

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

* [STEP1:Create an Ubuntu(22.04) T2 Large Instance](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step1create-an-ubuntu2204-t2-large-instance)
* [Step 2 — Install Jenkins, Docker and Trivy](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-2-install-jenkins-docker-and-trivy)
  + [2A — To Install Jenkins](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-2a-to-install-jenkins)
  + [2B — Install Docker](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-2b-install-docker)
  + [2C — Install Trivy](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-2c-install-trivy)
* [Step 3 — Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-3-install-plugins-like-jdk-sonarqube-scanner-maven-owasp-dependency-check)
  + [3A — Install Plugin](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-3a-install-plugin)
  + [3B — Configure Java and Maven in Global Tool Configuration](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-3b-configure-java-and-maven-in-global-tool-configuration)
  + [3C — Create a Job](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-3c-create-a-job)
* [Step 4 — Configure Sonar Server in Manage Jenkins](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-4-configure-sonar-server-in-manage-jenkins)
  + [Step 5 — Install OWASP Dependency Check Plugins](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-5-install-owasp-dependency-check-plugins)
* [Step 6 — Docker Image Build and Push](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-6-docker-image-build-and-push)
* [Step 8 — Kuberenetes Setup](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-8-kuberenetes-setup)
  + [Kubectl on Jenkins to be installed](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-kubectl-on-jenkins-to-be-installed)
  + [Part 1 ----------Master Node------------](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-part-1-master-node)
  + [----------Worker Node------------](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-worker-node)
  + [Part 2 ------------Both Master & Node ------------](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-part-2-both-master-andamp-node)
  + [Part 3 --------------- Master ---------------](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-part-3-master)
  + [----------Worker Node------------](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-worker-node-1)
* [Configuring mail server in Jenkins ( Gmail )](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-configuring-mail-server-in-jenkins-gmail)
* [STEP9:Access from a Web browser with](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step9access-from-a-web-browser-with)
* [Step 10: Terminate instances.](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-step-10-terminate-instances)
* [Complete Pipeline](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-complete-pipeline)
* [Trigger code](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-trigger-code)
* [CI-petshop-pipeline](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-ci-petshop-pipeline)
* [CD-petshop-pipeline](https://mrcloudbook.hashnode.dev/devsecops-project-complete-ci-cd-3-tier-app-petshop#heading-cd-petshop-pipeline)

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 and Push

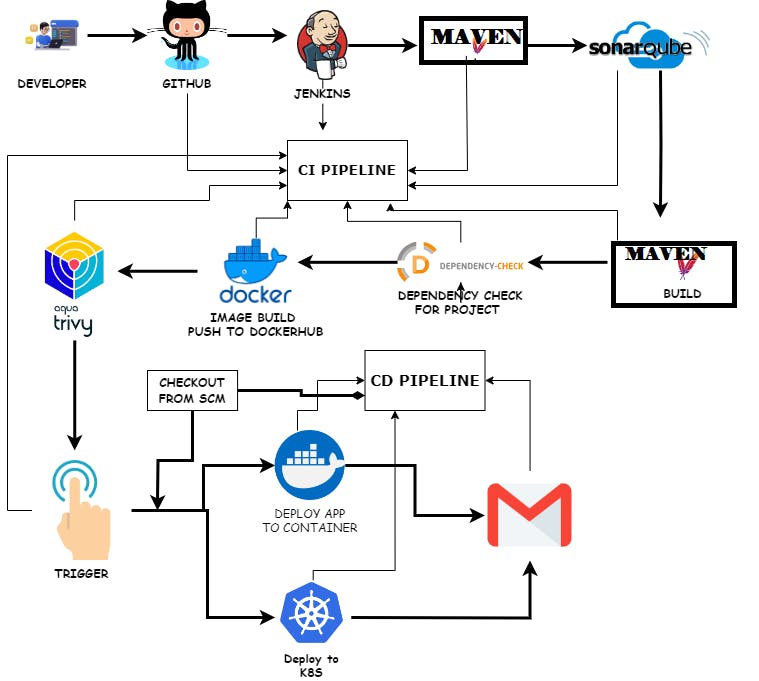
Step 7 — Deploy the image using Docker

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

Step 9 — Access the Real World Application

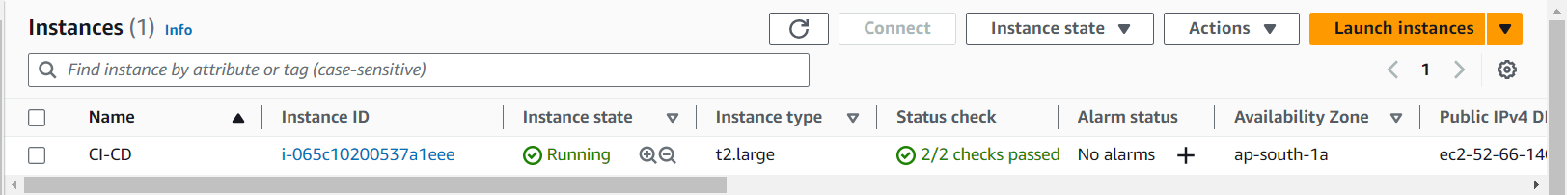
Step 10 — Terminate the AWS EC2 Instances.

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



**STEP1:Create an Ubuntu(22.04) T2 Large Instance**

Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group and open all ports (not best case to open all ports but just for learning purposes it's okay).



**Step 2 — Install Jenkins, Docker and Trivy**

**2A — To Install Jenkins**

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

vi jenkins.sh

#!/bin/bash

sudo apt update -y

#sudo apt upgrade -y

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

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

sudo apt update -y

sudo apt install temurin-17-jdk -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status jenkins

sudo chmod 777 jenkins.sh

./jenkins.sh # this will 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

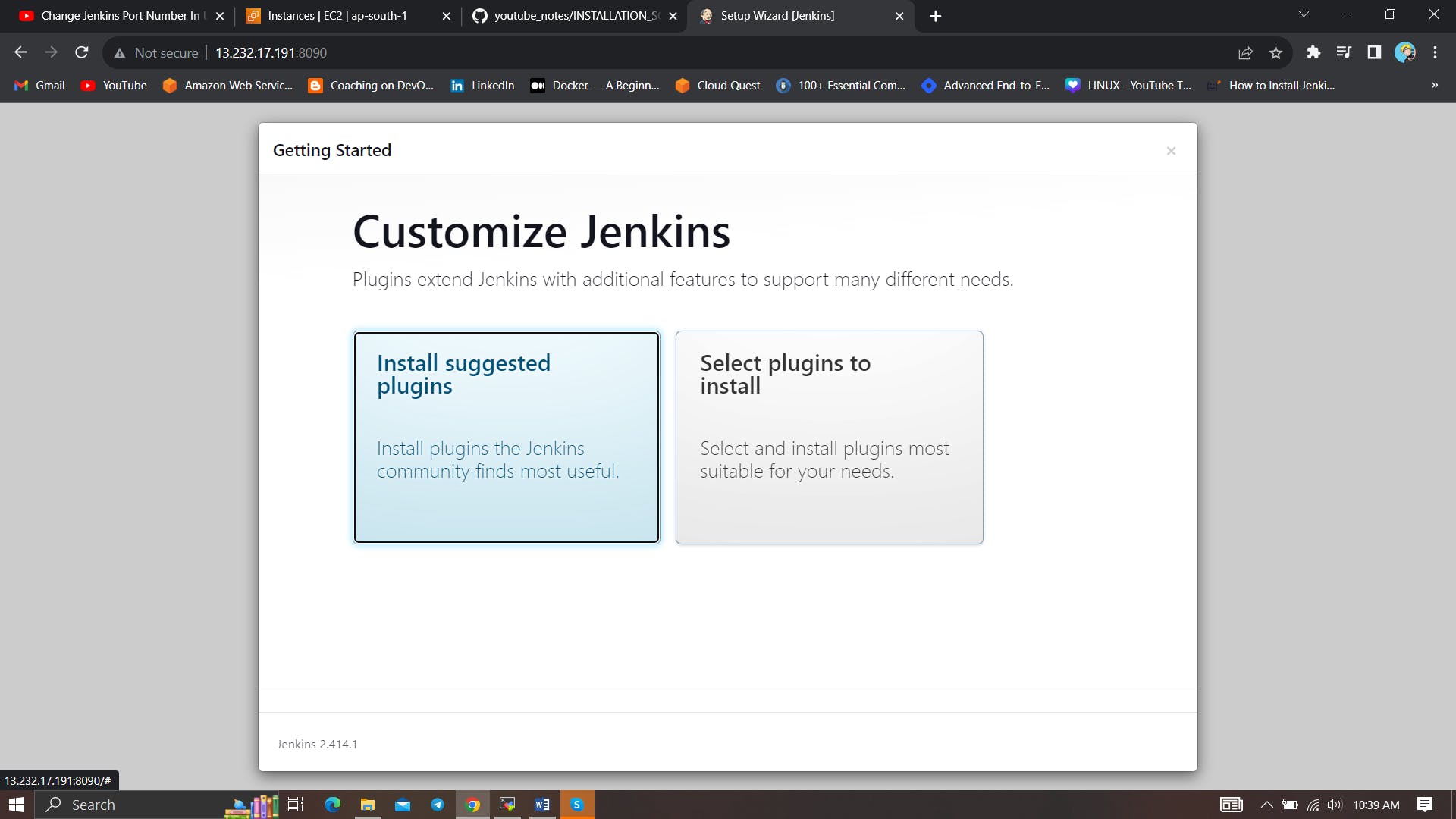
Now, grab your Public IP Address

<EC2 Public IP Address:8090>

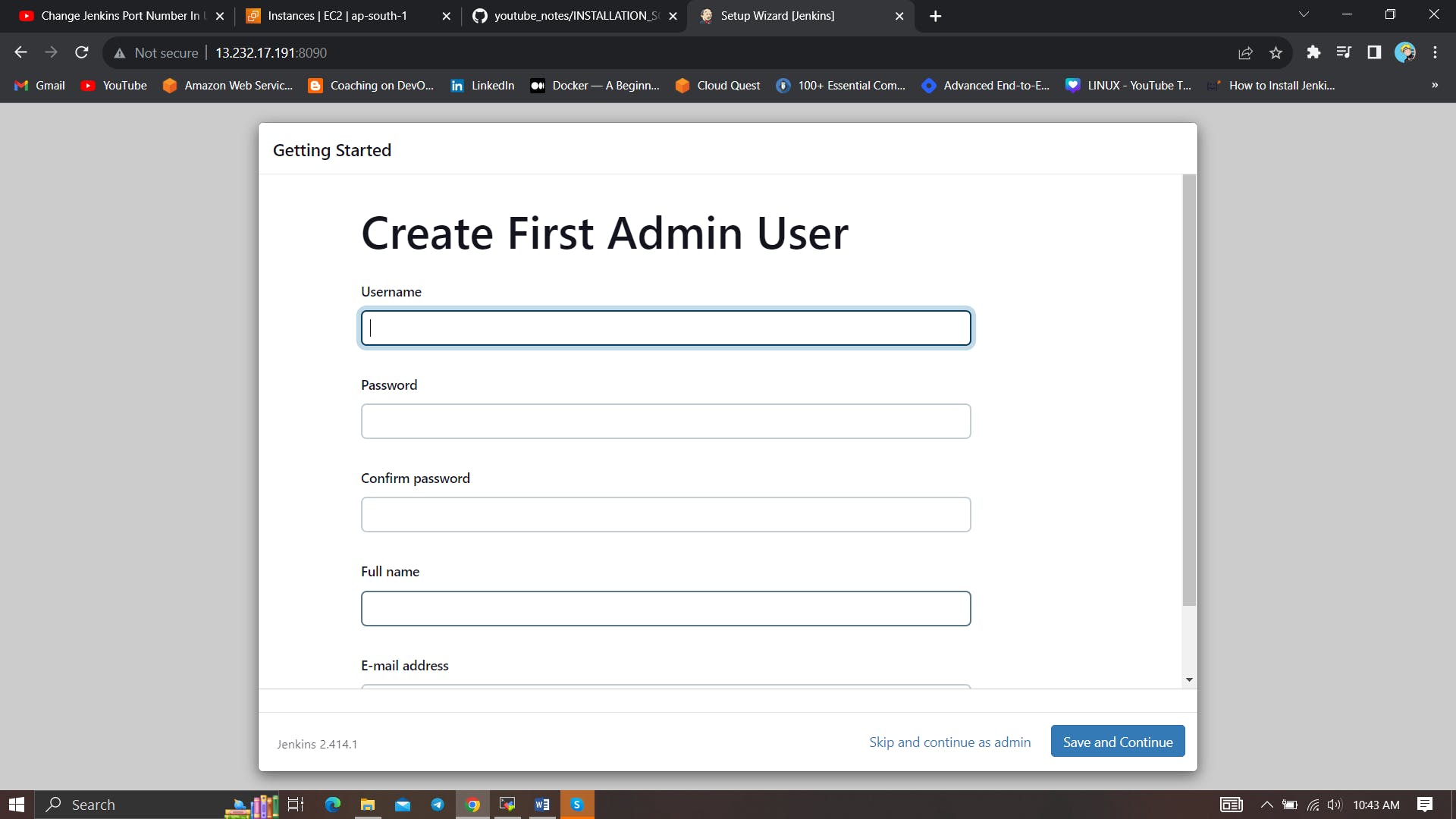
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



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

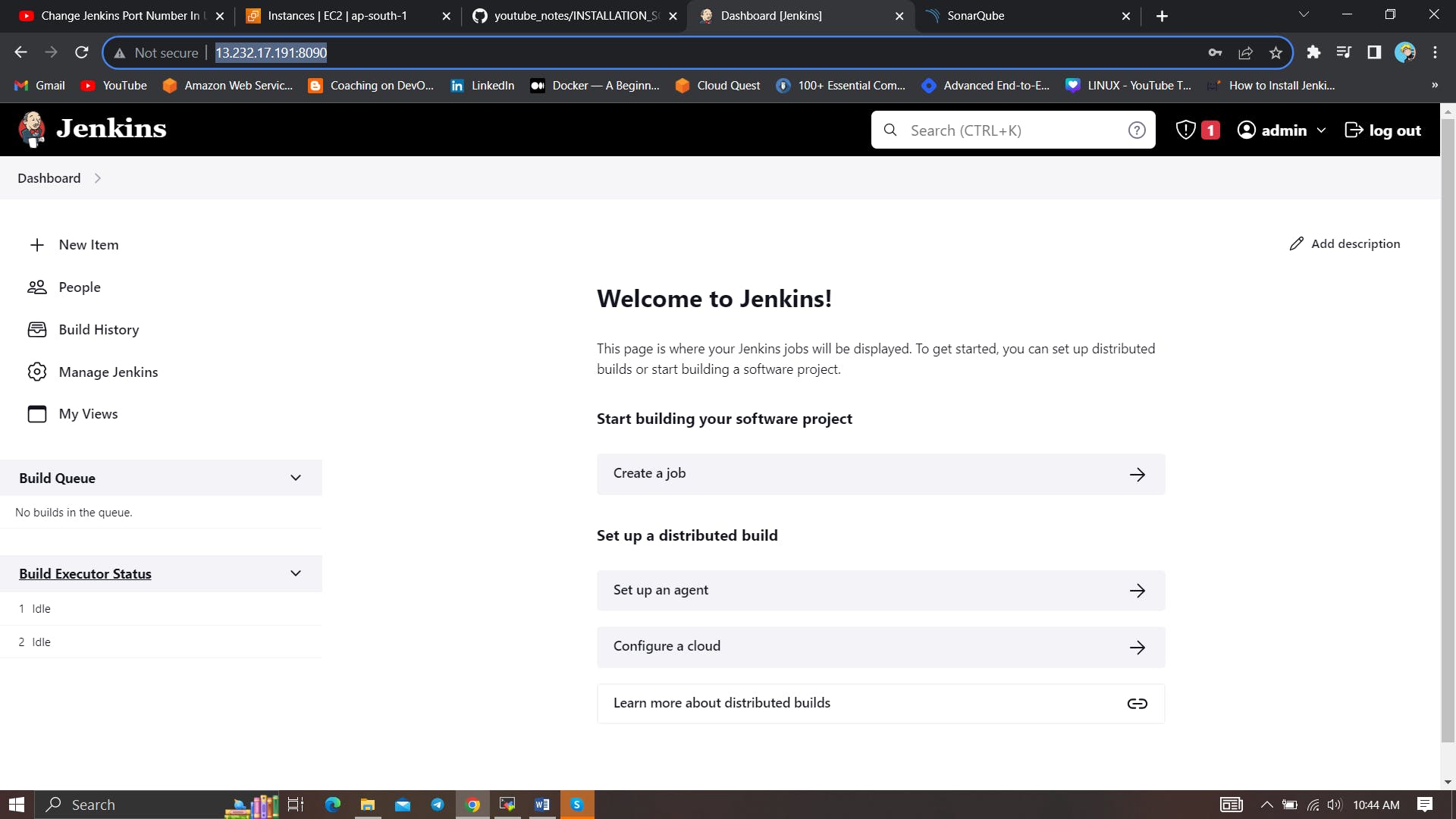


Jenkins will now get installed and install all the libraries.



