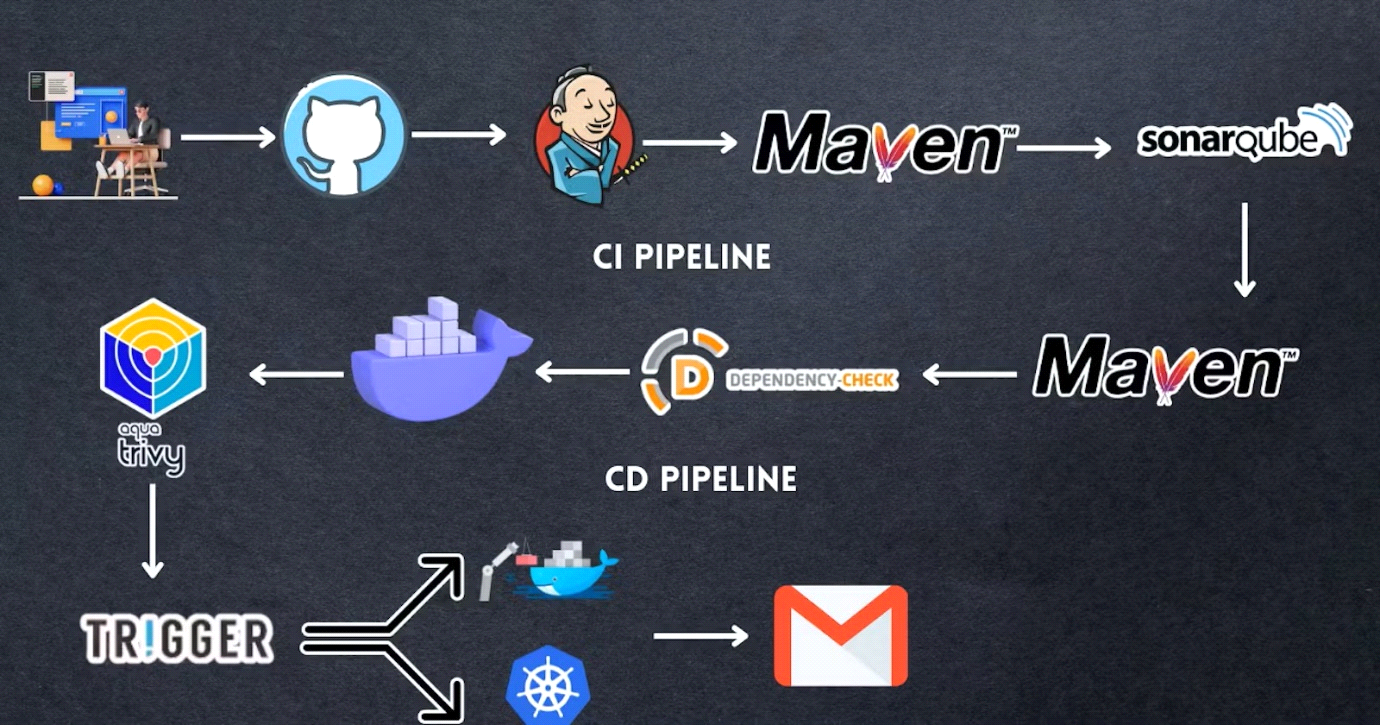
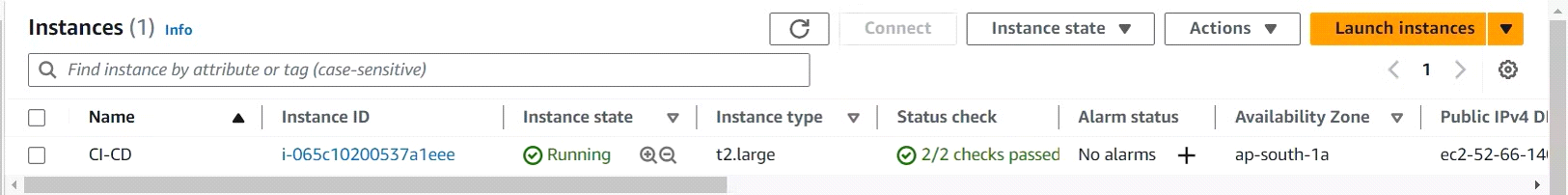
**DEVSECOPS Project: Complete CI-CD (3 tier app)-Pet shop**



STEP 1 — Create an Ubuntu 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.



Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

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

**apt update -y**

**apt install default -jdk**

**apt install maven**

**sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \**

[**https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key**](https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key)

**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**

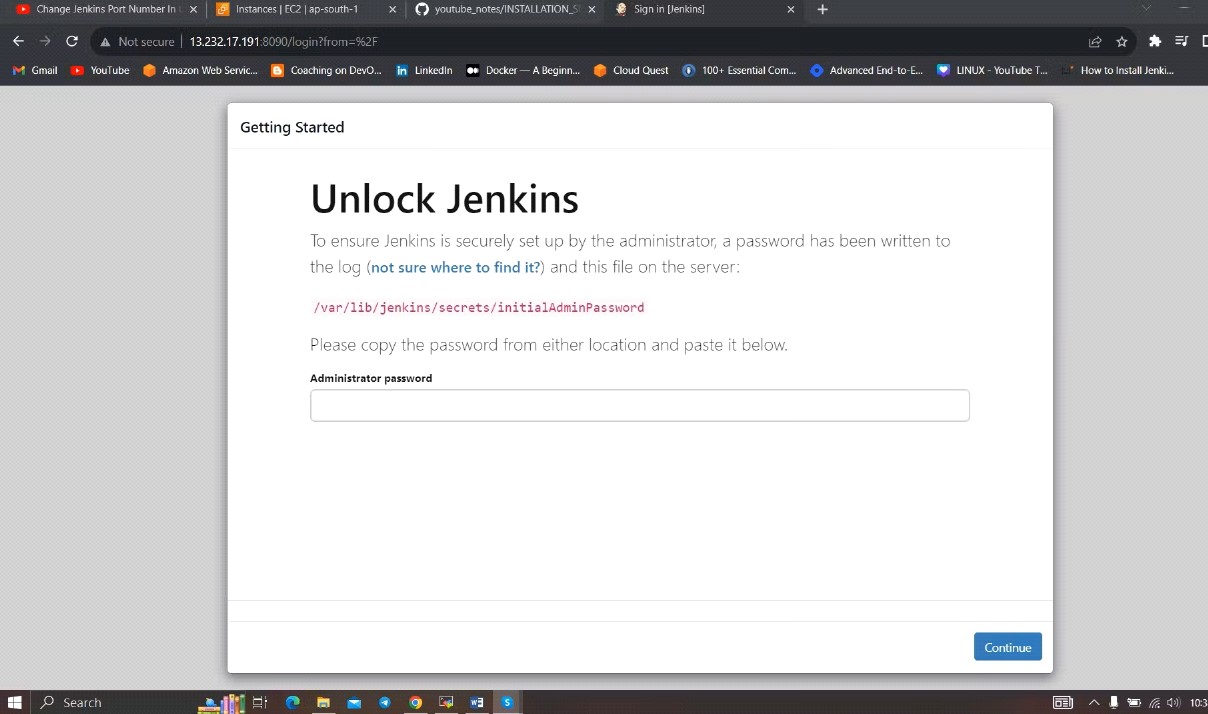
**sudo apt-get install jenkins**

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.

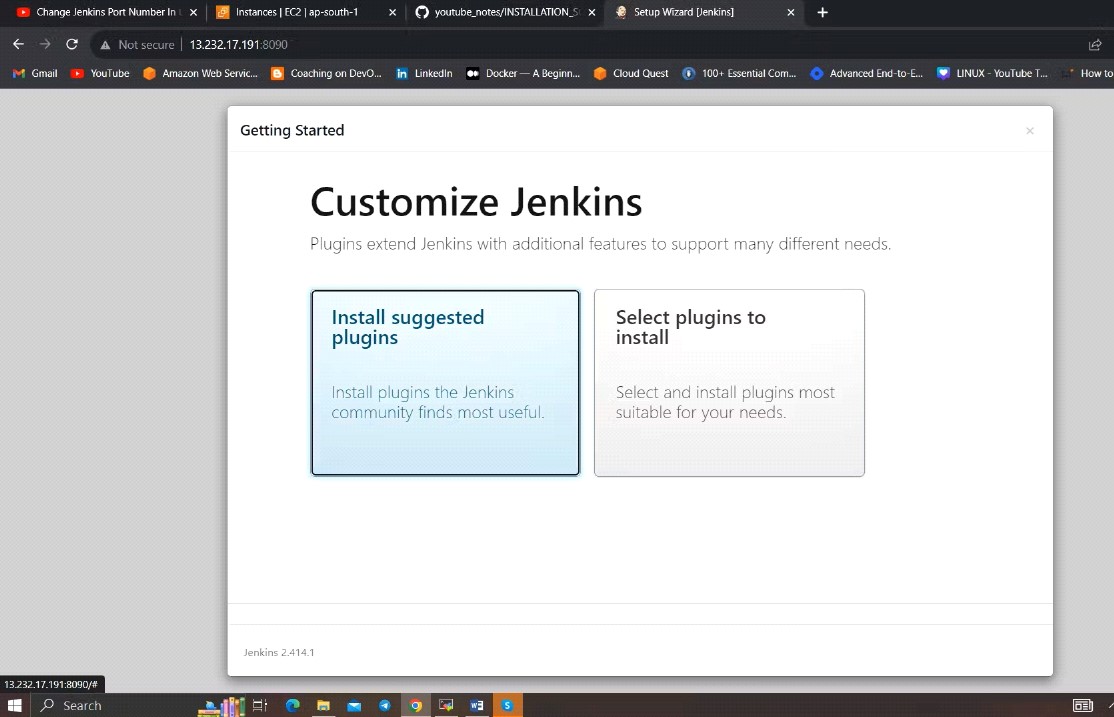
But for this Application case, we are running Jenkins on another port. so change the port to 8090

