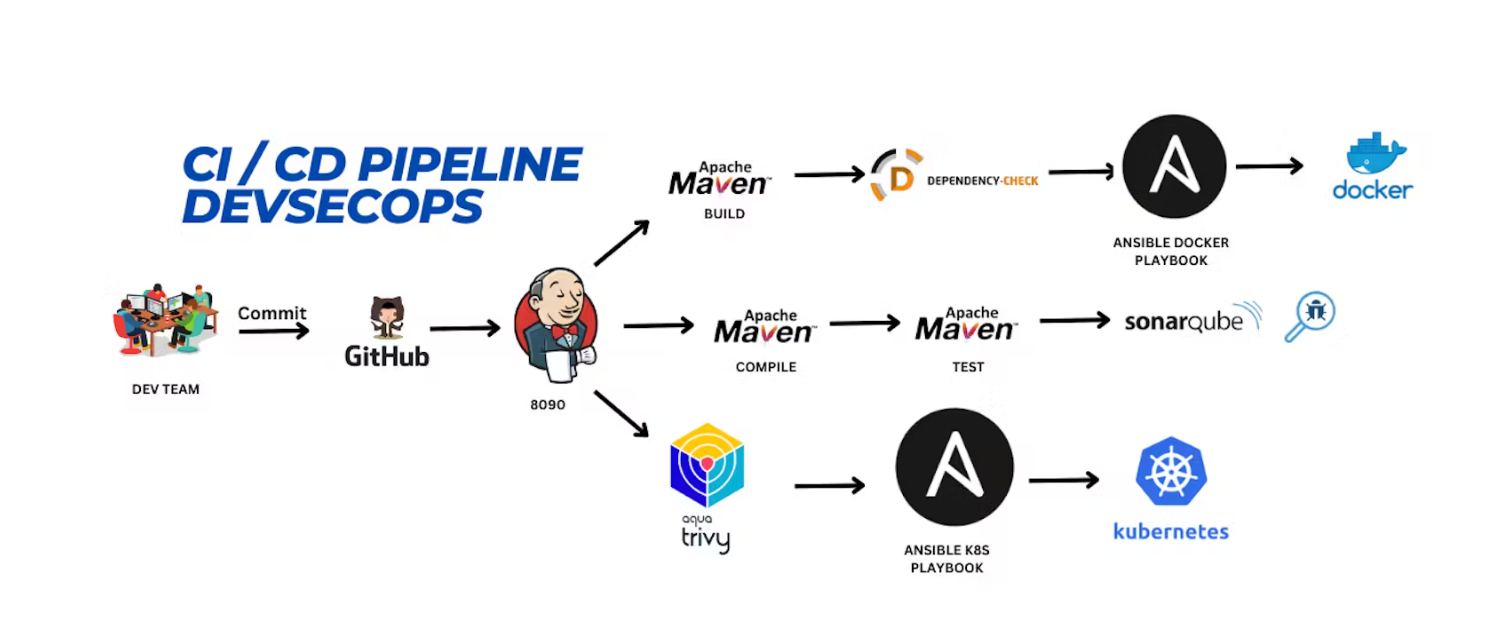
AWS CI DC PROJECT ANSIBLE



Launch instance

T2.large

Size: 30 GB

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

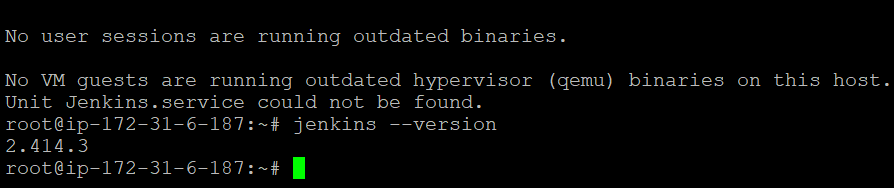
sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status Jenkins

sudo chmod 777 jenkins.sh

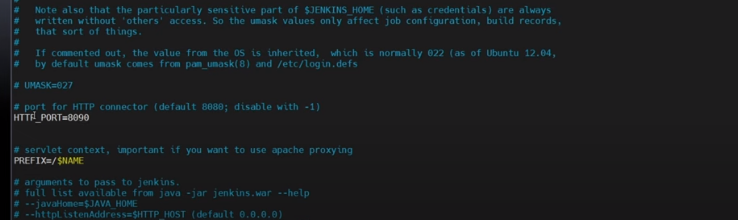
./jenkins.sh



login instance:

cd /etc/default

sudo vi jenkins



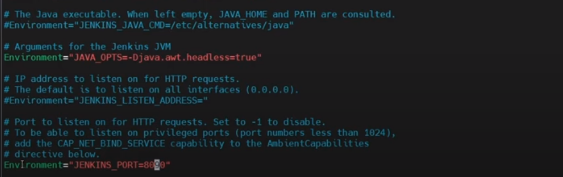
cd /lib/systemd/system

sudo vi jenkins.service #change Environments="Jenkins\_port=8090" save and exit

sudo systemctl daemon-reload

sudo systemctl restart jenkins

sudo systemctl status Jenkins



<Jenkins IP Address:8090>

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

sign in Jenkins:

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

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

**Sonarqube Installation:**

Launch EC2 instance 🡪 t2.medium 🡪 size: 30 GB

Login:

sudo apt-get update

sudo apt upgrade

## Add PostgresSQL repository

$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb\_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

$ wget -qO- https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo tee /etc/apt/trusted.gpg.d/pgdg.asc &>/dev/null

**## Install PostgreSQL**

$ sudo apt update

$ sudo apt-get -y install postgresql postgresql-contrib

sudo systemctl enable postgresql

**## Create Database for Sonarqube**

$ sudo passwd postgres

$ su - postgres

$ createuser sonar

$ psql

$ ALTER USER sonar WITH ENCRYPTED password 'sonar';

$ CREATE DATABASE sonarqube OWNER sonar;

$ grant all privileges on DATABASE sonarqube to sonar;

