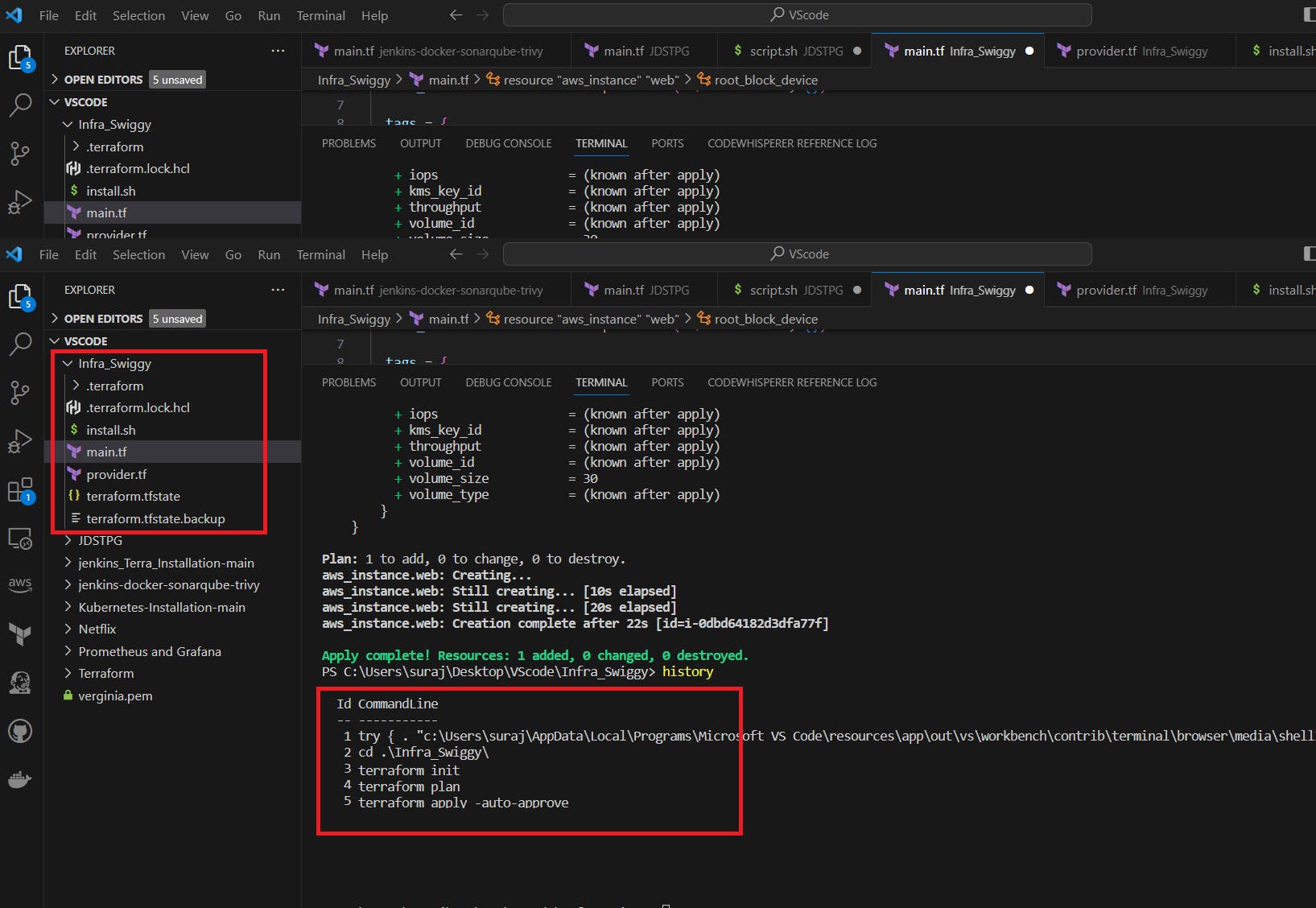
**Deploying a Swiggy clone app using Kubernetes.**

**Let's use Terraform to create an EC2 instance for Jenkins, Docker, and SonarQube.**



[main.tf](http://main.tf/)**:**

resource "aws\_instance" "web" {

ami = "ami-0287a05f0ef0e9d9a" #change ami id for different region

instance\_type = "t2.large"

key\_name = "mumbai" #change key name as per your setup

vpc\_security\_group\_ids = [aws\_security\_group.Jenkins-sg.id]

user\_data = templatefile("./install.sh", {})

tags = {

Name = "Jenkins-sonarqube"

}

root\_block\_device {

volume\_size = 30

}

}

resource "aws\_security\_group" "Jenkins-sg" {

name = "Jenkins-sg"

description = "Allow TLS inbound traffic"

ingress = [

for port in [22, 80, 443, 8080, 9000, 3000] : {

description = "inbound rules"

from\_port = port

to\_port = port

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = []

prefix\_list\_ids = []

security\_groups = []

self = false

}

]

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "jenkins-sg"

}

}

[provider.tf](http://provider.tf/)

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "ap-south-1" #change your region

}

[install.sh](http://install.sh/)

#!/bin/bash

sudo apt update -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

#install docker

sudo apt-get update

sudo apt-get install docker.io -y

sudo usermod -aG docker ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

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

#install trivy

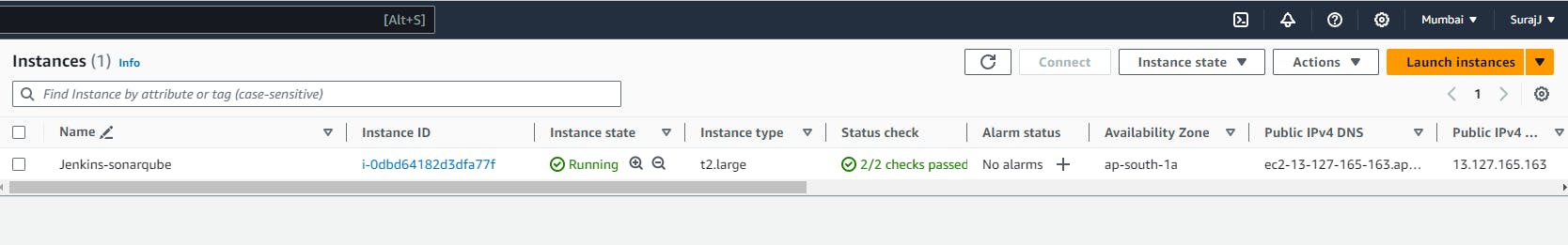
sudo apt-get install wget apt-transport-https gnupg lsb-release -y

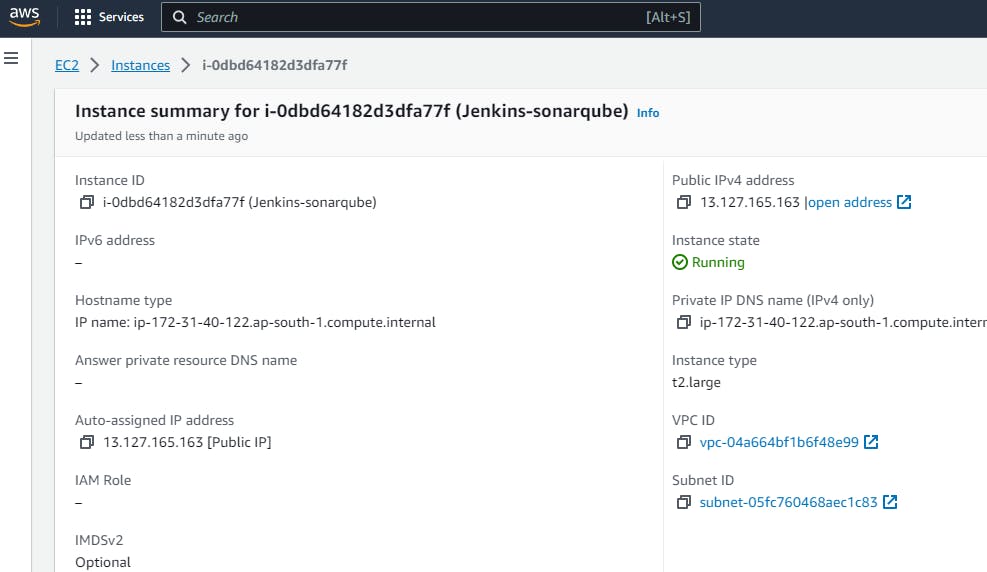
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null

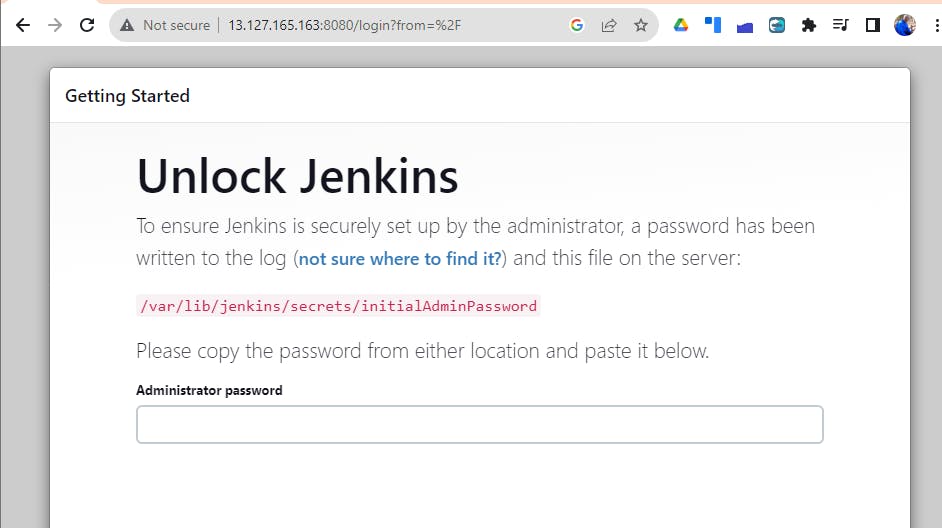
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list