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

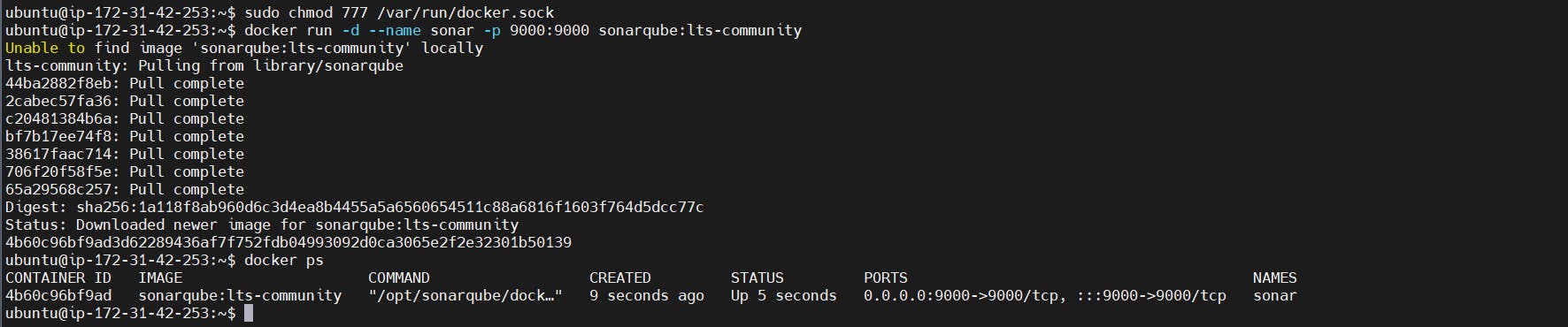
sudo usermod -aG docker $USER #my case is ubuntu

newgrp docker

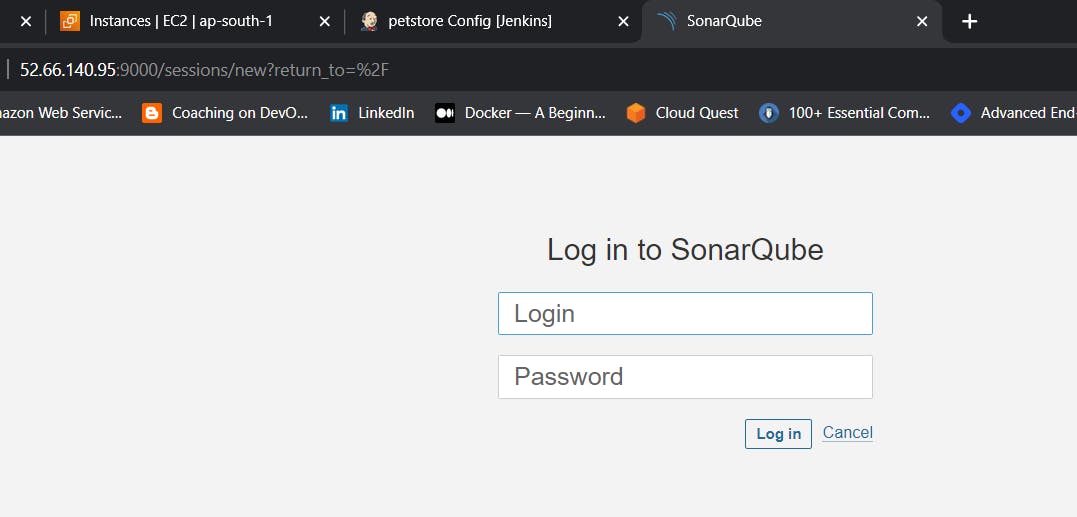
sudo chmod 777 /var/run/docker.sock

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



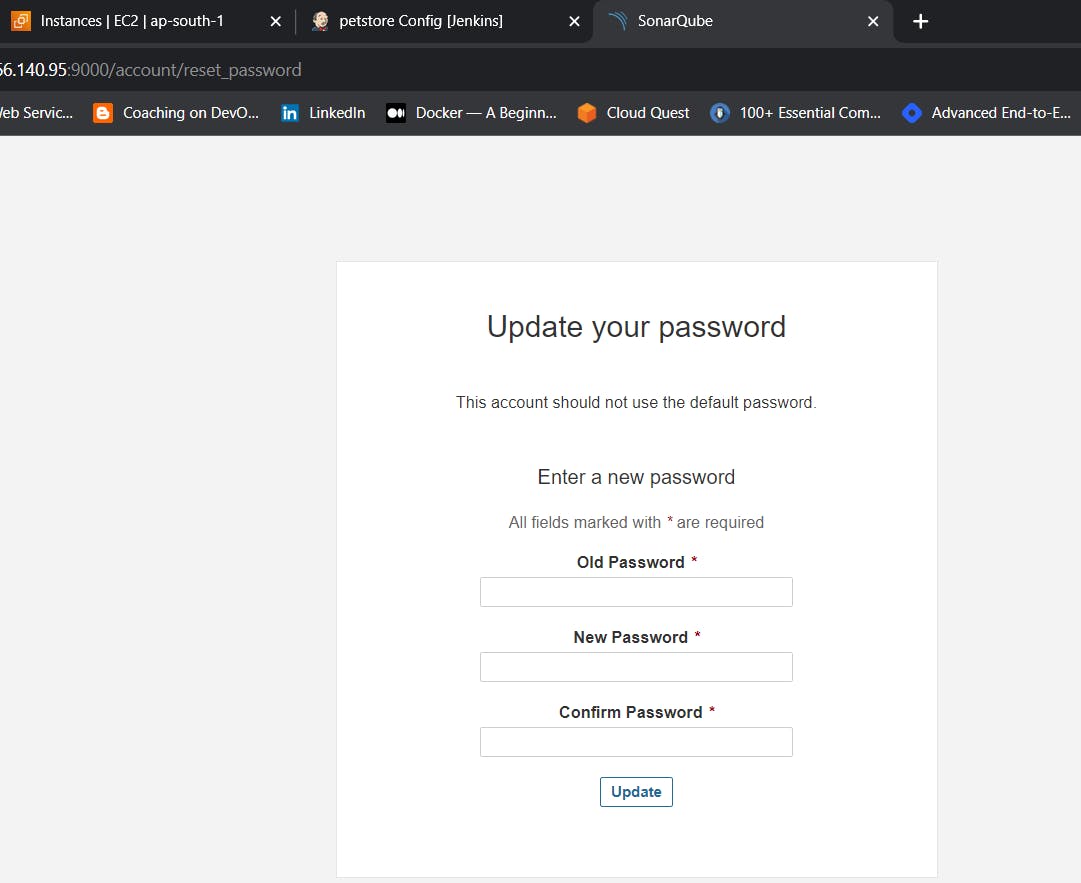
Now our sonarqube is up and running



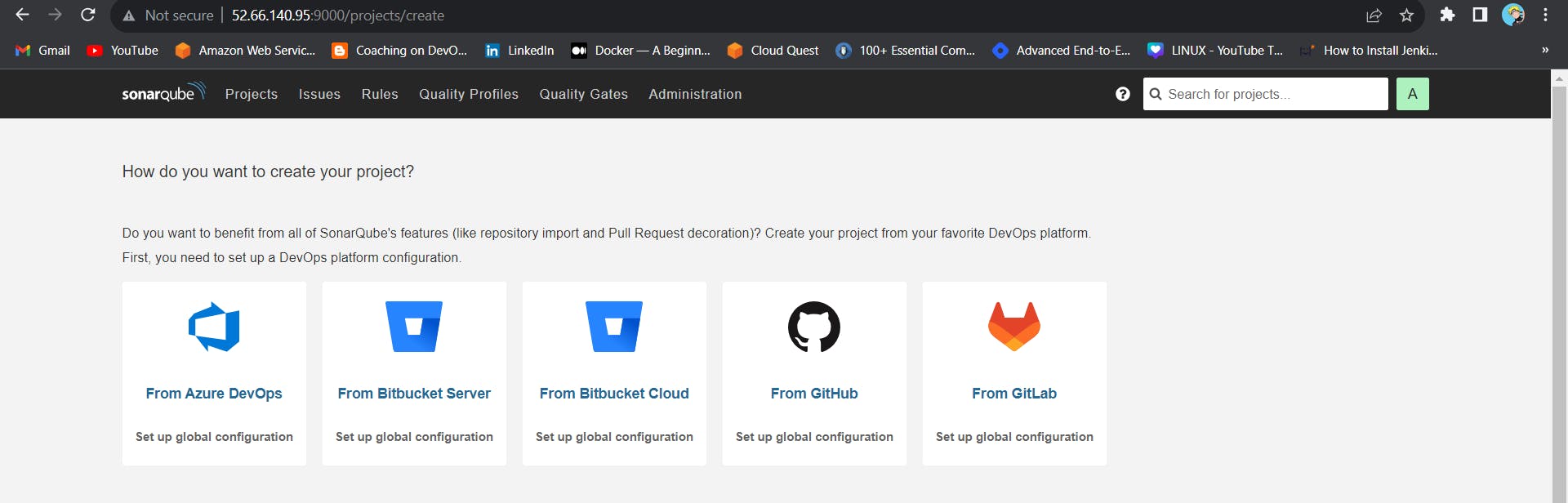
Enter username and password, click on login and change password

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**

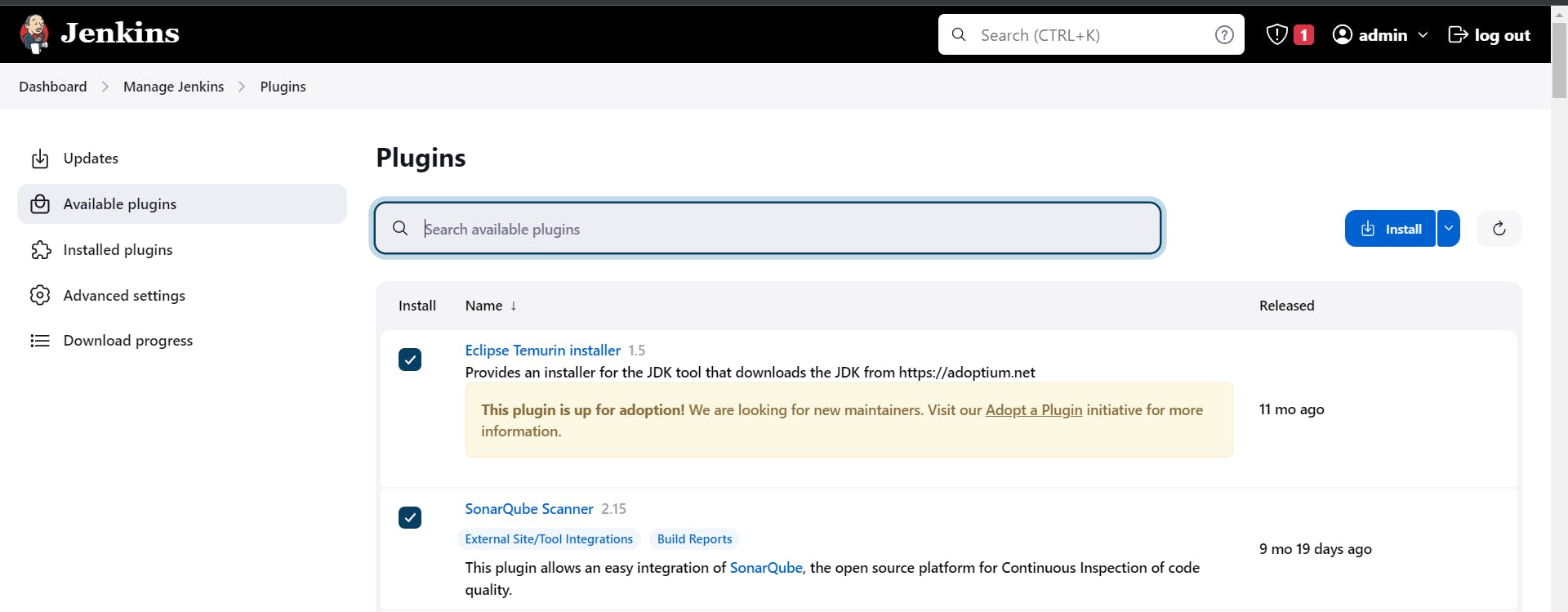
**3A — Install Plugin**

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

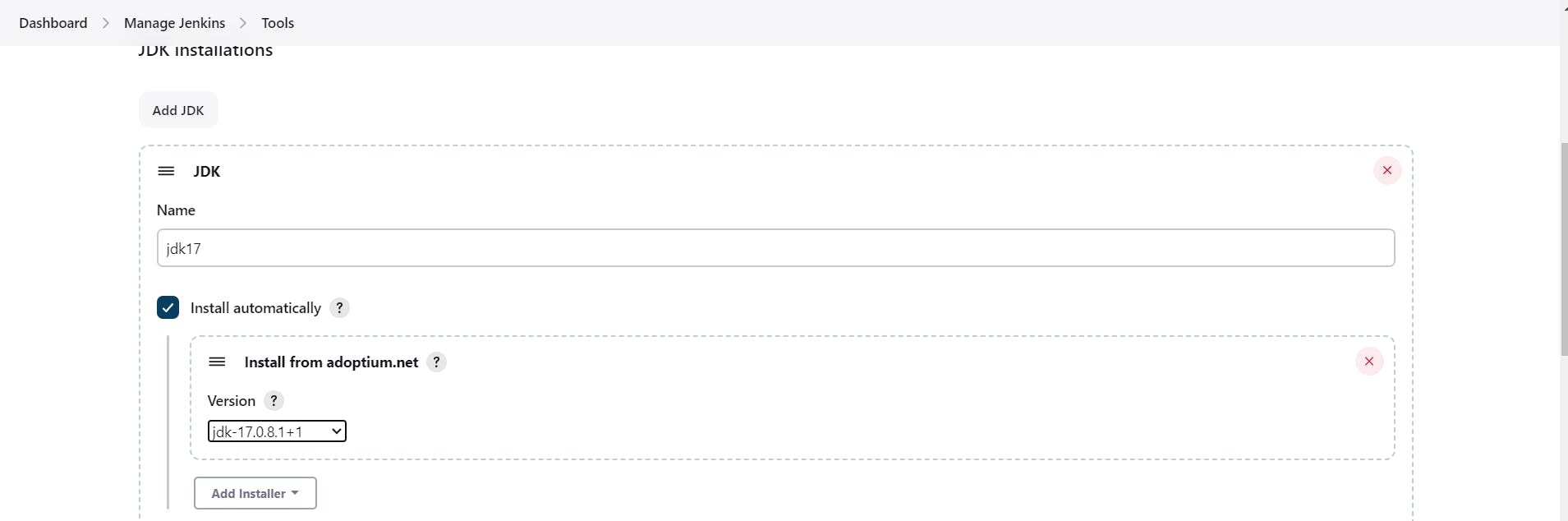
1 → Eclipse Temurin Installer (Install without restart)

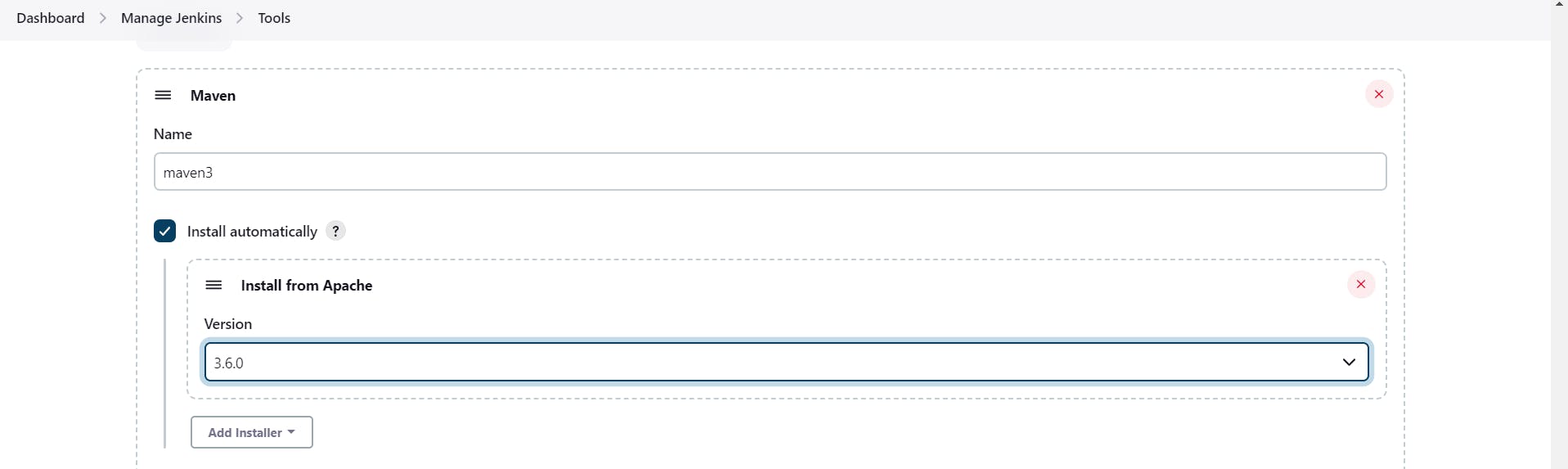
2 → SonarQube Scanner (Install without restart)



**3B — Configure Java and Maven in Global Tool Configuration**

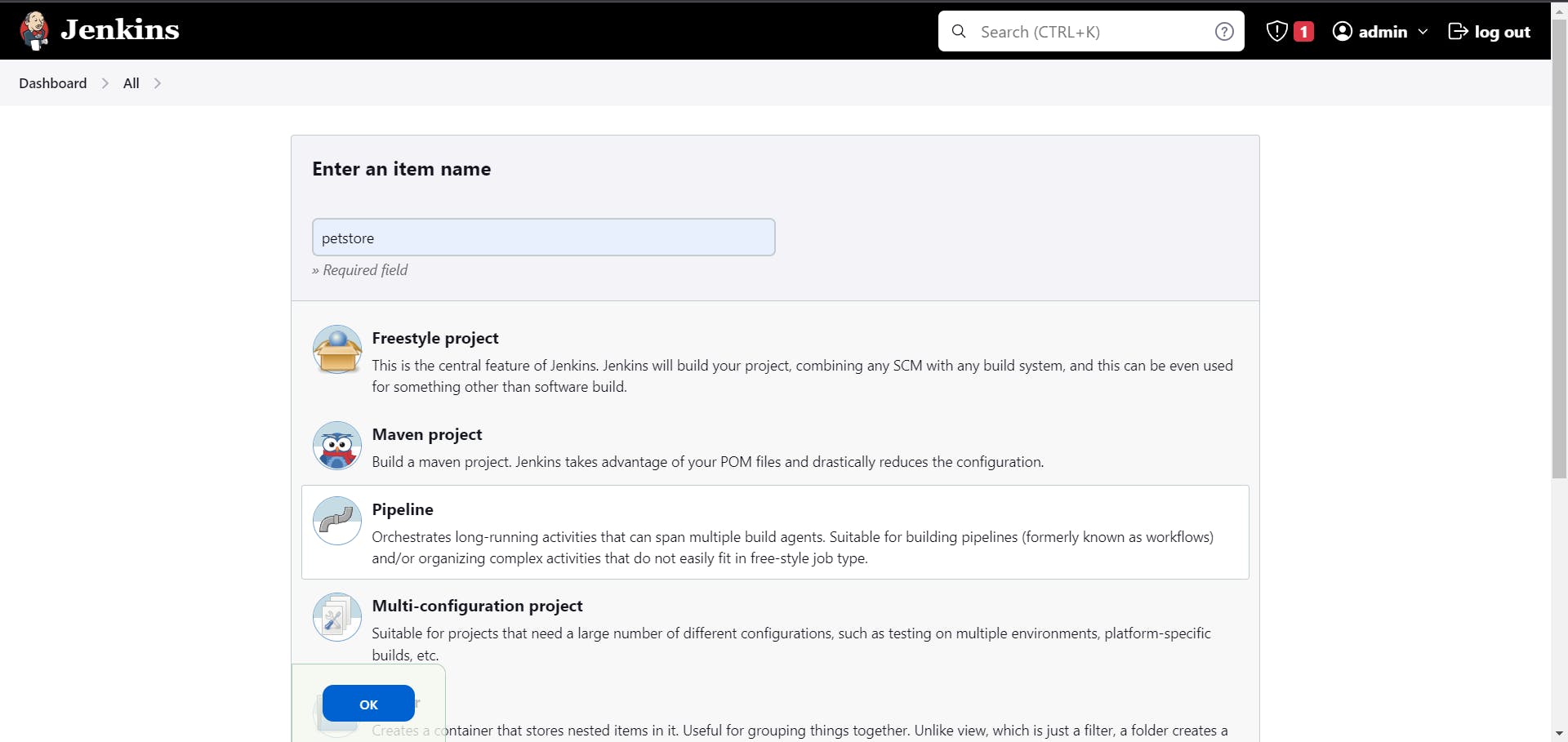
Goto Manage Jenkins → Tools → Install JDK(17) and Maven3(3.6.0) → Click on Apply and Save





**3C — Create a Job**

Label it as PETSHOP, click on Pipeline and OK.



