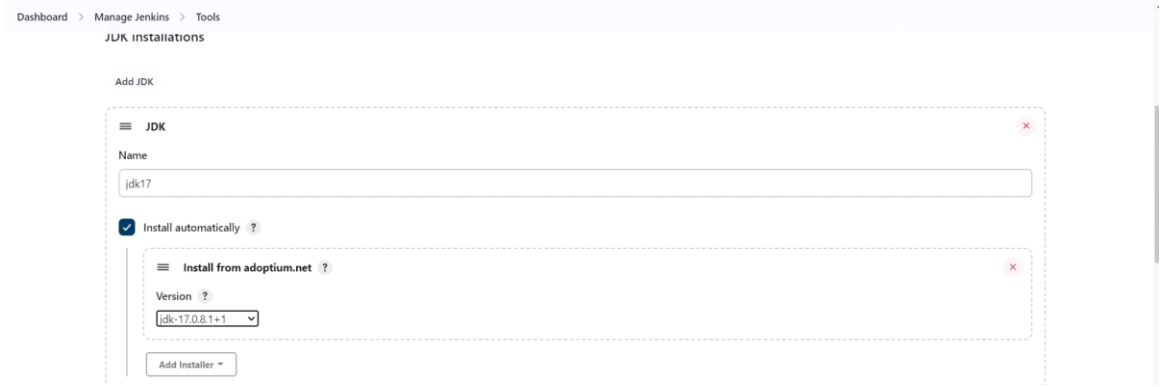
1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