$ \q

$ exit

**## Add Adoptium repository**

$ sudo bash

$ 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

**## Install Java 17**

$ apt update

$ apt install temurin-17-jdk

$ update-alternatives --config java

$ /usr/bin/java --version

$ exit

**## Linux Kernel Tuning**

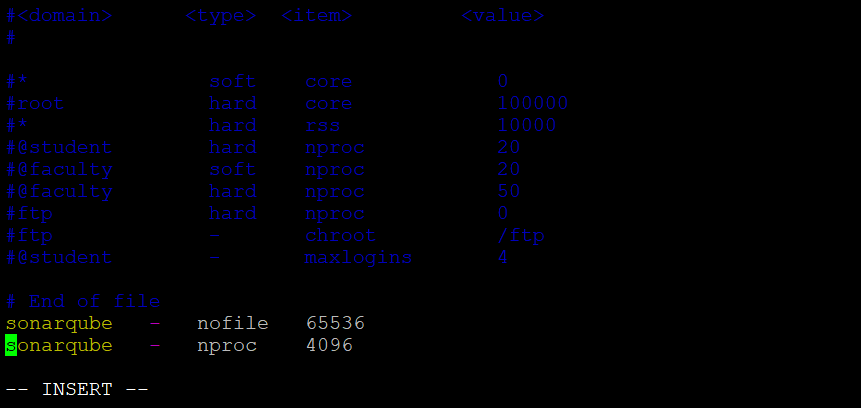
# Increase Limits

$ sudo vim /etc/security/limits.conf

//Paste the below values at the bottom of the file

sonarqube - nofile 65536

sonarqube - nproc 4096



**# Increase Mapped Memory Regions**

sudo vim /etc/sysctl.conf

//Paste the below values at the bottom of the file

vm.max\_map\_count = 262144

sudo init 6

security group: all port

**#### Sonarqube Installation ####**

**## Download and Extract**

$ sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip

$ sudo apt install unzip

$ sudo unzip sonarqube-9.9.0.65466.zip -d /opt

$ sudo mv /opt/sonarqube-9.9.0.65466 /opt/sonarqube

**## Create user and set permissions**

$ sudo groupadd sonar

$ sudo useradd -c "user to run SonarQube" -d /opt/sonarqube -g sonar sonar

$ sudo chown sonar:sonar /opt/sonarqube -R

**## Update Sonarqube properties with DB credentials**

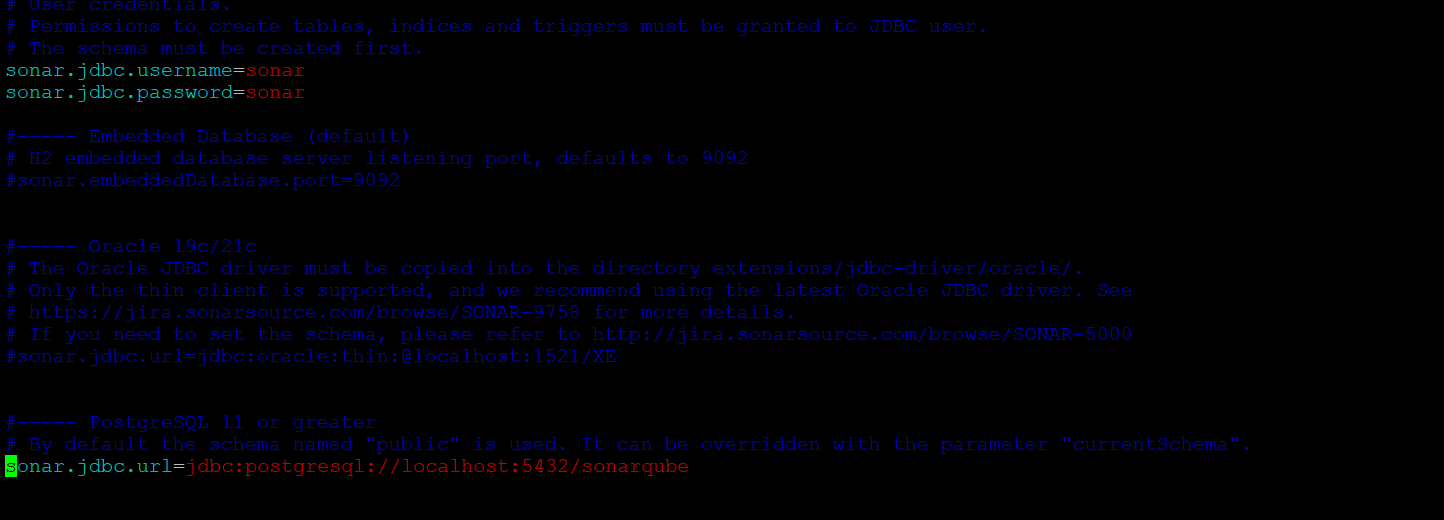
$ sudo vim /opt/sonarqube/conf/sonar.properties

//Find and replace the below values, you might need to add the sonar.jdbc.url

sonar.jdbc.username=sonar

sonar.jdbc.password=sonar

sonar.jdbc.url=jdbc:postgresql://localhost:5432/sonarqube



**## Create service for Sonarqube**

$ sudo vim /etc/systemd/system/sonar.service

//Paste the below into the file

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar

Group=sonar

Restart=always

LimitNOFILE=65536

LimitNPROC=4096

[Install]