Enter this in Pipeline Script,

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

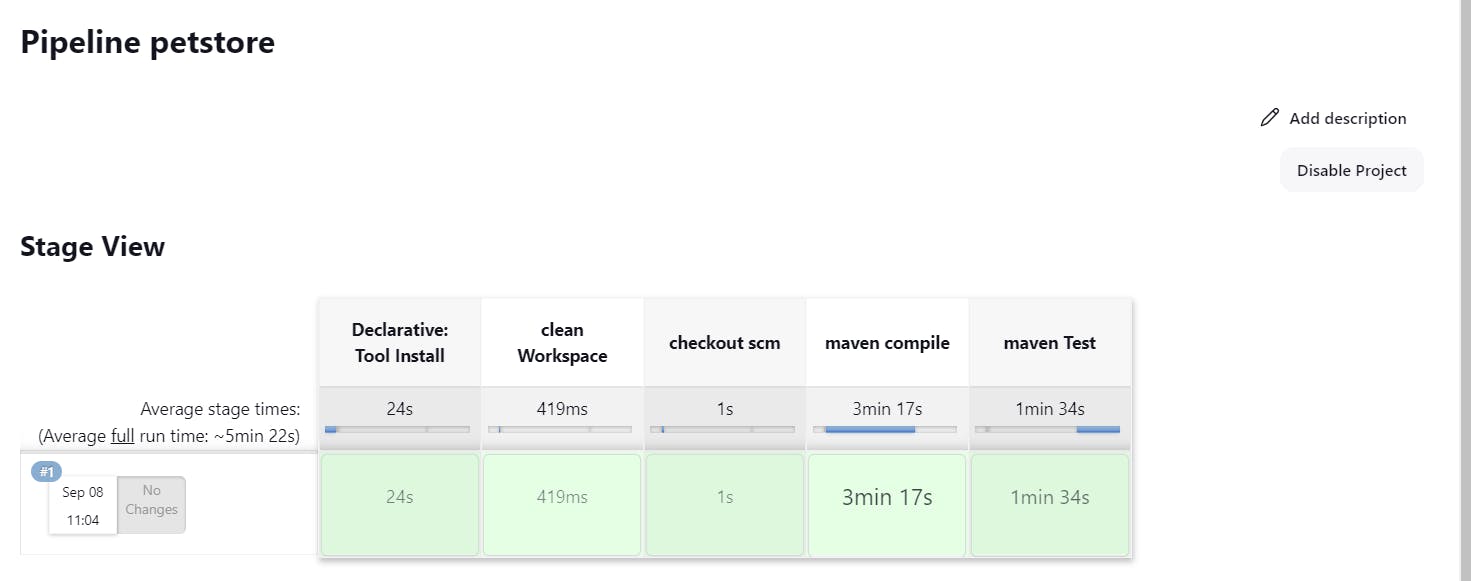
}

}

}

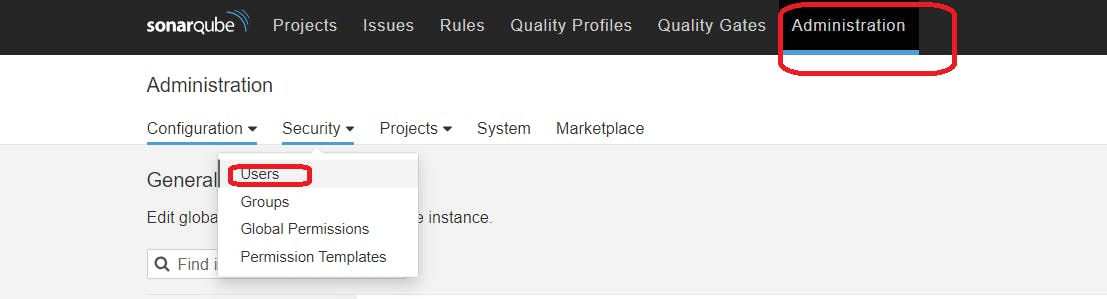
}

The stage view would look like this,

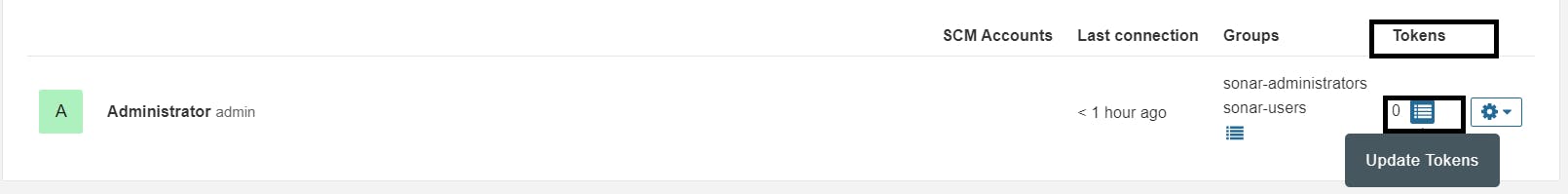


**Step 4 — Configure Sonar Server in Manage Jenkins**

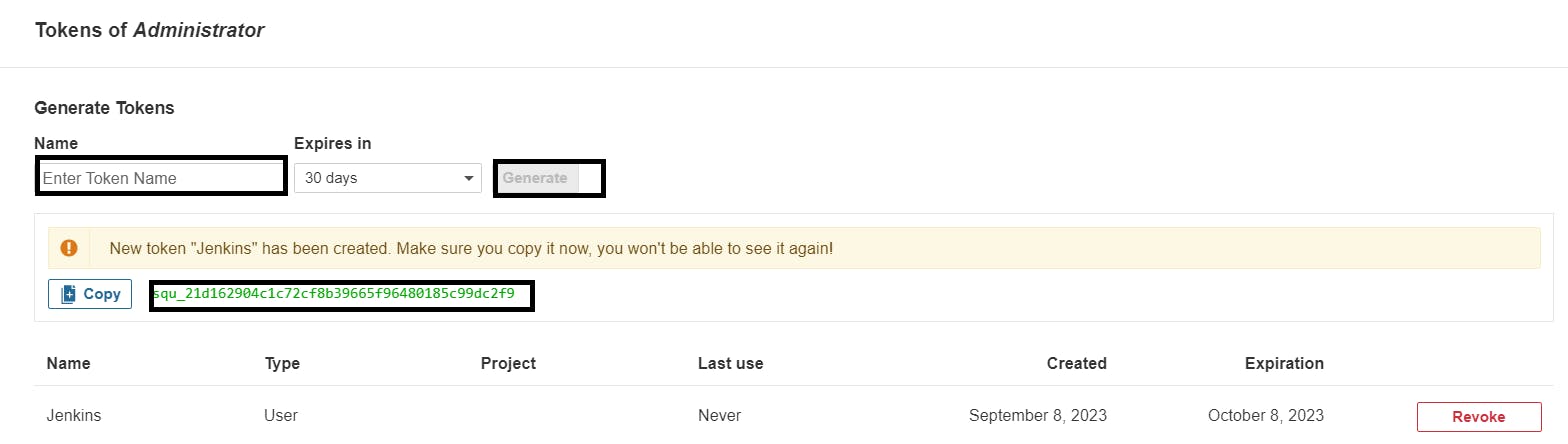
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

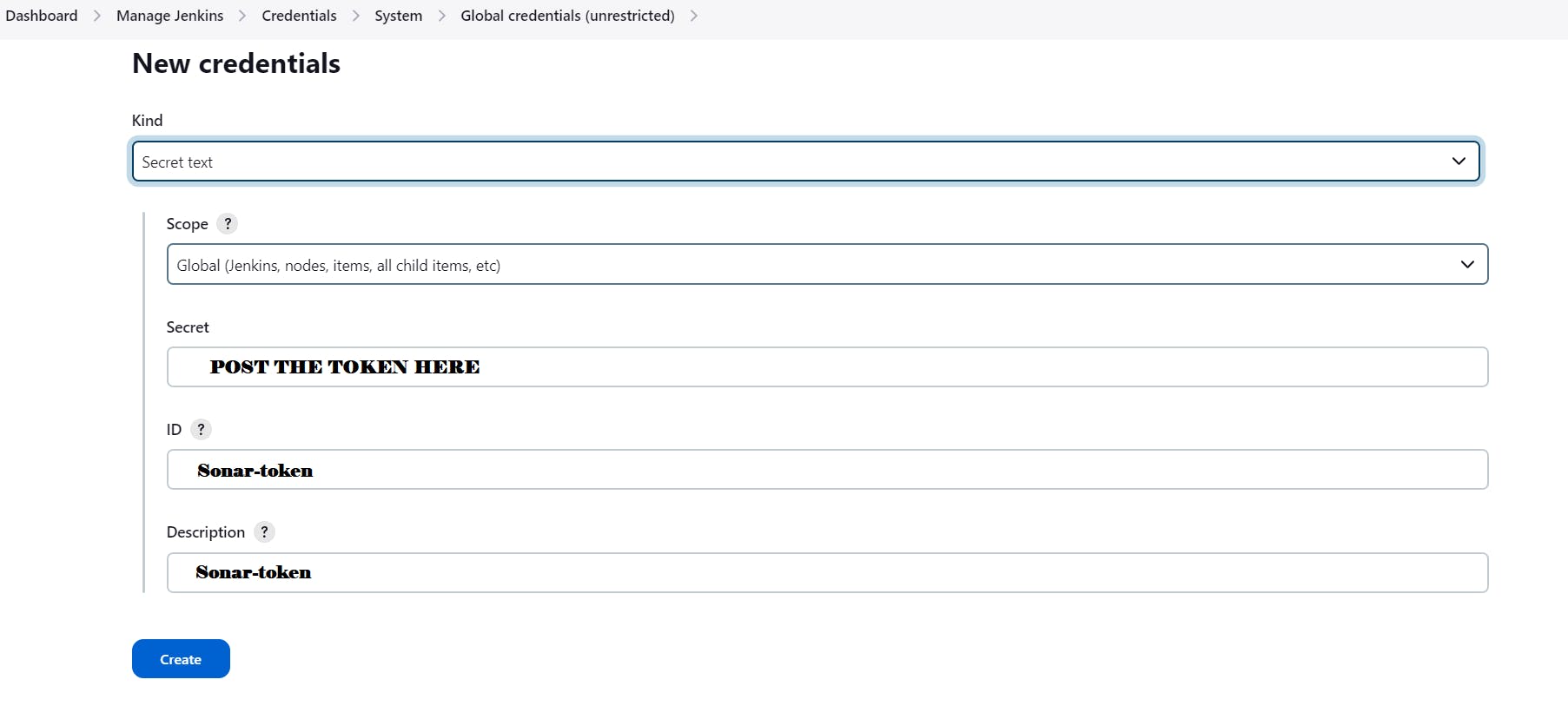


Create a token with a name and generate

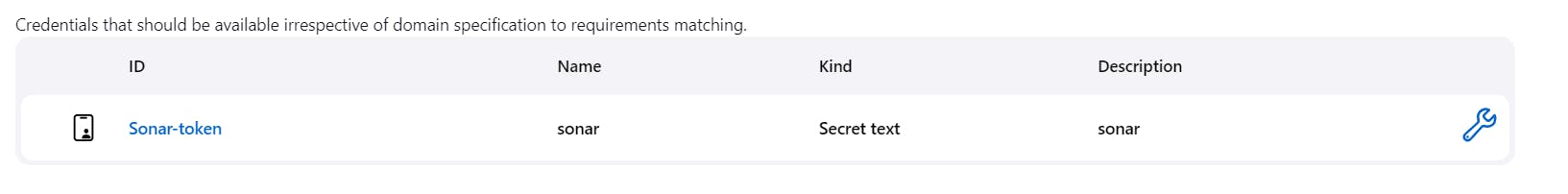


copy Token

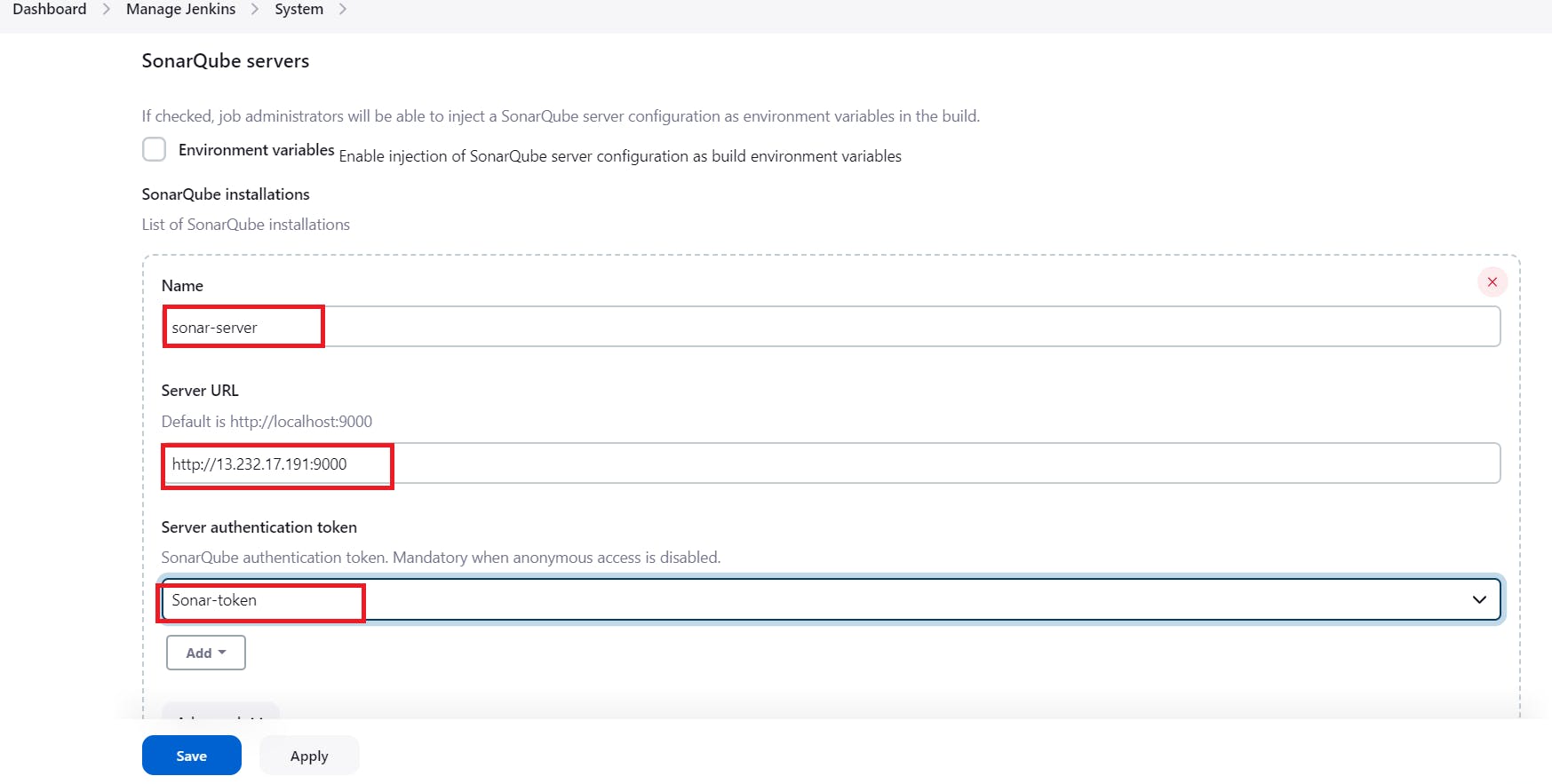
Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this