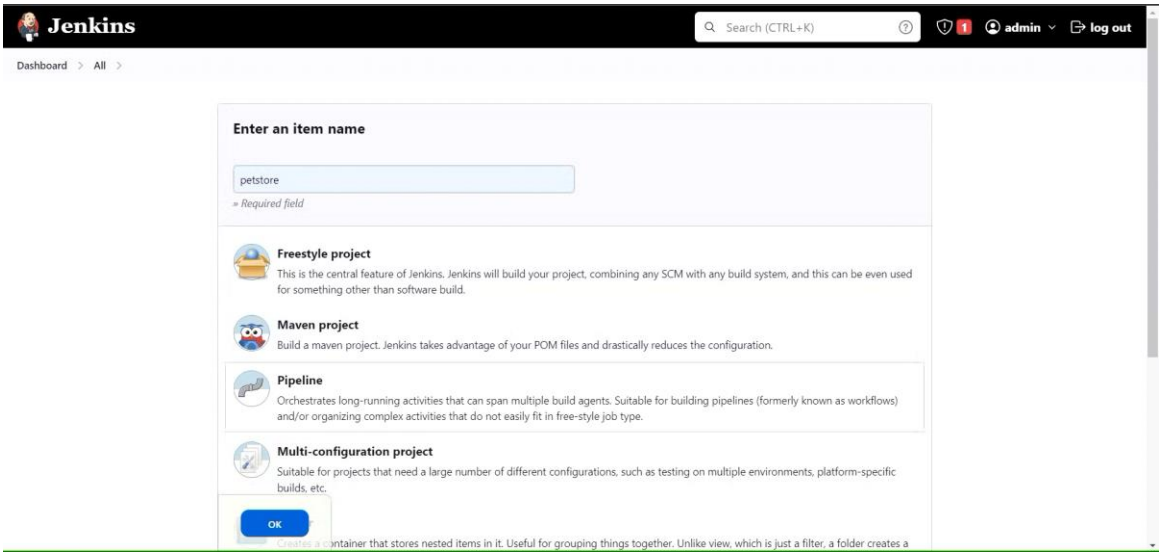


3B — Configure Java and Maven in Global Tool Configuration

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



3C — Create a Job

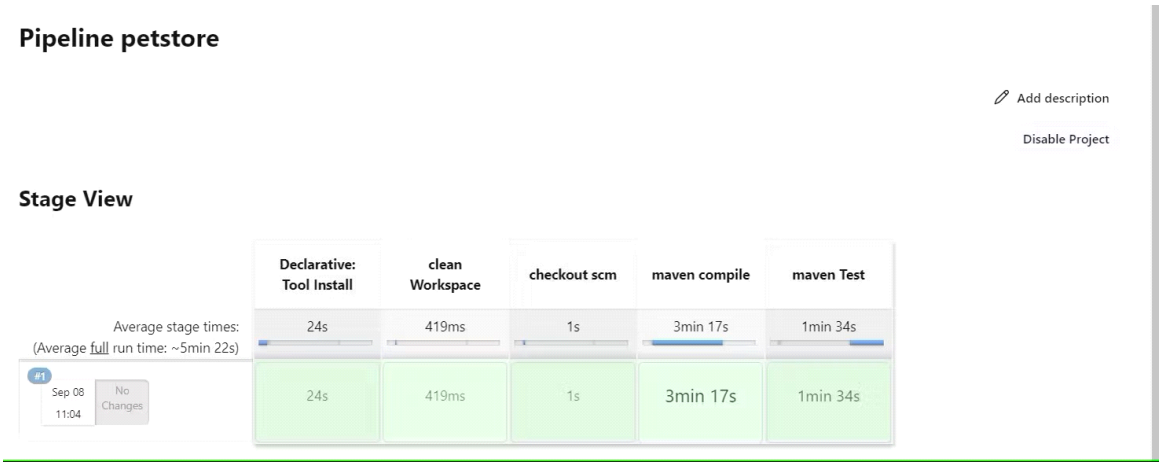


Enter this in Pipeline Script,

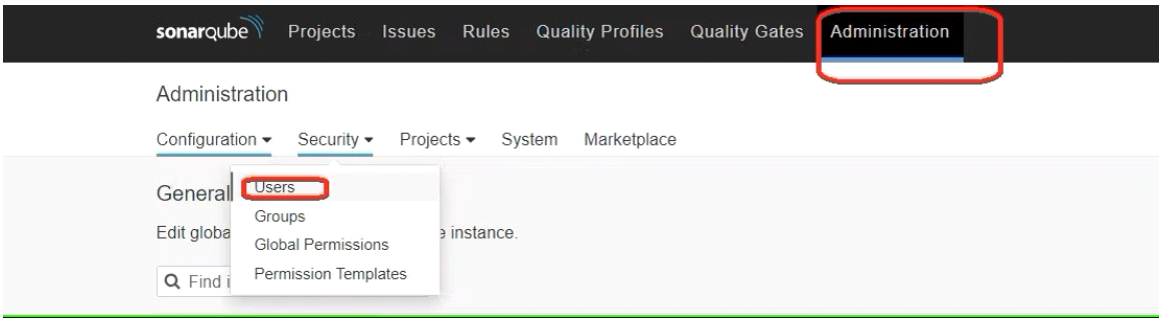
```
pipeline{
  agent any
  tools {
    jdk 'jdk17'
    maven 'maven3'
  }
  stages{
    stage ('clean Workspace'){
      steps{
        cleanWs()
      }
    }
    stage ('checkout scm') {
      steps {
        git ' https://github.com/Venn1991/jpetstore-6.git'
      }
```

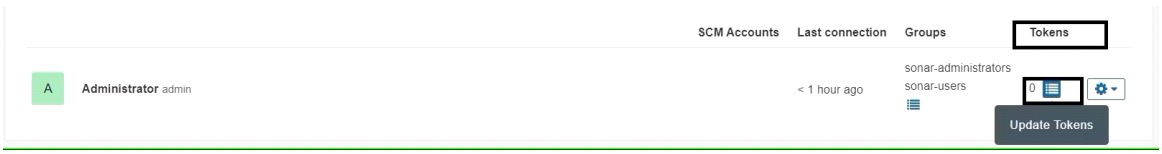
```
    }
    stage ('maven compile') {
      steps {
        sh 'mvn clean compile'
      }
    }
    stage ('maven Test') {
      steps {
        sh 'mvn test'
      }
    }
  }
}
```