Now, grab your Public IP Address

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



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

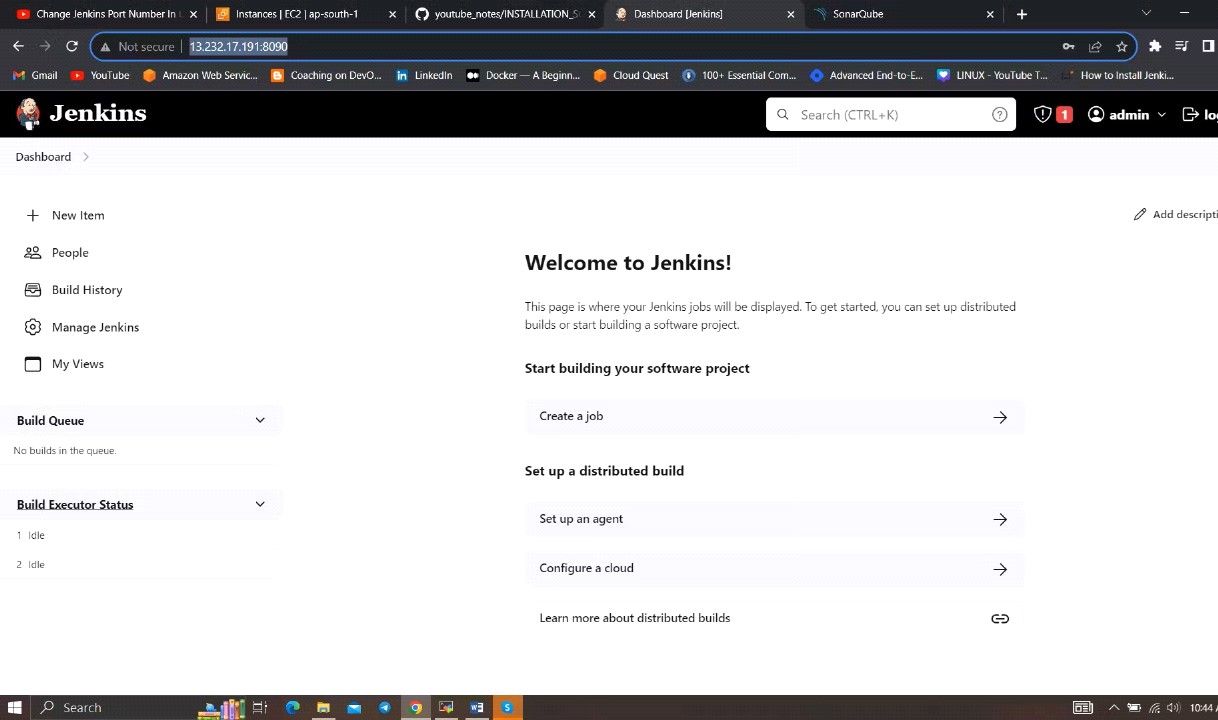


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



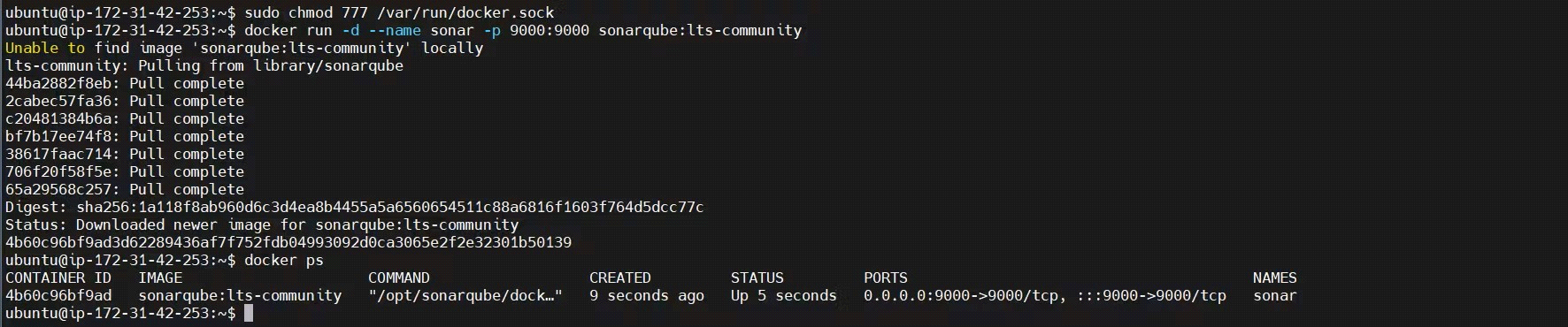
2B — Install Docker

**apt update -y**

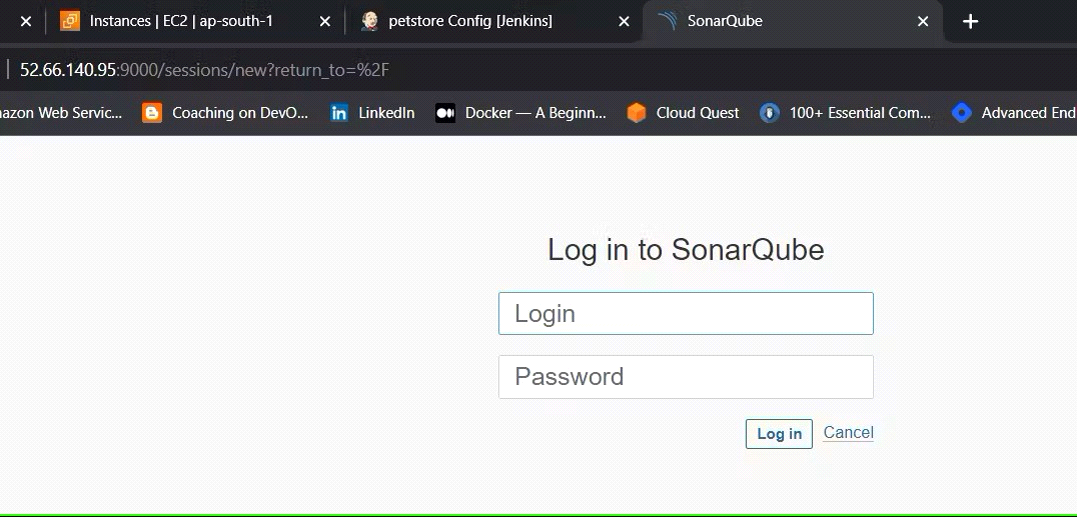
**apt install docker.io -y**

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



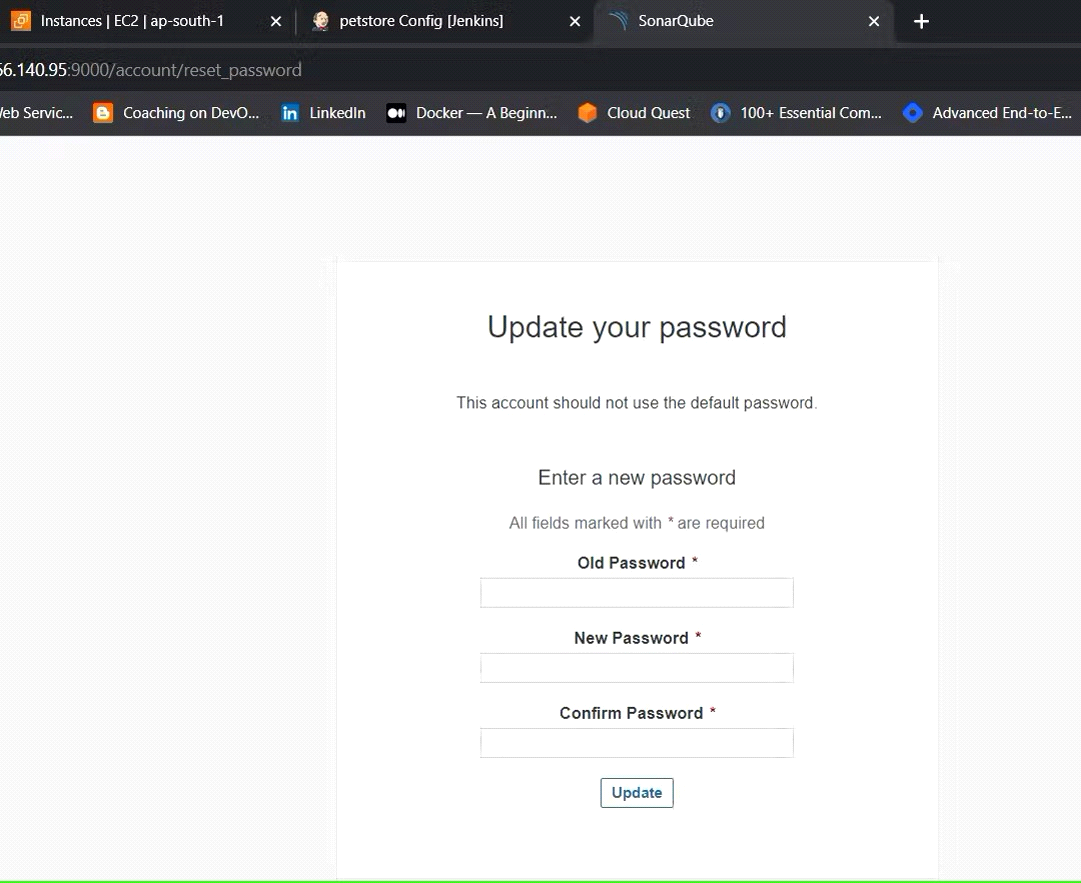
Now our SonarQube is up and running