You will this page once you click on create



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

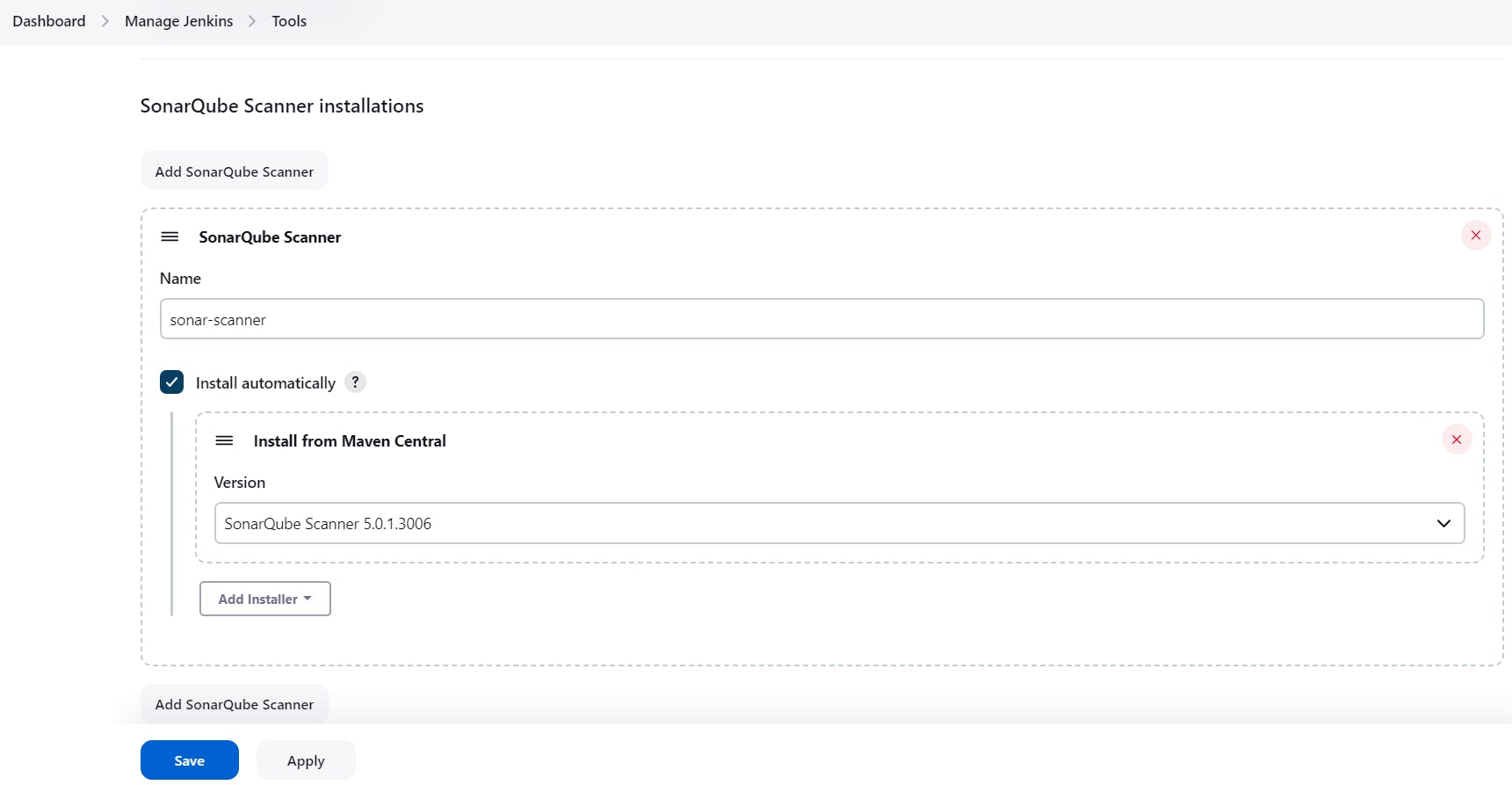


Click on Apply and Save

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

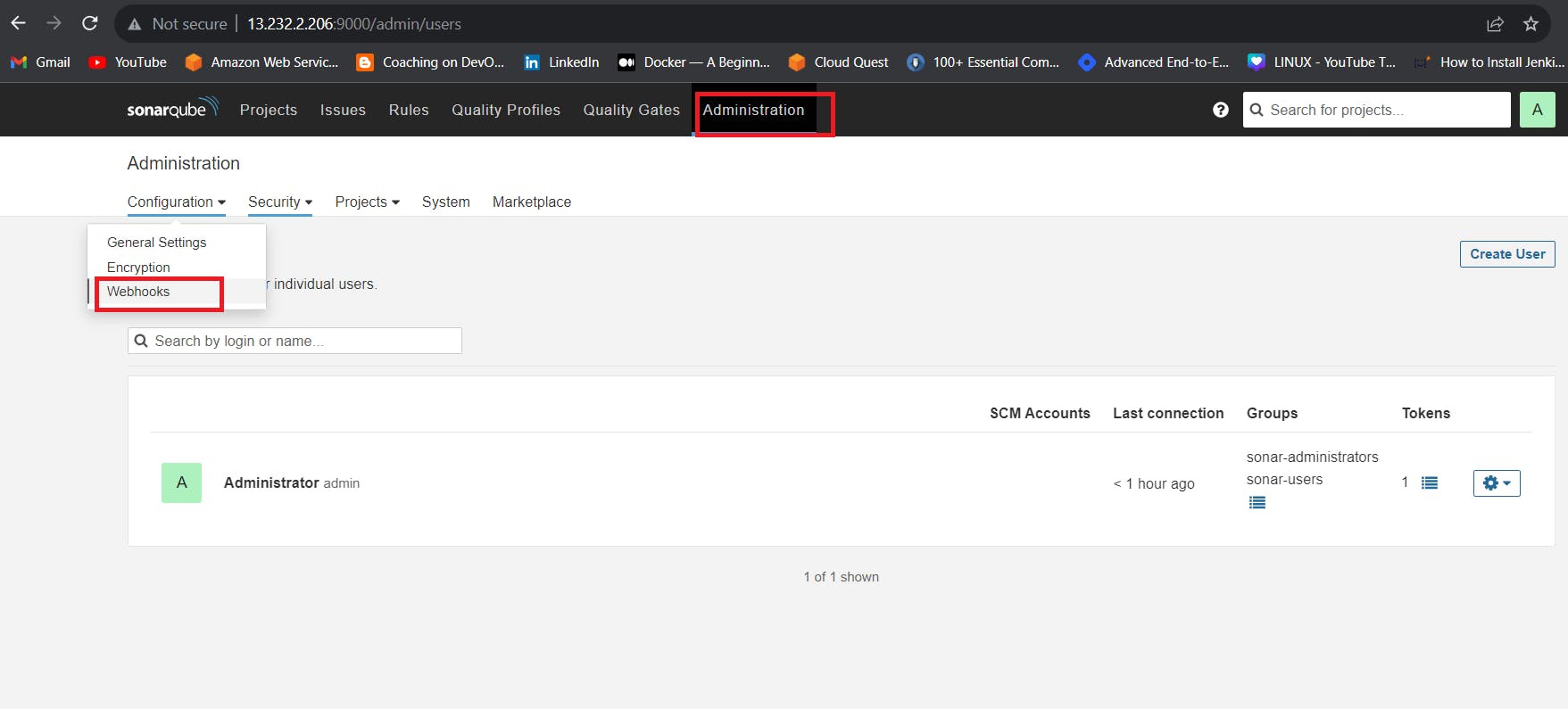
**Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

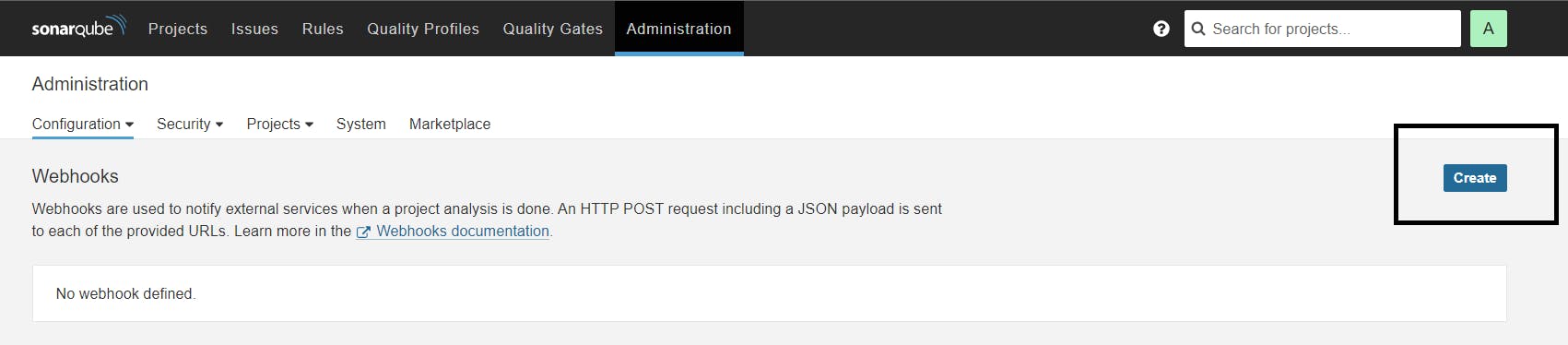


In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



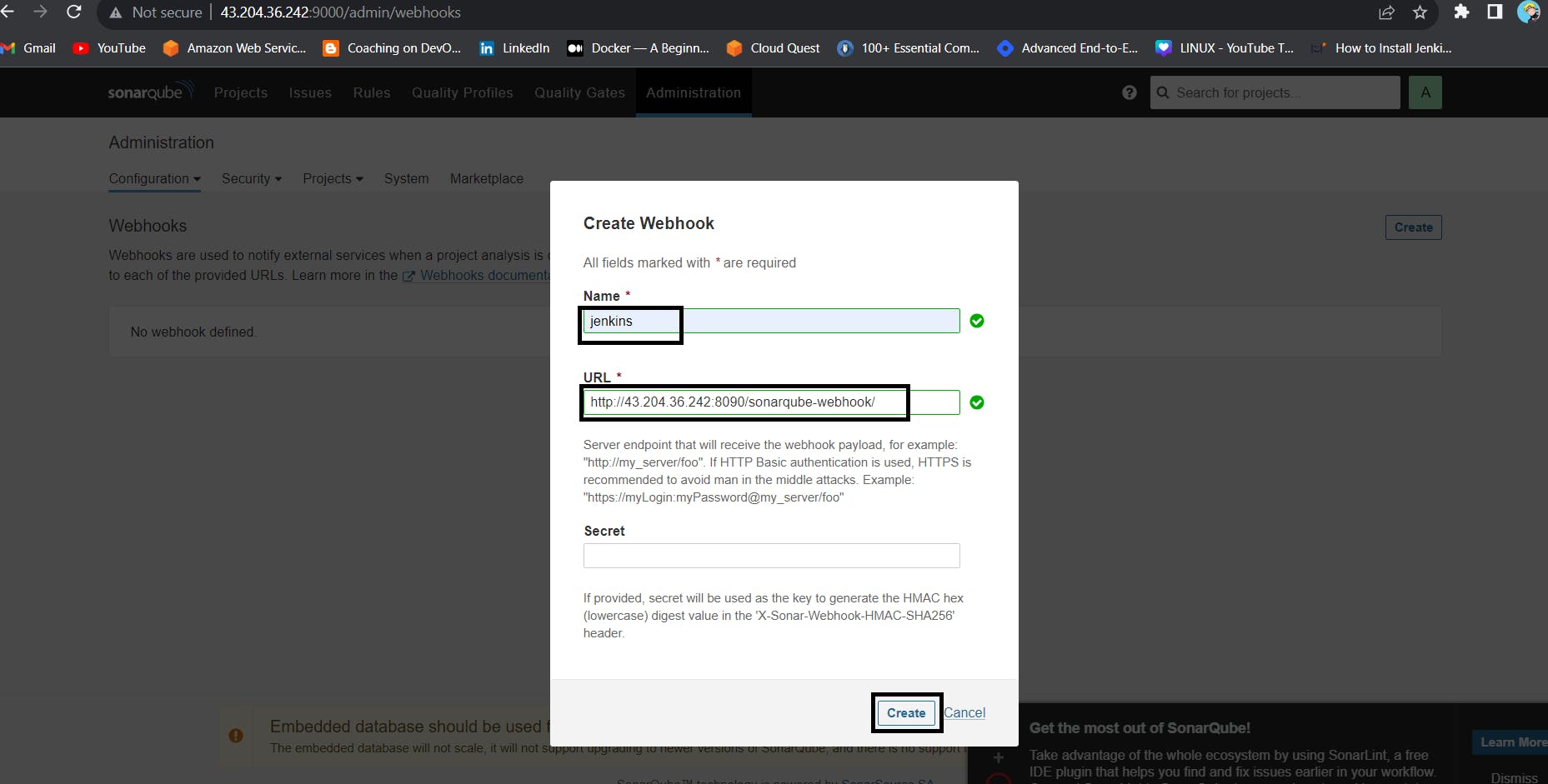
Click on Create



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.

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

#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 {

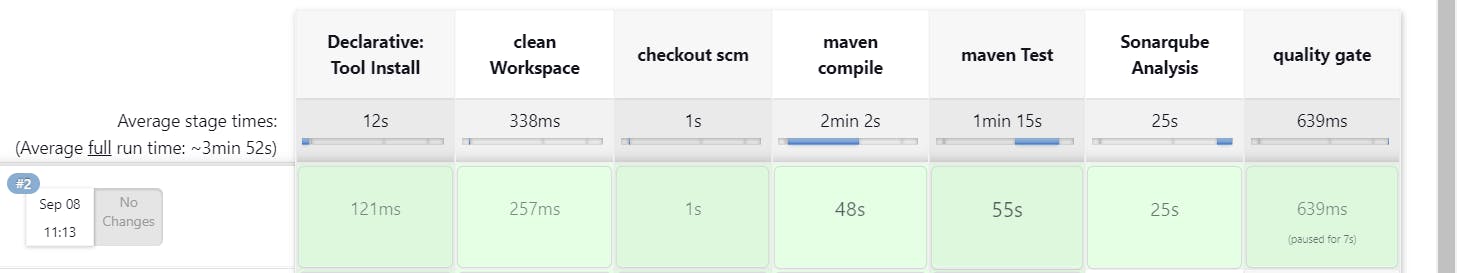
script {

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

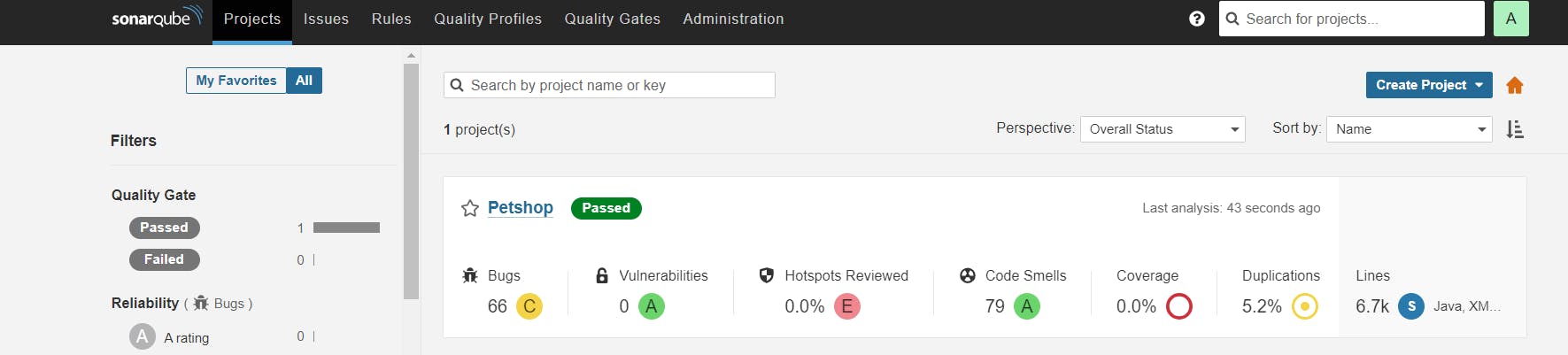
}

}

}



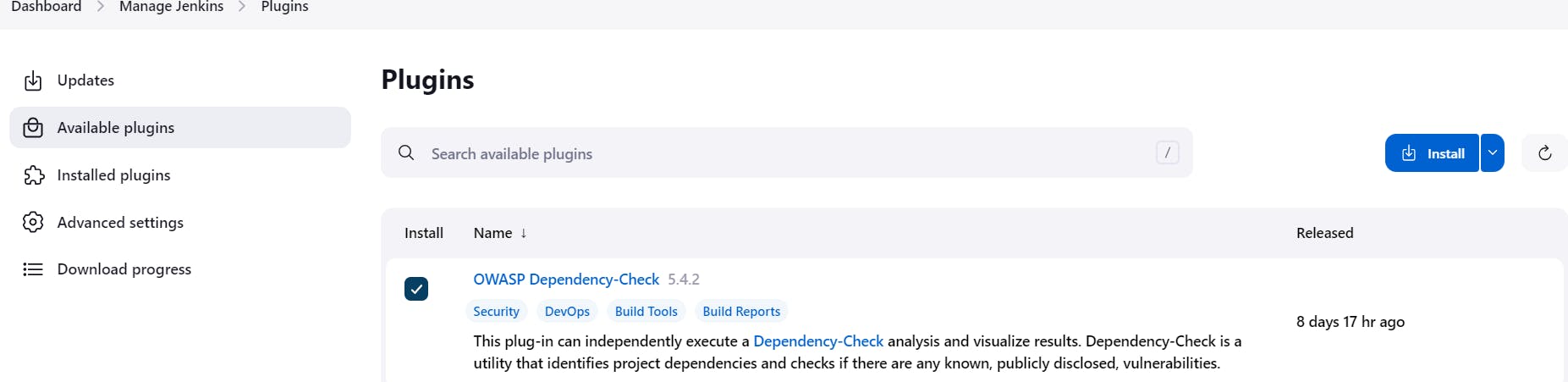
To see the report, you can go to Sonarqube Server and go to Projects.



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.

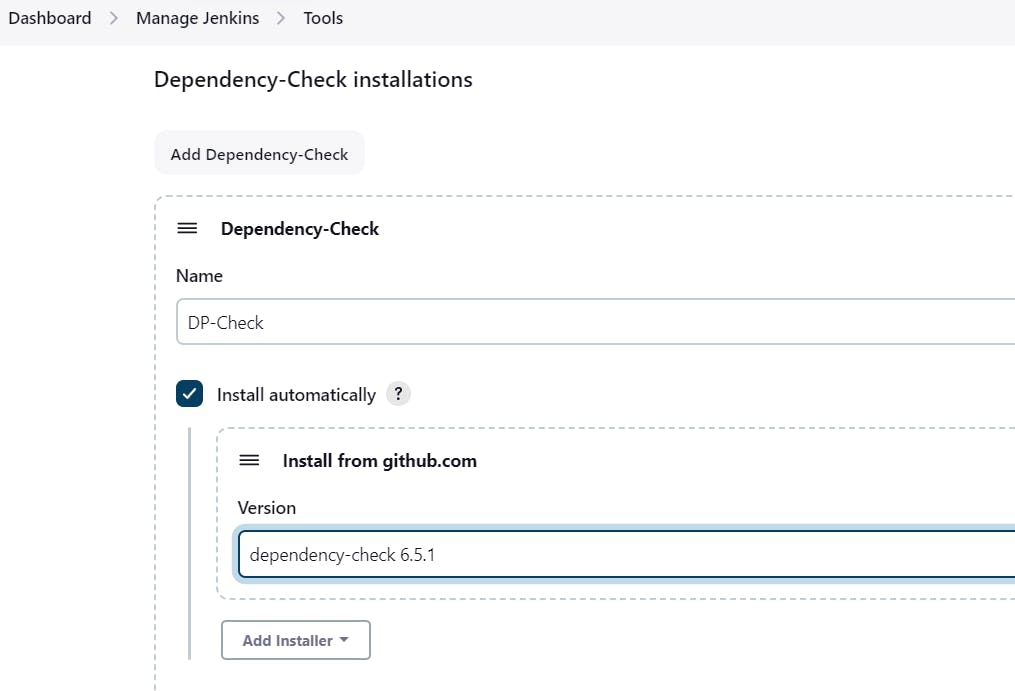
**Step 5 — Install OWASP Dependency Check Plugins**

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{

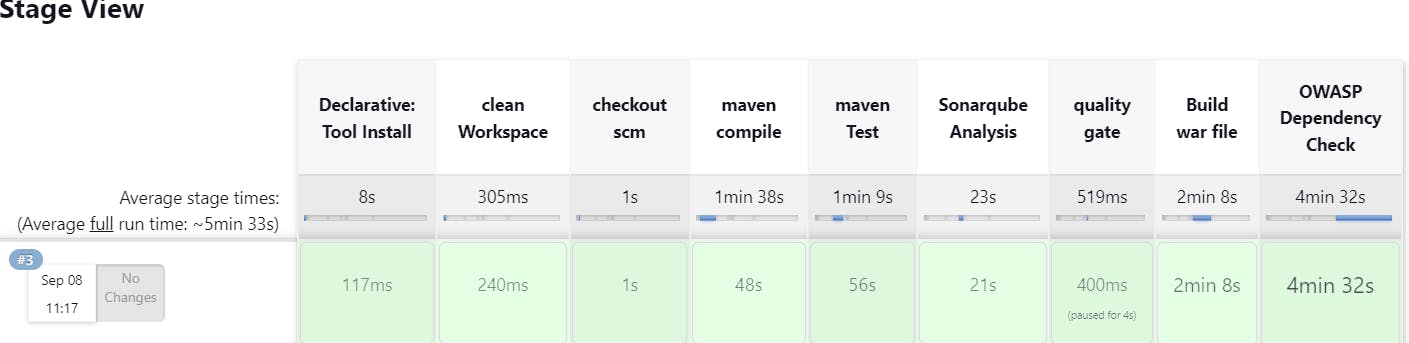
dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