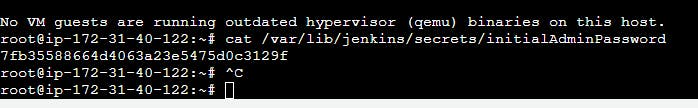
sudo apt-get update

sudo apt-get install trivy -y

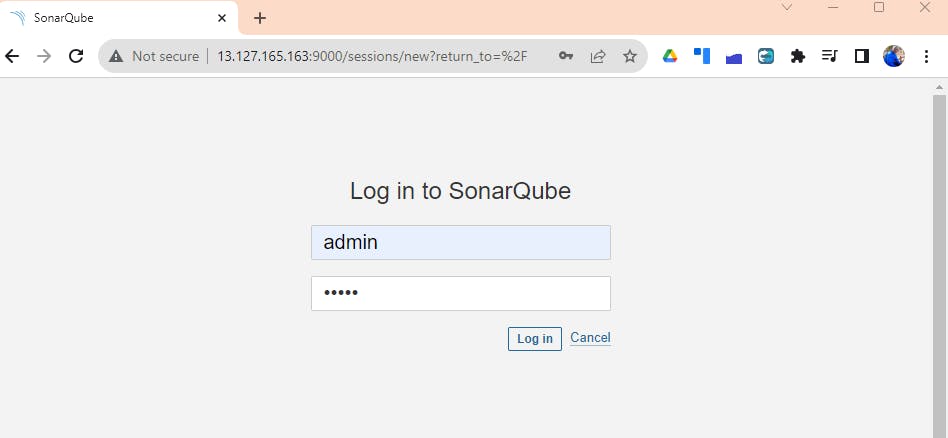








Use > admin >admin

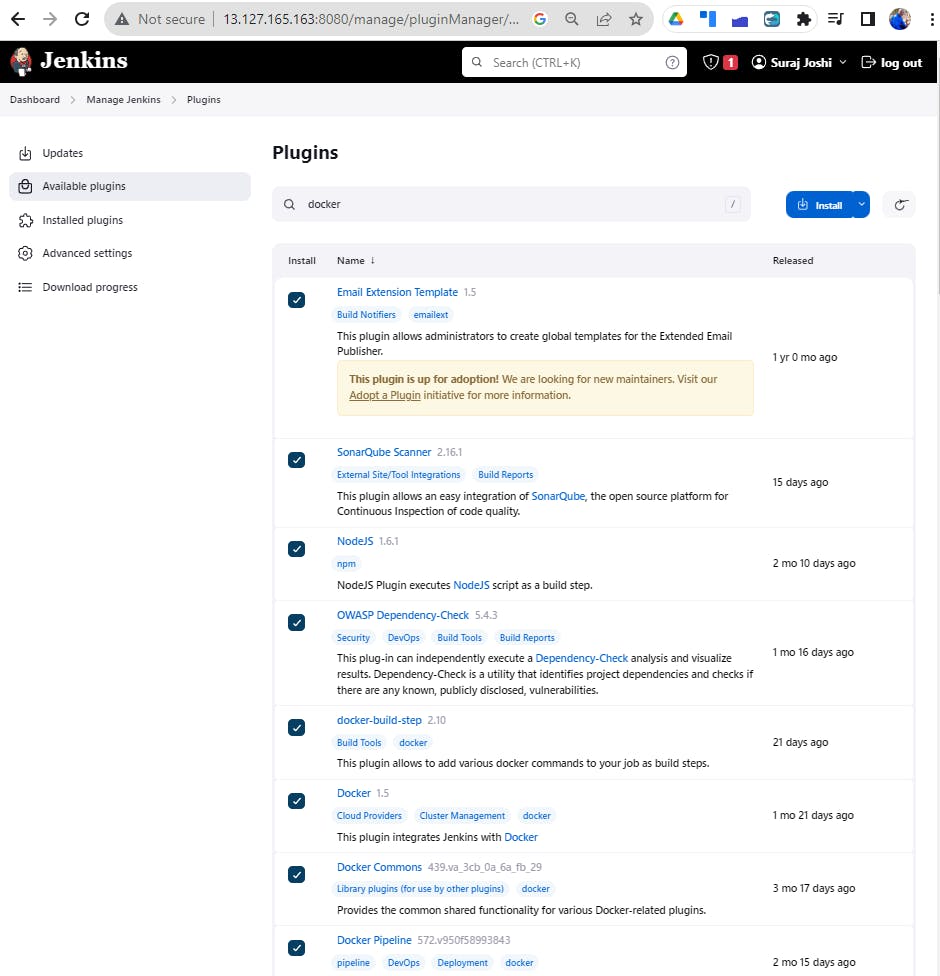


**Now Install the Plugins without restarting Jenkins -**

**Eclipse Temurin installer**  
**Sonarqube Scanner**  
**NodeJs**  
**OWASP Dependency Check**  
**Docker**  
**Docker Commons**  
**Docker Pipeline**  
**Docker API &**  
**Docker-build-step**

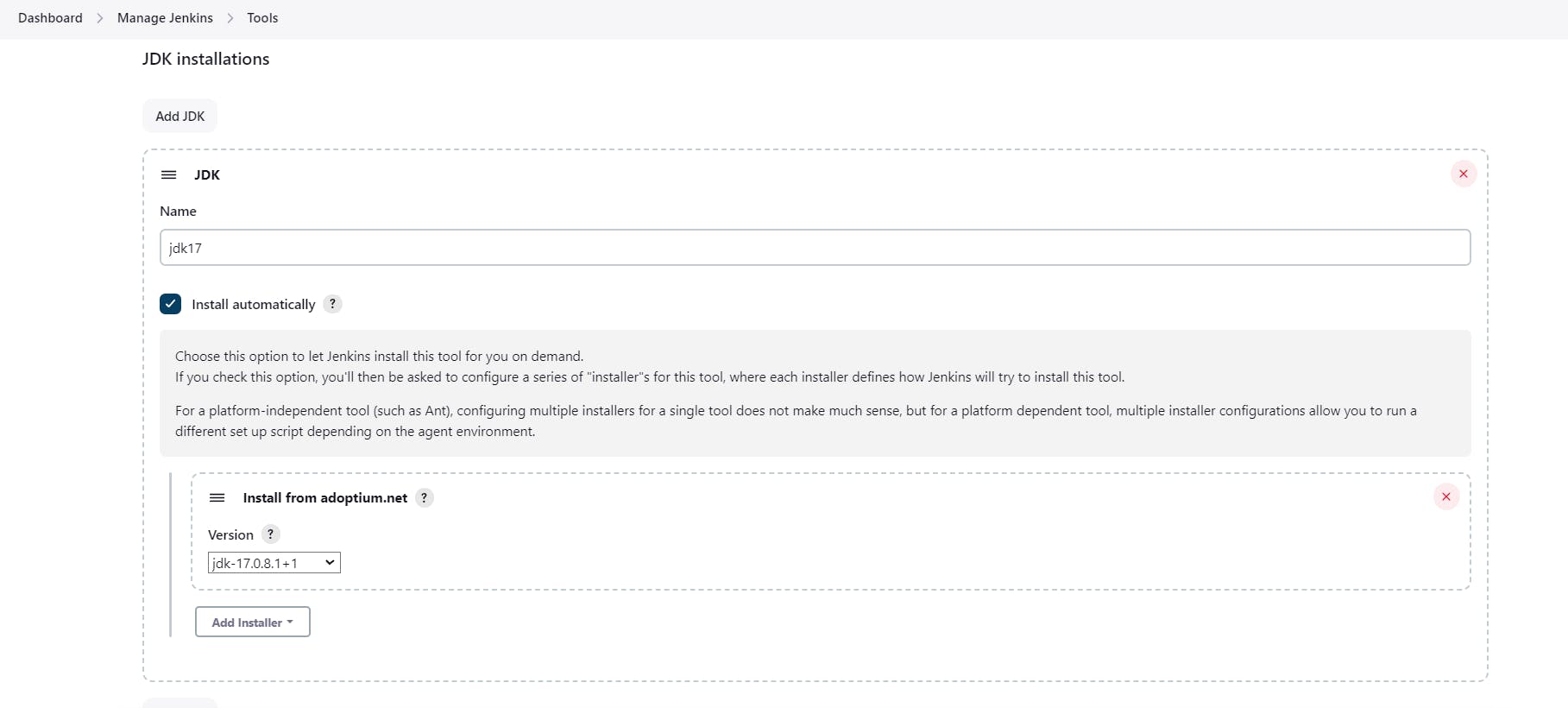
**Install Each plugin like the below:-**