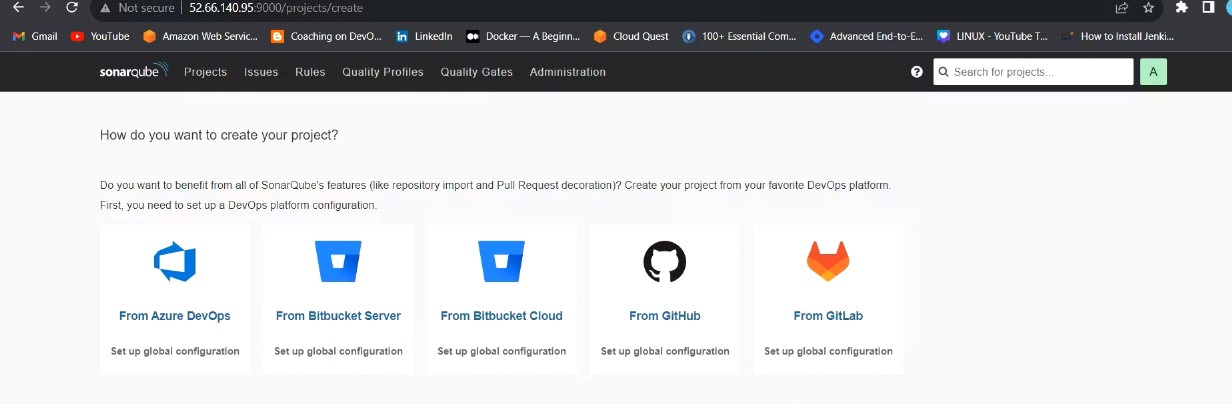
Enter username and password, click on login and change password

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**](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**](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

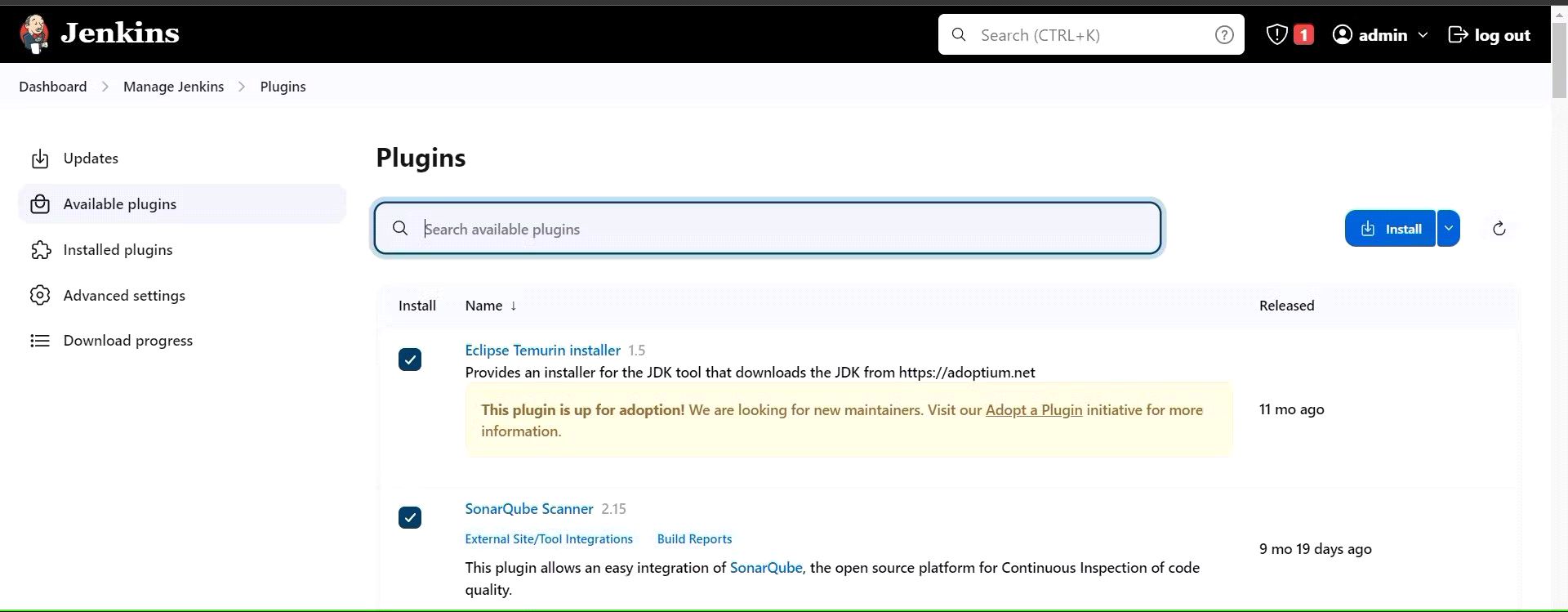
3A — Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

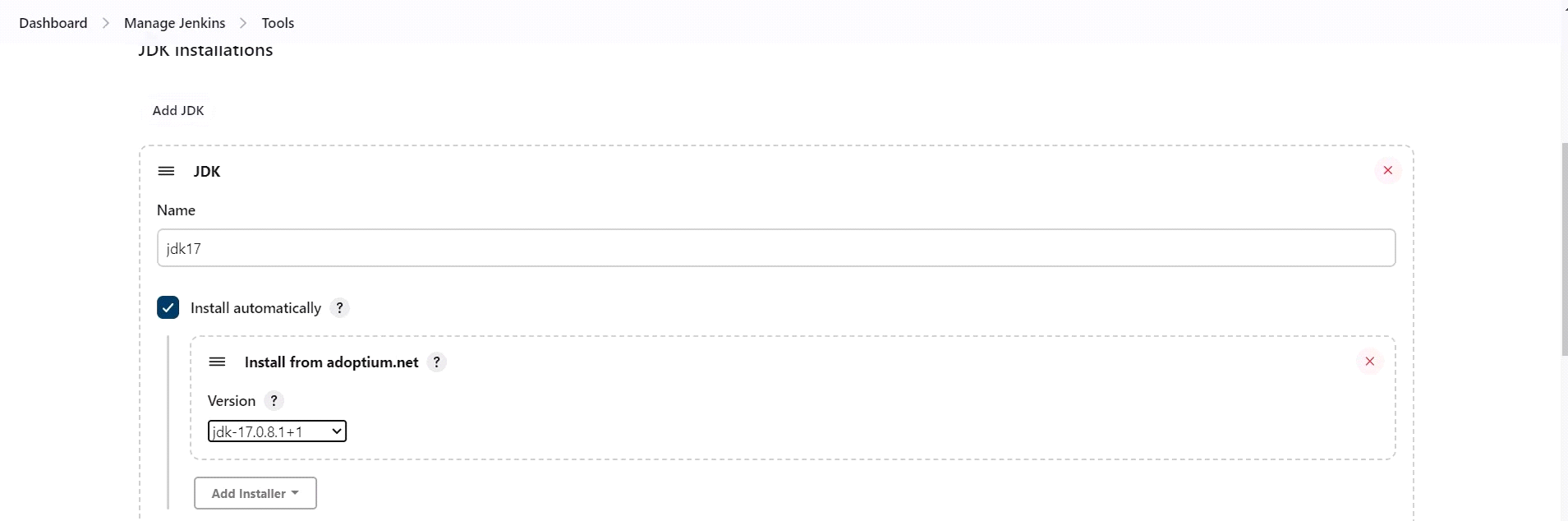
1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)



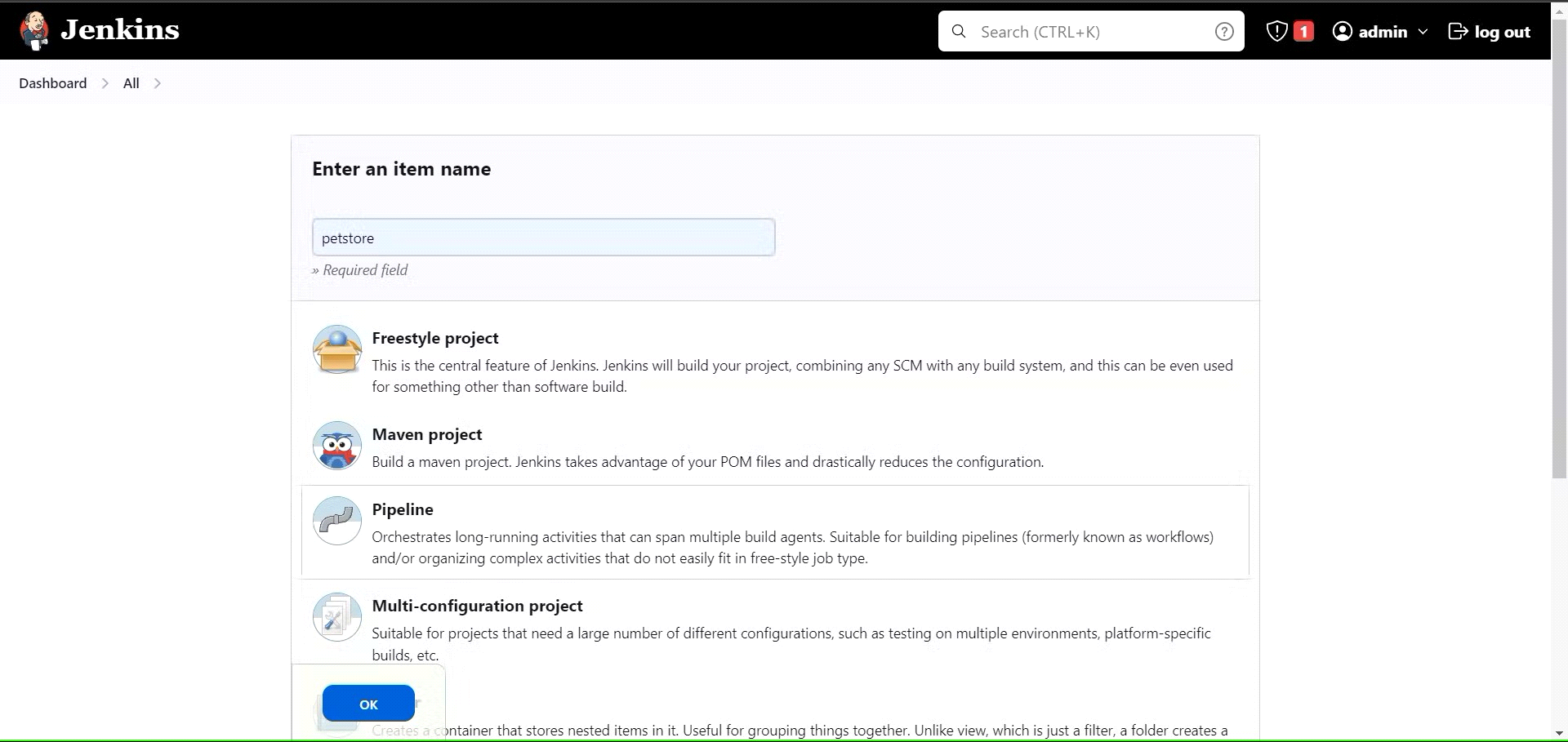
3B — Configure Java and Maven in Global Tool Configuration