WantedBy=multi-user.target

**## Start Sonarqube and Enable service**

$ sudo systemctl start sonar

$ sudo systemctl enable sonar

$ sudo systemctl status sonar

**login sonarqube:**

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

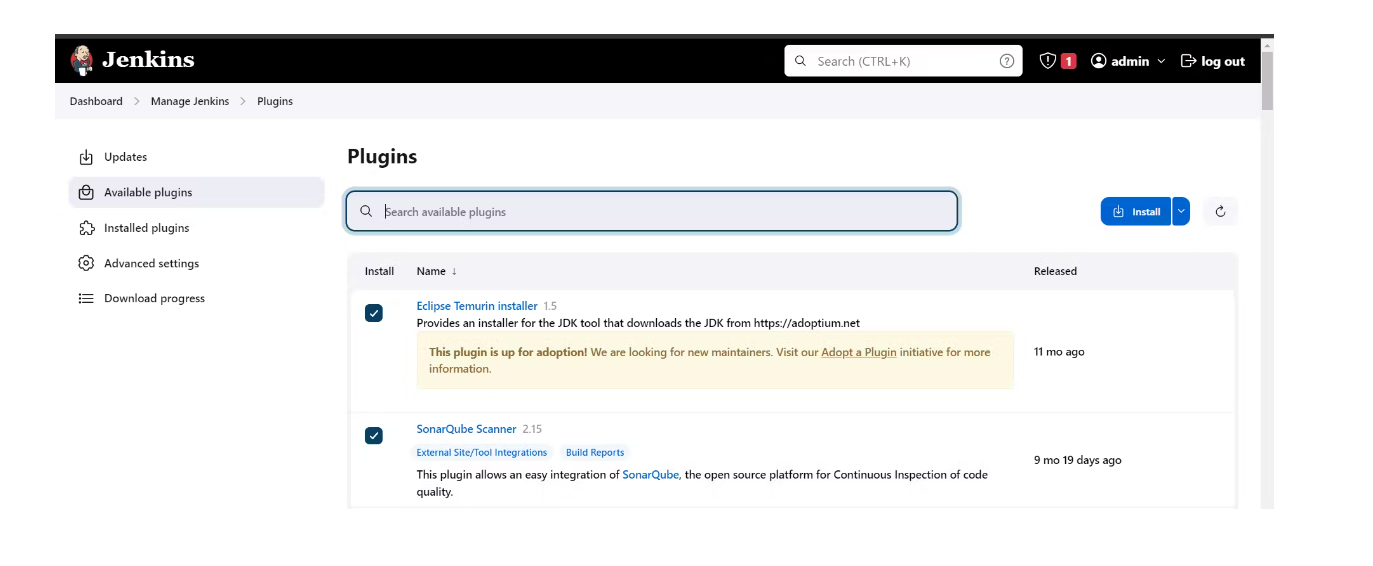
sudo apt-get update

sudo apt-get install trivy -y

**permission:**

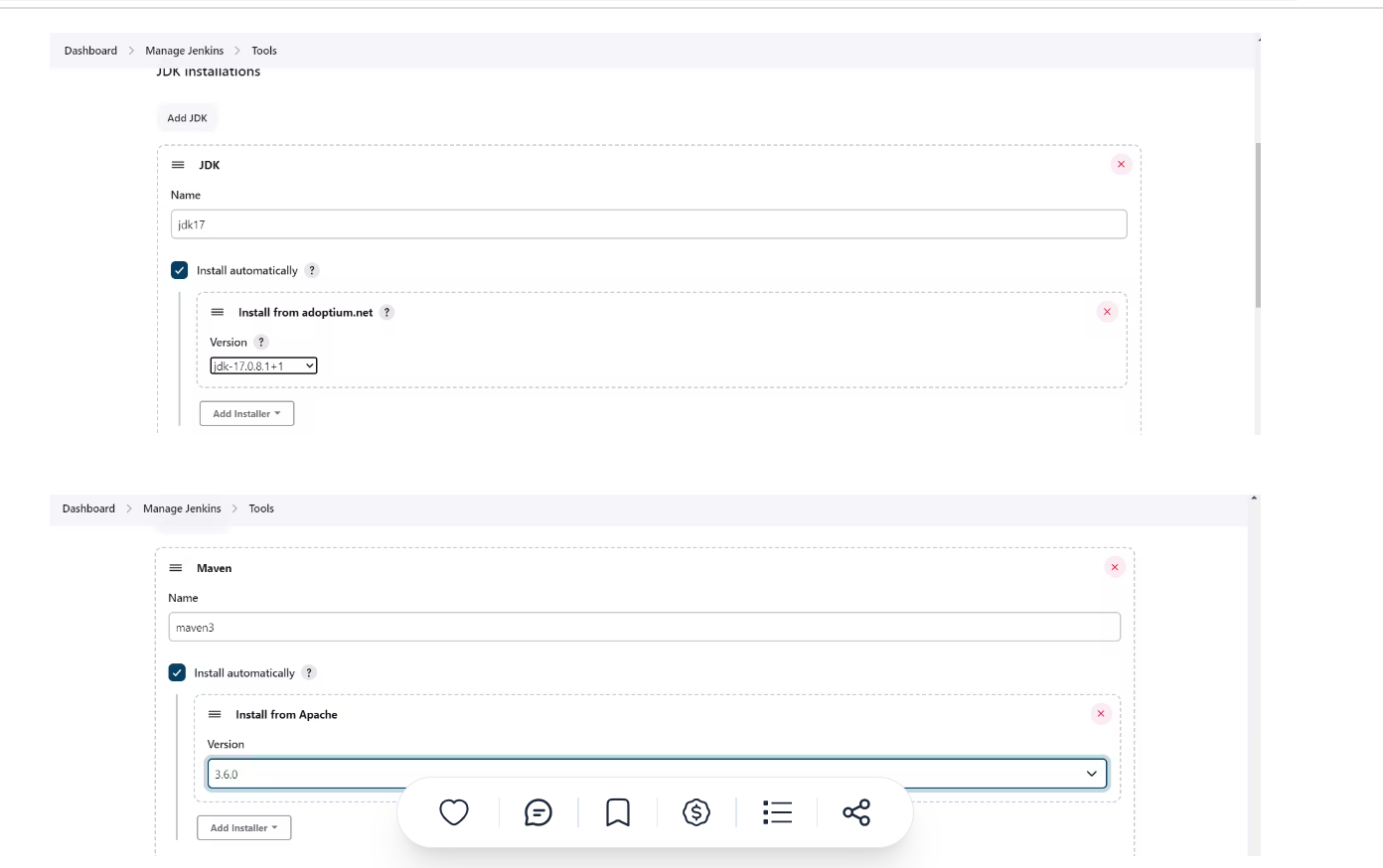
sudo chmod 777 trivy.sh

./trivy.sh



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

My-pipeline click on Pipeline and OK.



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'

}

}

}

}

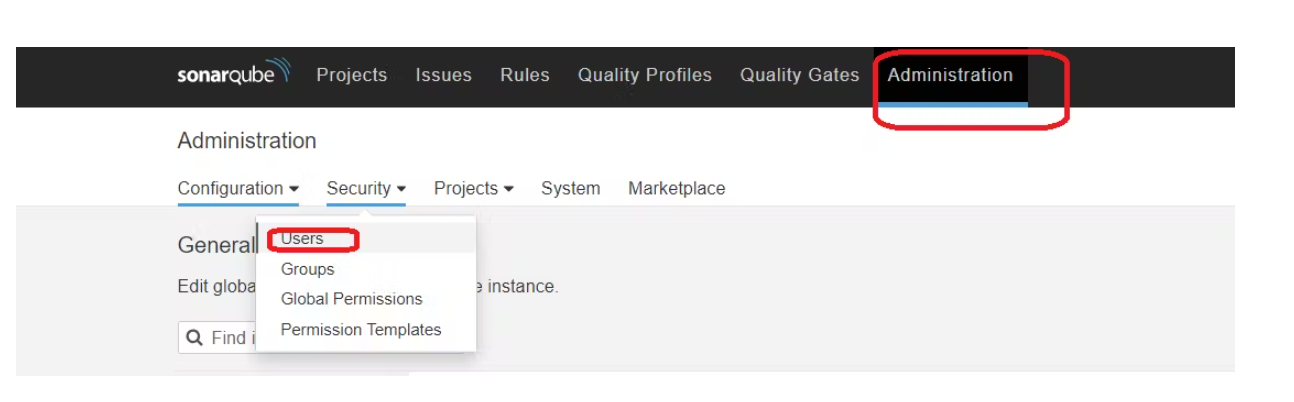
Click pipeline syntax

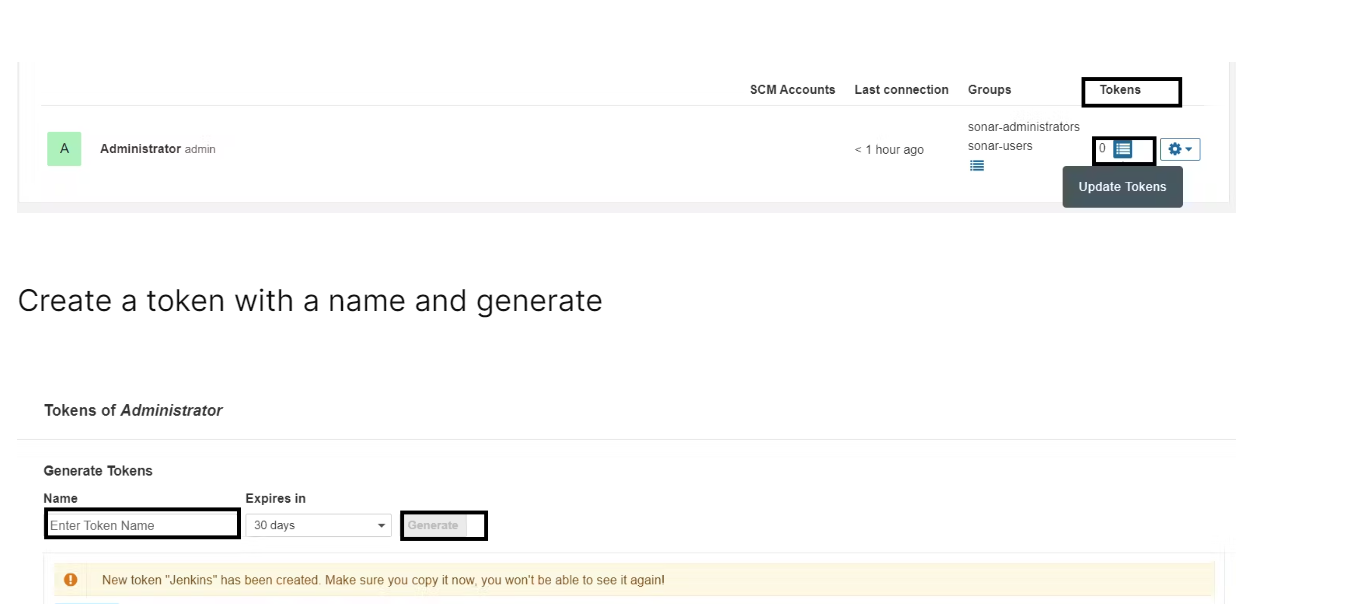
Seleck Git:git

Repository URL:

Generate

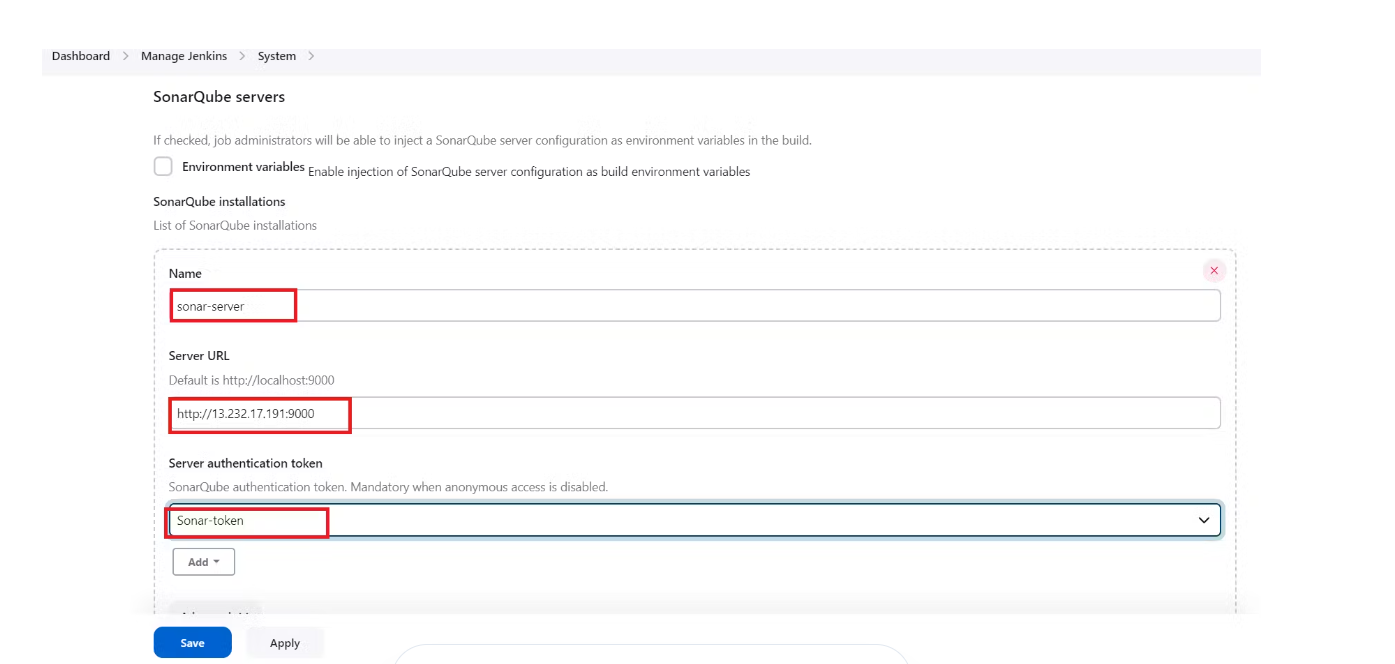
Apply and save





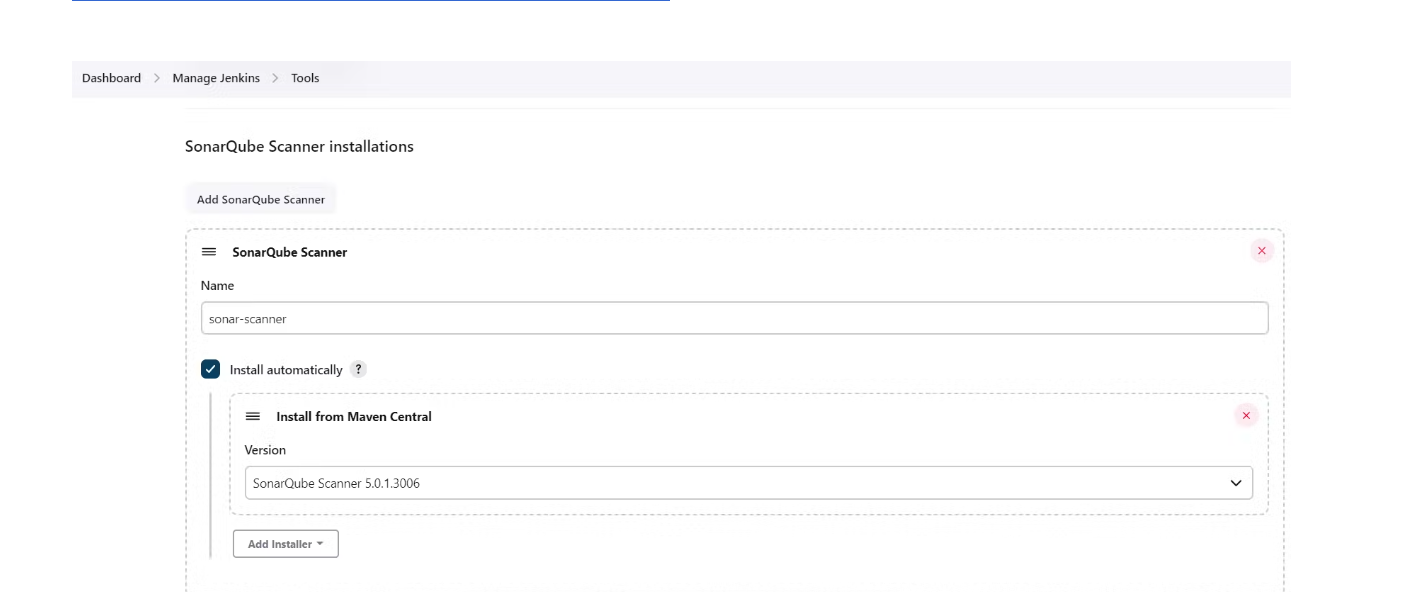


Now, go to Dashboard → Manage Jenkins → System and Add like the below image.