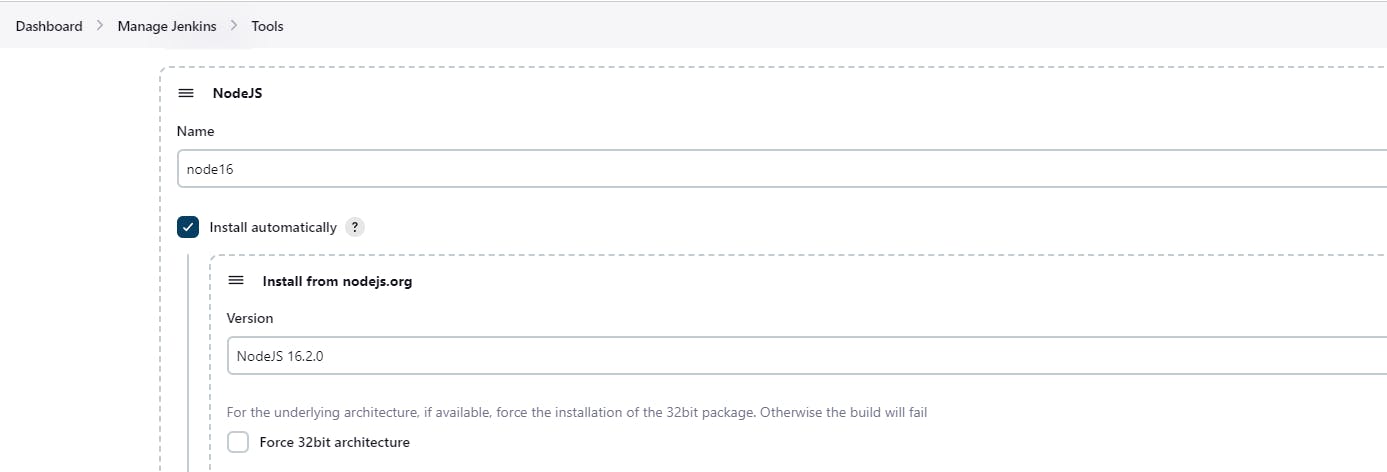


**Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save**

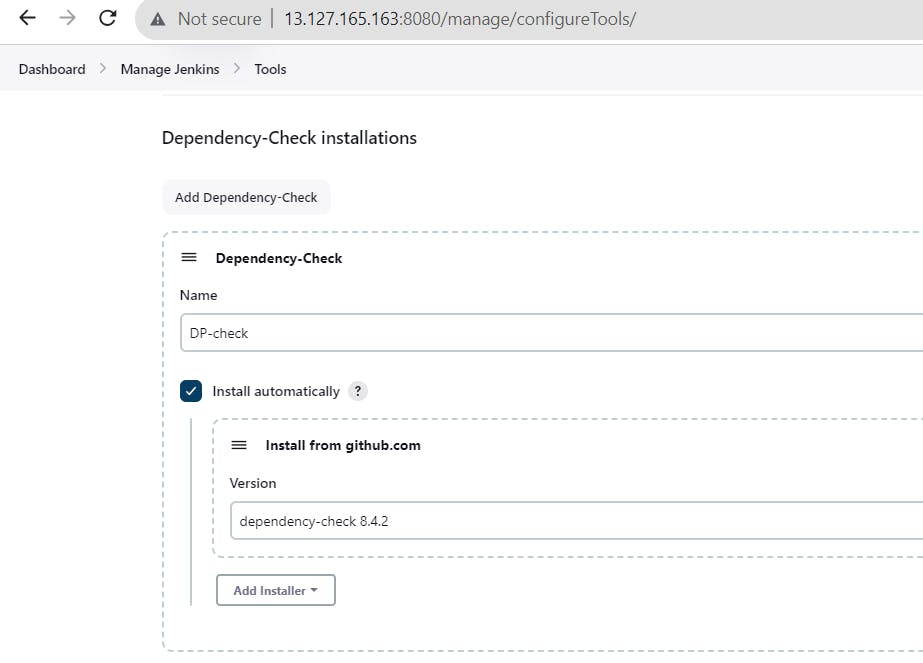
JDK



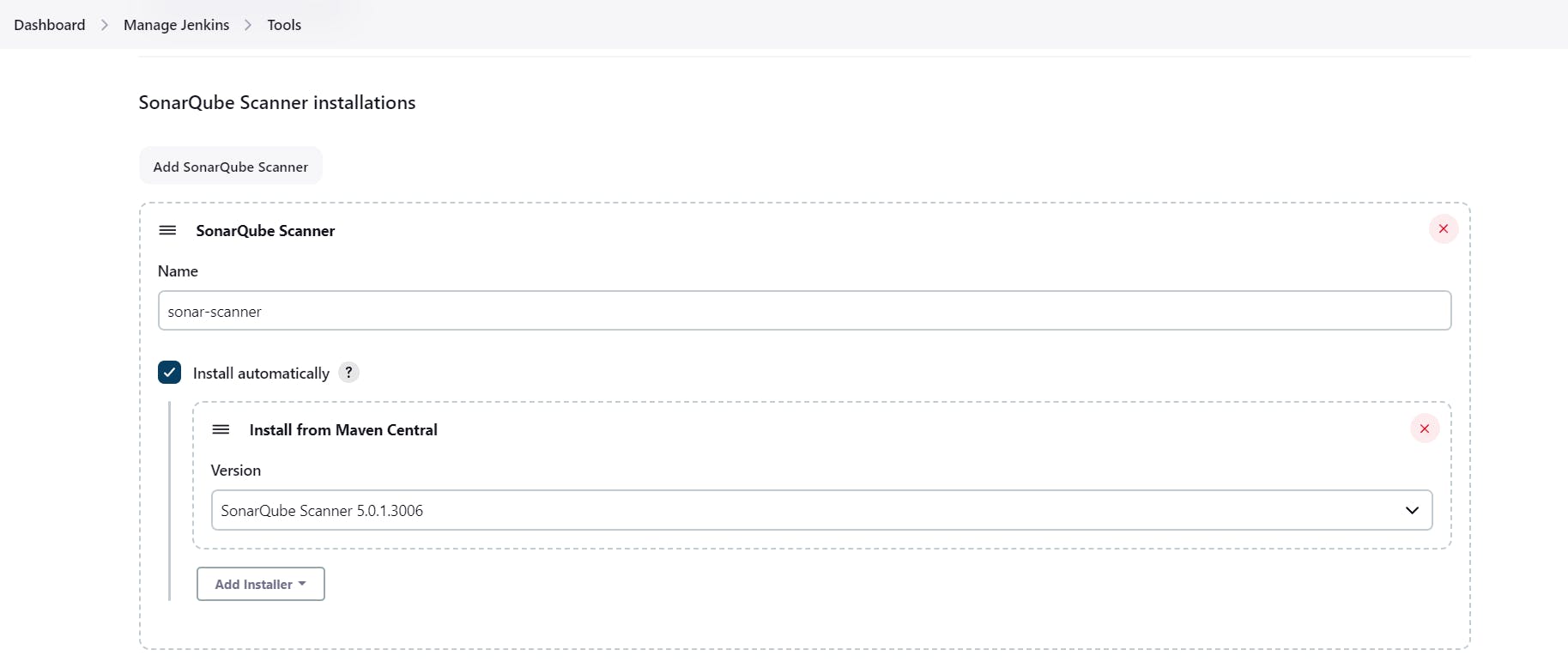
**NodeJS**



**Dependency-Check**

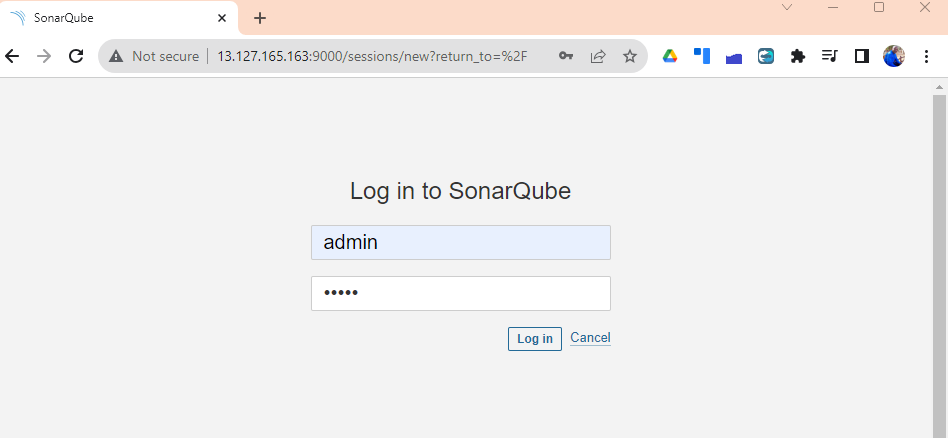


**SonarQube Scanner installations :**

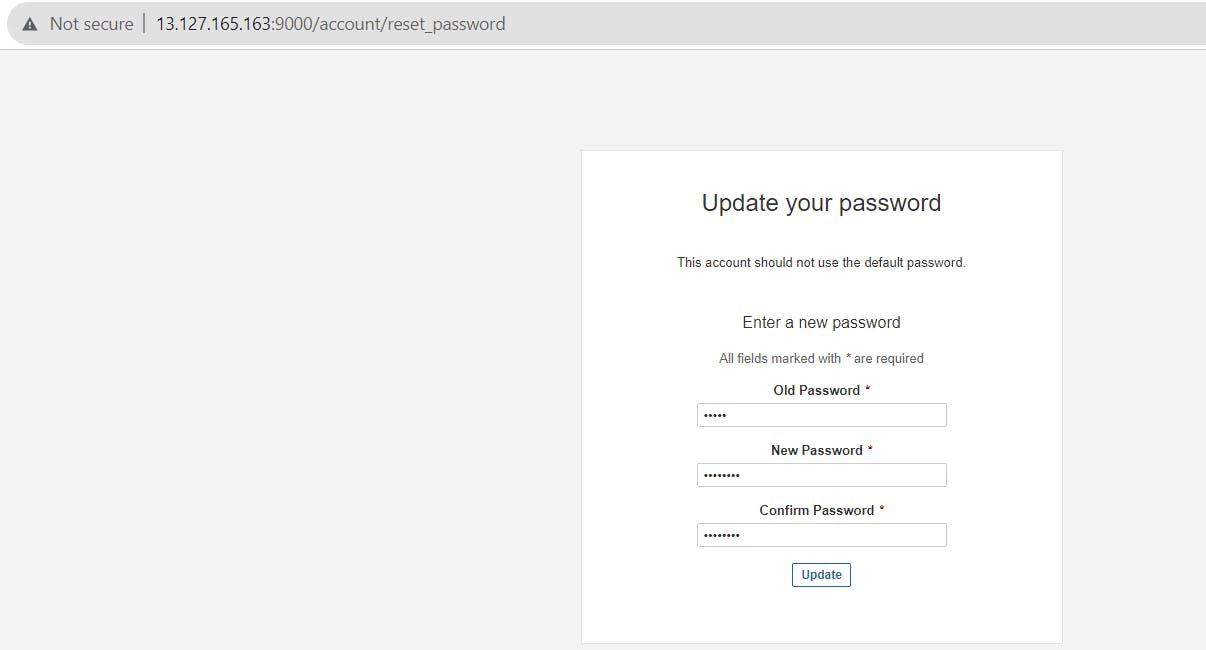


Retrieve the Public IP Address of your EC2 Instance, and access Sonarqube running on Port 9000, please follow these steps:

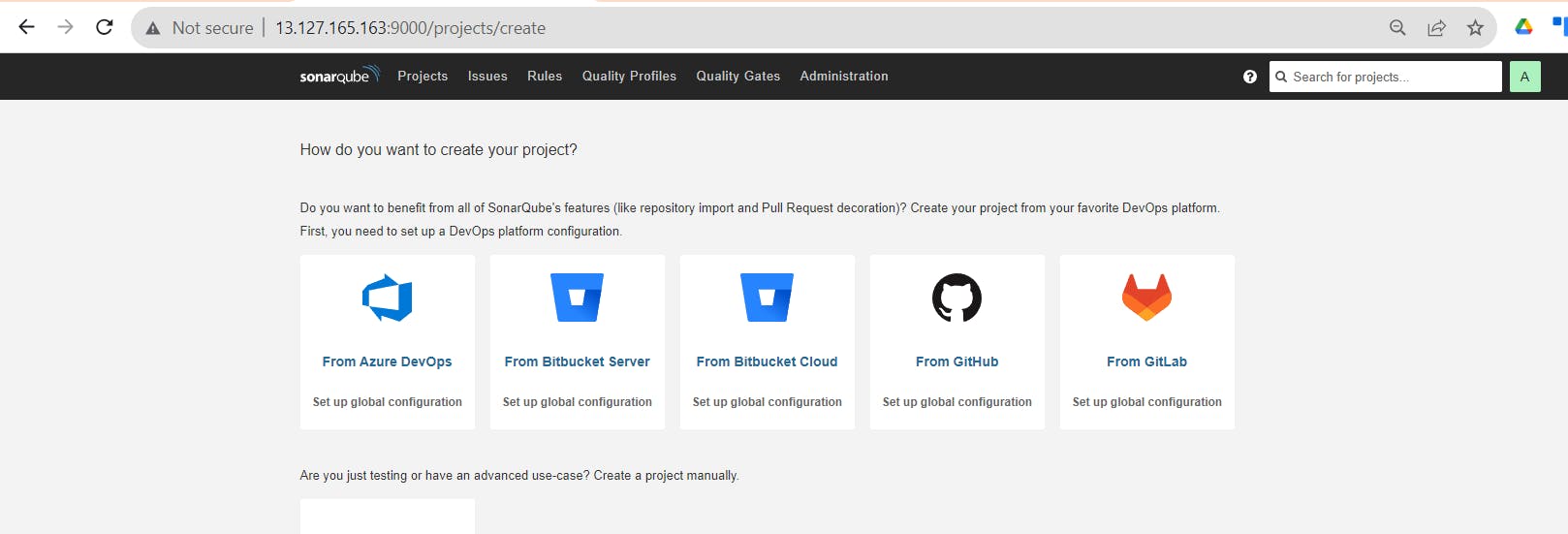
1. Obtain the Public IP Address of your EC2 Instance.
2. Using a web browser, go to your Sonarqube Server's Public IP address followed by port 9000 (e.g., <Public IP>:9000).  
   **Use admin --> admin**



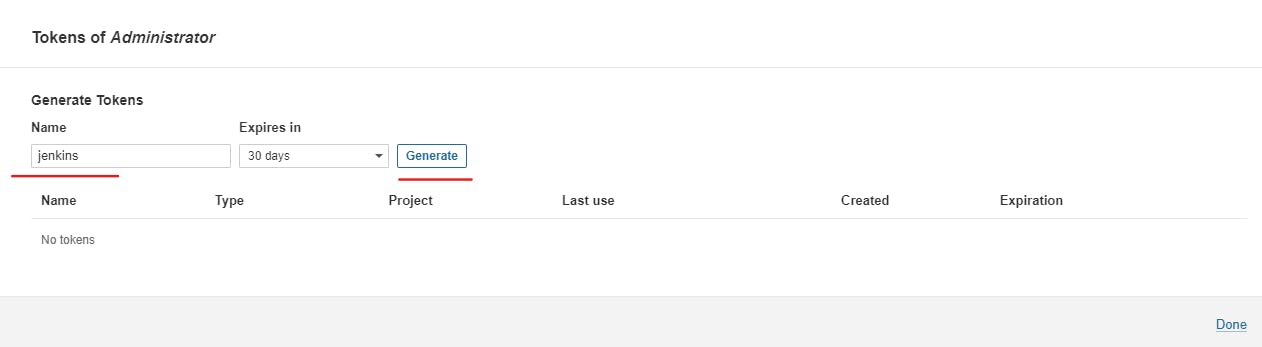
Setup your Password:



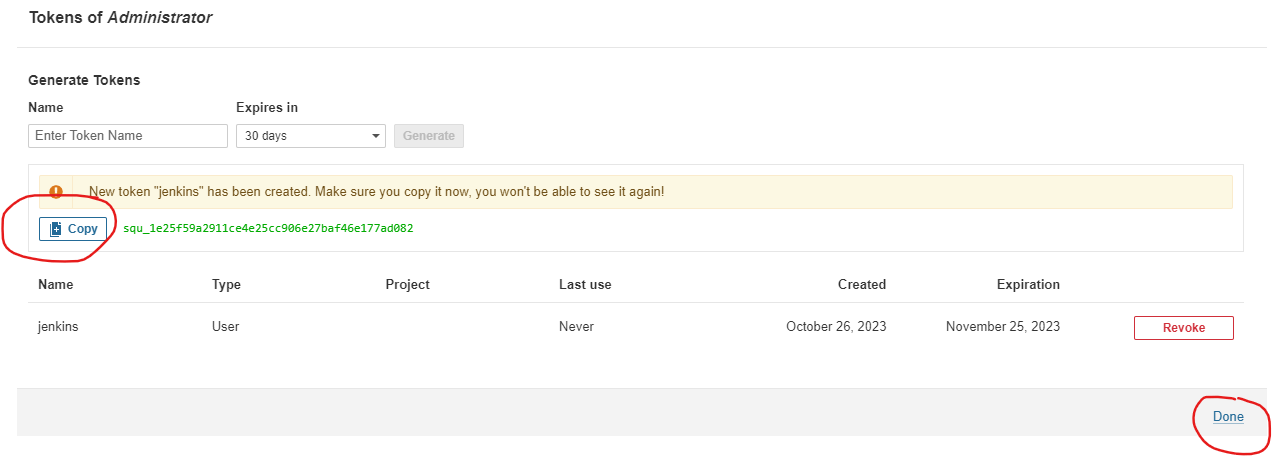
1. Navigate to the Sonarqube Administration section.



1. In the Administration section, locate and click on the "Security" option.
2. Within the Security section, find and select "Users."
3. Click on the "Tokens" option.
4. Provide a name for the token to easily identify its purpose.