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

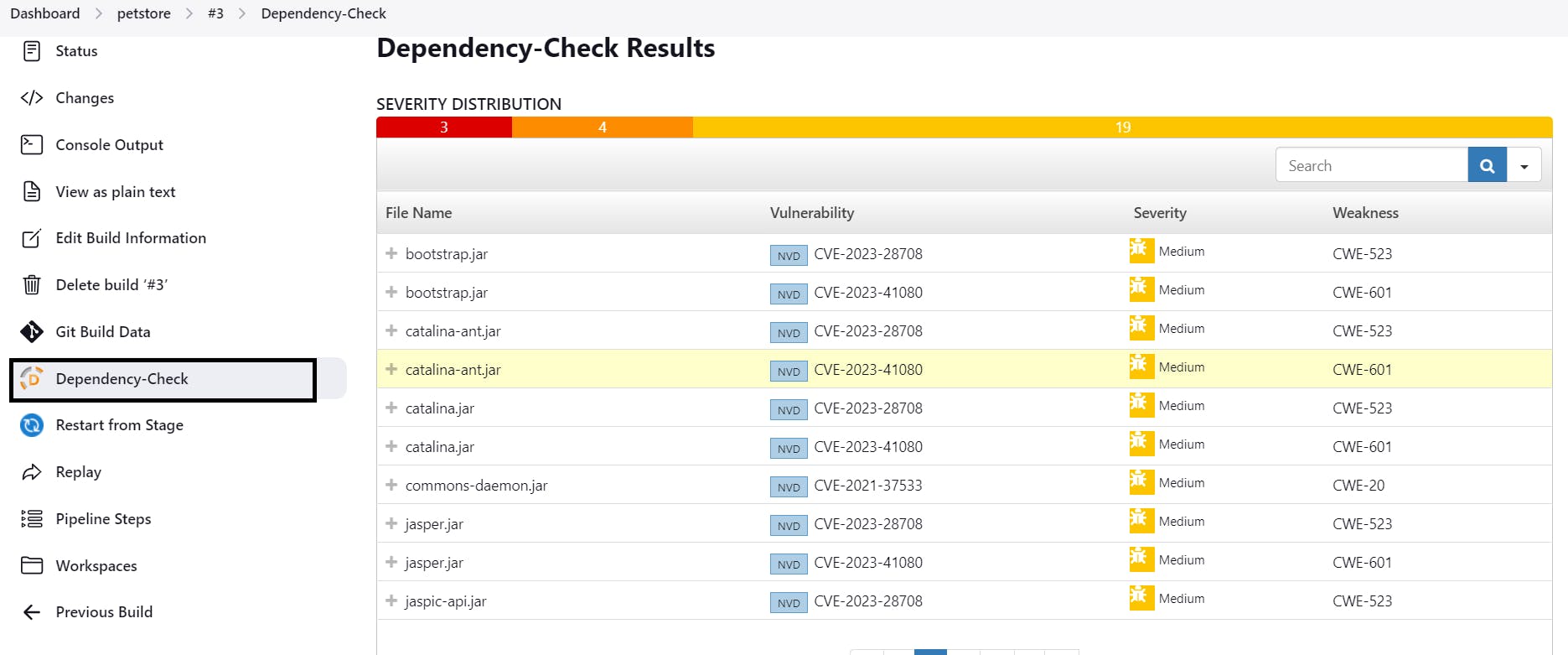
}

}

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

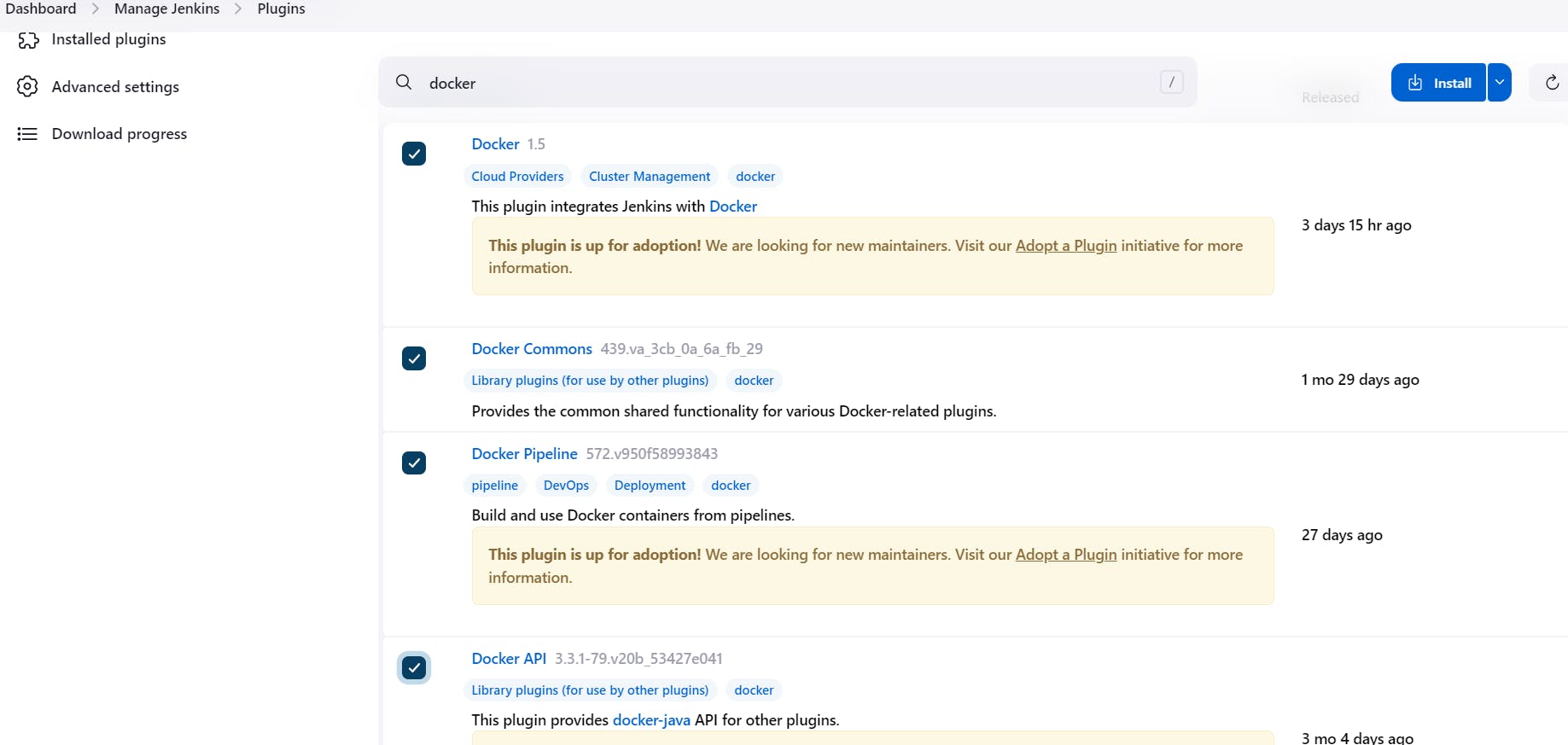
Docker Commons

Docker Pipeline

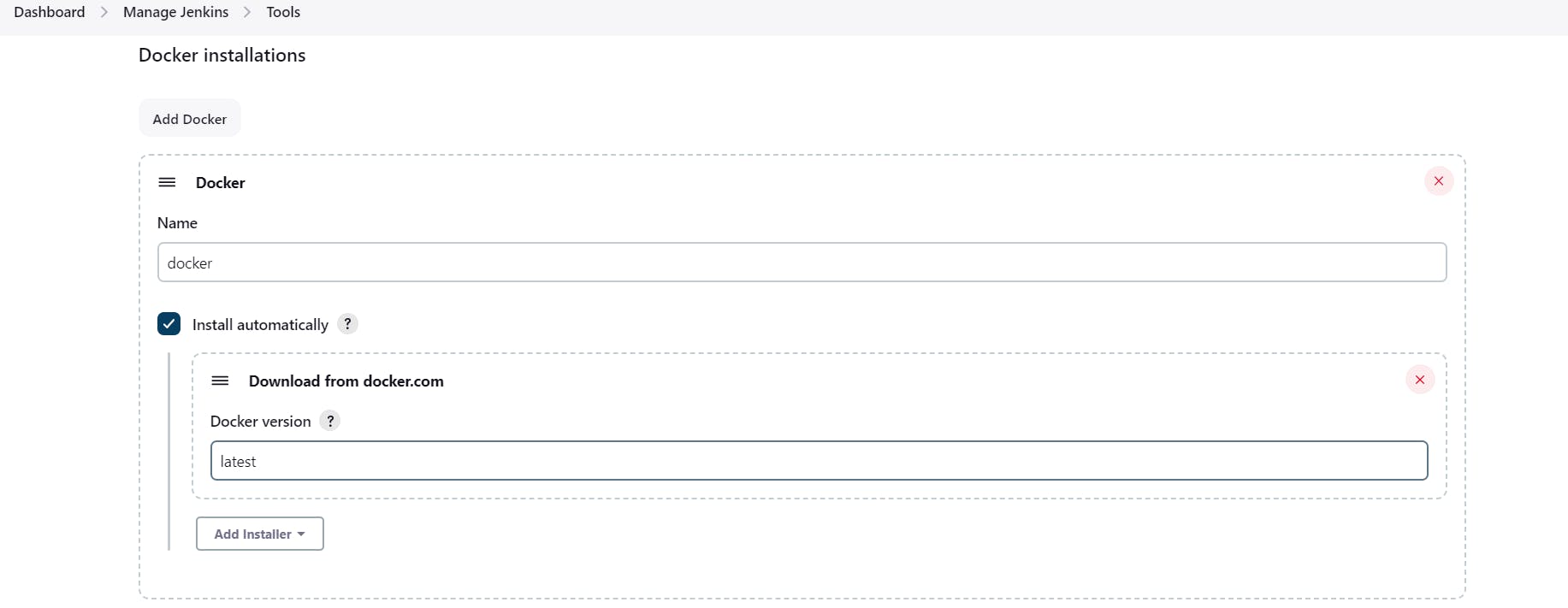
Docker API

docker-build-step

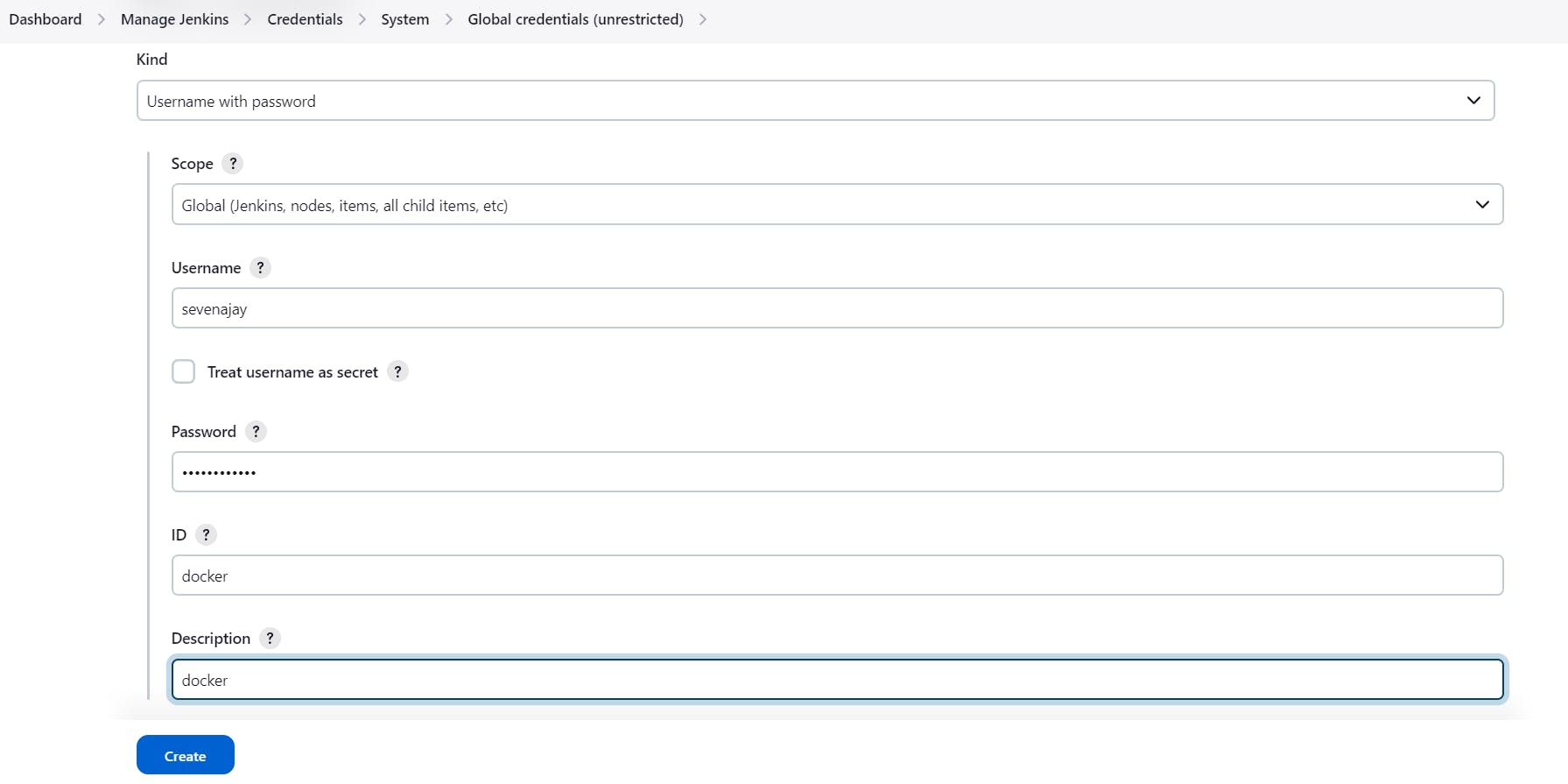
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