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/DasariSankirthana/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

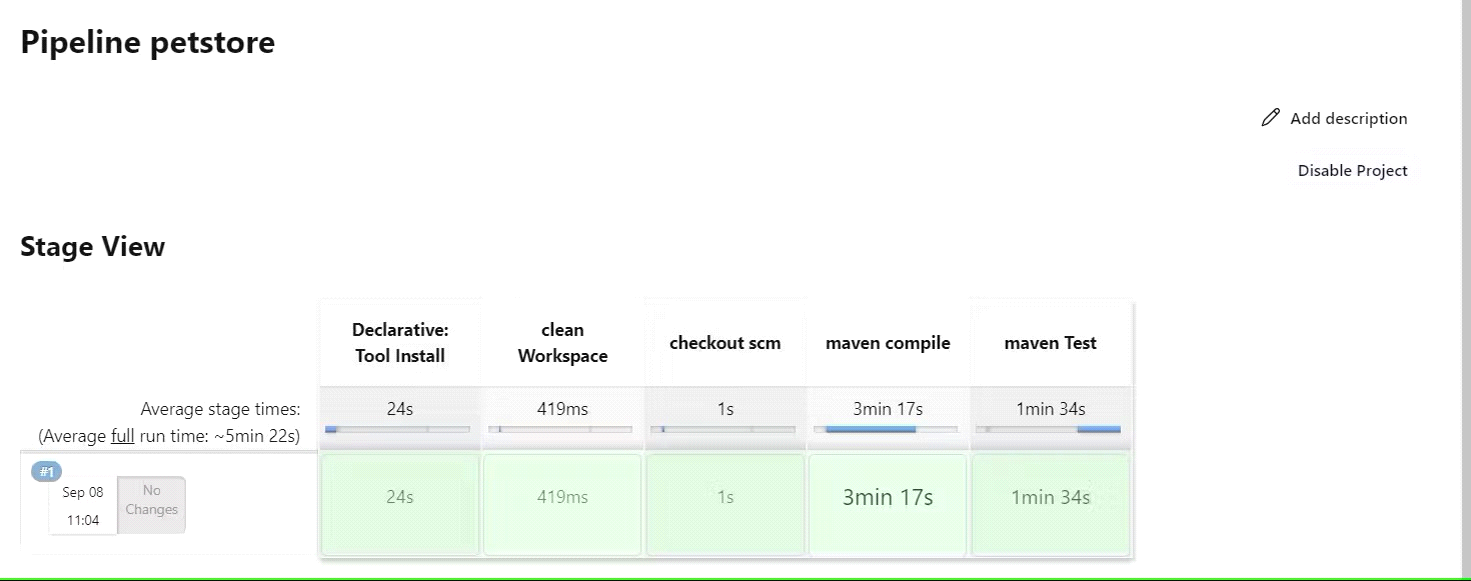
}

}

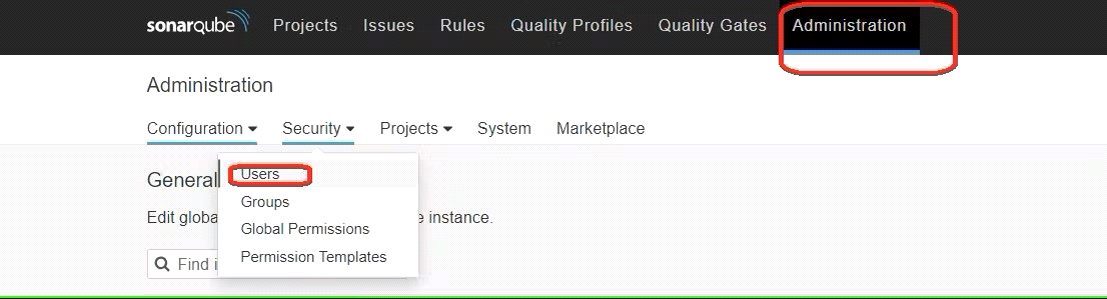
}

}

The stage view would look like this,

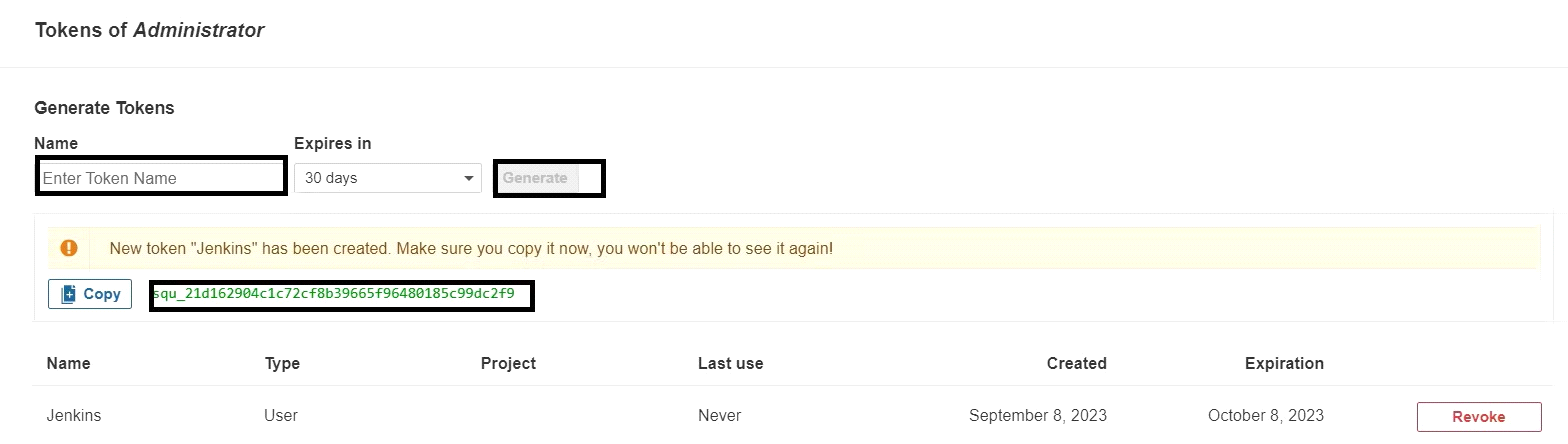


Step 4 — Configure Sonar Server in Manage Jenkins





Create a token with a name and generate

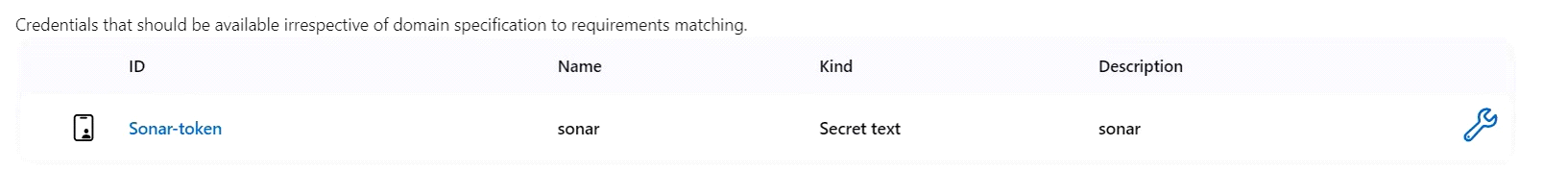


copy Token

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.