The stage view would look like this,

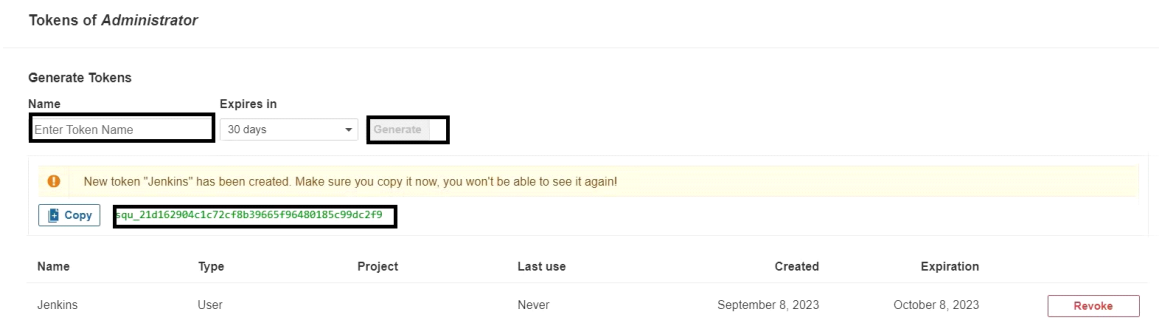


Step 4 — Configure Sonar Server in Manage Jenkins



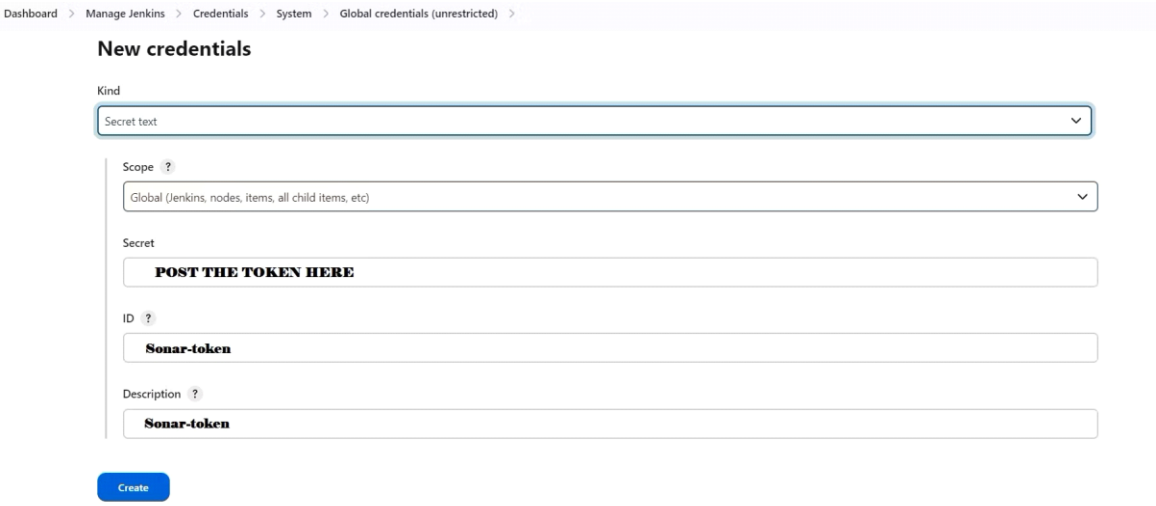


Create a token with a name and generate

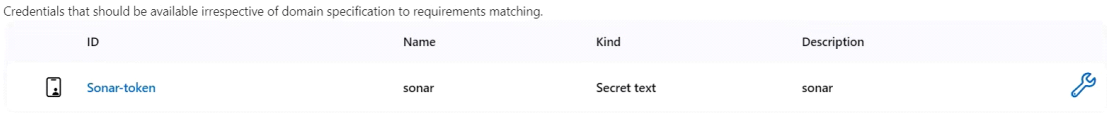


copy Token

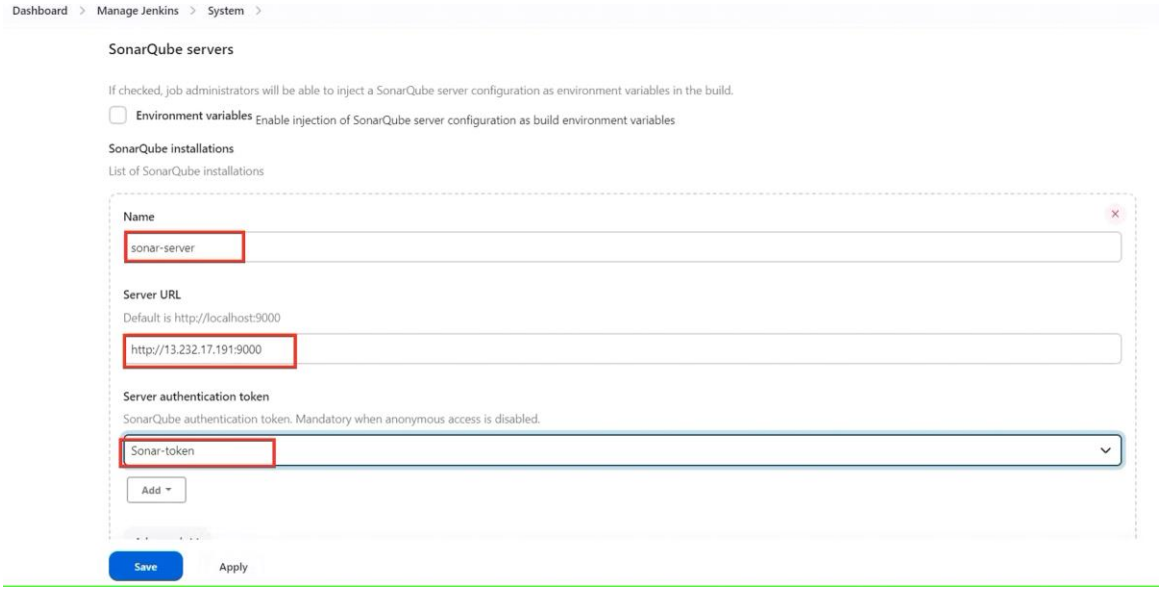
Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this