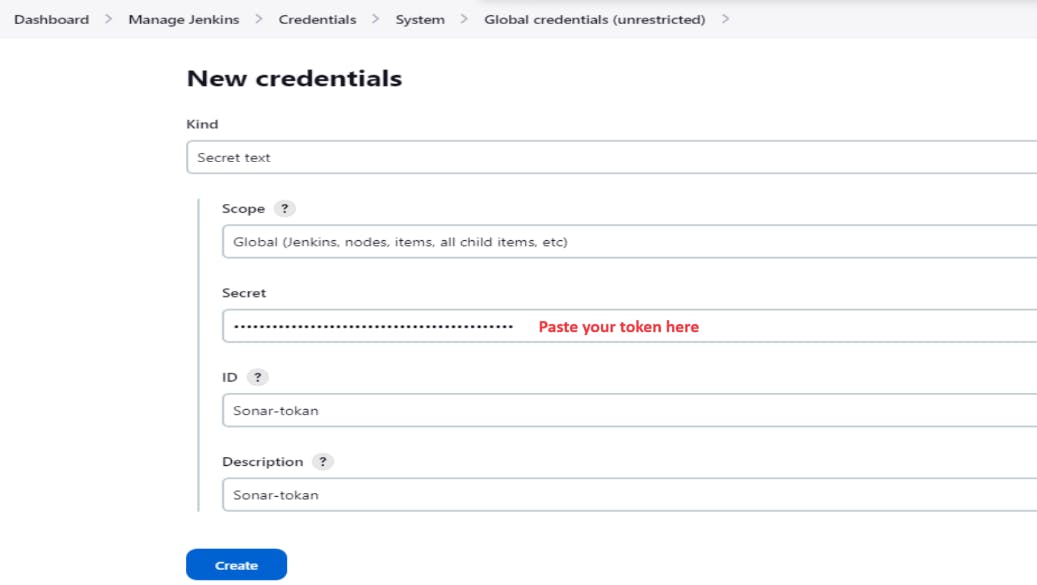


Click on the "Generate Token" button to create the token.

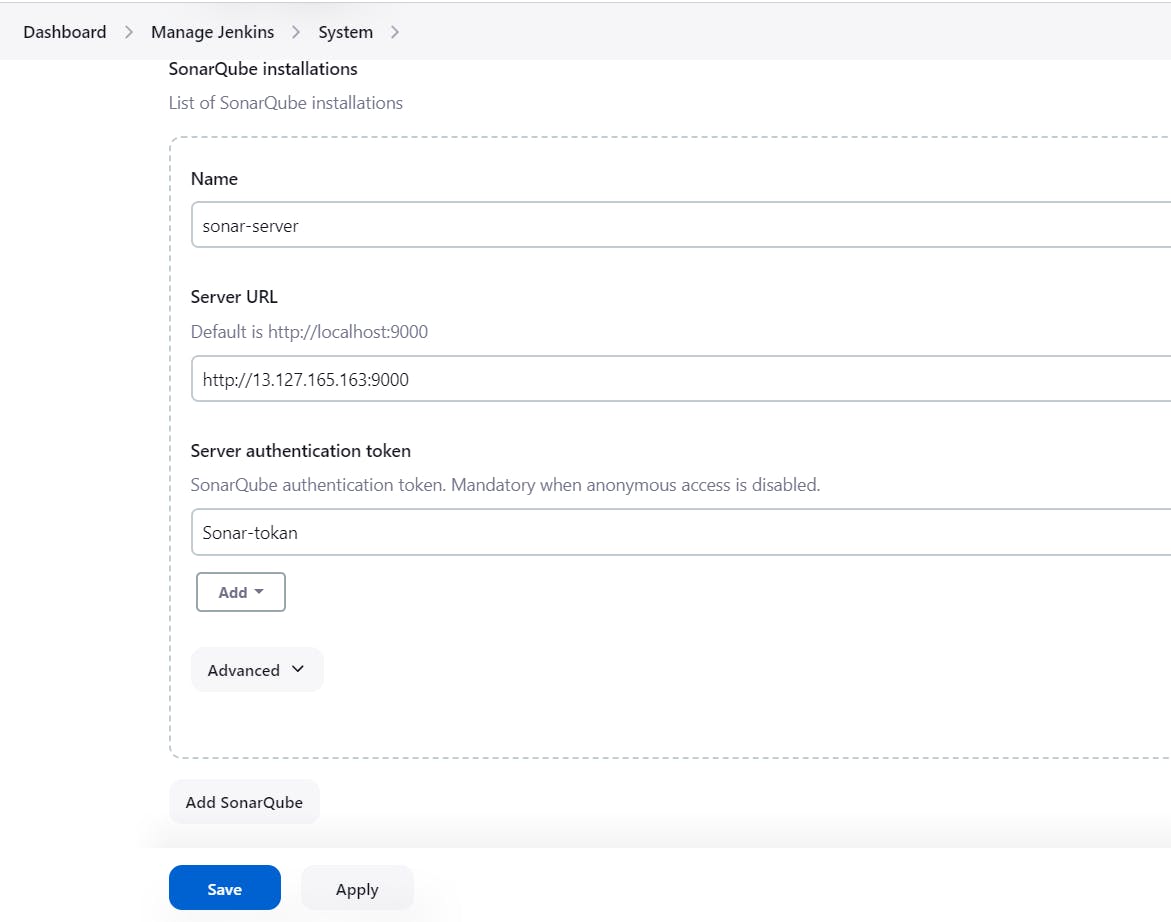


copy this token & keep it with you.

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

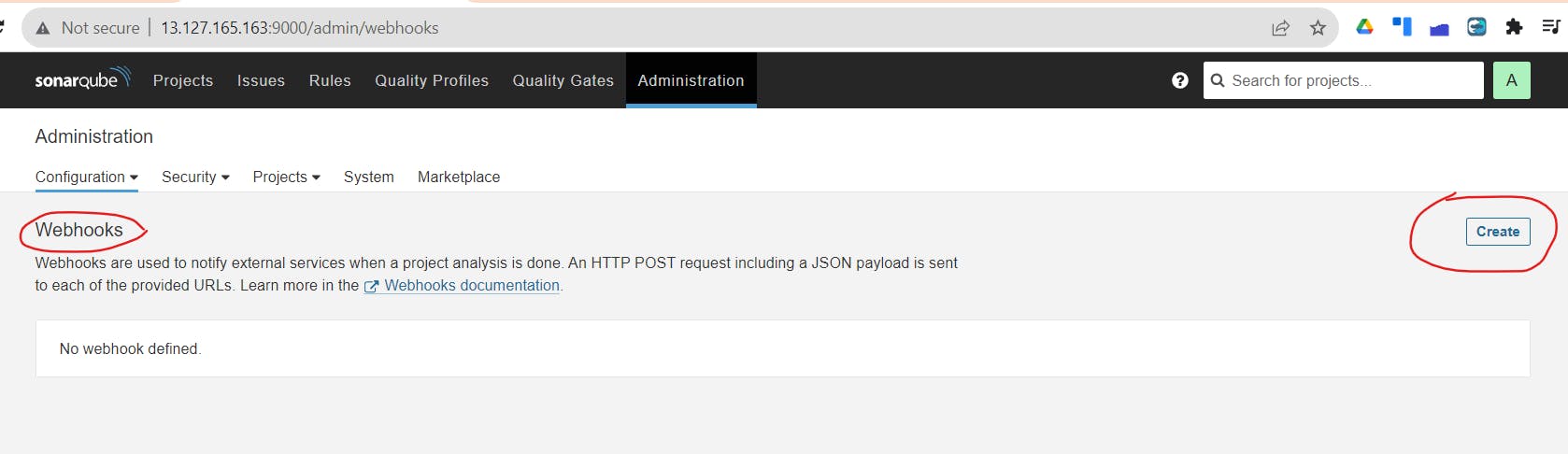


Now, go to Dashboard → Manage Jenkins → System

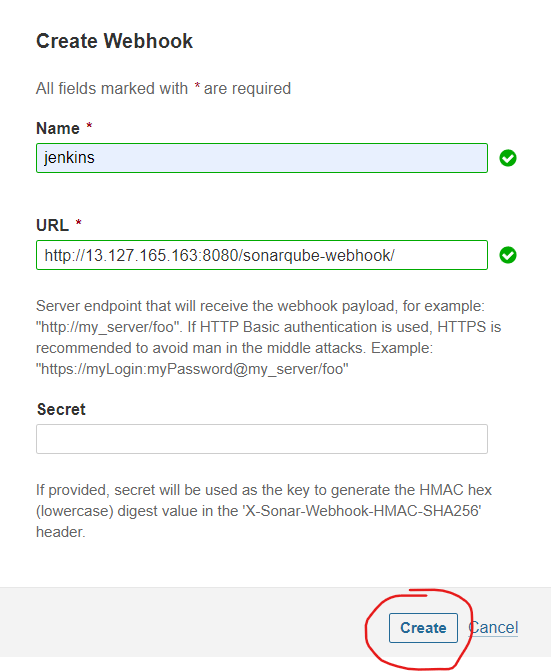


In the Sonarqube Dashboard add a quality gate also

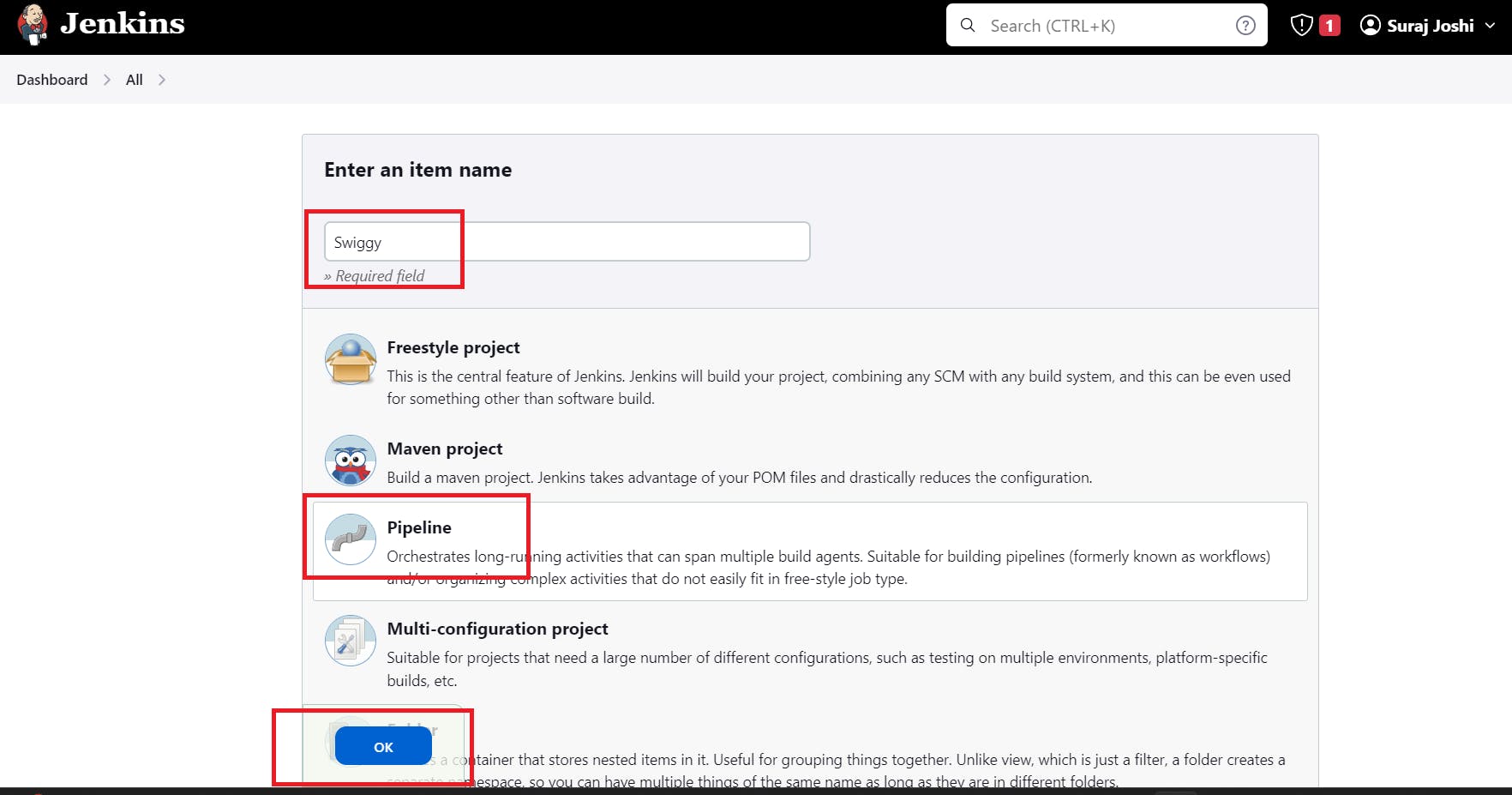
Administration--> Configuration-->Webhooks