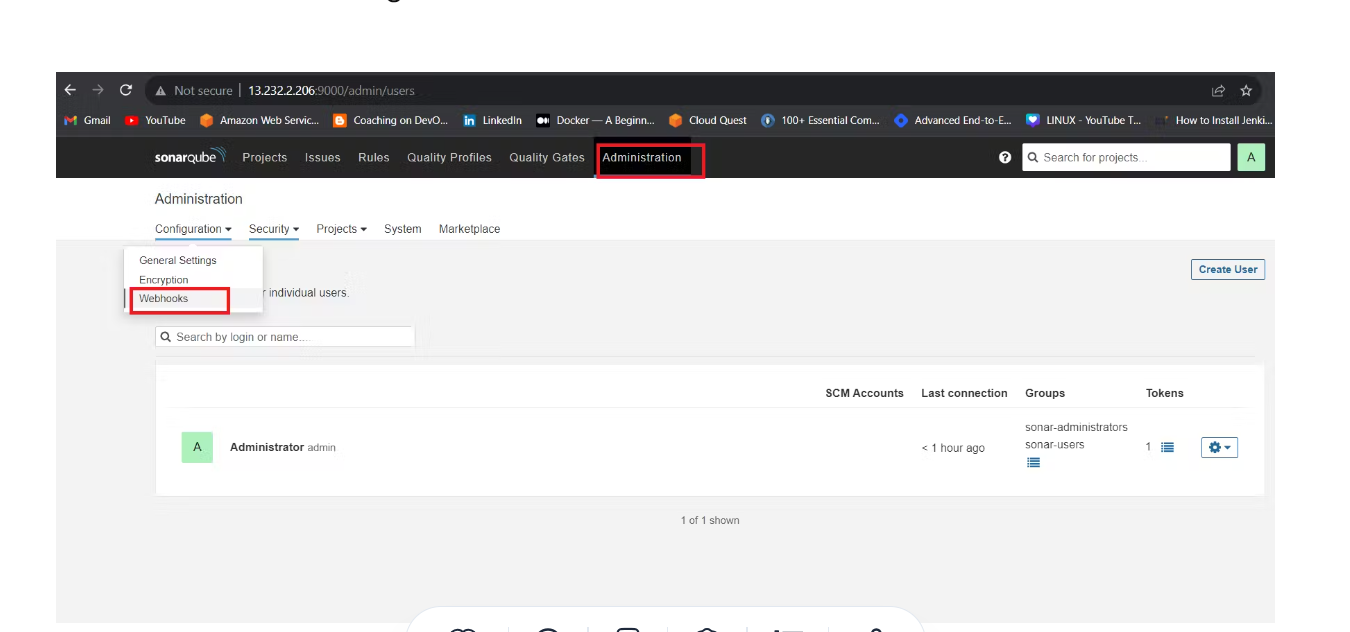
Click on Apply and Save

We will install a sonar scanner in the tools.



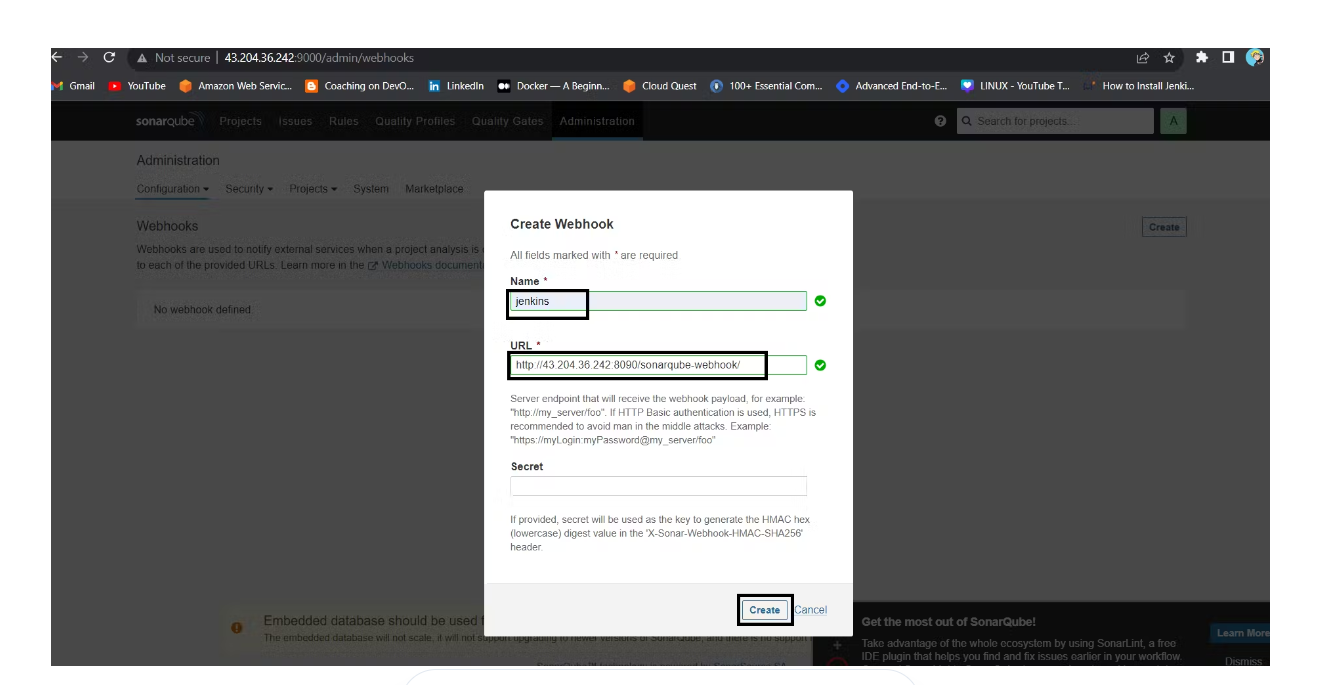
In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



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

http://13.126.228.42:8090/sonarqube-webhook/



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'