You will this page once you click on create



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

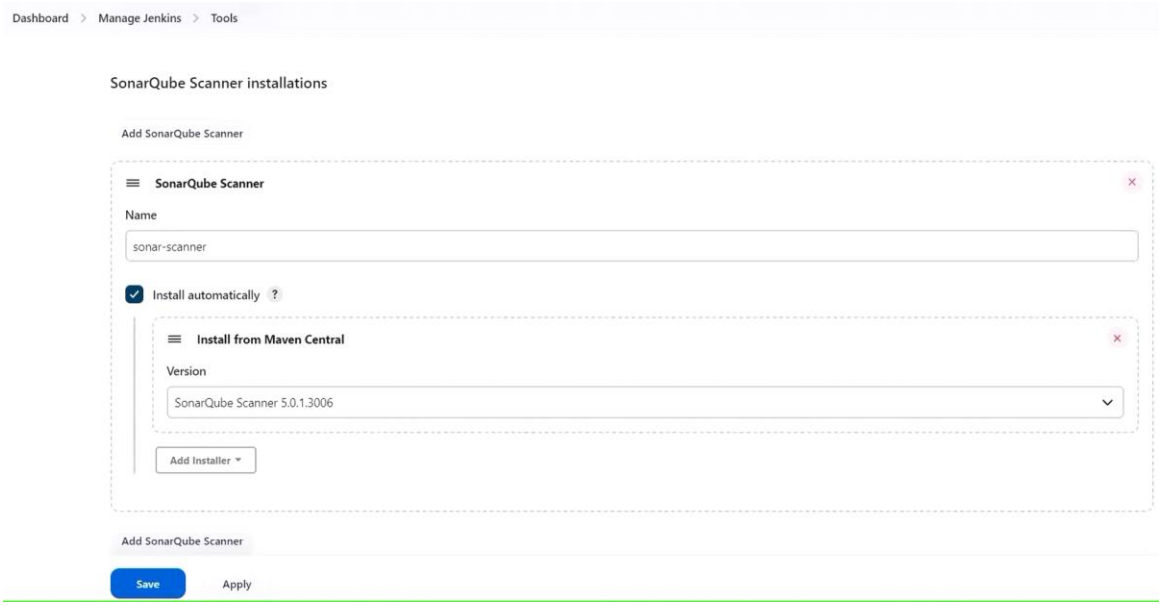


Click on Apply and Save

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

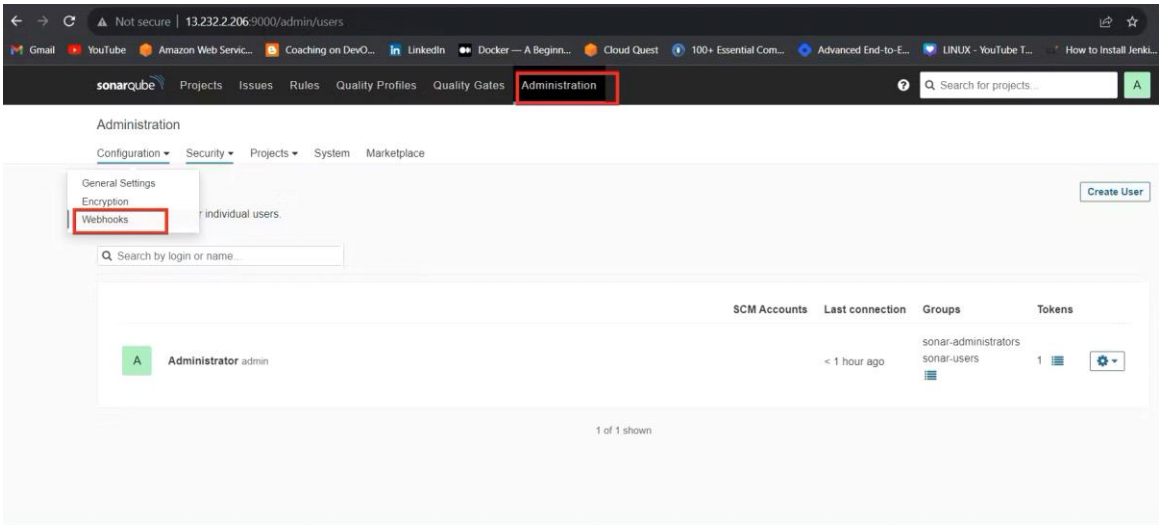
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