Add details Link below & create



Now Go to Dashboard > New Item



Now, let's navigate to our pipeline configuration and include the script

COPY

COPY

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

agent any

tools{

jdk 'jdk17'

nodejs 'node16'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages {

stage('clean workspace'){

steps{

cleanWs()

}

}

stage('Checkout from Git'){

steps{

git branch: 'main', url: 'https://github.com/surajpjoshi/Swiggy-clone.git'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

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

-Dsonar.projectKey=Swiggy '''

}

}

}

stage("quality gate"){

steps {

script {

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

}

}

}

stage('Install Dependencies') {

steps {

sh "npm install"

}

}

stage('OWASP FS SCAN') {

steps {

dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'

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

}

}

stage('TRIVY FS SCAN') {

steps {

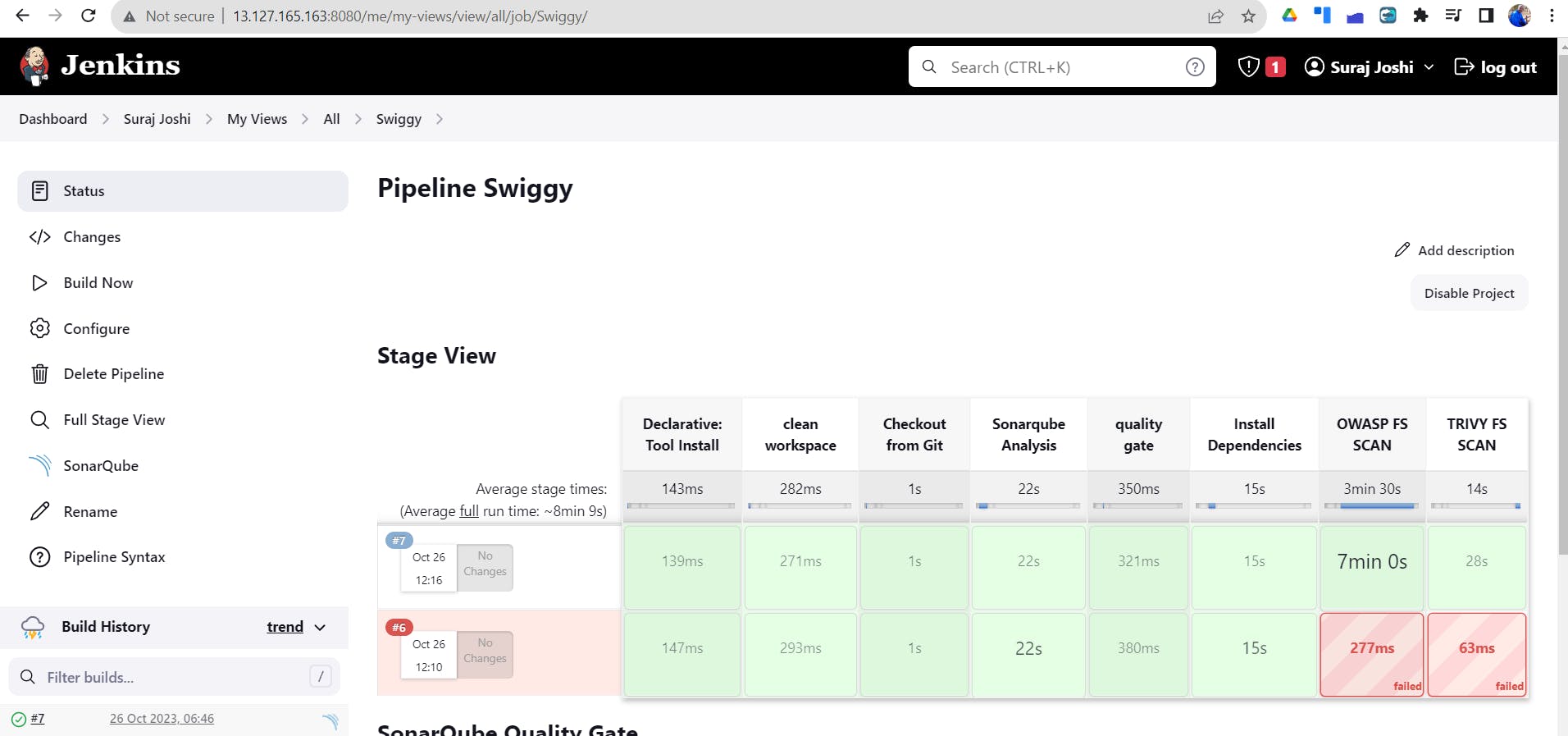
sh "trivy fs . > trivyfs.txt"

}

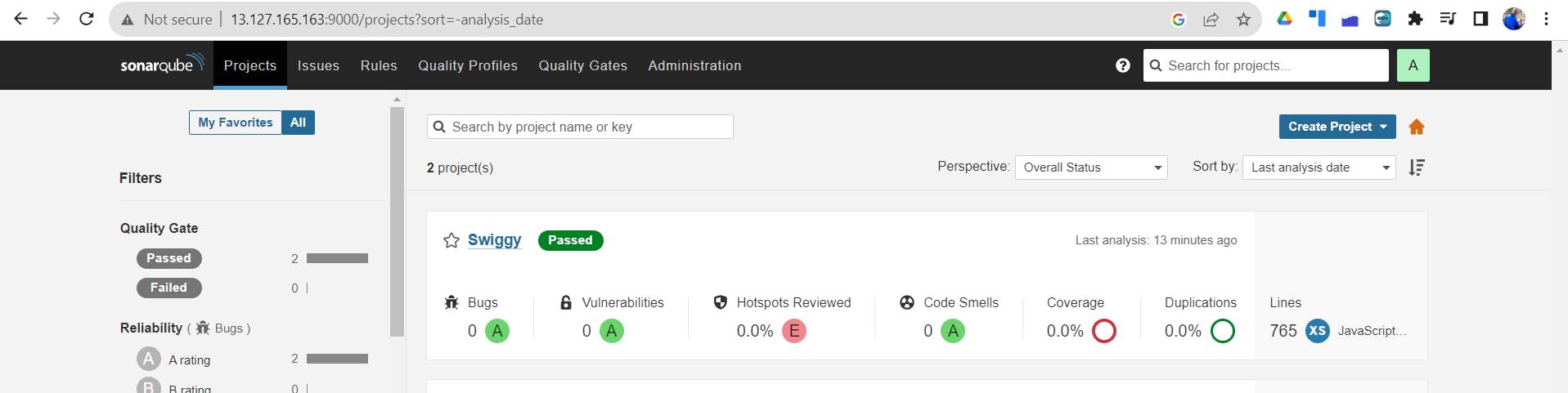
}

}

}



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



**Docker Image Build and Push**

COPY

COPY

stage("Docker Build & Push"){

steps{

script{

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

sh "docker build -t swiggy ."

sh "docker tag swiggy surajpjoshi/swiggy:latest "

sh "docker push surajpjoshi/swiggy:latest "

}

}

}

}

stage("TRIVY"){

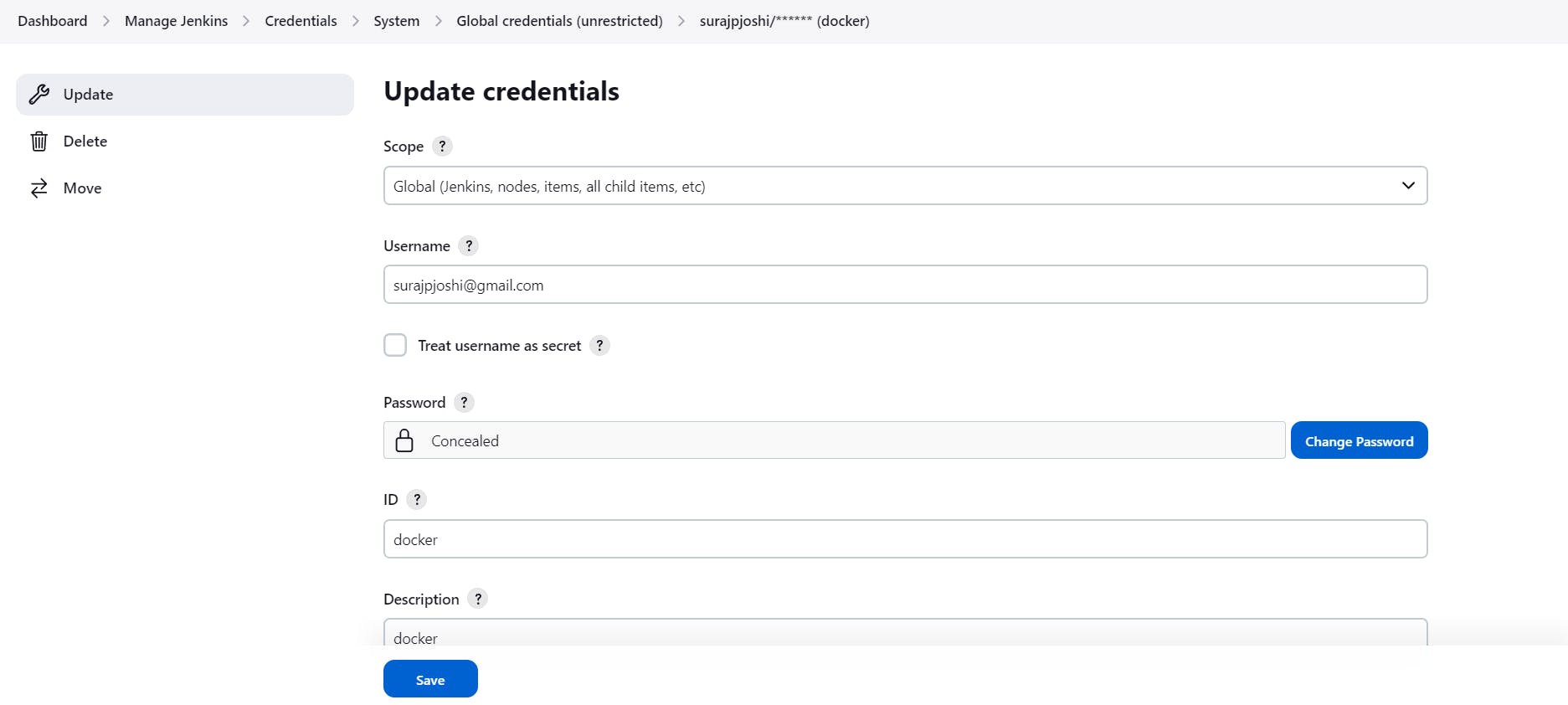
steps{

sh "trivy image surajpjoshi/swiggy:latest > trivyimage.txt"