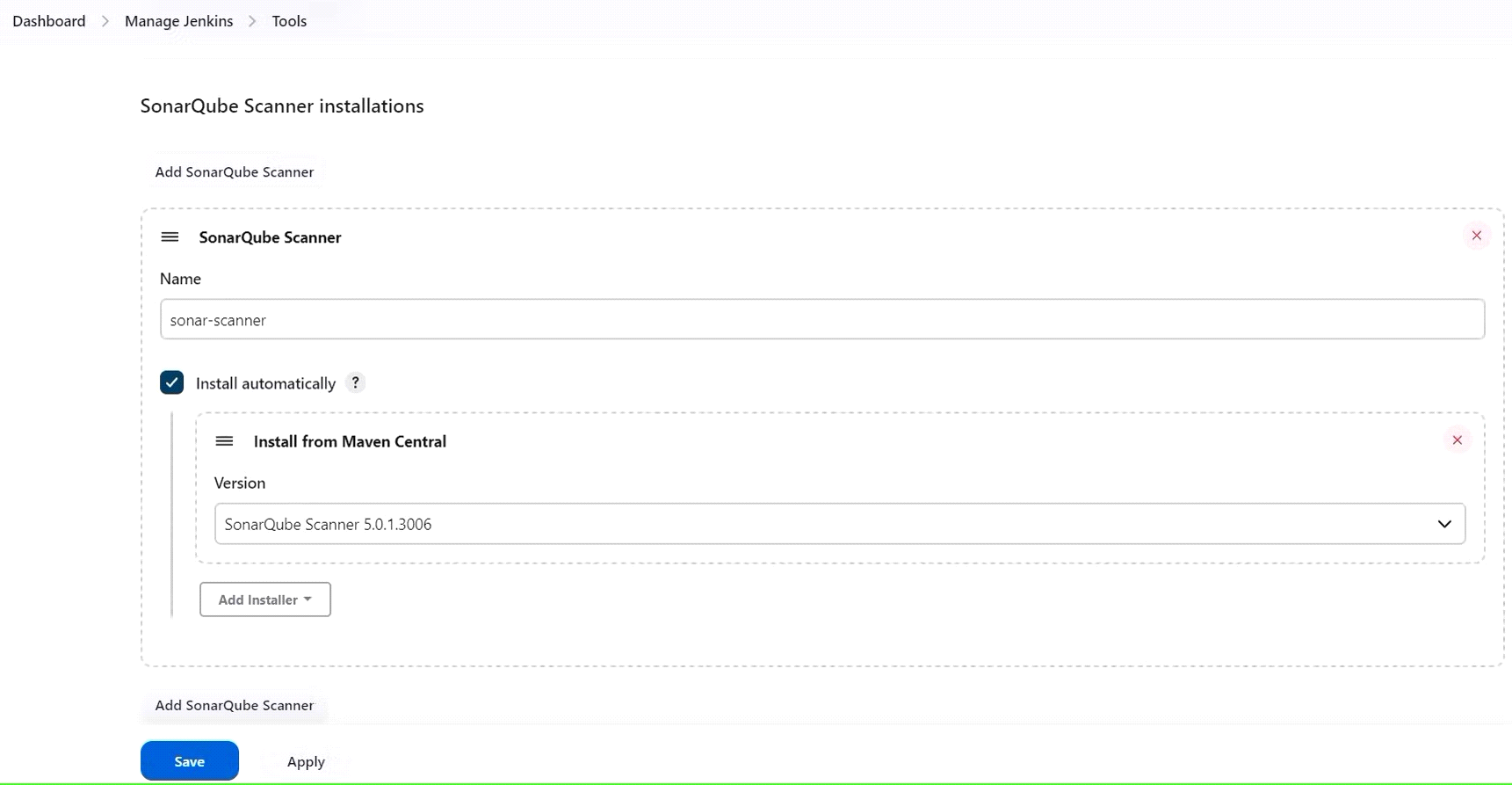


Click on Apply and Save

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

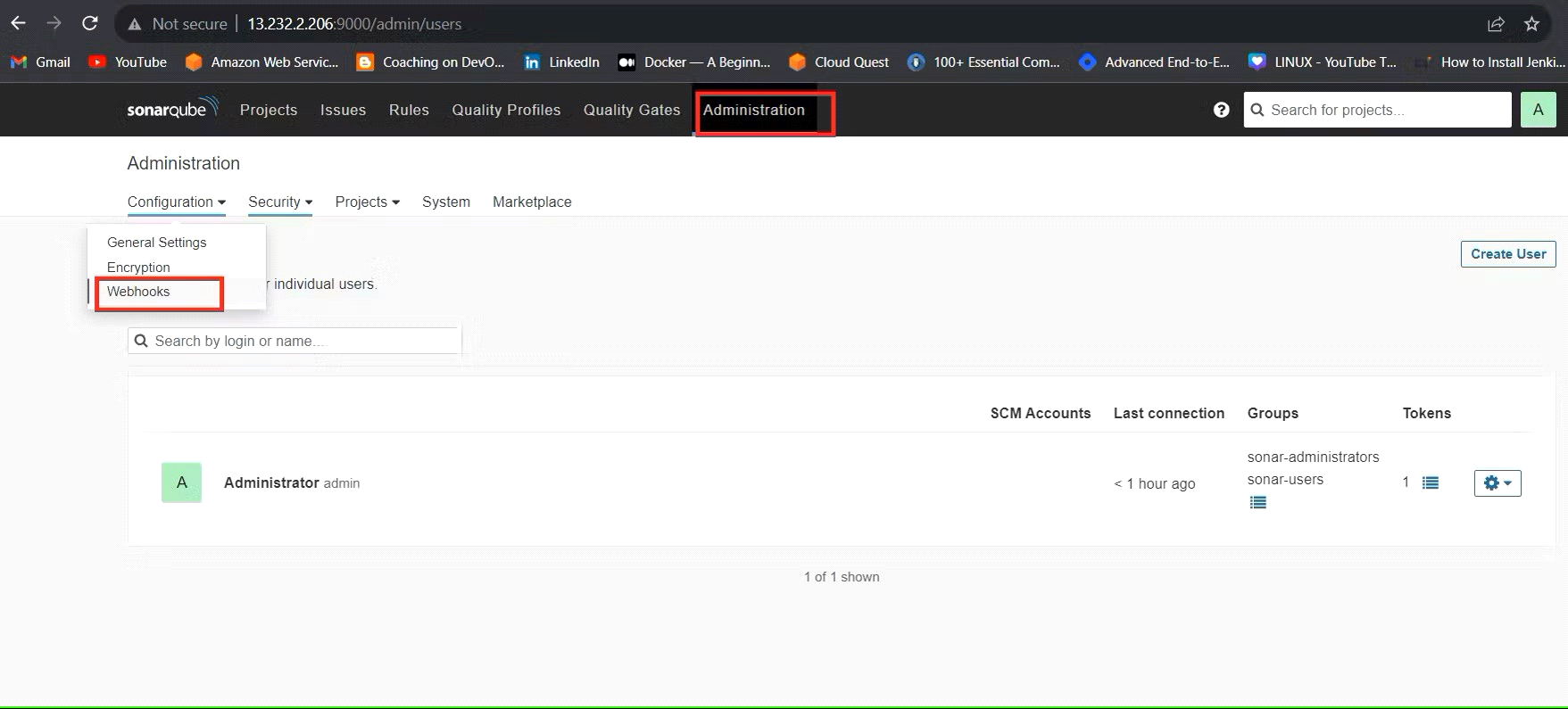
**Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