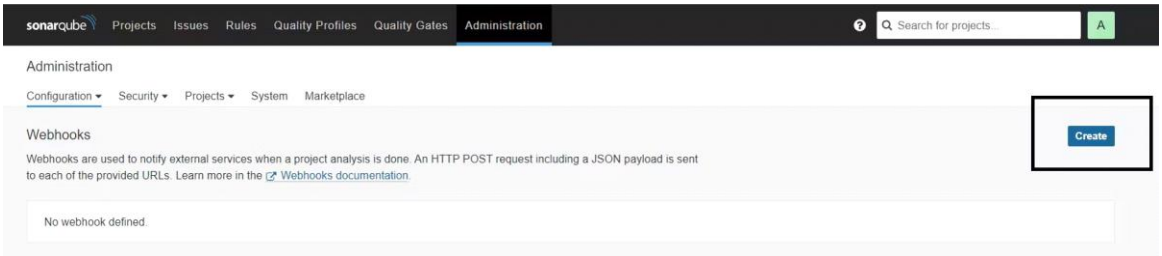


In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



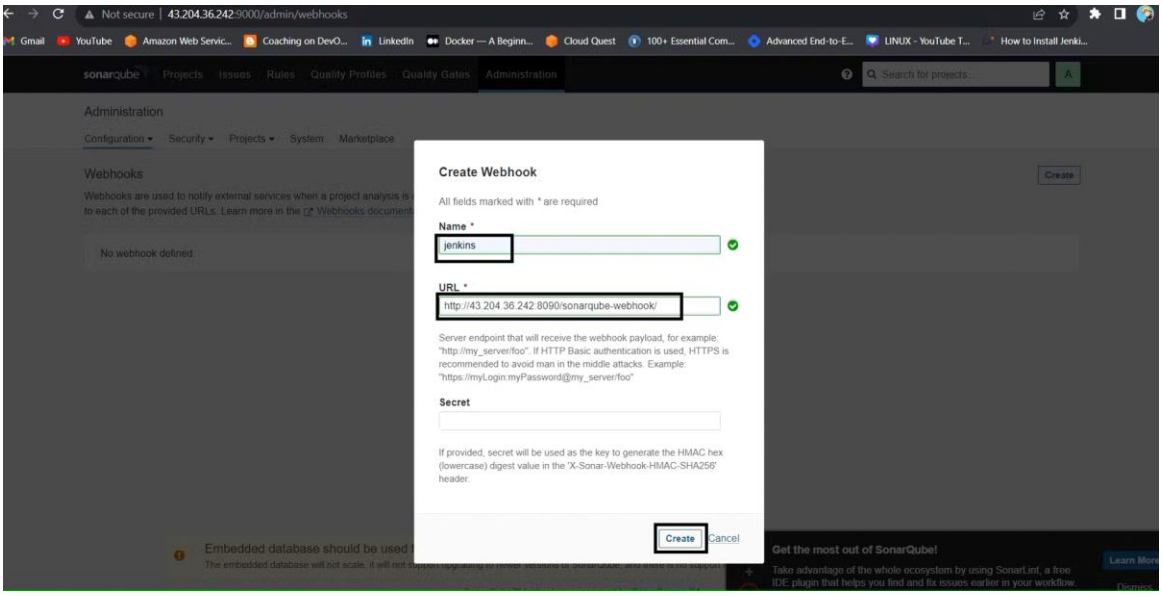
Click on Create



Add details

#in url section of quality gate

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

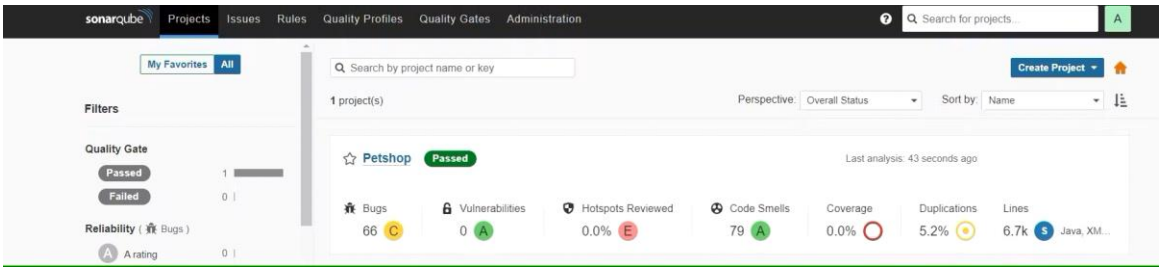


Let's go to our Pipeline and add Sonarqube Stage in our Pipeline Script.

```
#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.



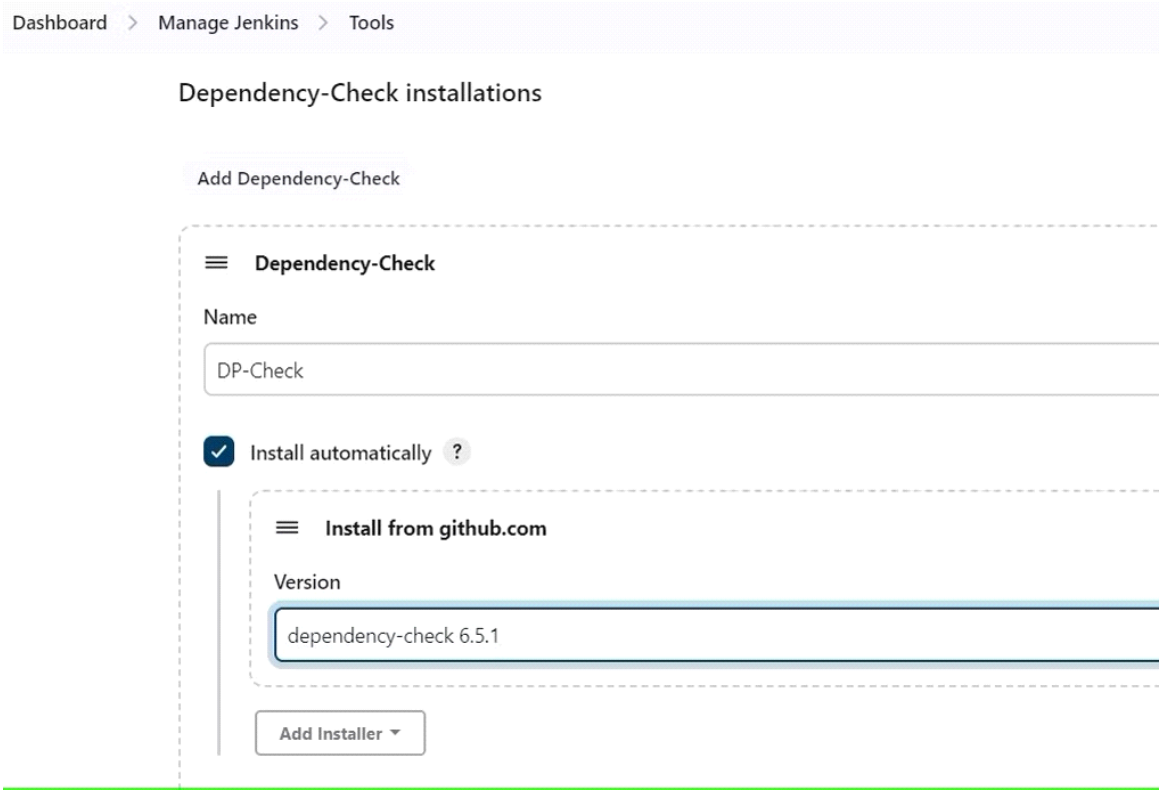
You can see the report has been generated and the status shows as passed. You can see that there are 6.7k lines. To see a detailed report, you can go to issues.