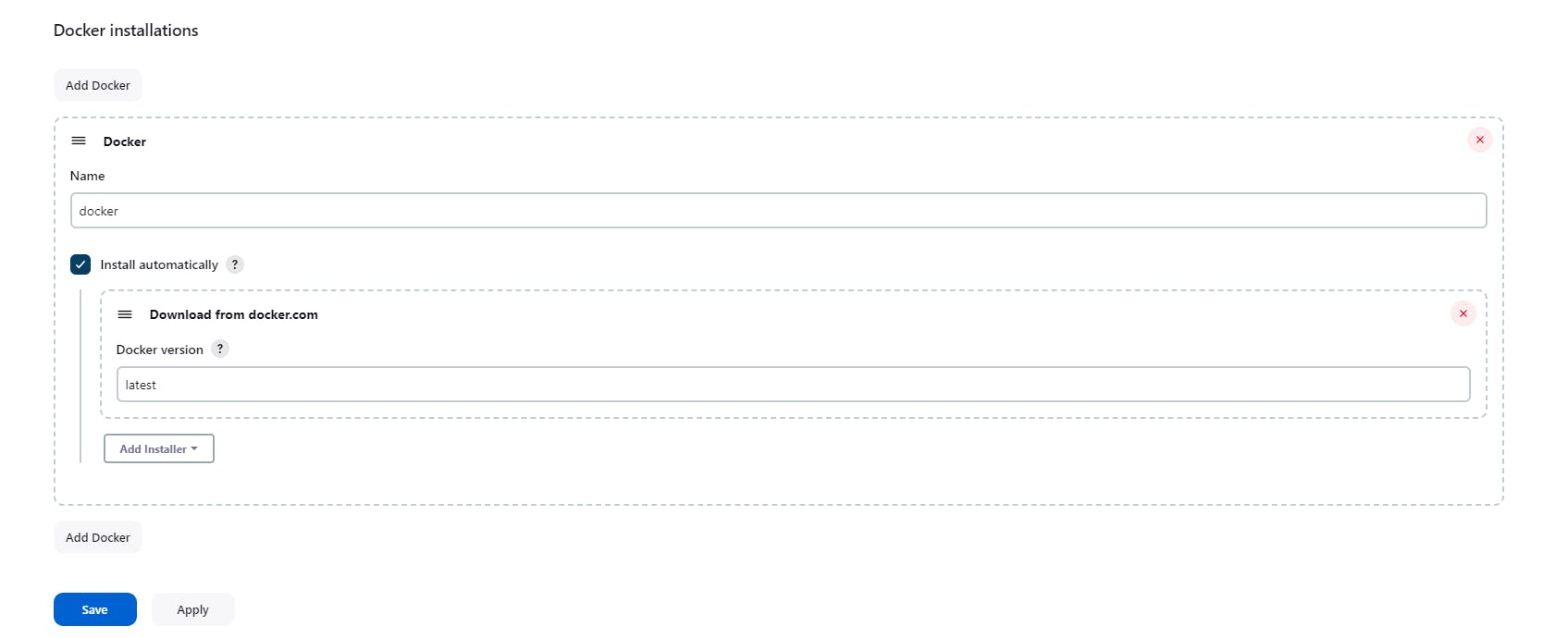
}

}

Add DockerHub Username and Password under Global Credentials



Now, goto Dashboard → Manage Jenkins → Tools →



Add this stage to Pipeline Script

COPY

COPY

stage("Docker Build & Push"){

steps{

script{

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

sh "docker build -t swiggy ."

sh "docker tag swiggy surajpjoshi/swiggy:latest "

sh "docker push surajpjoshi/swiggy:latest "

}

}

}

}

stage("TRIVY"){

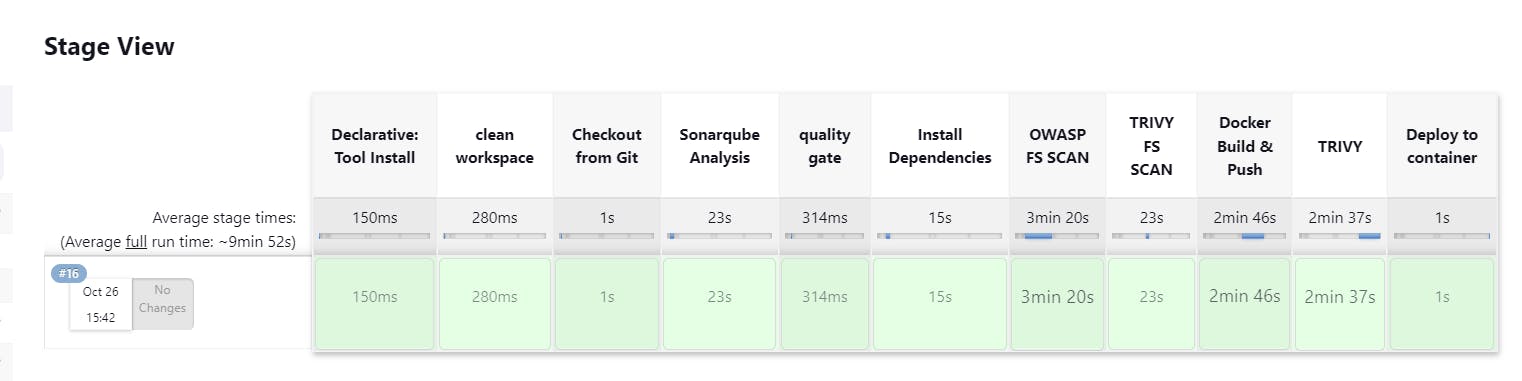
steps{

sh "trivy image surajpjoshi/swiggy:latest > trivyimage.txt"