Add DockerHub Username and Password under Global Credentials



Add this stage to Pipeline Script

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

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

}

}

stage ('Deploy to container'){

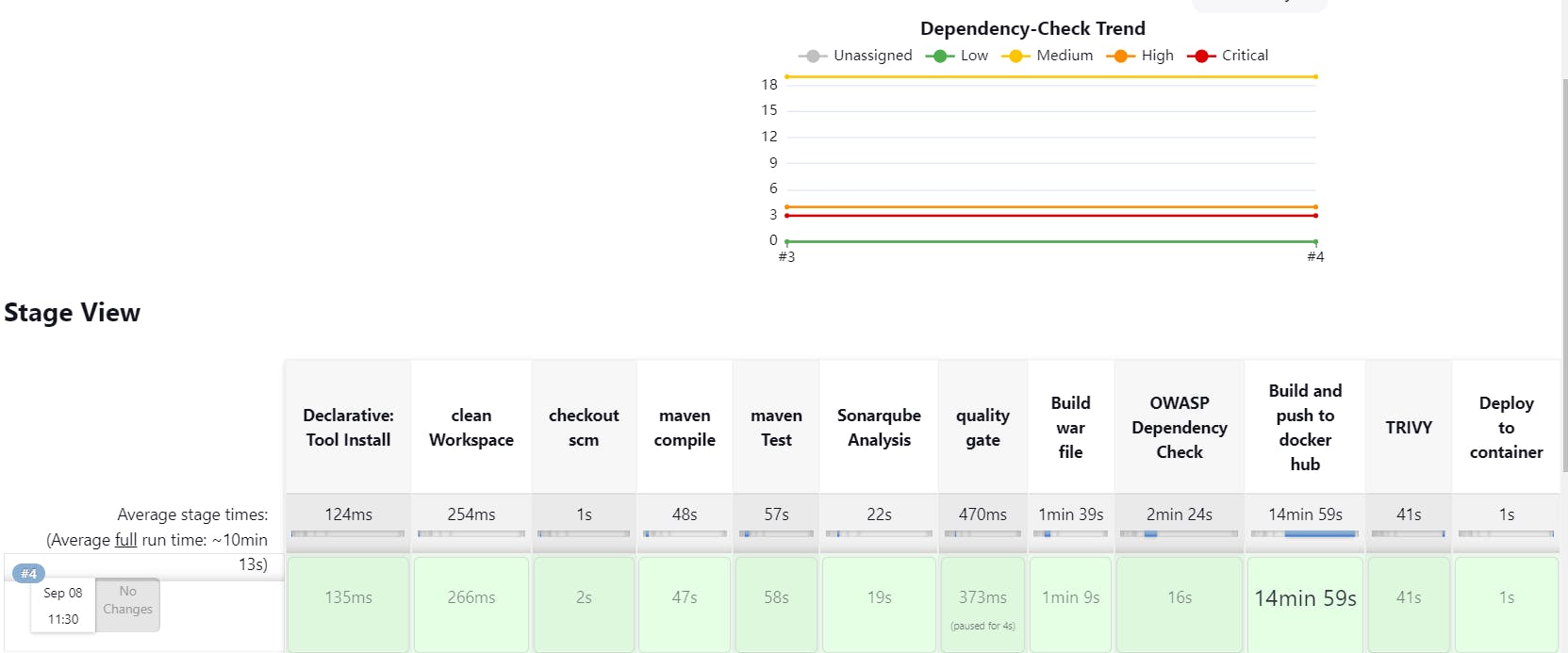
steps{

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

}

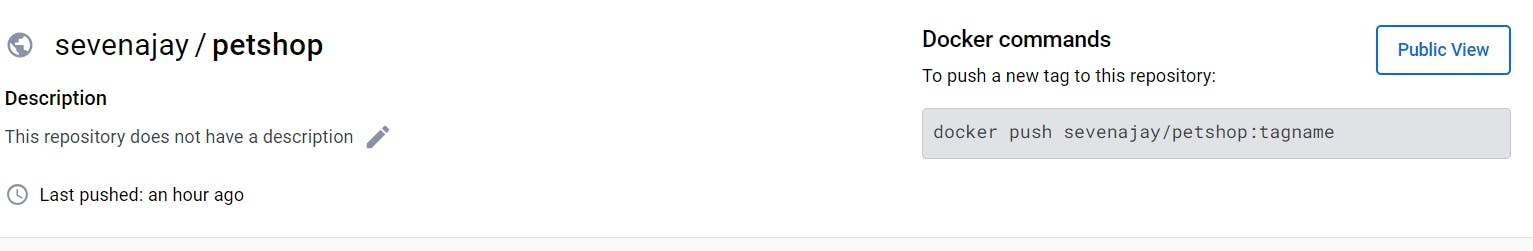
}

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



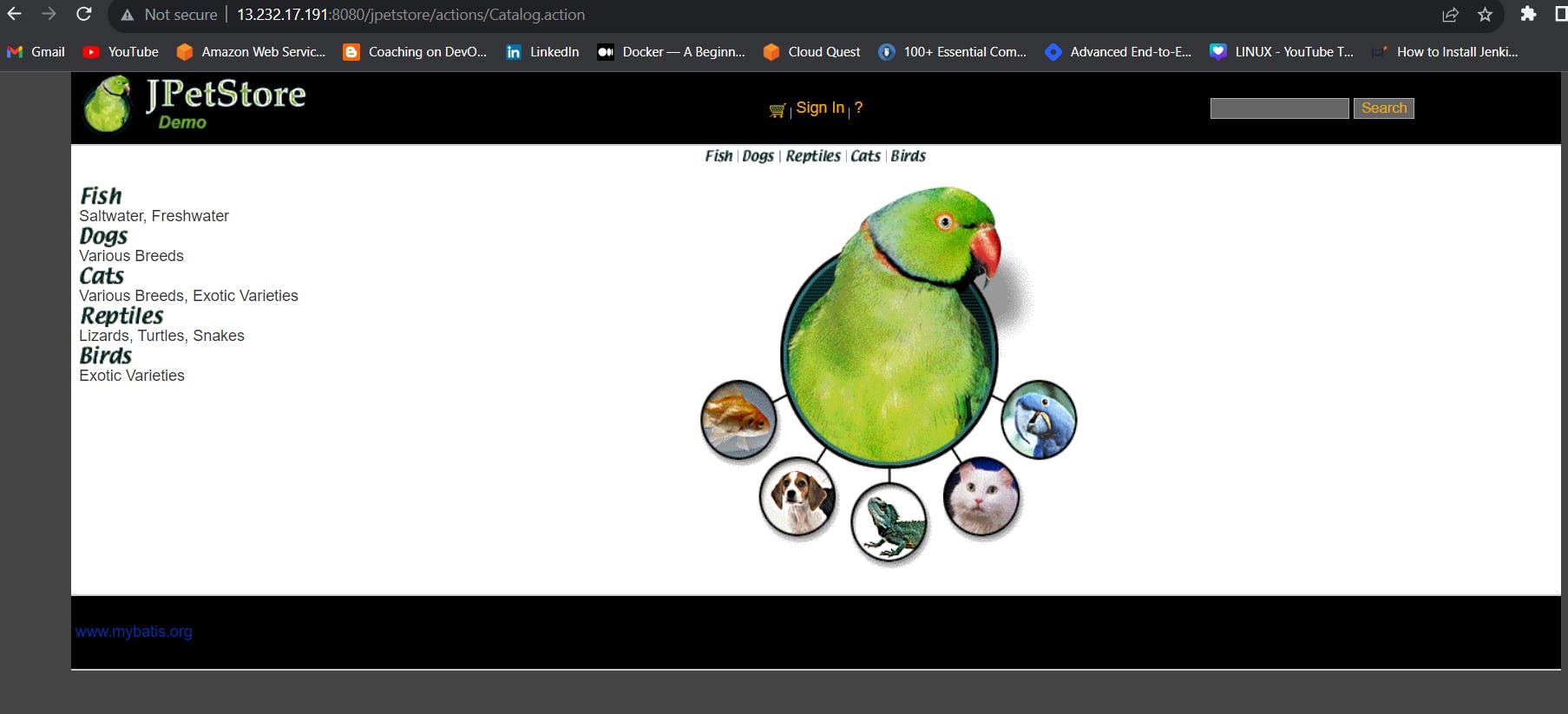
Now, when you do

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



<Ec2-public-ip:8080/jpetstore>

You will get this output



**Step 8 — Kuberenetes 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**

COPY

COPY

sudo apt update

sudo apt install curl

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

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

kubectl version --client

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

sudo su

hostname master

bash

clear

**----------Worker Node------------**

sudo su

hostname worker

bash

clear

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

sudo apt-get update

sudo apt-get install -y docker.io

sudo usermod –aG docker Ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

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

sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF

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

EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo snap install kube-apiserver

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

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

# in case your in root exit from it and run below commands

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

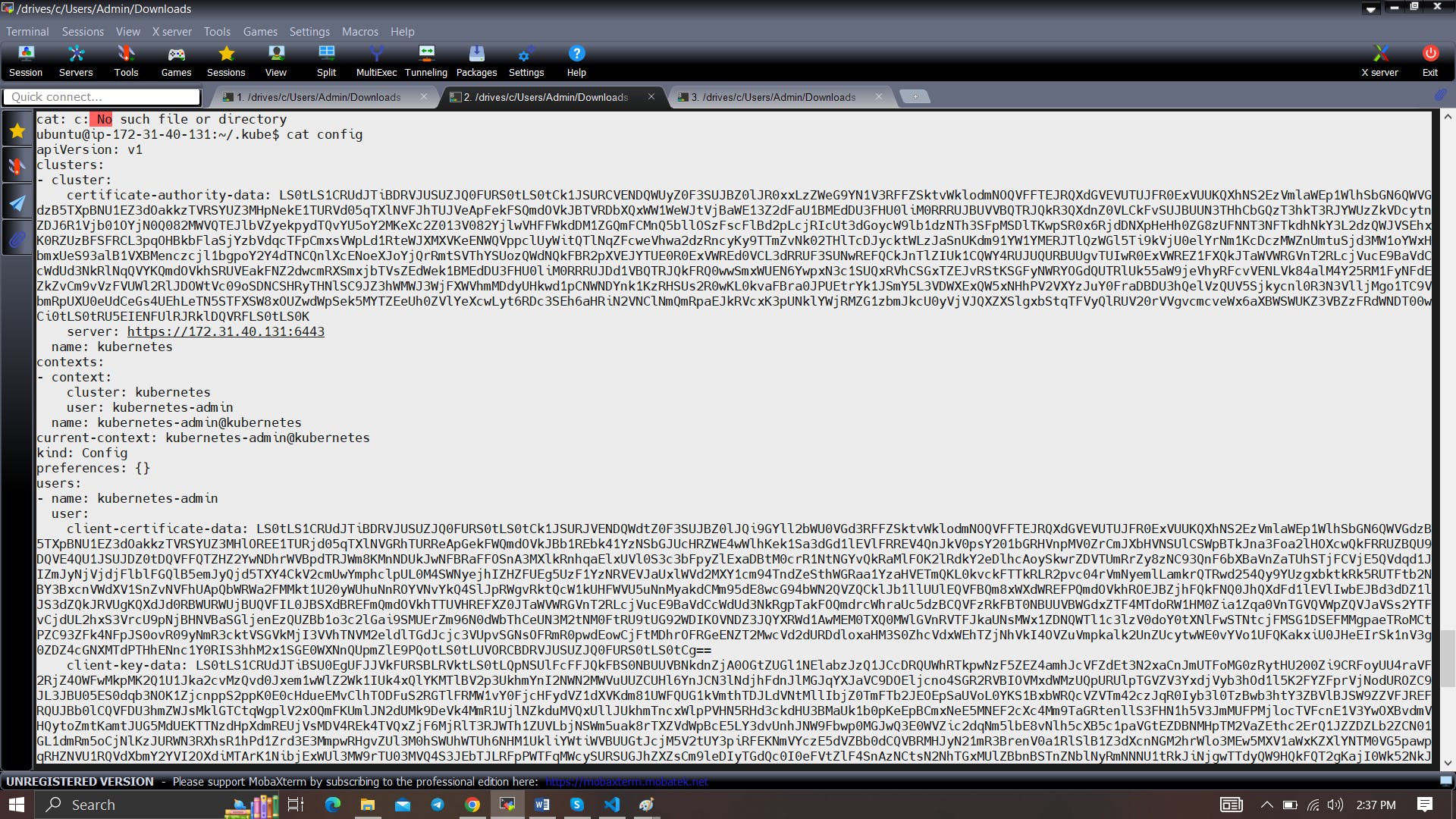
sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

**----------Worker Node------------**

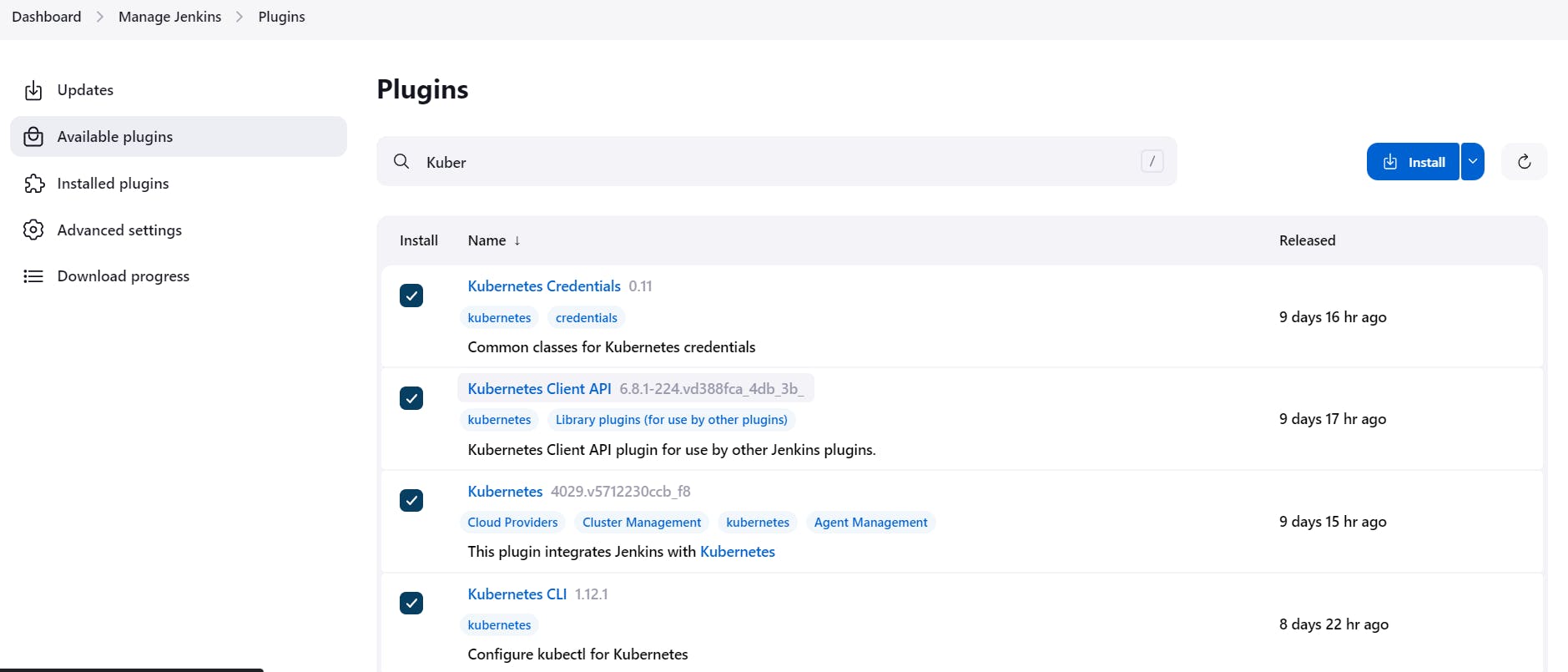
sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash <hash>

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

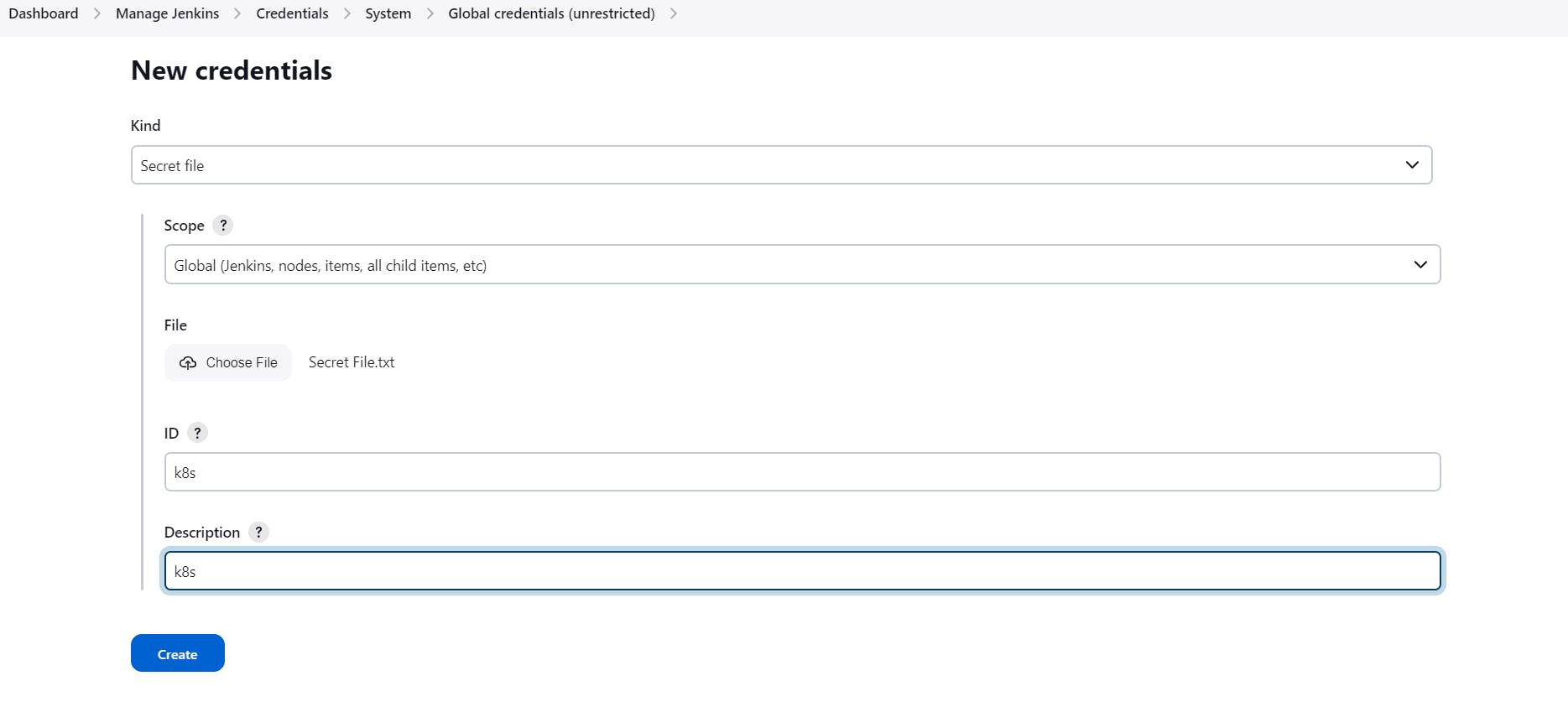


copy it and save it in documents or another folder save it as secret-file.txt

Install Kubernetes Plugin, Once it's installed successfully

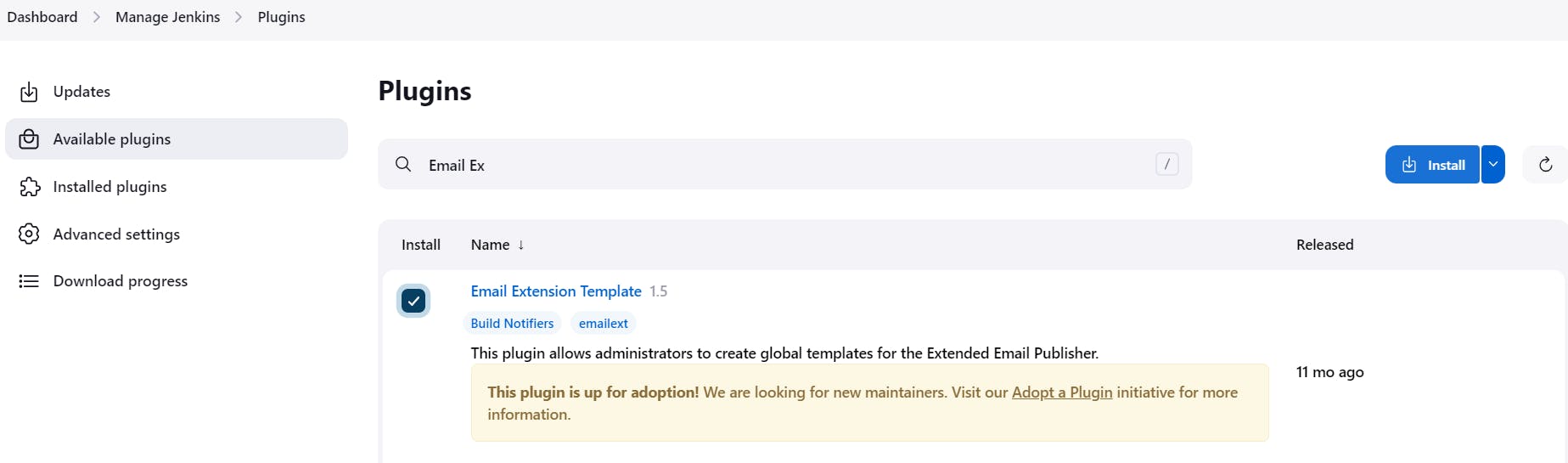


goto manage Jenkins --> manage credentials --> Click on Jenkins global --> add credentials



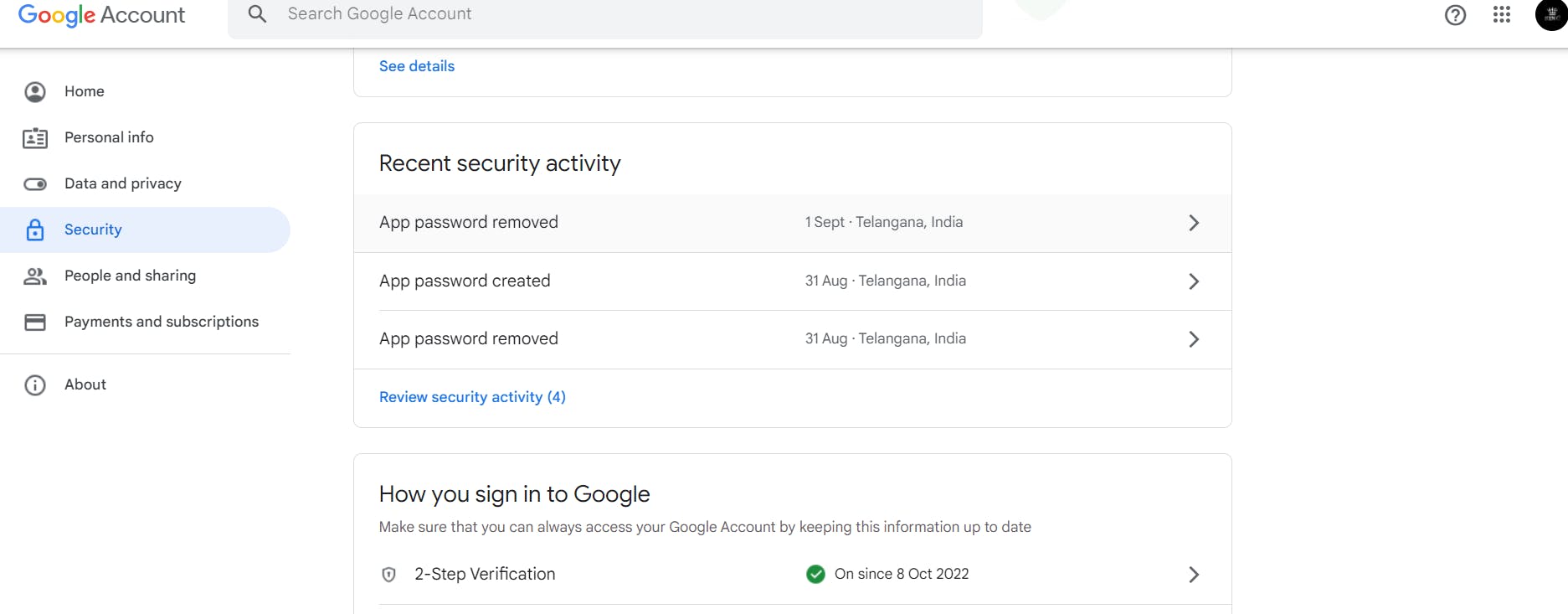
**Configuring mail server in Jenkins ( Gmail )**