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 →



Click on Apply and Save here.

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

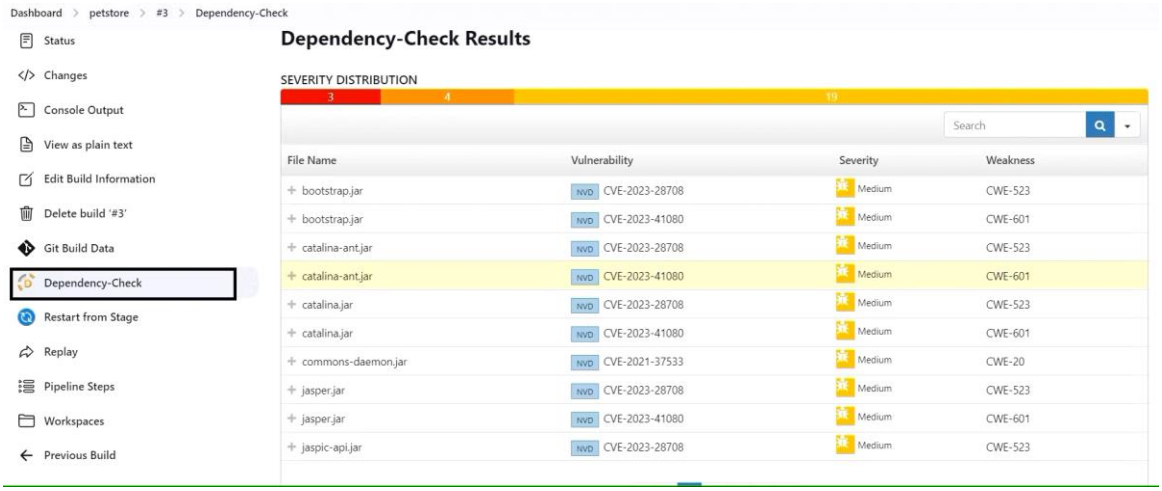
```
stage ('Build war file'){
        steps{
            sh 'mvn clean install -DskipTests=true'
        }
    }
    stage("OWASP Dependency Check"){
        steps{
            dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'
            dependencyCheckPublisher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