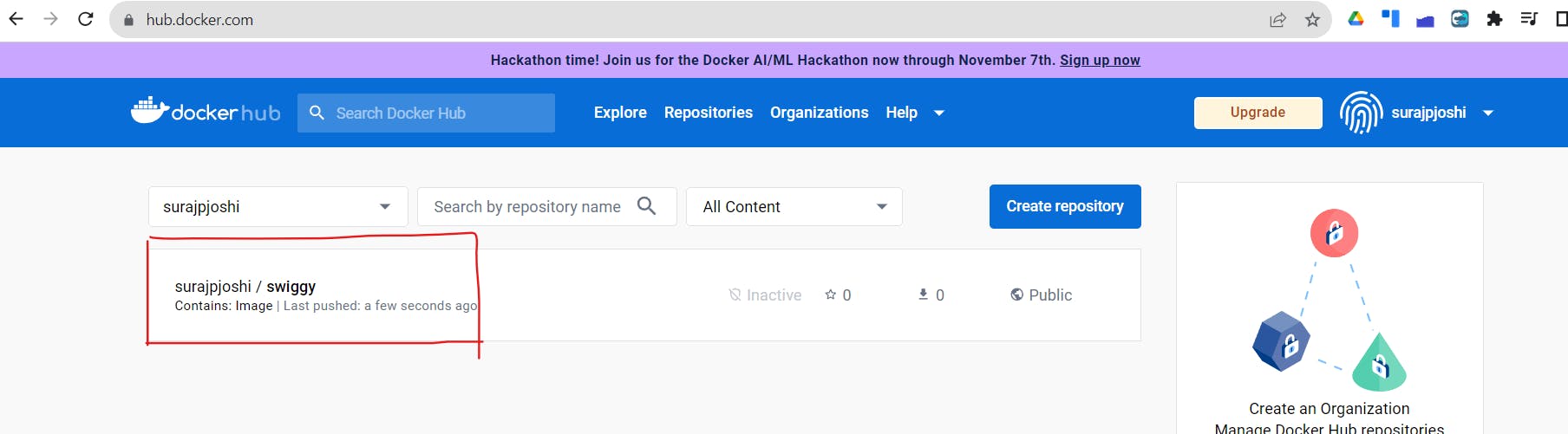
}

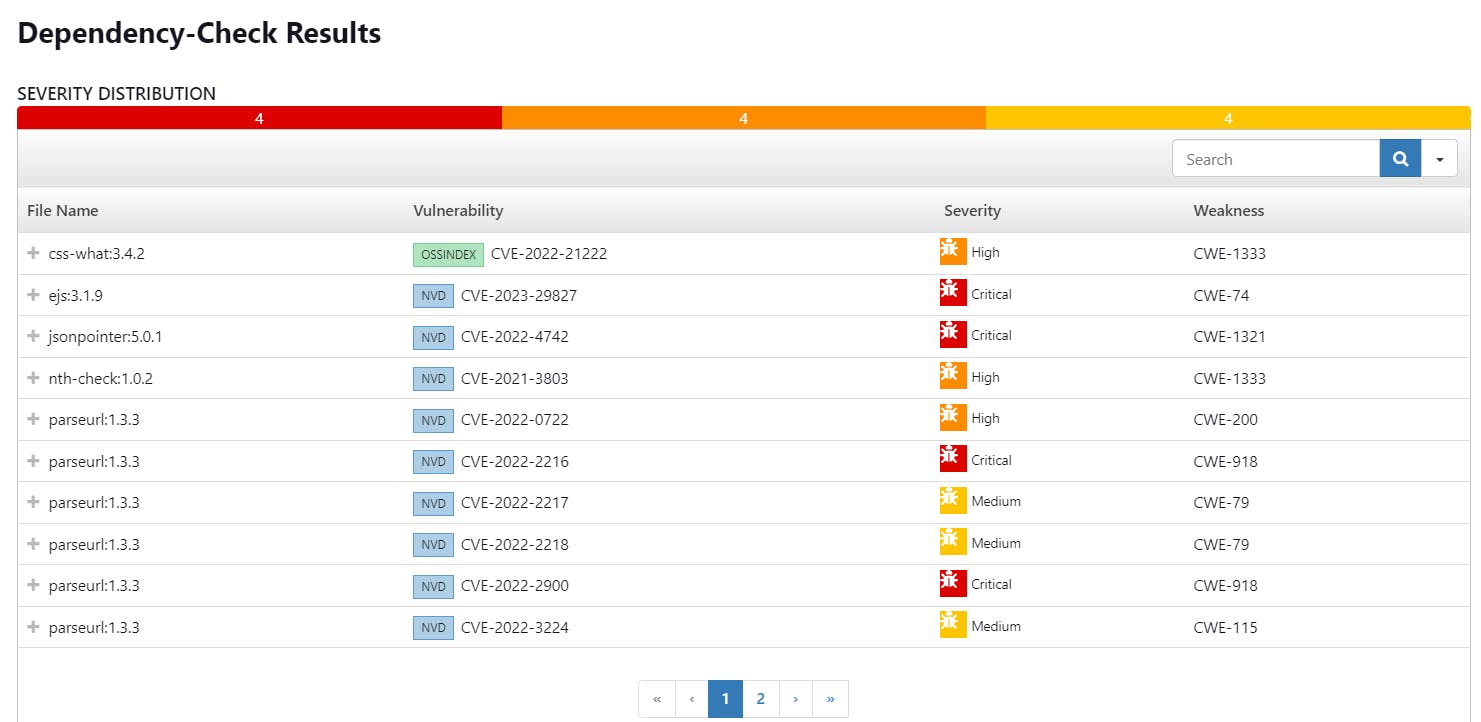
}

**Stage View -**

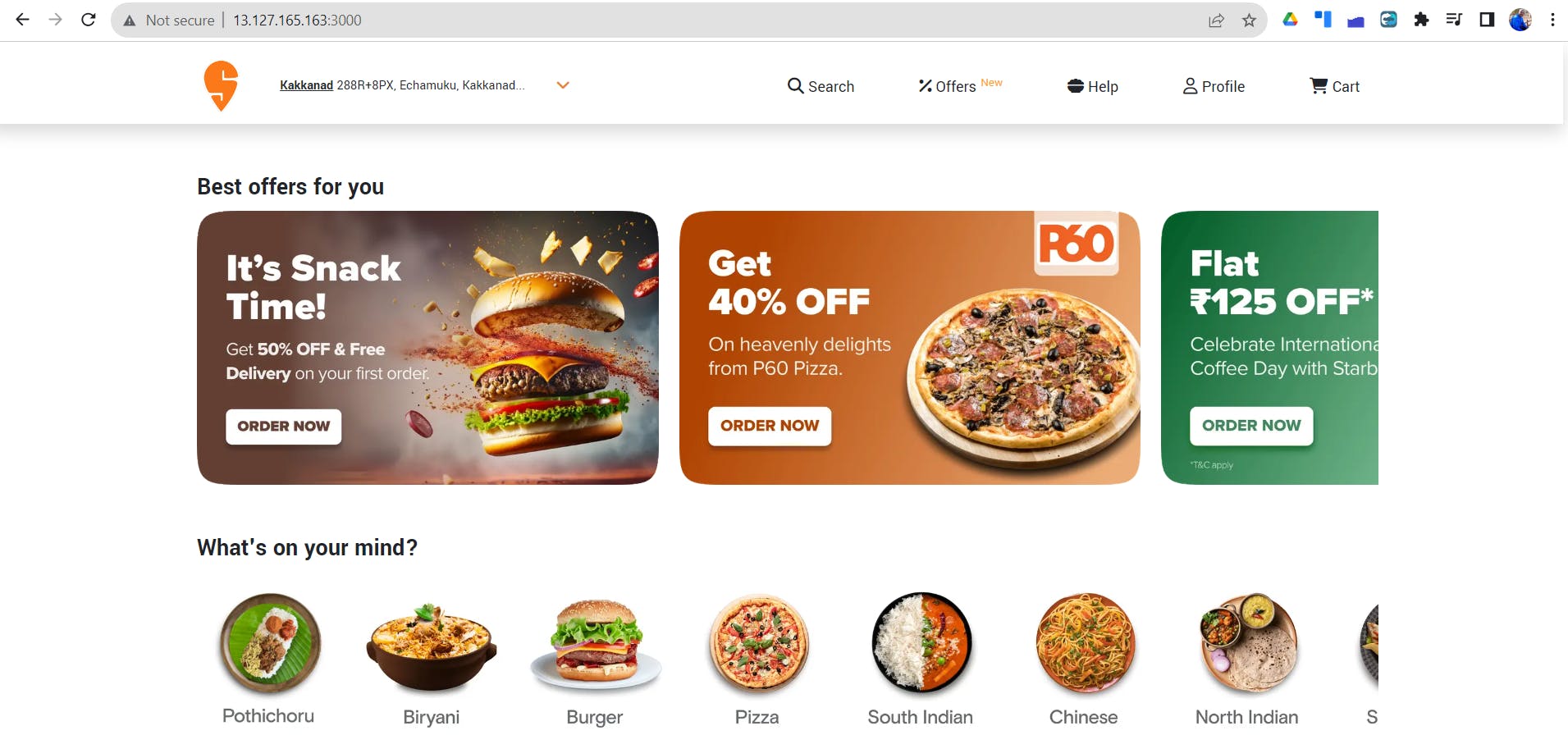


log in to Dockerhub, you will see a new image is created





[http://13.127.165.163:3000](http://13.127.165.163:3000/)



**Let's set Infra for Kubernetes using Terraform to deploy the Swiggy app :**

1. **Kubectl is to be installed on Jenkins also**

COPY

COPY

sudo apt update

sudo apt install curl

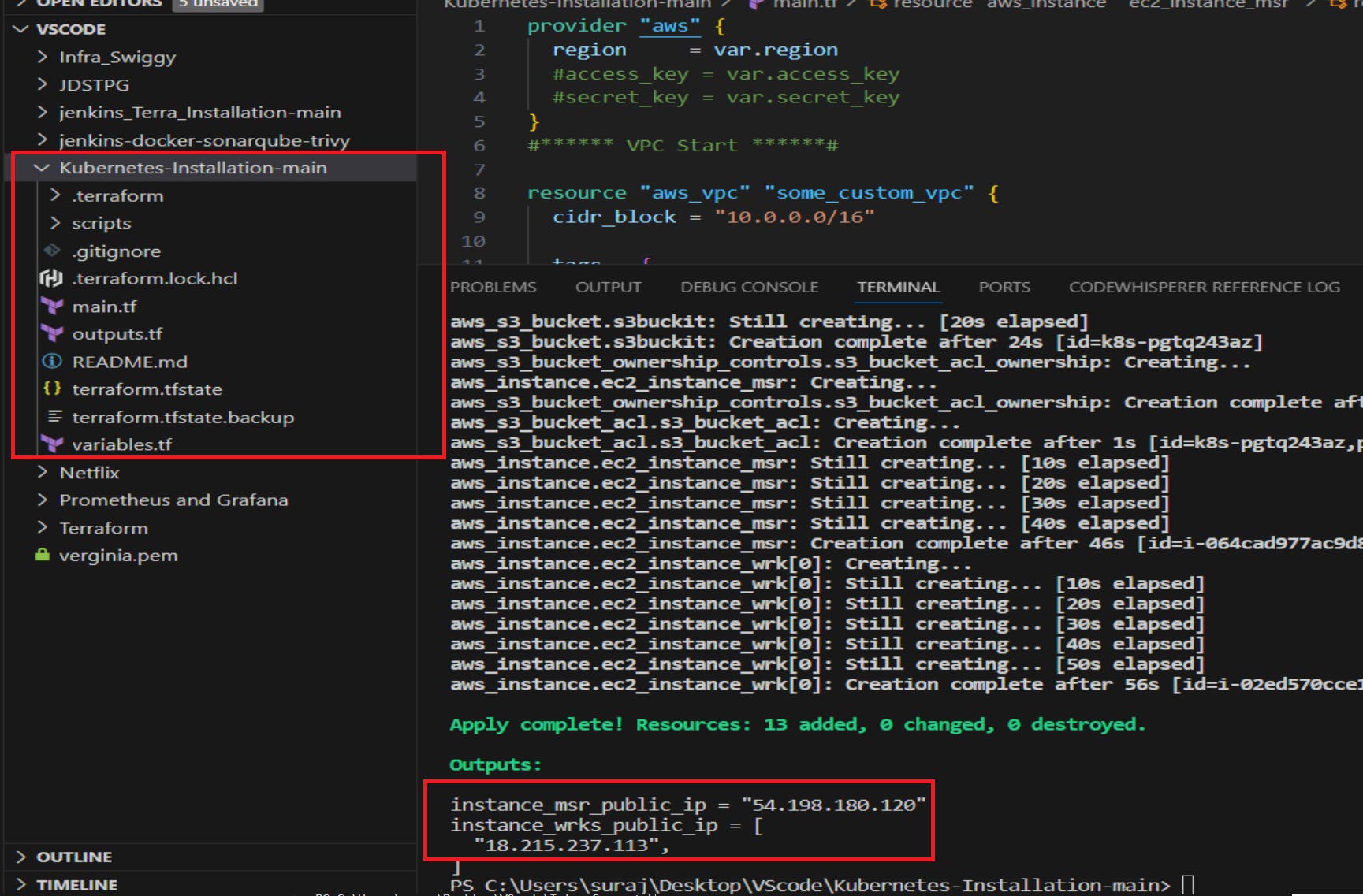
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

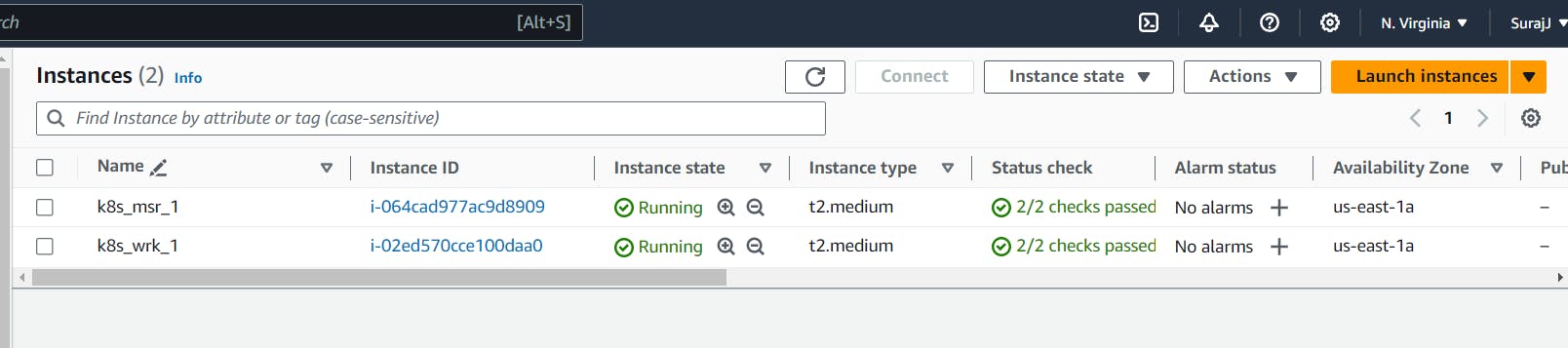
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