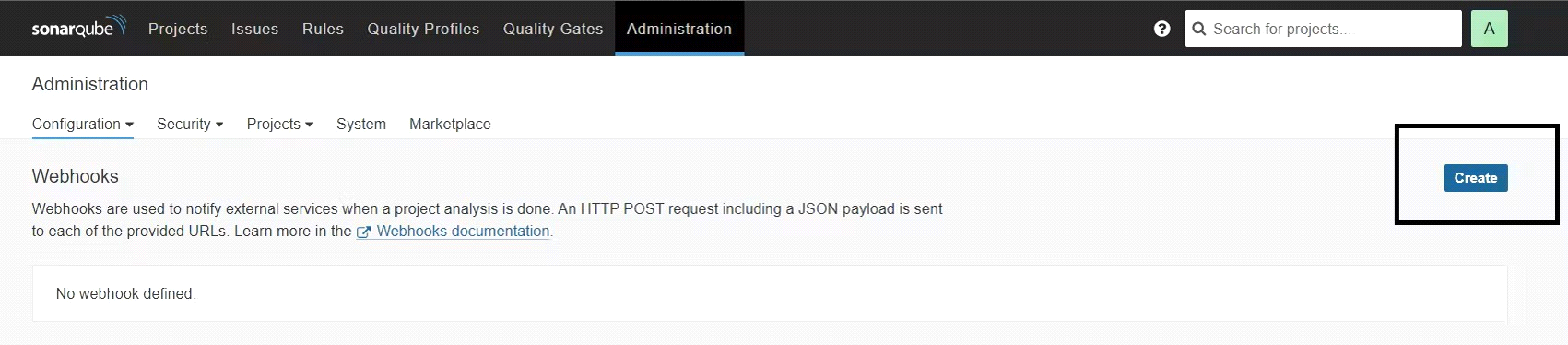


In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



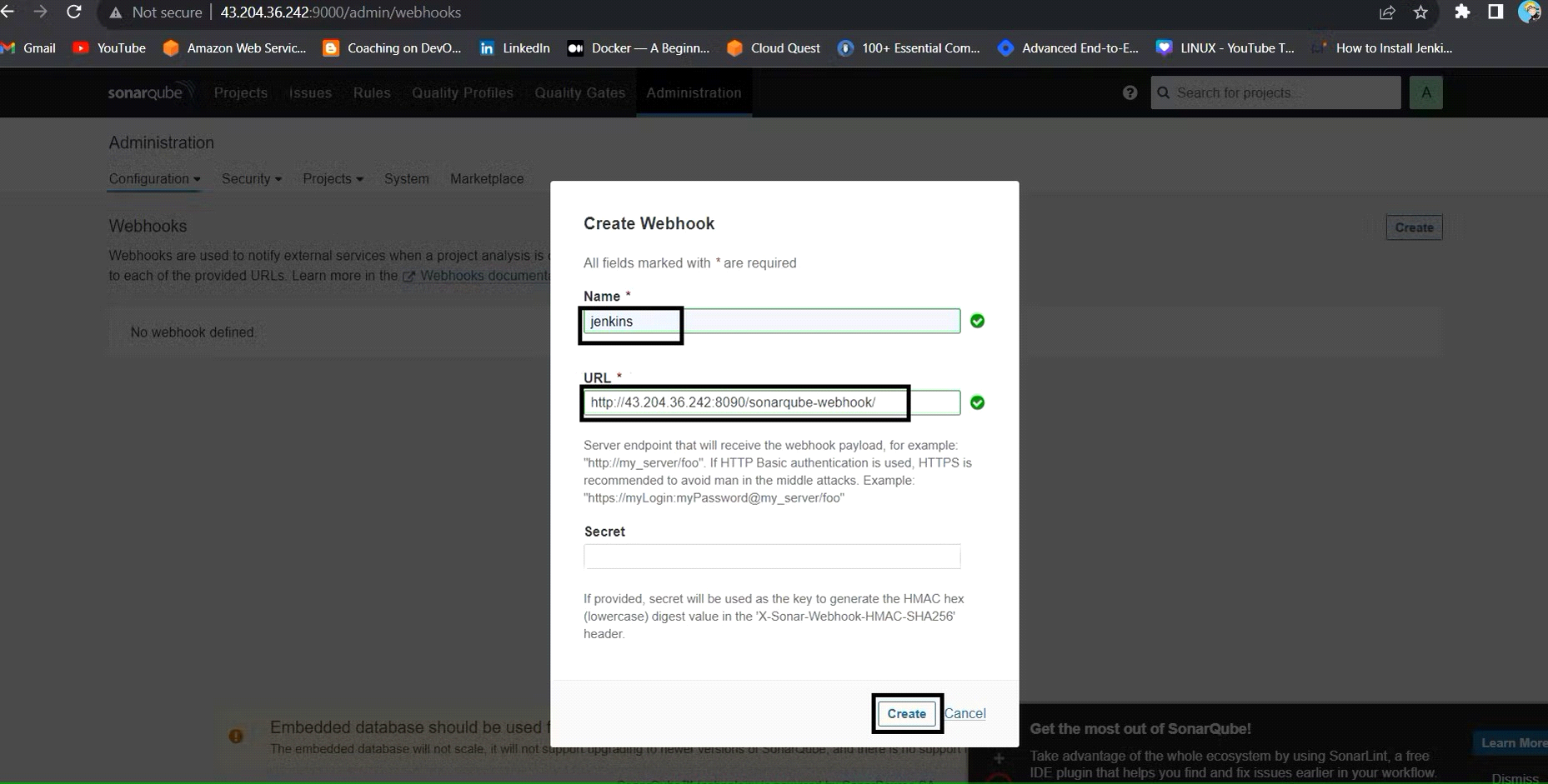
Click on Create



Add details

#in url section of quality gate

<[http://jenkins-public-ip:8090](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 {

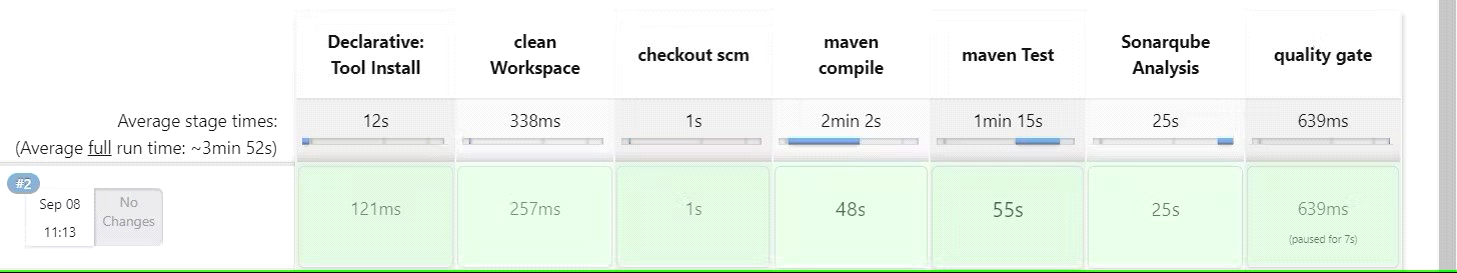
waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

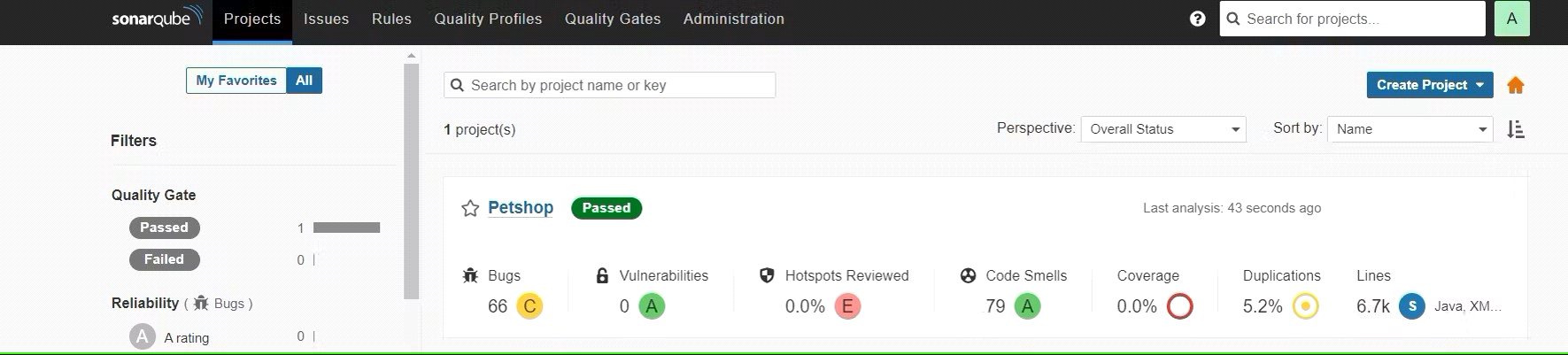
}

}

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



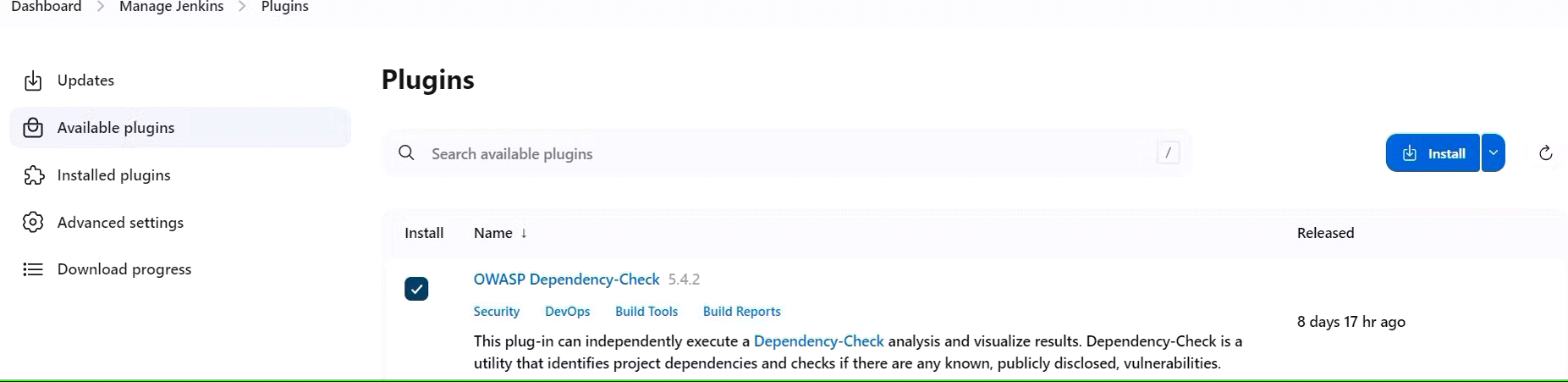
To see the report, you can go to Sonarqube Server and go to Projects.



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.

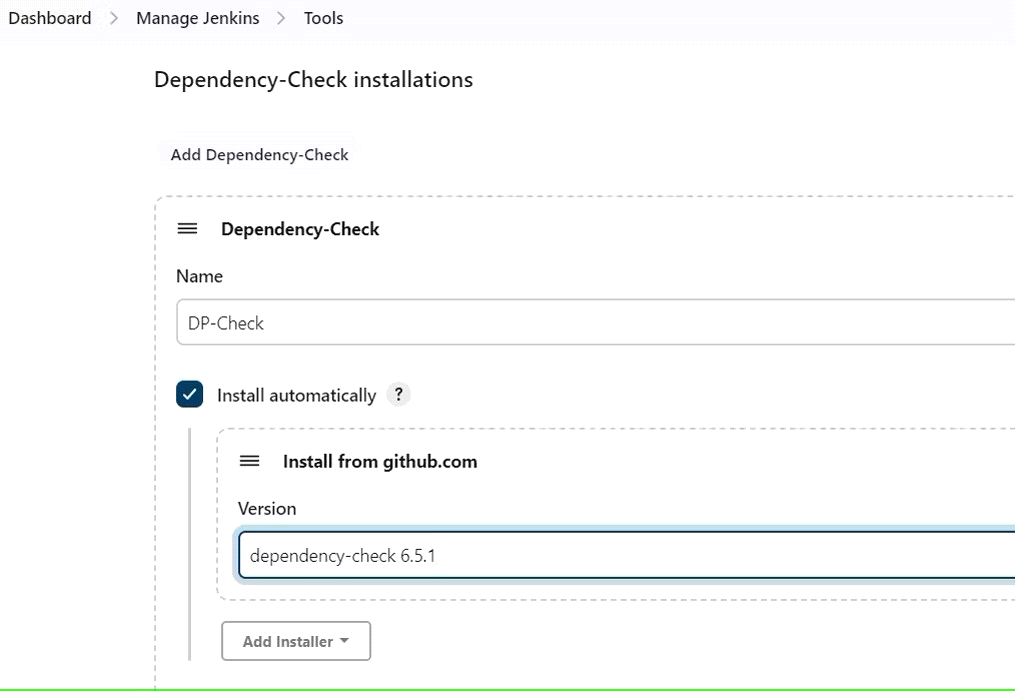
Step 5 — Install OWASP Dependency Check Plugins