Install Email Extension Plugin in Jenkins



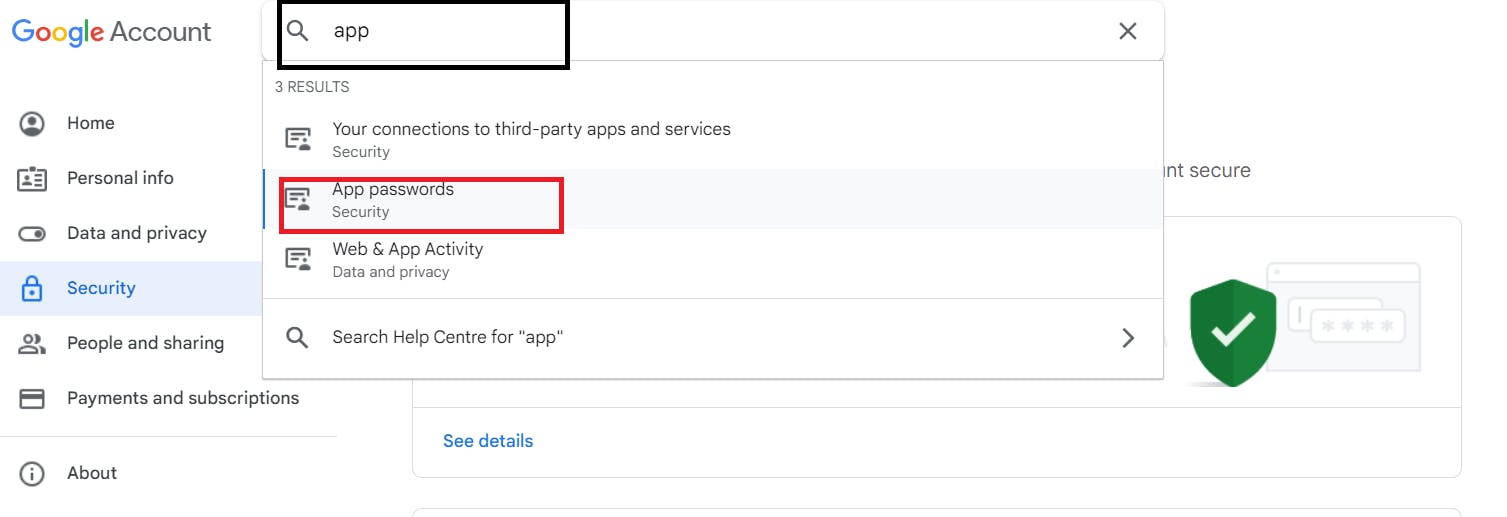
Go to your Gmail and click on your profile

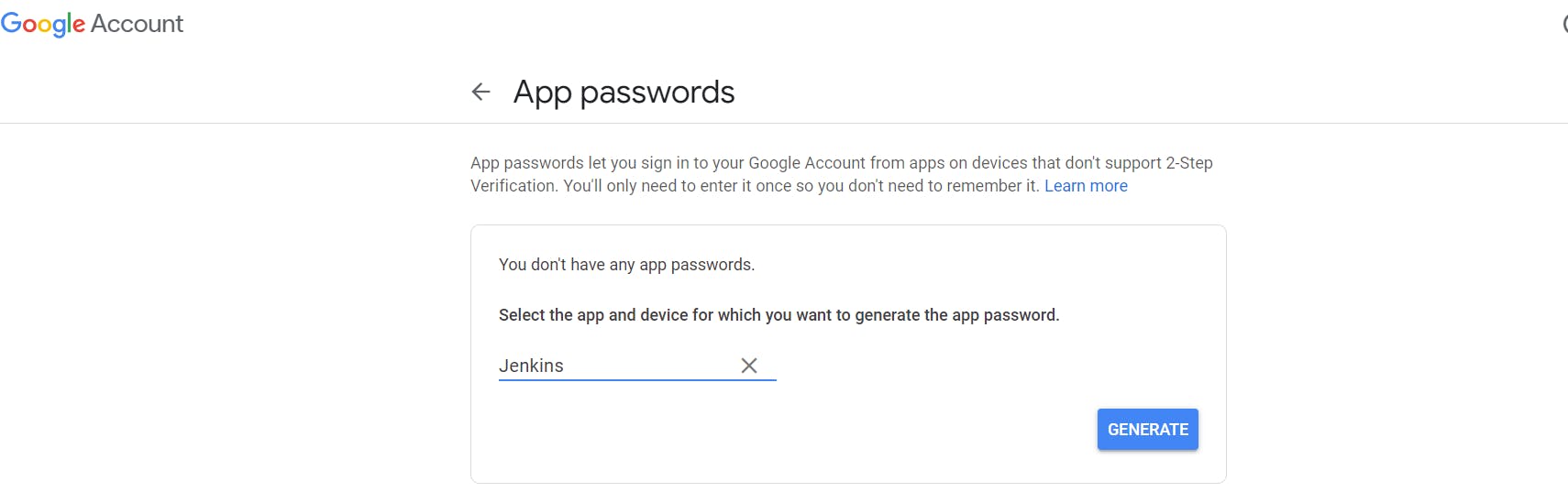
Then click on Manage Your Google Account --> click on the security tab on the left side panel you will get this page



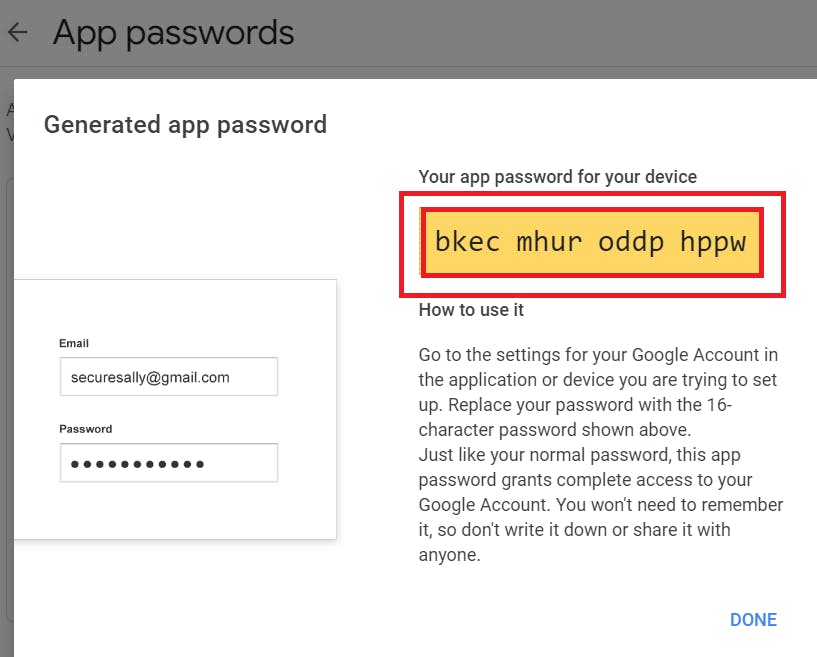
2-step verification should be enabled.

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

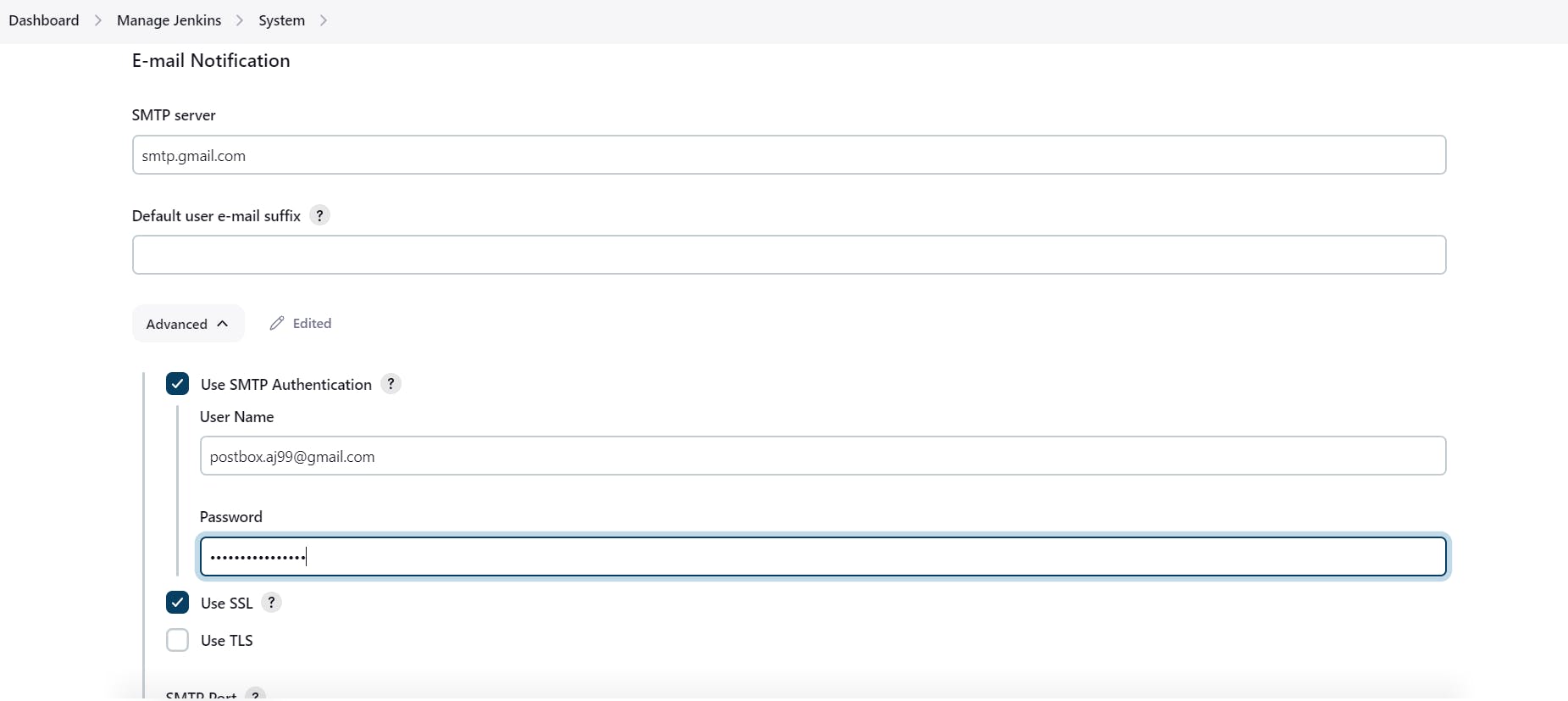


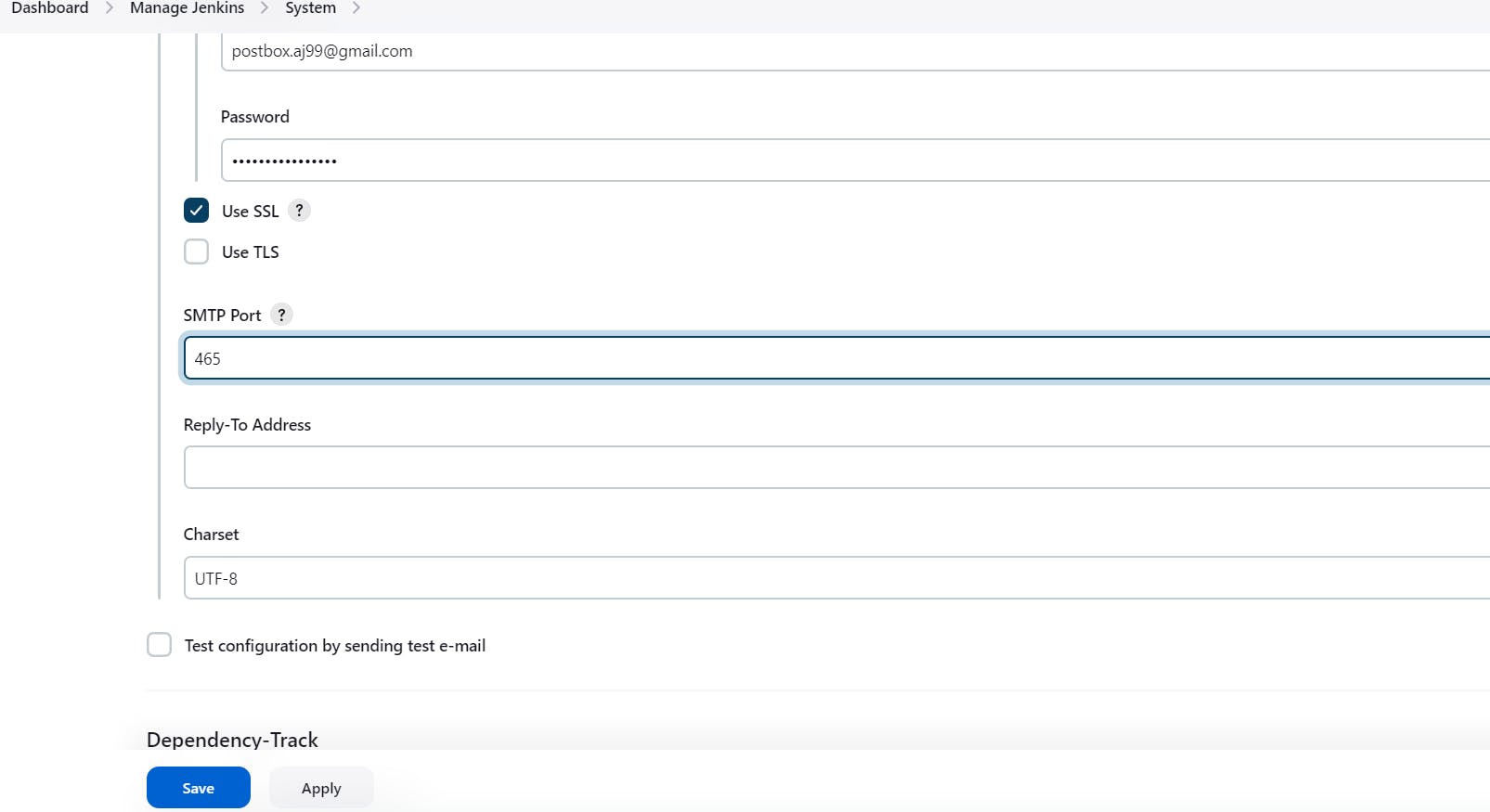


Click on other and provide your name and click on Generate and copy the password



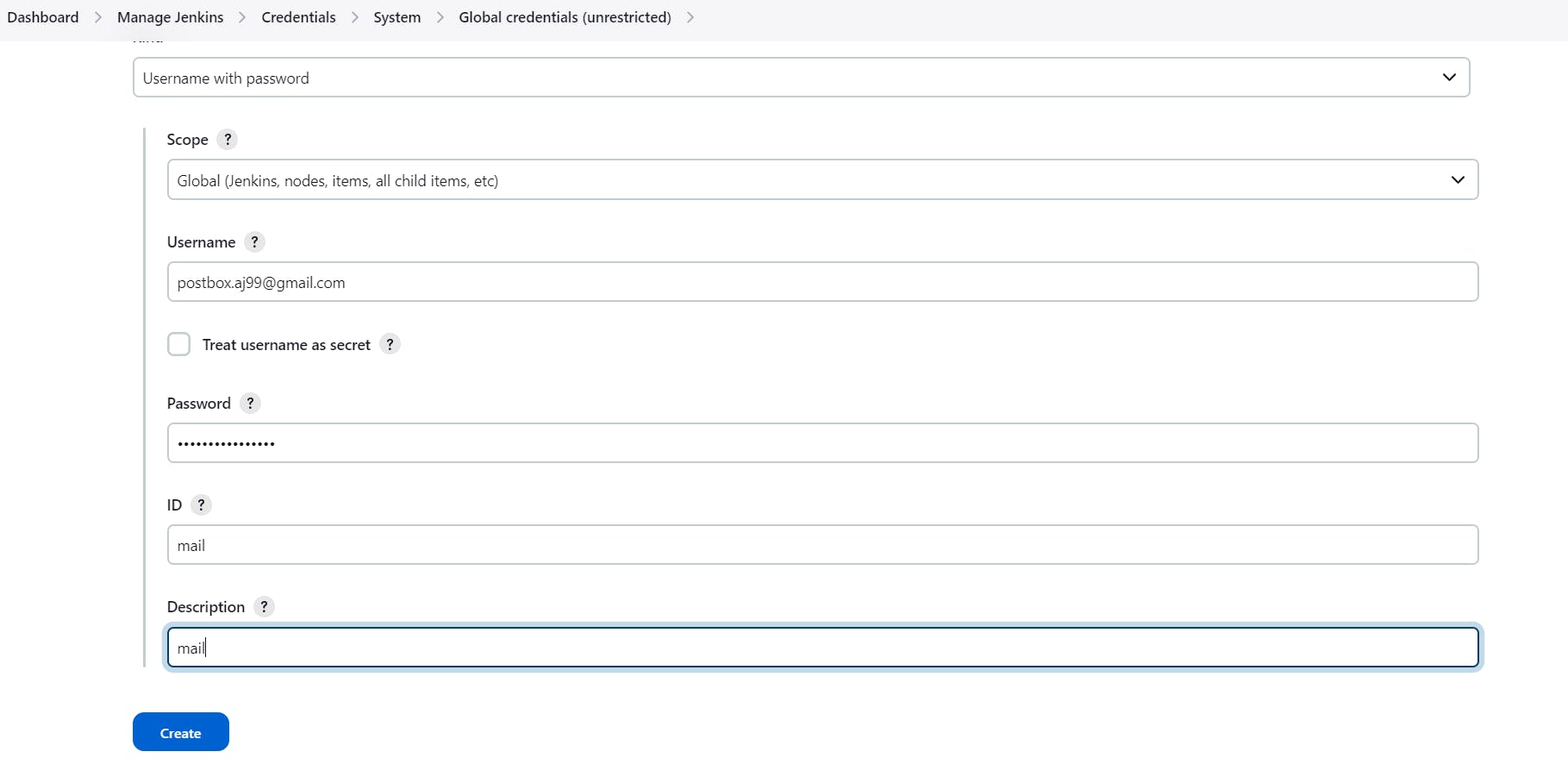
Once the plugin is installed in Jenkins, click on manage Jenkins --> configure system there under the E-mail Notification section configure the details as shown in the below image





Click on Apply and save.