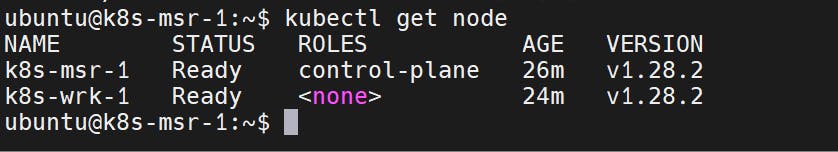
kubectl version --client

**2. Using the below GitHub repo configure Kubernetes**

<https://github.com/surajpjoshi/Kubernetes-Installation.git>

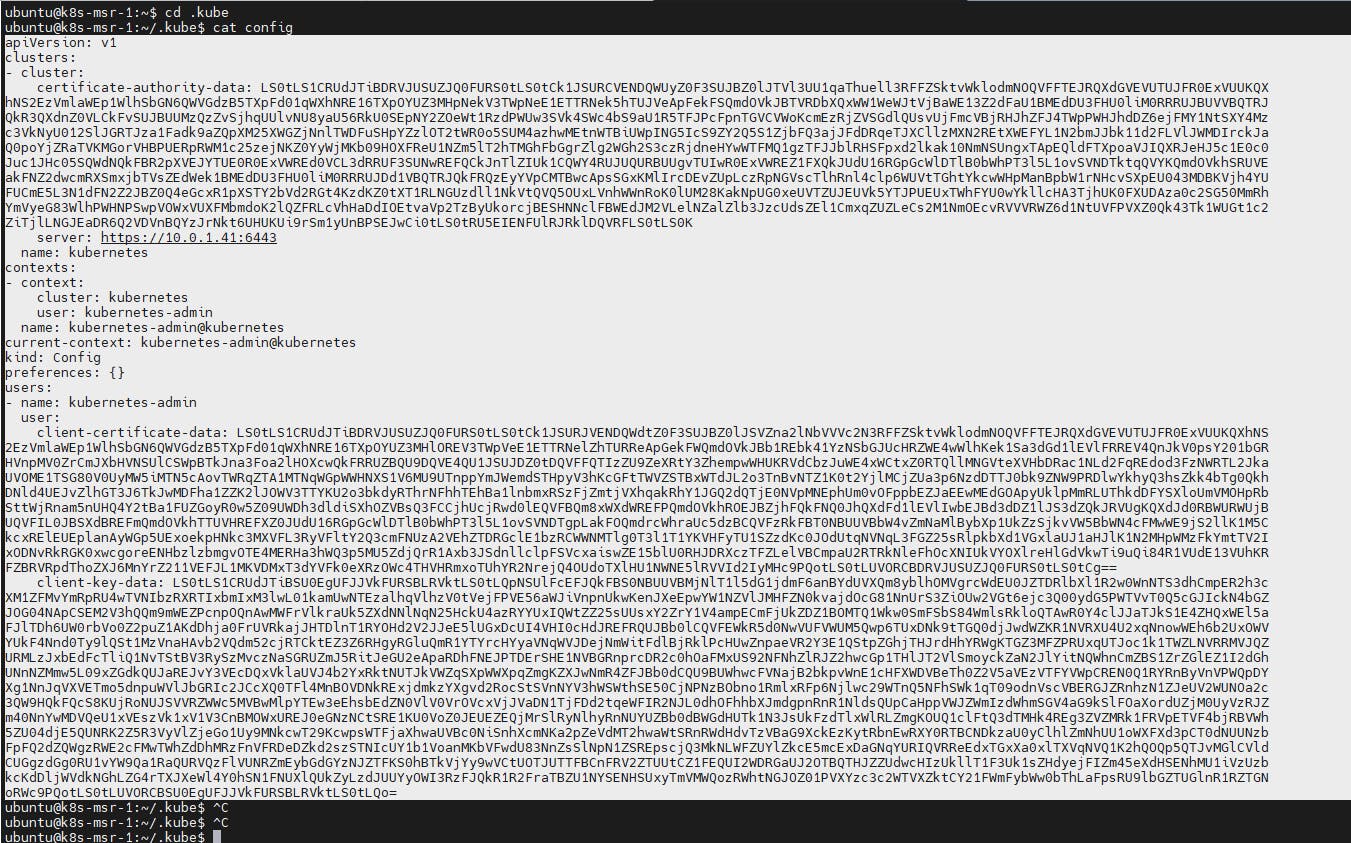






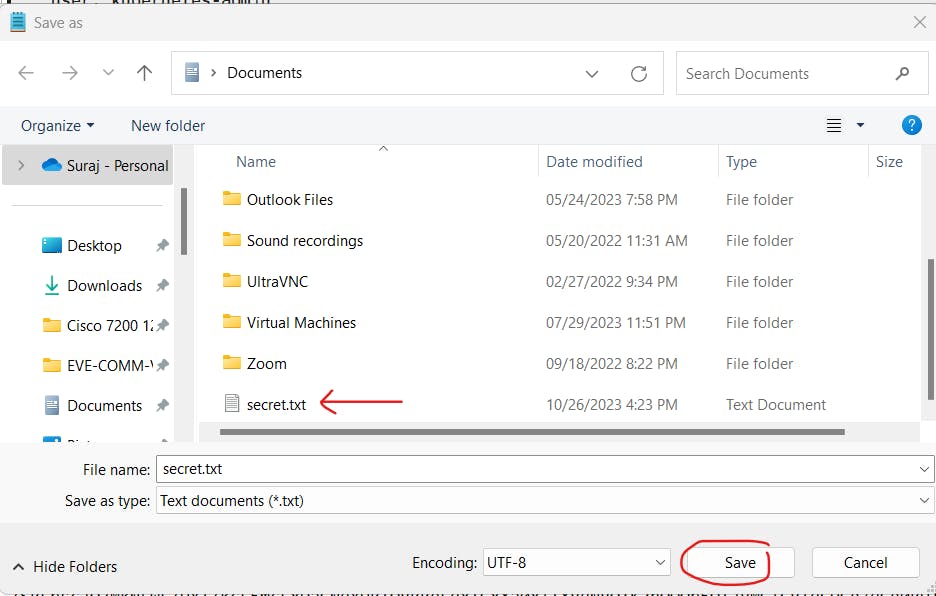
Go to Master & type the below commands to get the certificate configuration



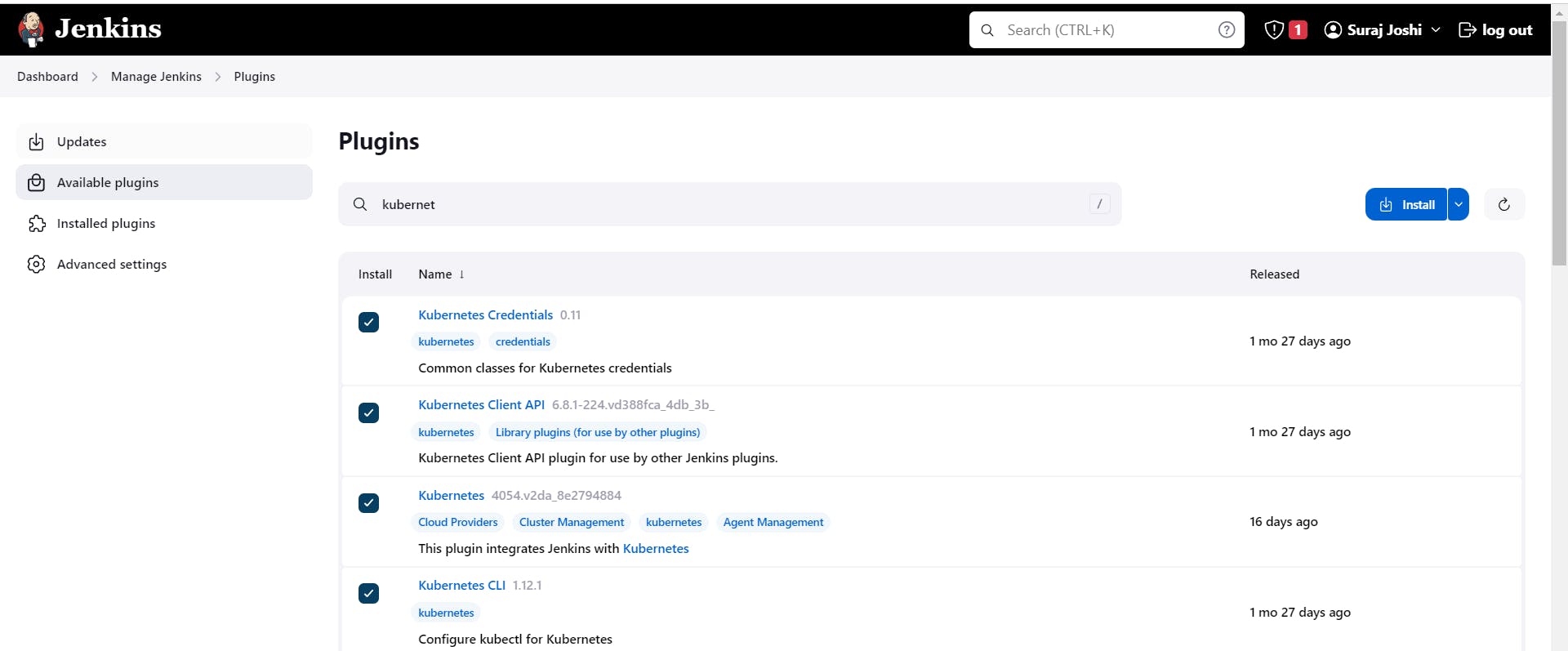


copy it and save it in documents or another folder save it as secret-file.txt

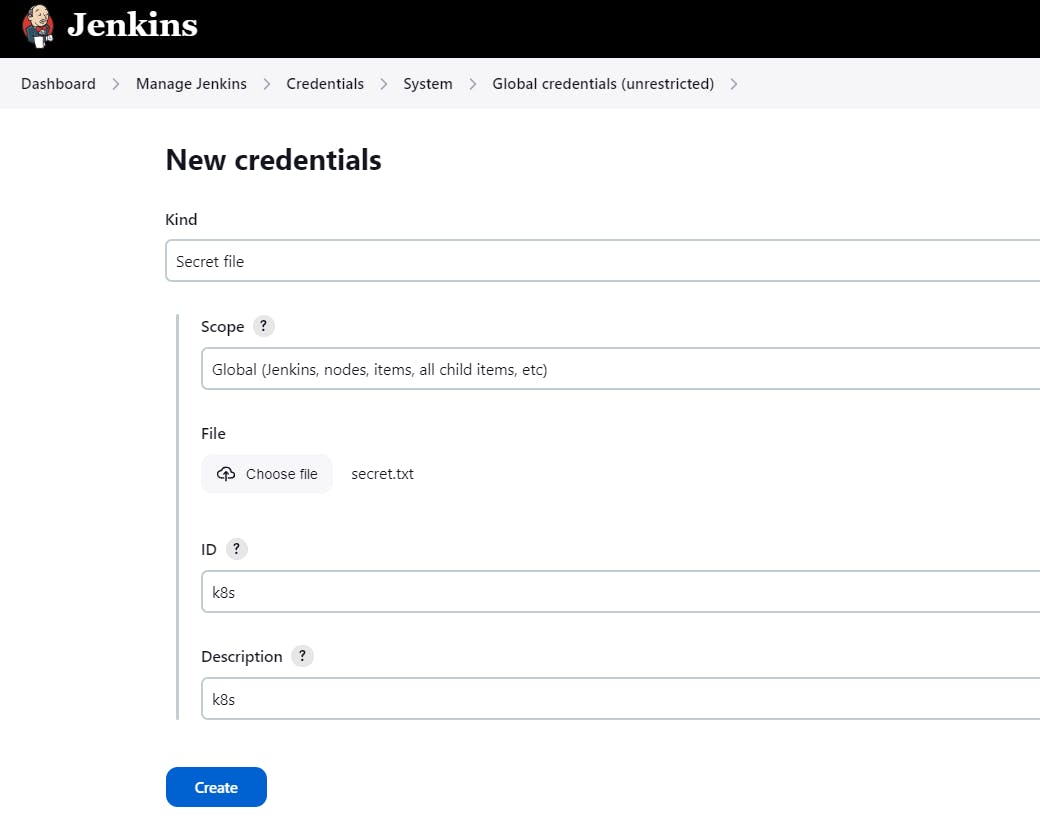
Note: create a secret-file.txt in your file explorer save the config in it and use this at the kubernetes credential section.



**Add the below plugins for Kubernetes**



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



Add Below stage to our pipeline

COPY

COPY

stage('Deploy to kubernets'){

steps{

script{

dir('Kubernetes') {

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

sh 'kubectl apply -f deployment.yml'

sh 'kubectl apply -f service.yml'

}

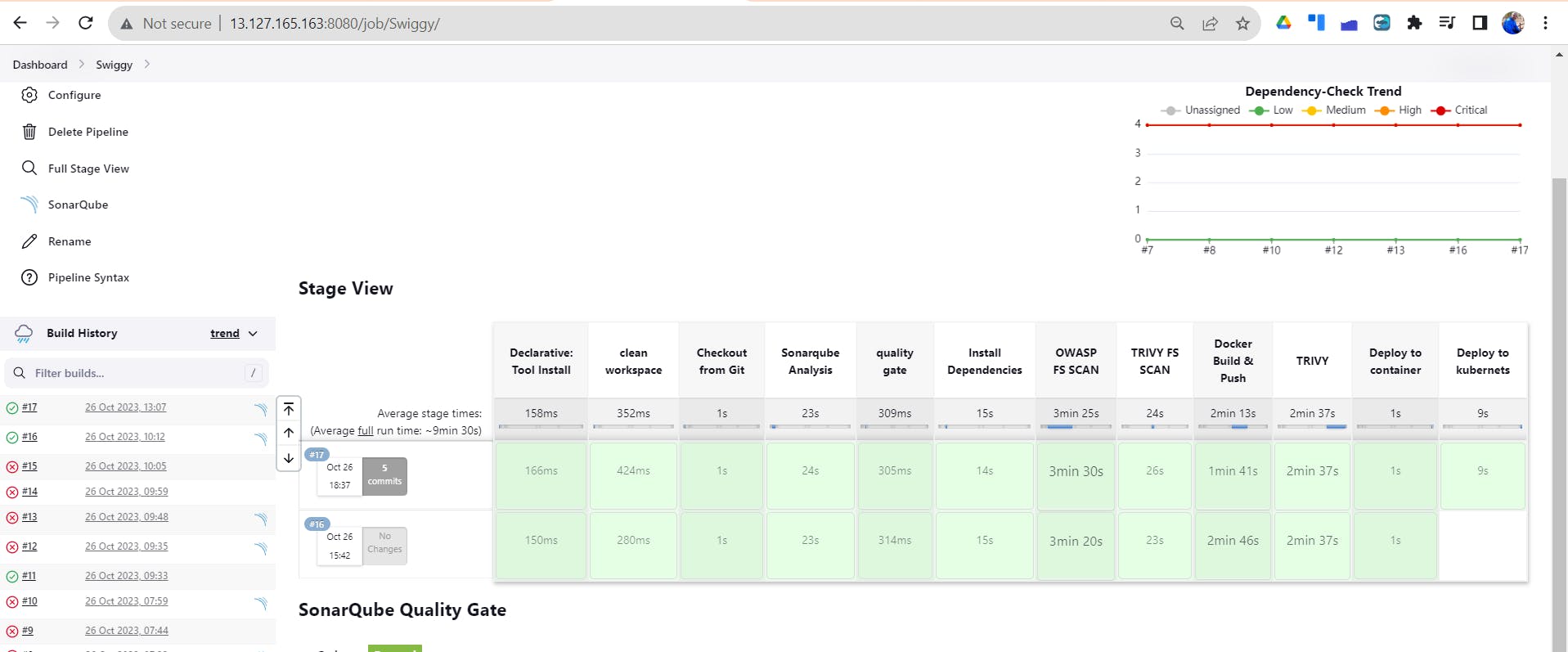
}

}

}

}

Build Now:



**Final output after deployment to Kubernetes**

[http://18.215.237.113:30007](http://18.215.237.113:30007/)