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{

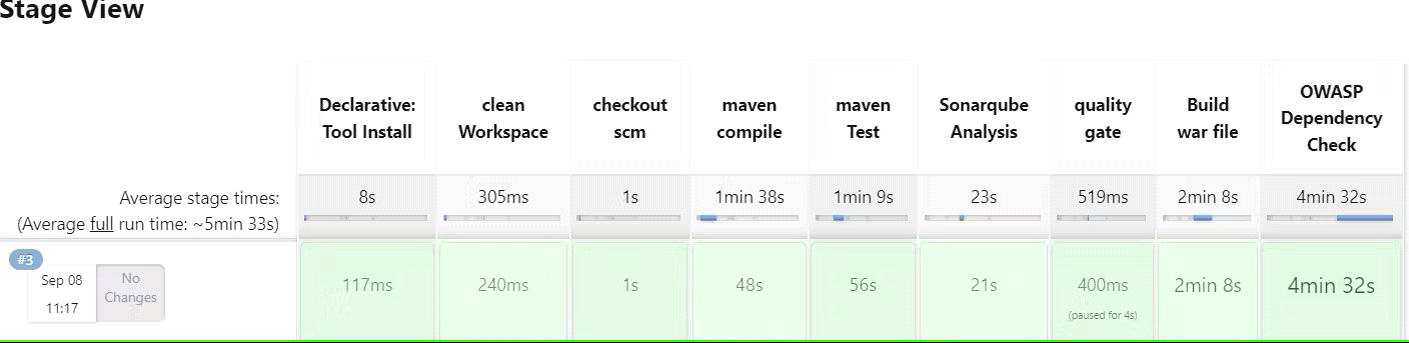
dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