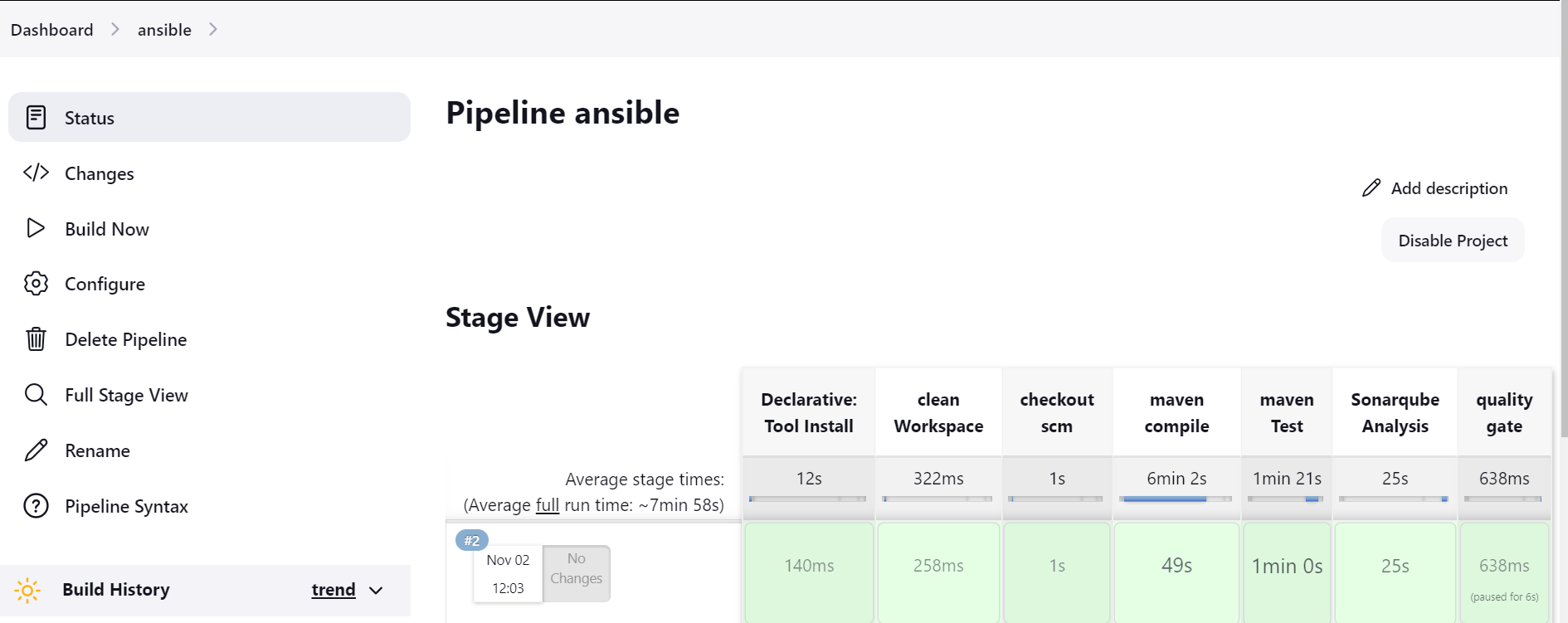
}

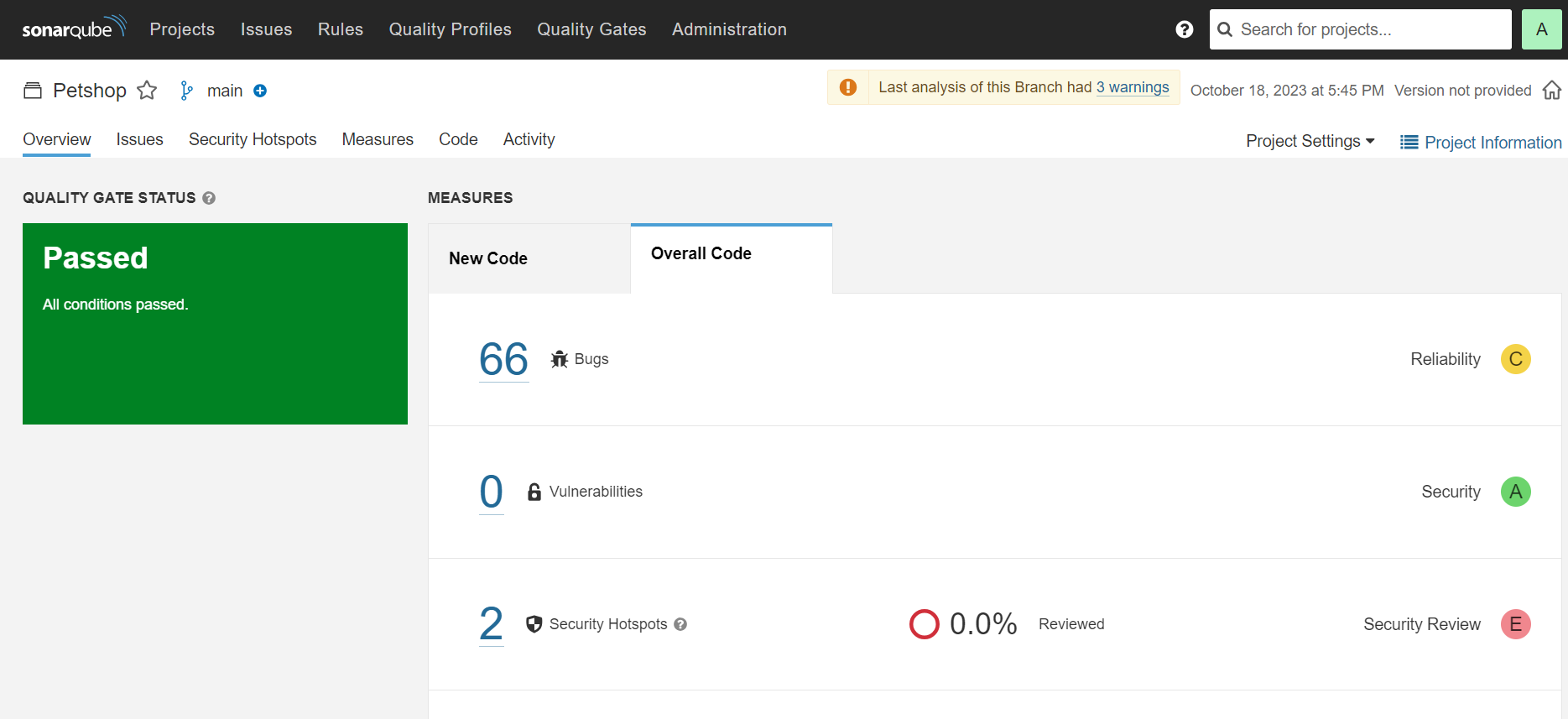
}

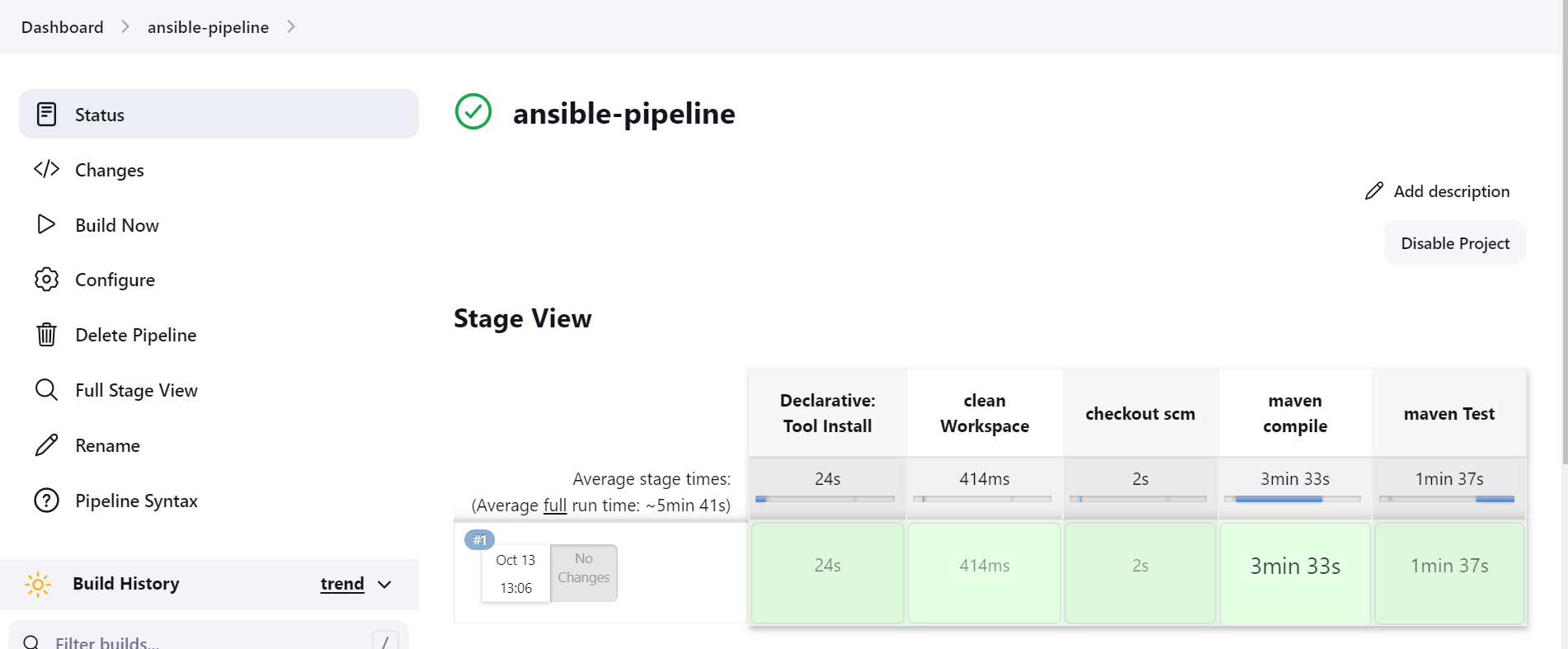
}

}

}

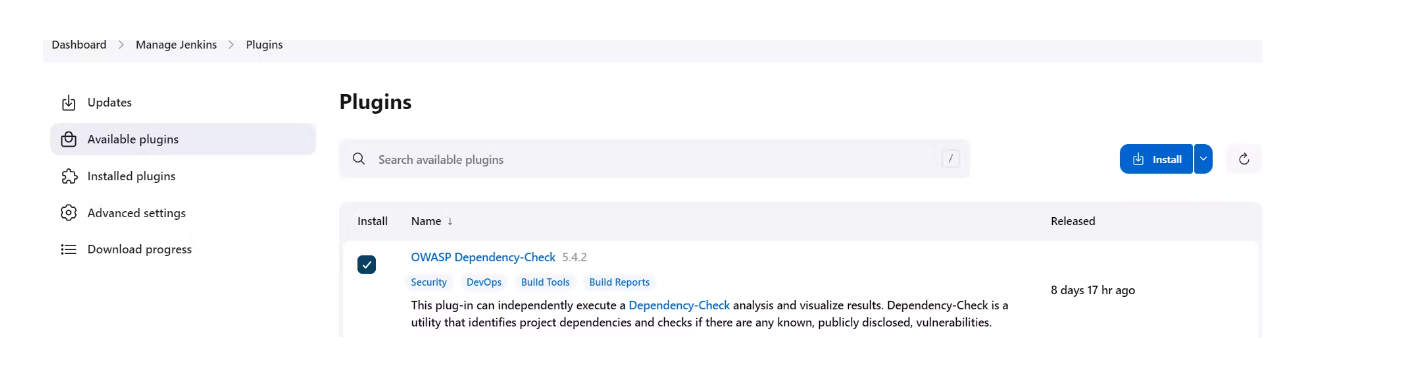






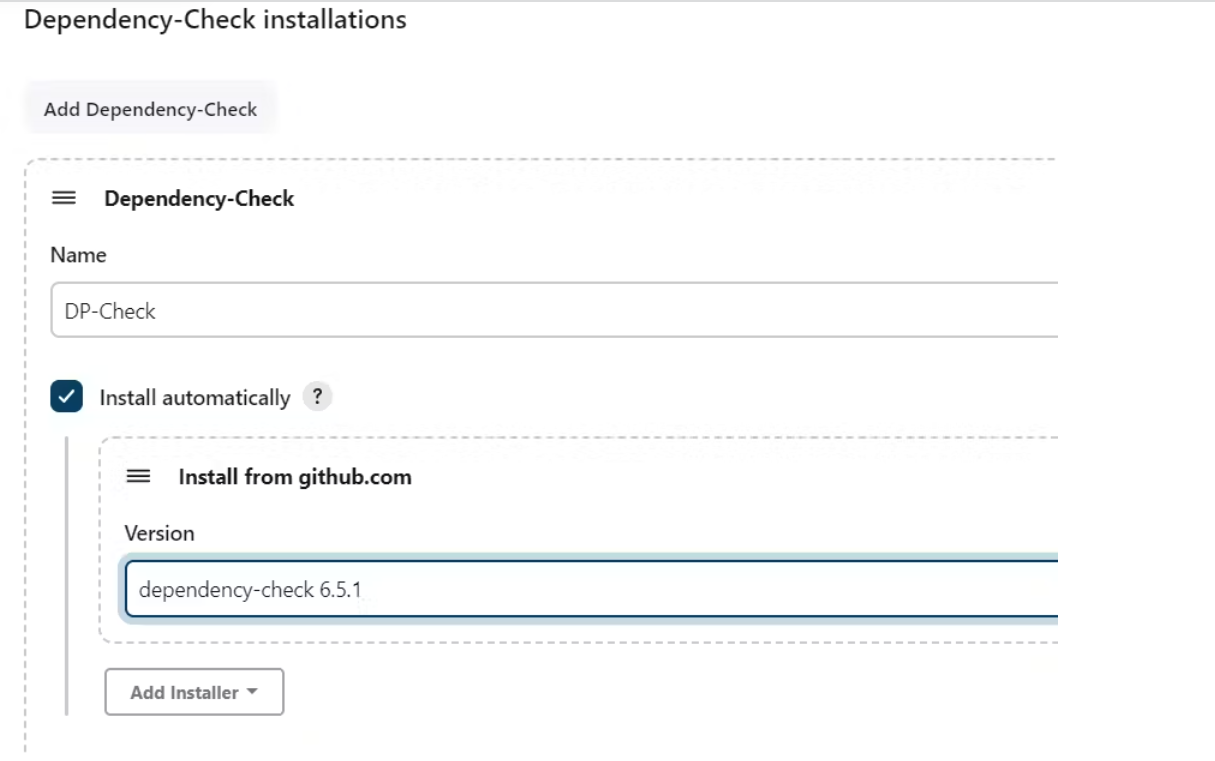
Build now

Install e Dependency Check Plugins



First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →



Administrator 🡪 security 🡪 token generate

squ\_364f678d015c9c6f13edc915c450e8a39b918063

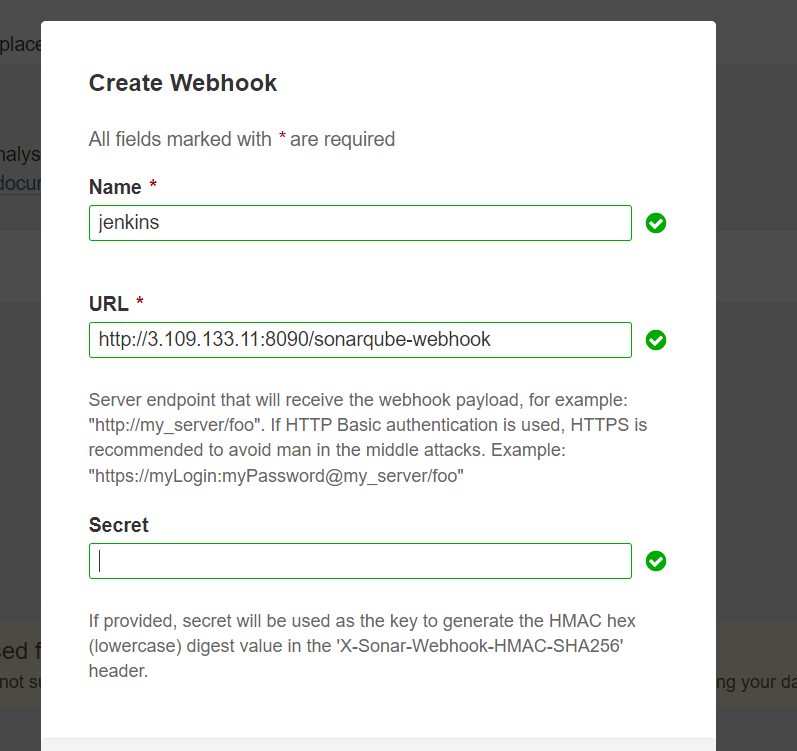
credentials 🡪 kind secret

ID: Sonar-token

Secret: squ\_364f678d015c9c6f13edc915c450e8a39b918063

Maage Jenkins 🡪 system 🡪 sonarqube

Sonarqube 🡪 Administrator 🡪 [Configuration](http://3.109.133.11:9000/admin/webhooks) 🡪 webook



pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

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'

}

}

}

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 ('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'