The stage view would look like this,

Stage View

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

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



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`

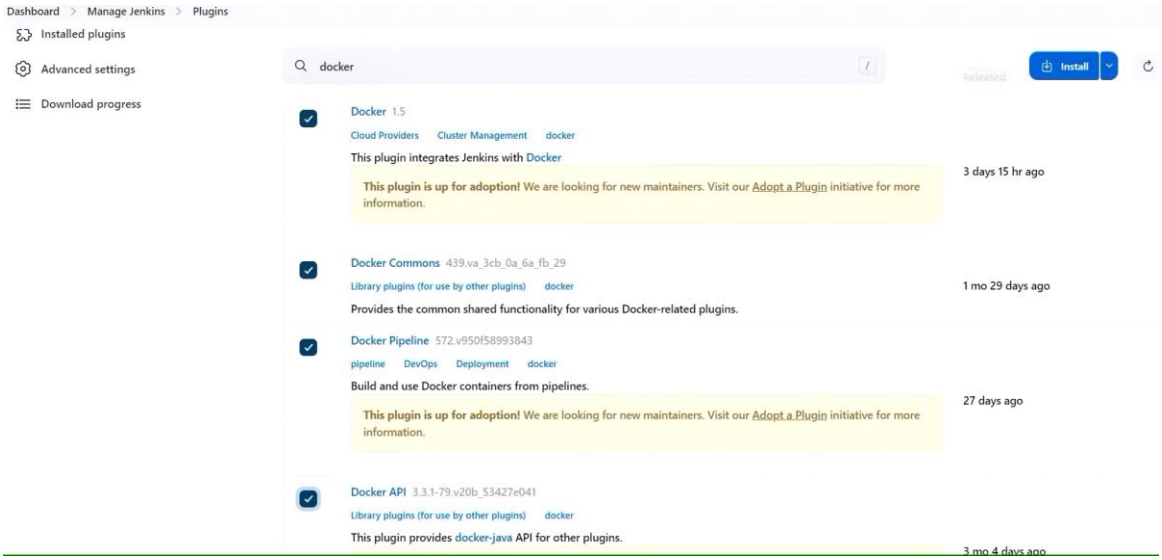
`Docker Commons`

`Docker Pipeline`

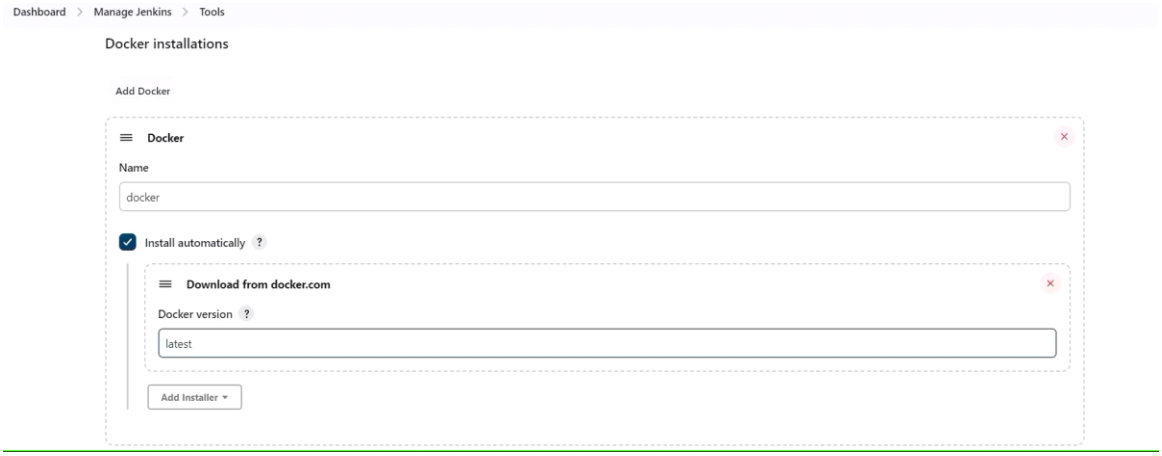
`Docker API`

`docker-build-step`

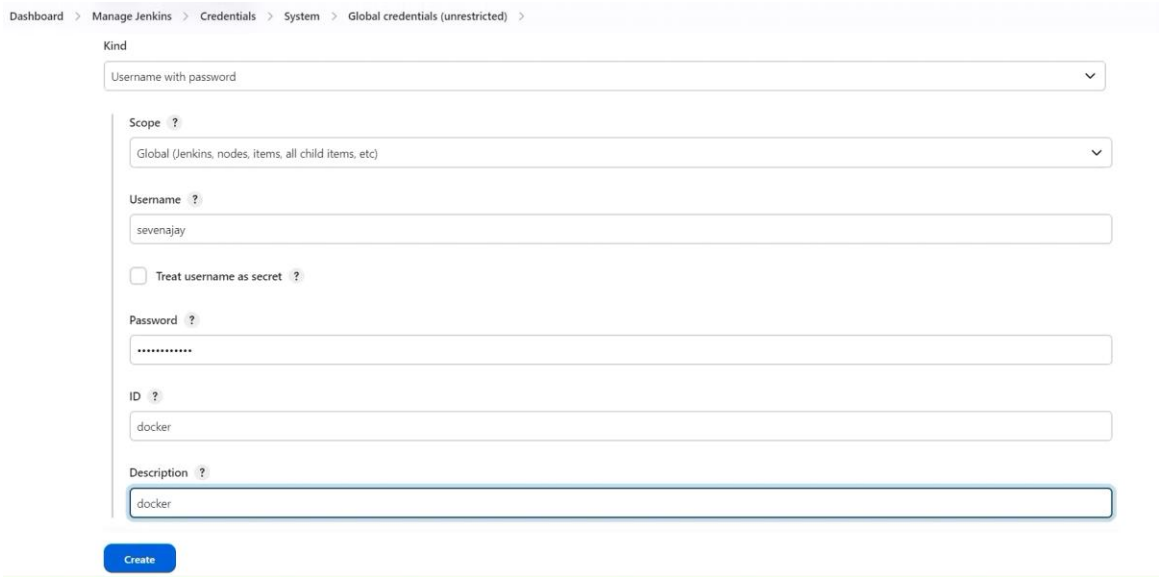
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