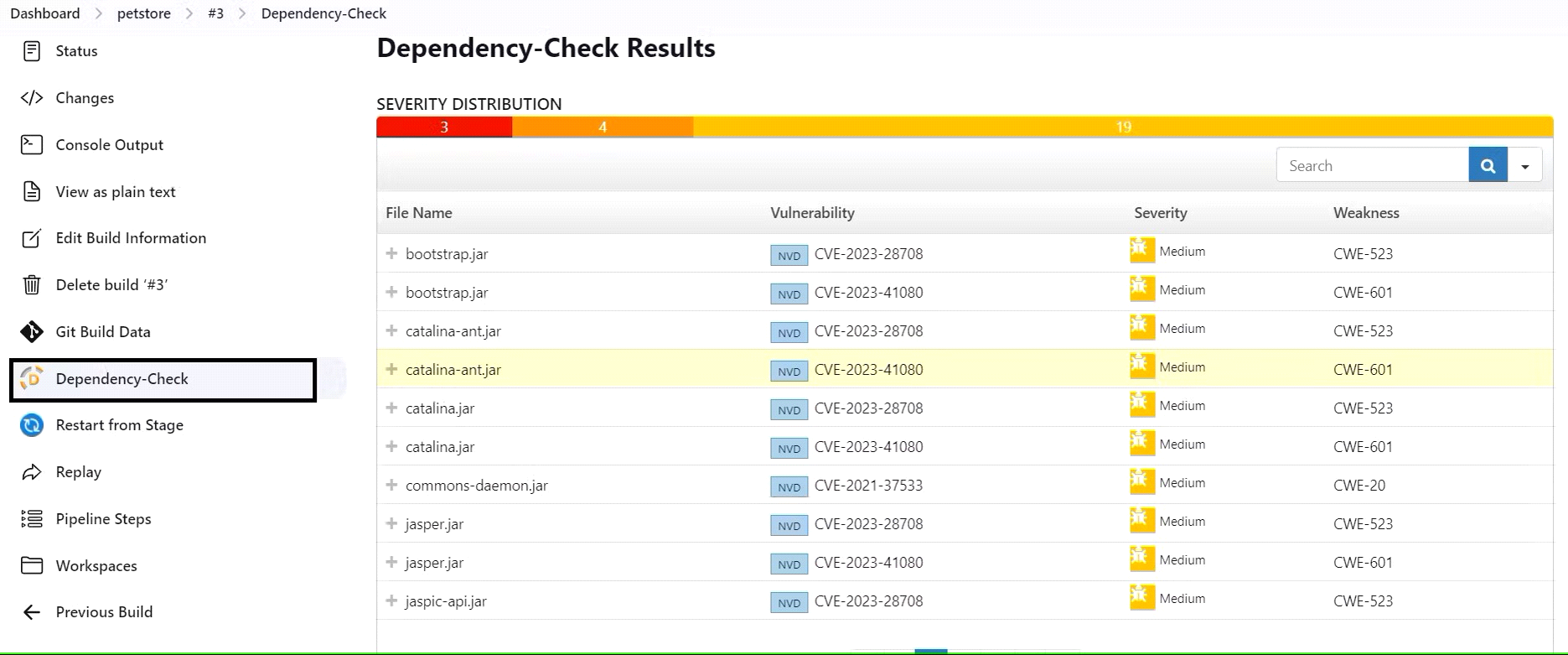
}

}

The stage view would look like this,



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

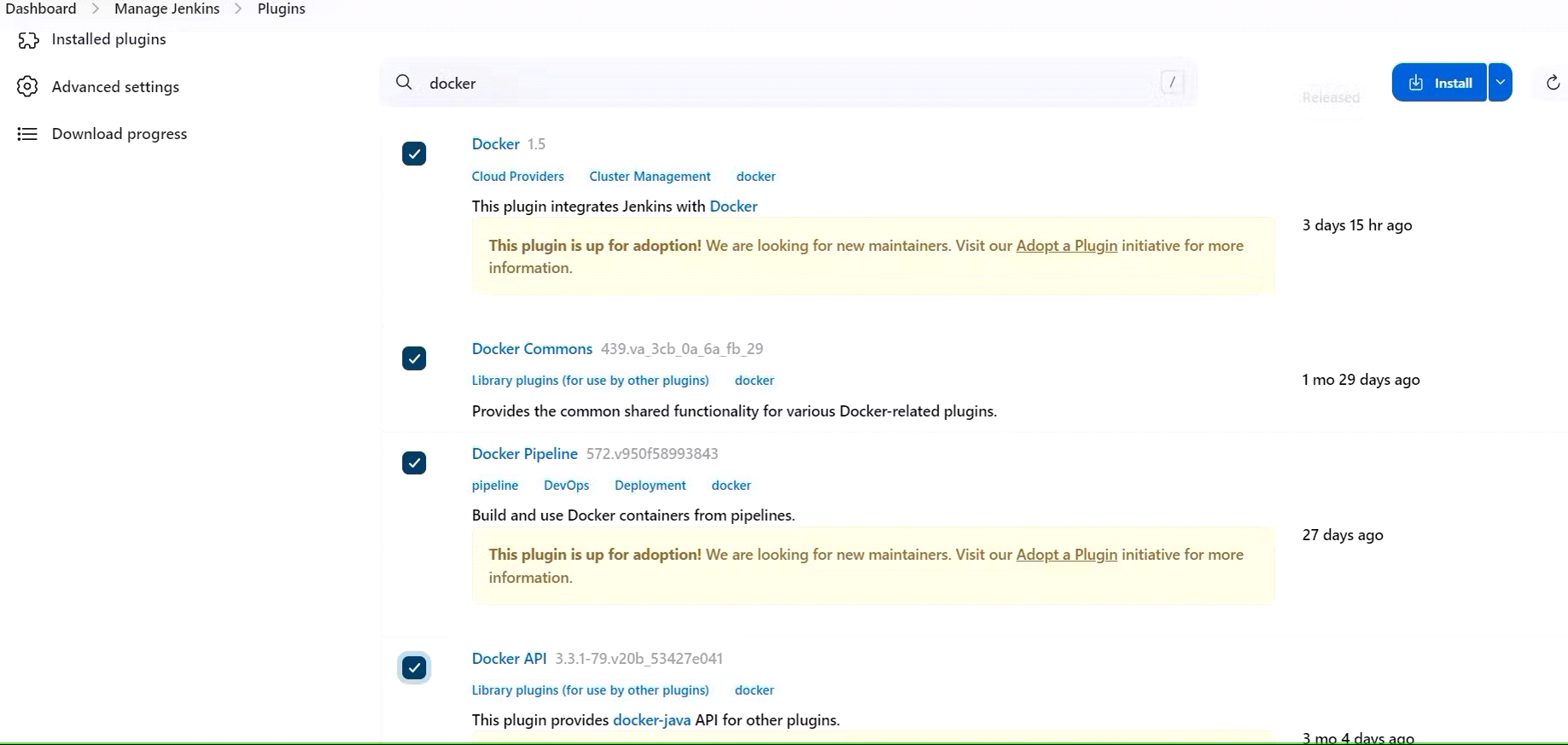
Docker

Docker commons

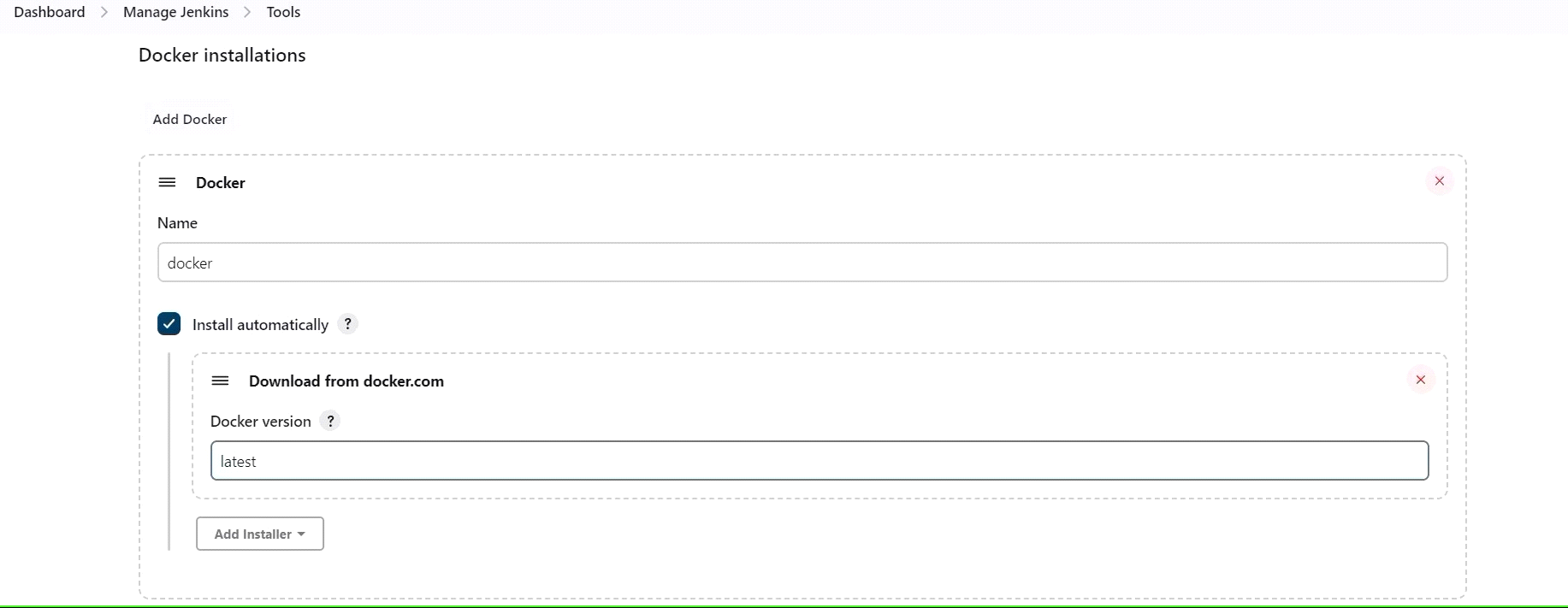
Docker Pipeline

Docker-build-setup

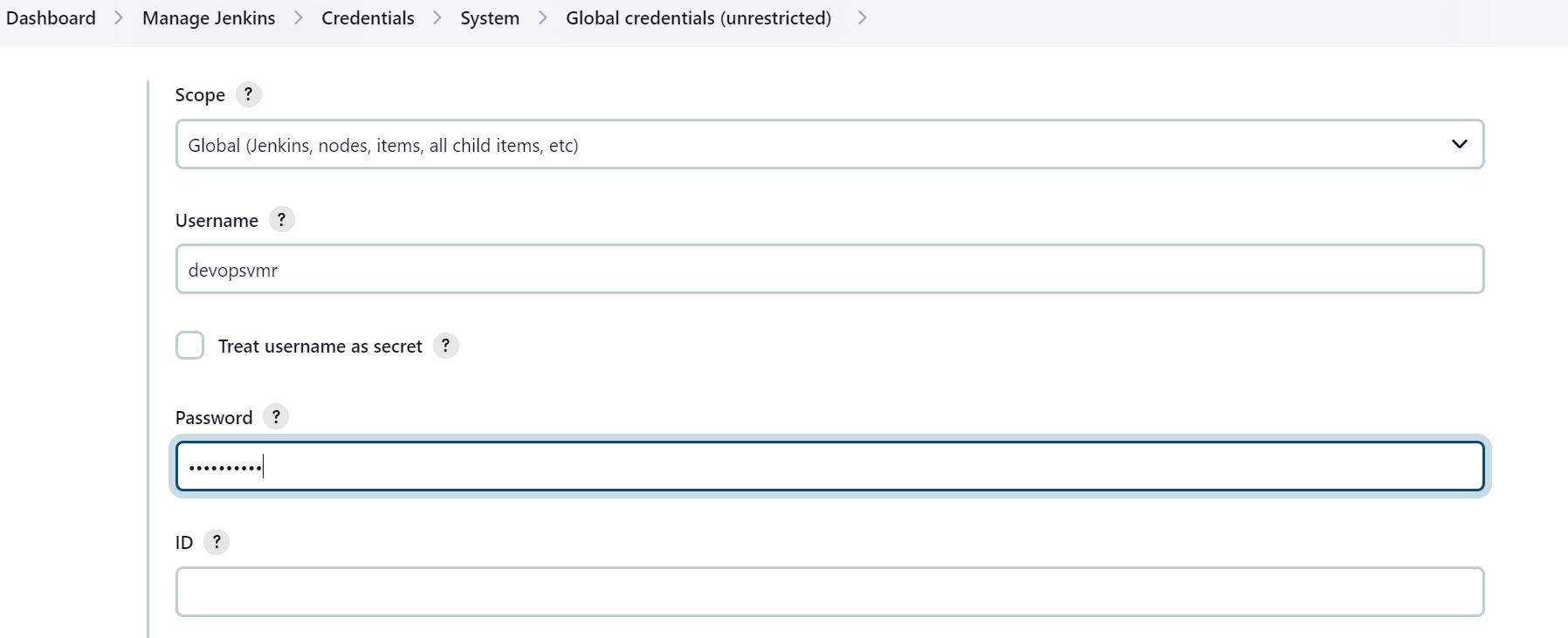
and click on install without restart



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 devopsvmr/petshop:latest"

sh "docker push devopsvmr/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image devopsvmr/petshop:latest > trivy.txt"

}

}

stage ('Deploy to container'){

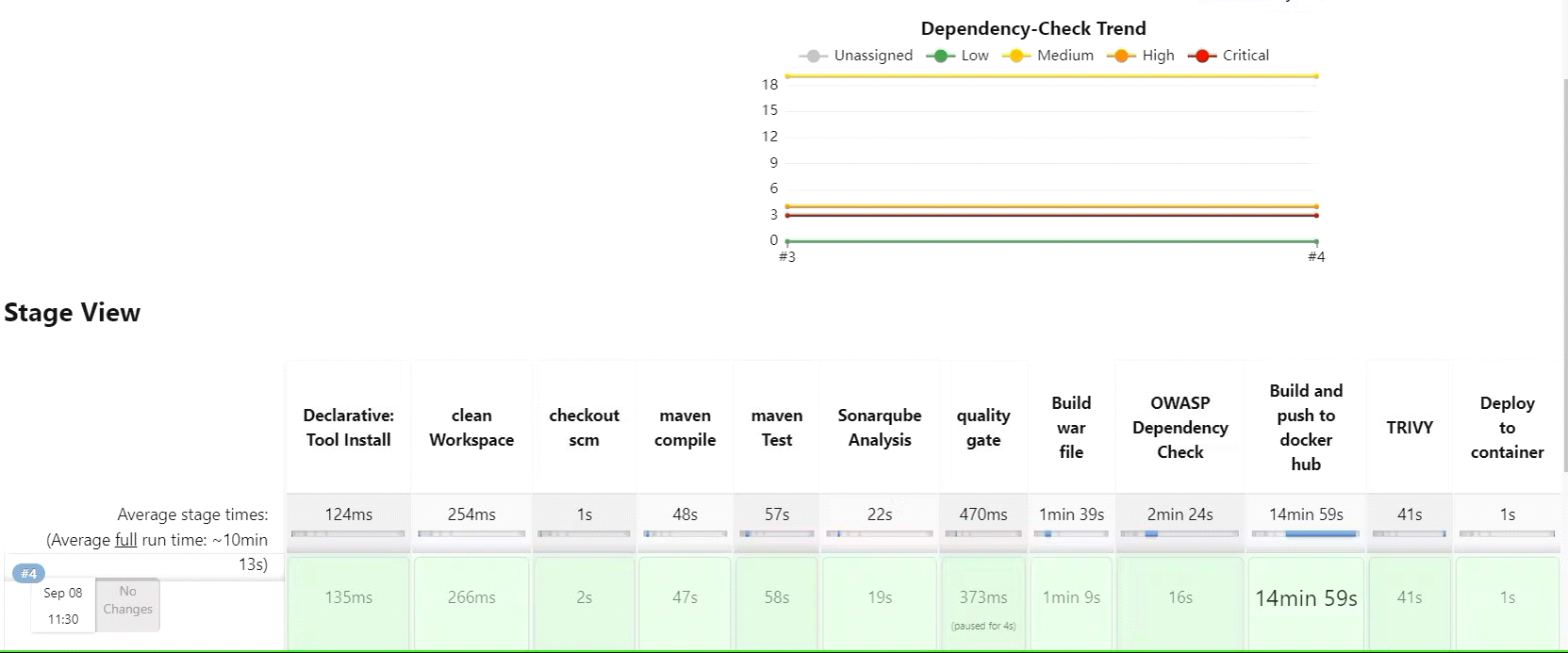
steps{

sh 'docker run -d --name pet1 -p 8080:8080 devopsvmr/petshop:latest'

}

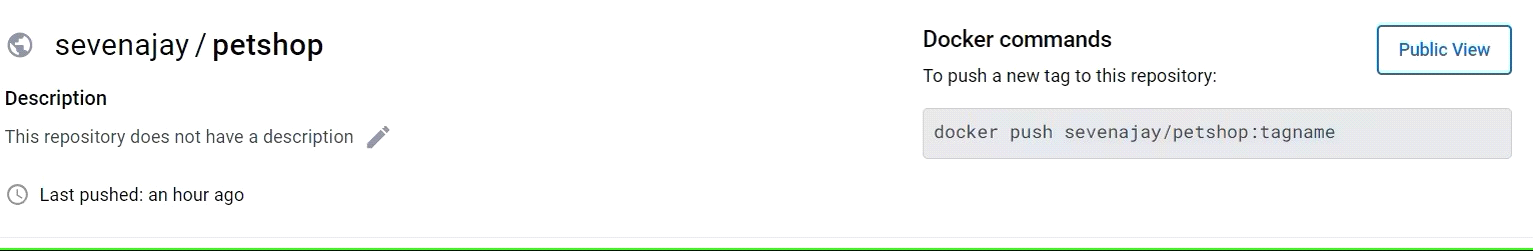
}

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