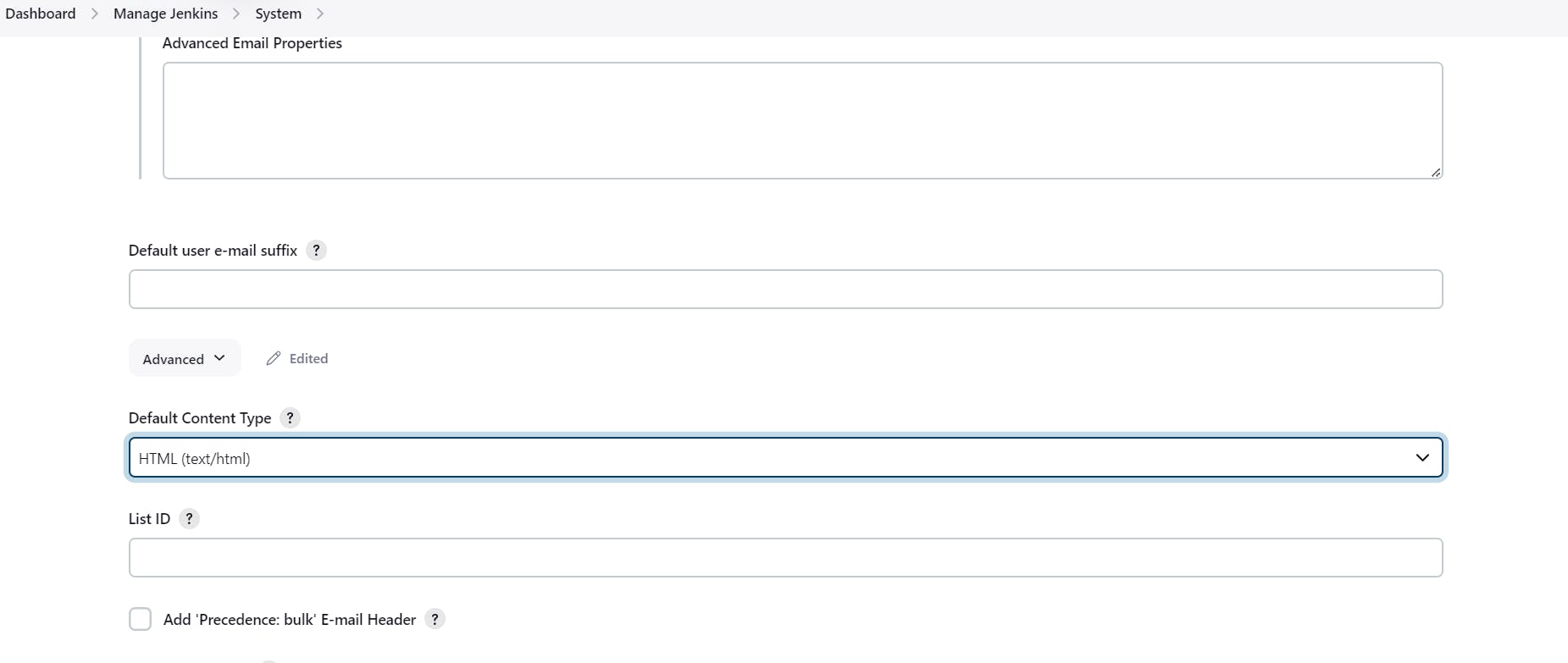
Click on Manage Jenkins--> credentials and add your mail username and generated password

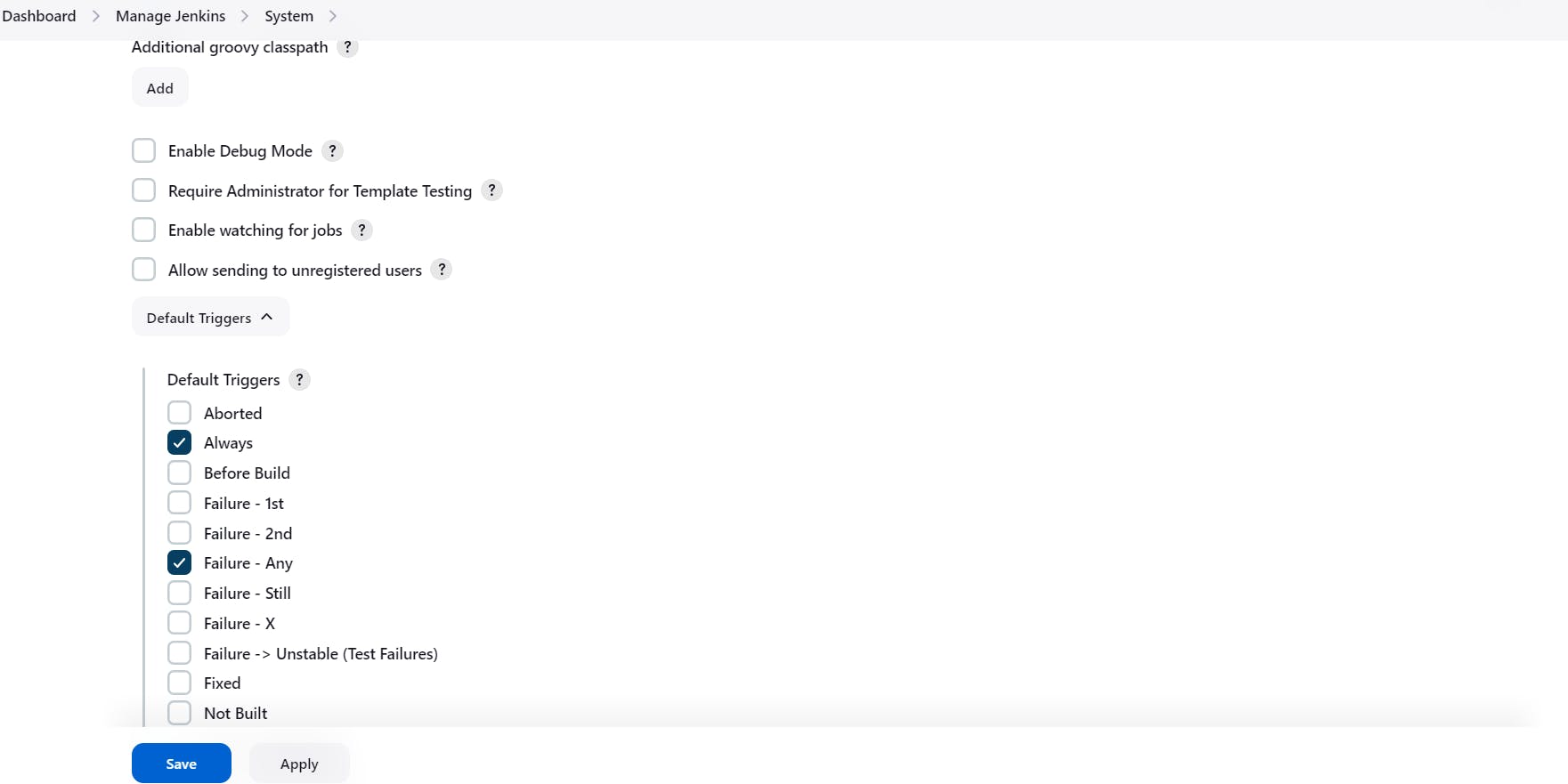


This is to just verify the mail configuration

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







Click on Apply and save.

final step to deploy on the Kubernetes cluster and email pipeline

stage('K8s'){

steps{

script{

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

#post block after stages

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

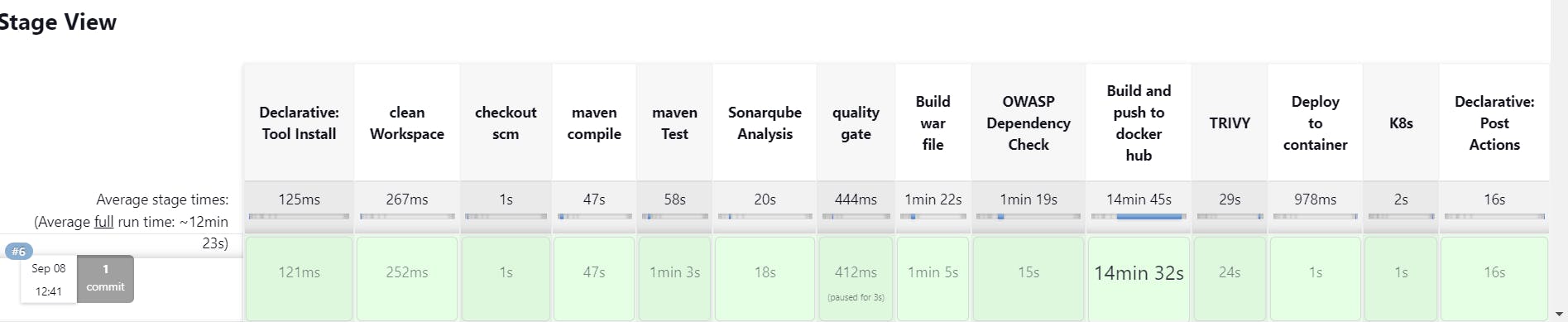
to: 'postbox.aj99@gmail.com',

attachmentsPattern: 'trivy.txt'

}

}

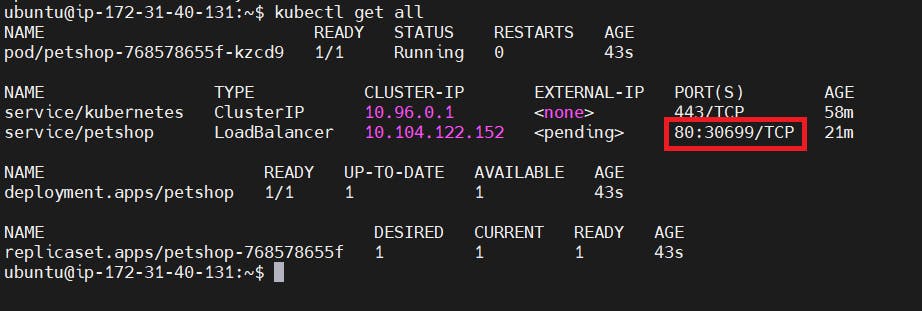
stage view



In the Kubernetes cluster give this command

kubectl get all

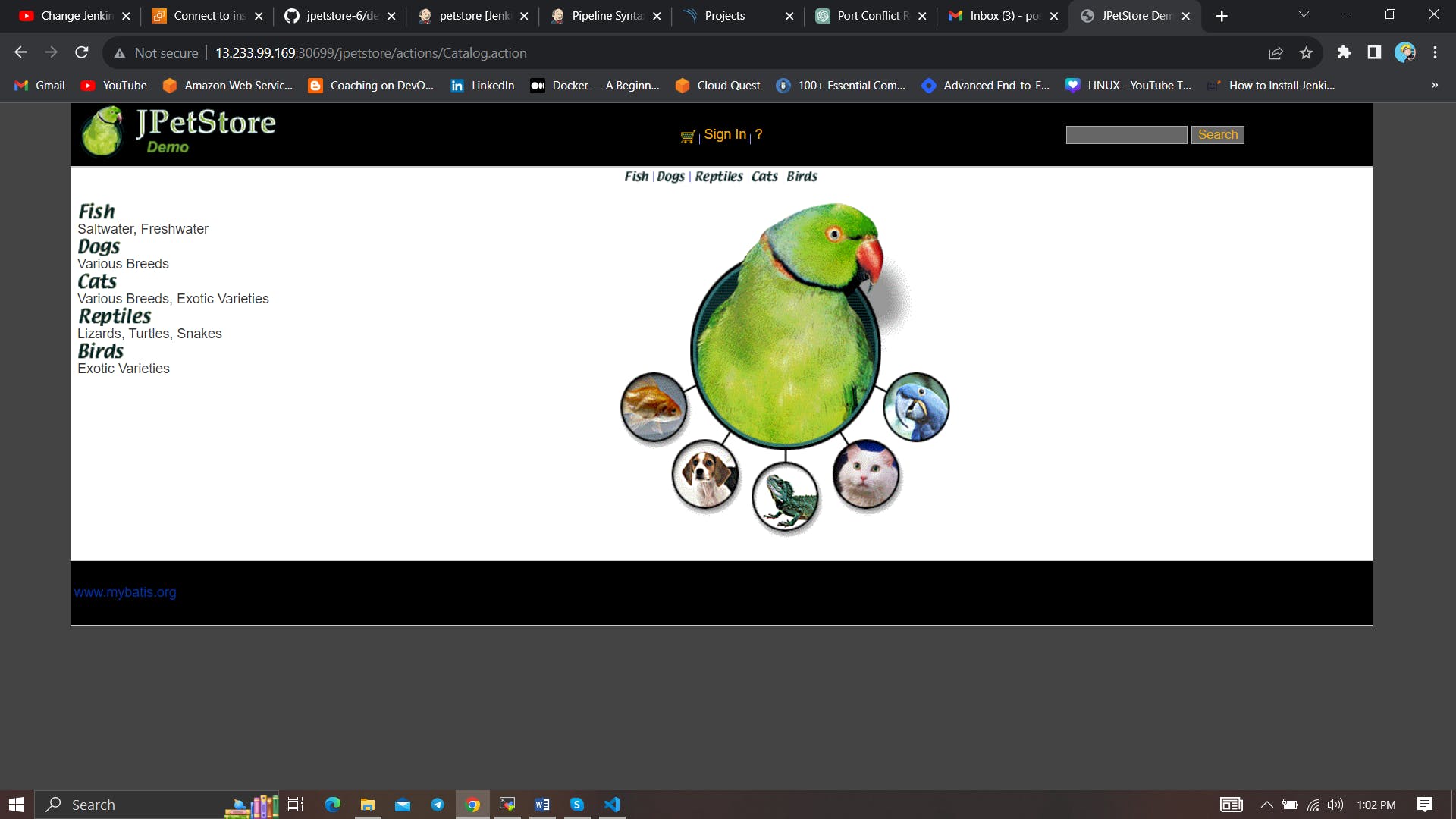
kubectl get svc



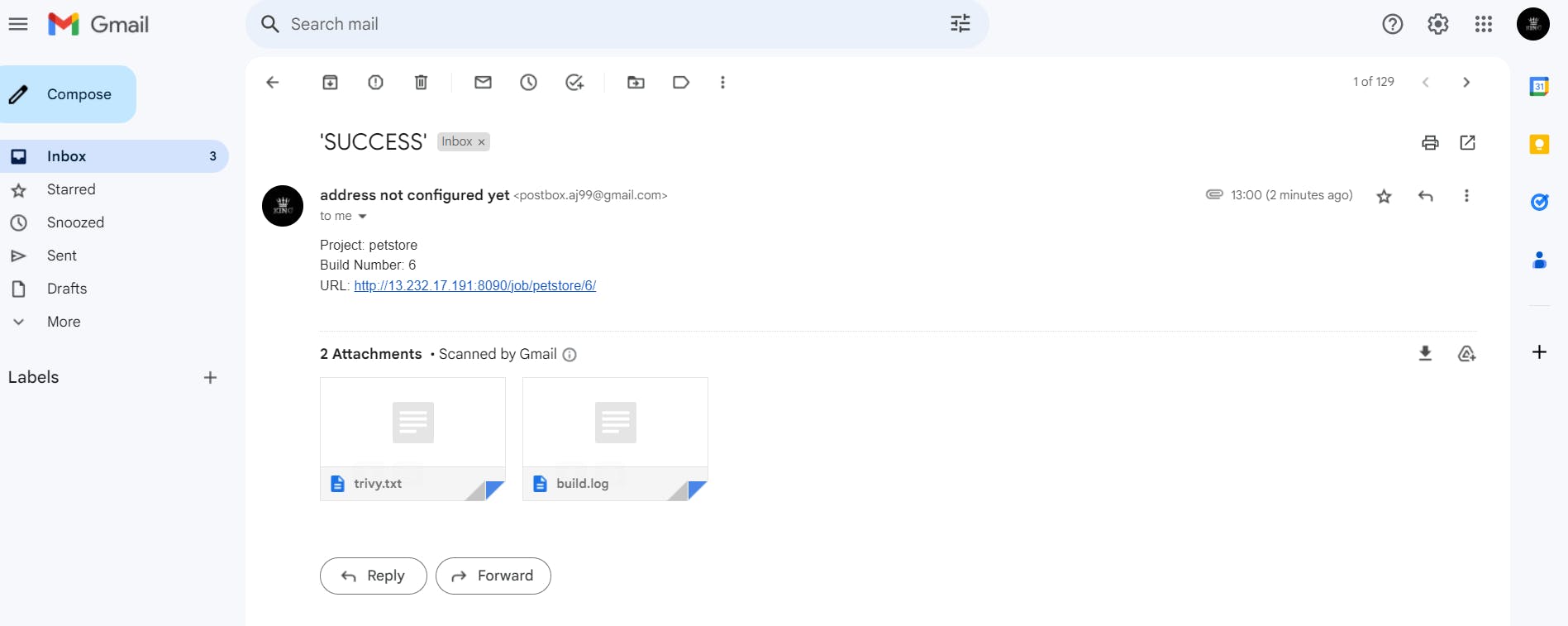
**STEP9:Access from a Web browser with**

<public-ip-of-slave:30699>

output:



Mail



**Step 10: Terminate instances.**

**Complete Pipeline**

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Petshop \

-Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop '''

}

}

}

stage("quality gate"){

steps {

script {

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

}

}

}

stage ('Build war file'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("OWASP Dependency Check"){

steps{

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

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

}

}

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

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

}

}

stage ('Deploy to container'){

steps{

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

}

}

stage('K8s'){

steps{

script{

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

}

post {

always {

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'

}

}

}

**Trigger code**

**CI-petshop-pipeline**

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('maven compile') {

steps {

sh 'mvn clean compile'

}

}

stage ('maven Test') {

steps {

sh 'mvn test'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Petshop \

-Dsonar.java.binaries=. \

-Dsonar.projectKey=Petshop '''

}

}

}

stage("quality gate"){

steps {

script {

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

}

}

}

stage ('Build war file'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("OWASP Dependency Check"){

steps{

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

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

}

}

stage ('Build and push to docker hub'){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh "docker build -t petshop ."

sh "docker tag petshop sevenajay/petshop:latest"

sh "docker push sevenajay/petshop:latest"

}

}

}

}

stage("TRIVY"){

steps{

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

}

}

stage("Trigger deployment"){

steps{

// Trigger the deployment pipeline and wait for it to complete

build job: 'CD-petshop', wait: true

}

}

}

post {

always {

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**

pipeline{

agent any

stages{

stage ('clean Workspace'){

steps{

cleanWs()

}

}

stage ('checkout scm') {

steps {

git 'https://github.com/Aj7Ay/jpetstore-6.git'

}

}

stage ('Deploy to container'){

steps{

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

}

}

stage('K8s'){

steps{

script{

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yaml'

}

}

}

}

}

post {

always {

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'

}

}

}