Add DockerHub Username and Password under Global Credentials

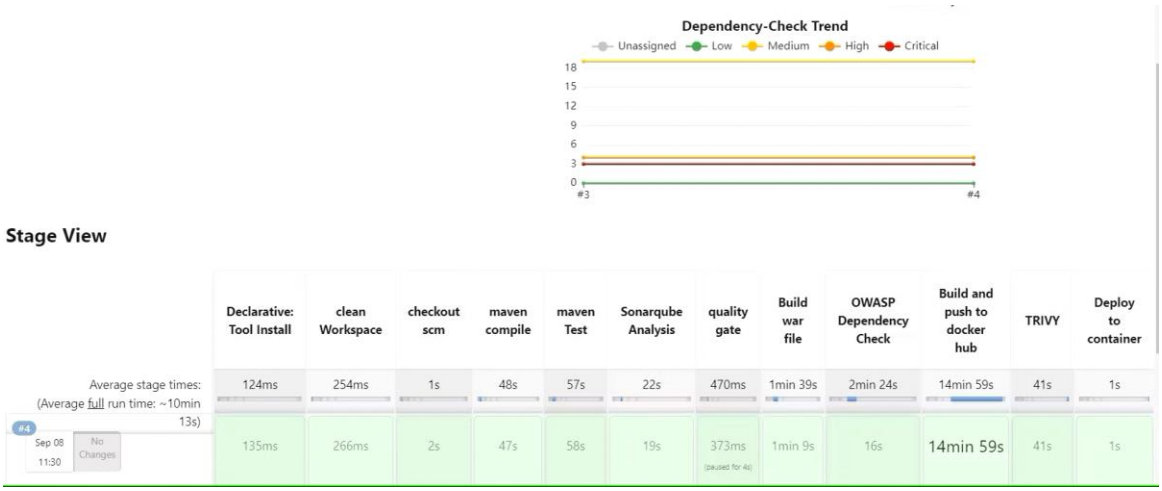


Add this stage to Pipeline Script

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

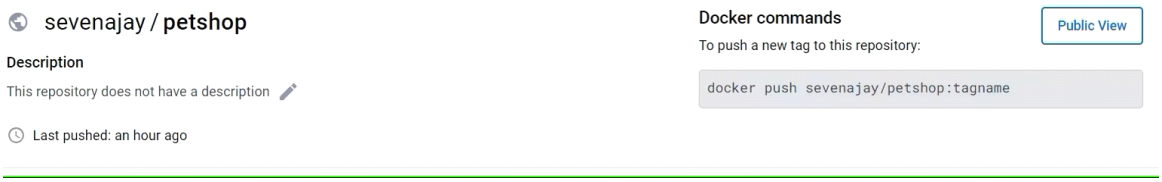
```
        steps{
            sh 'docker run -d --name pet1 -p 8080:8080 sevenajay/petshop: latest'
        }
    }
```

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



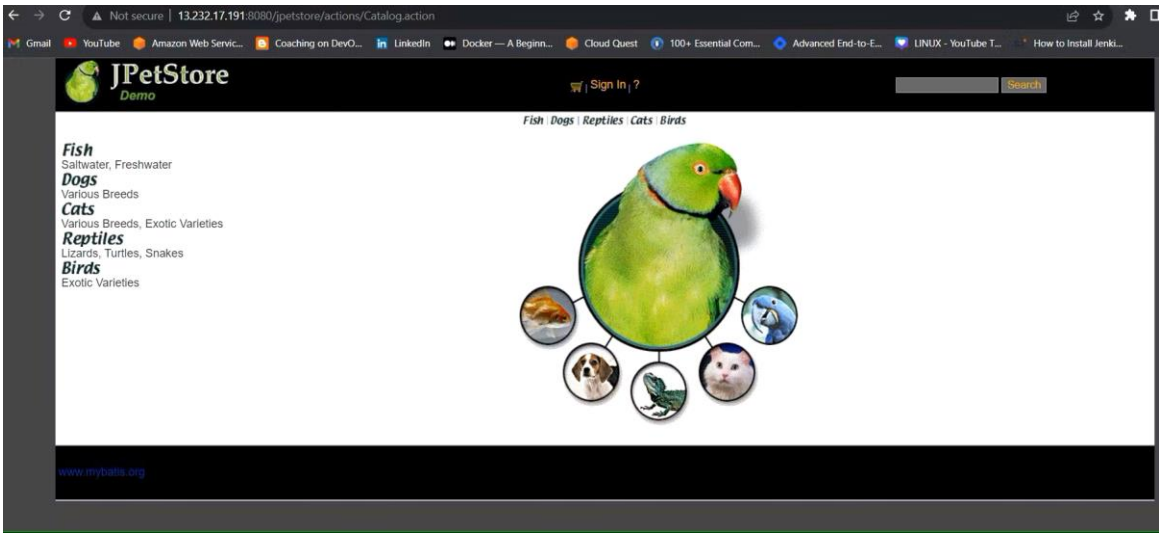
Now, when you do

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



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

You will get this output


Not secure | 13.232.17.191:8080/jpetstore/actions/Catalog.action
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...
JPetStore
Demo
Sign In | ?
Search
Fish | Dogs | Reptiles | Cats | Birds
Fish
Saltwater, Freshwater
Dogs
Various Breeds
Cats
Various Breeds, Exotic Varieties
Reptiles
Lizards, Turtles, Snakes
Birds
Exotic Varieties
www.mybatis.org