}

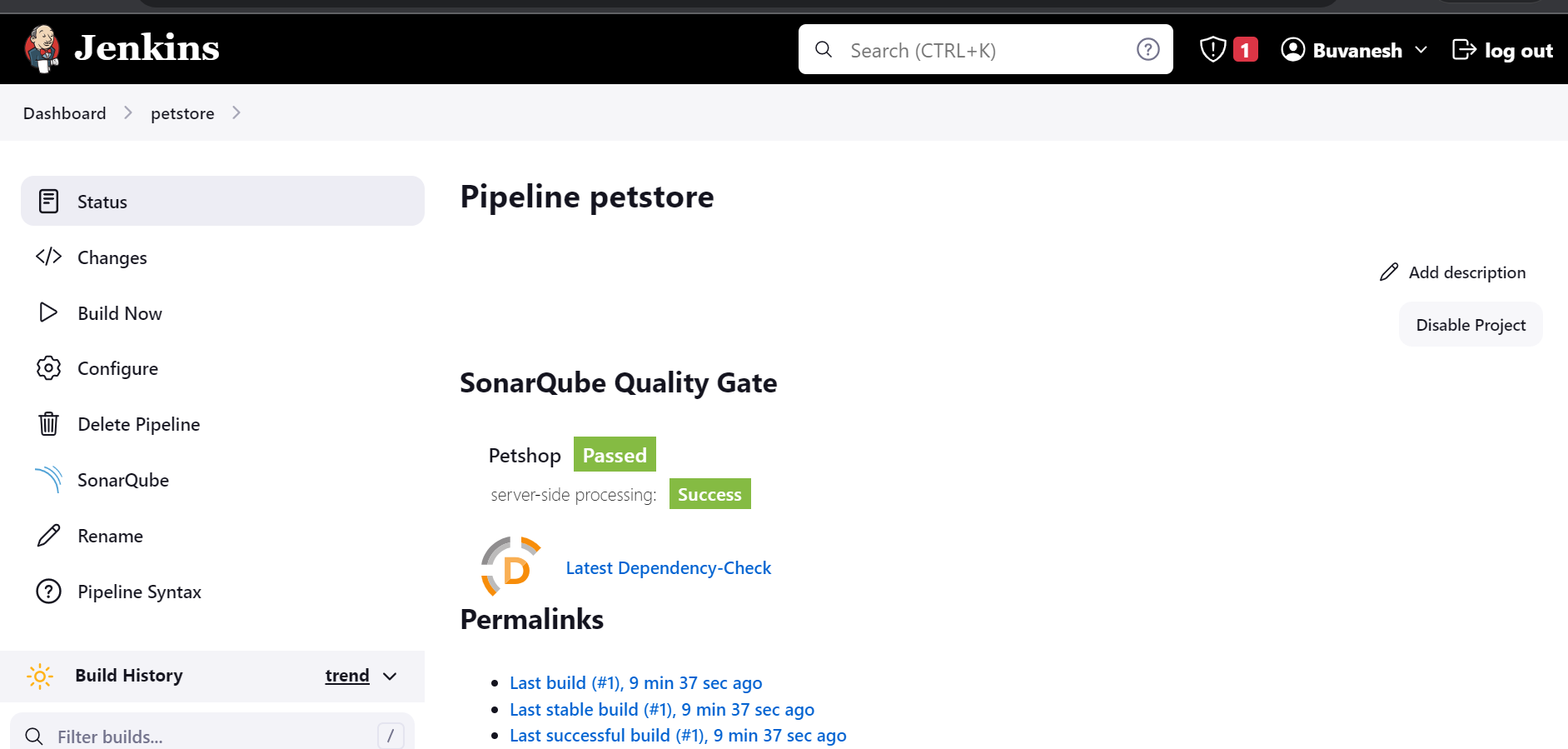
}

}

}

Save and apply

Build now



We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

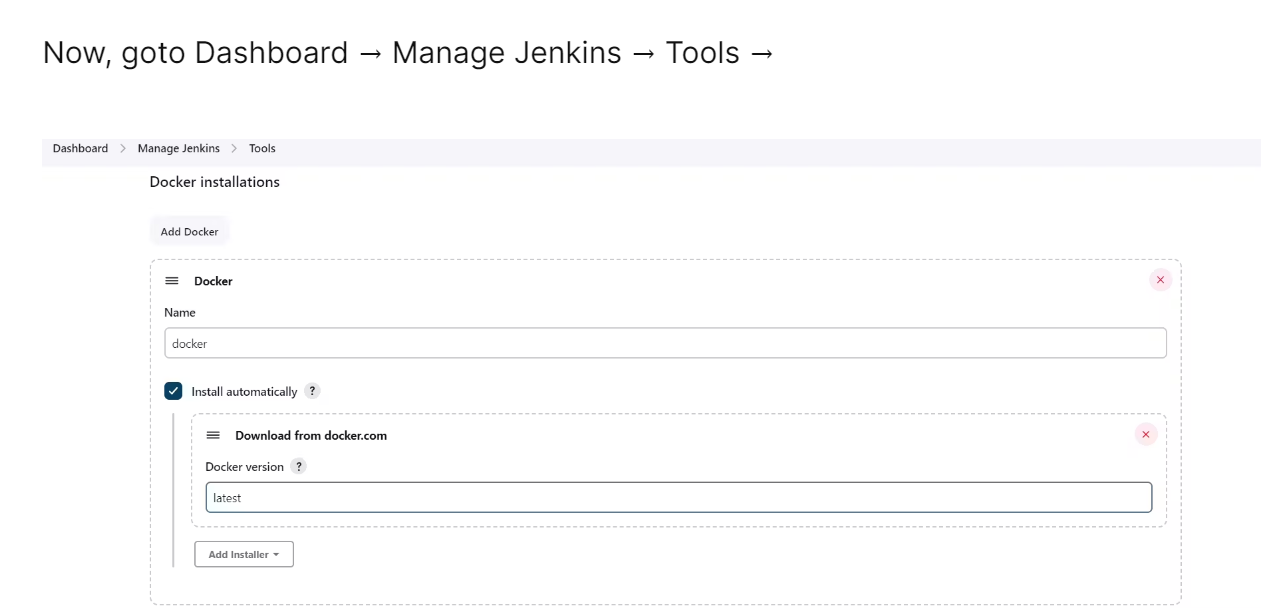
Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step





sudo apt-get update

sudo apt install software-properties-common

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install python3

sudo apt install ansible -y

sudo apt install ansible-core -y

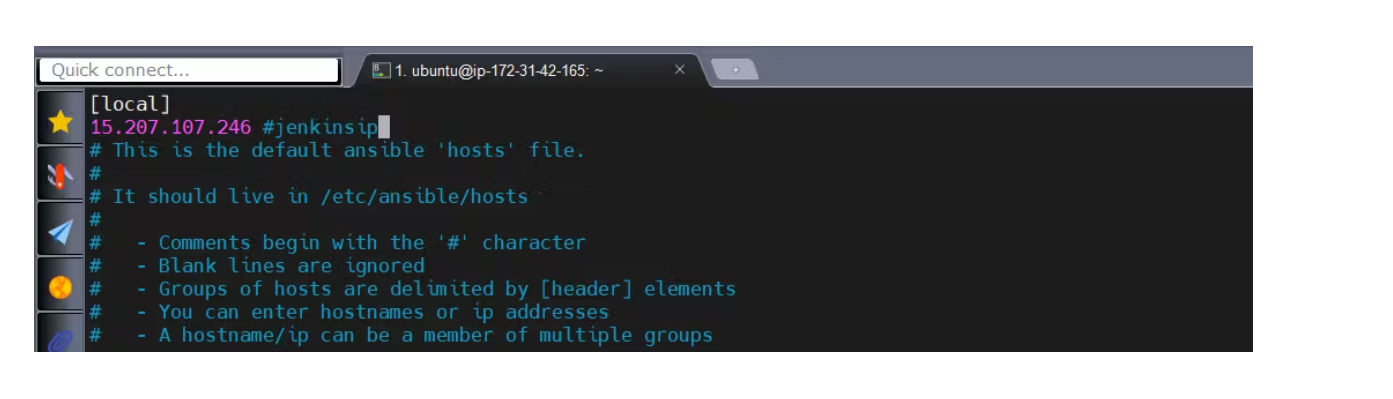
cd /etc/ansible

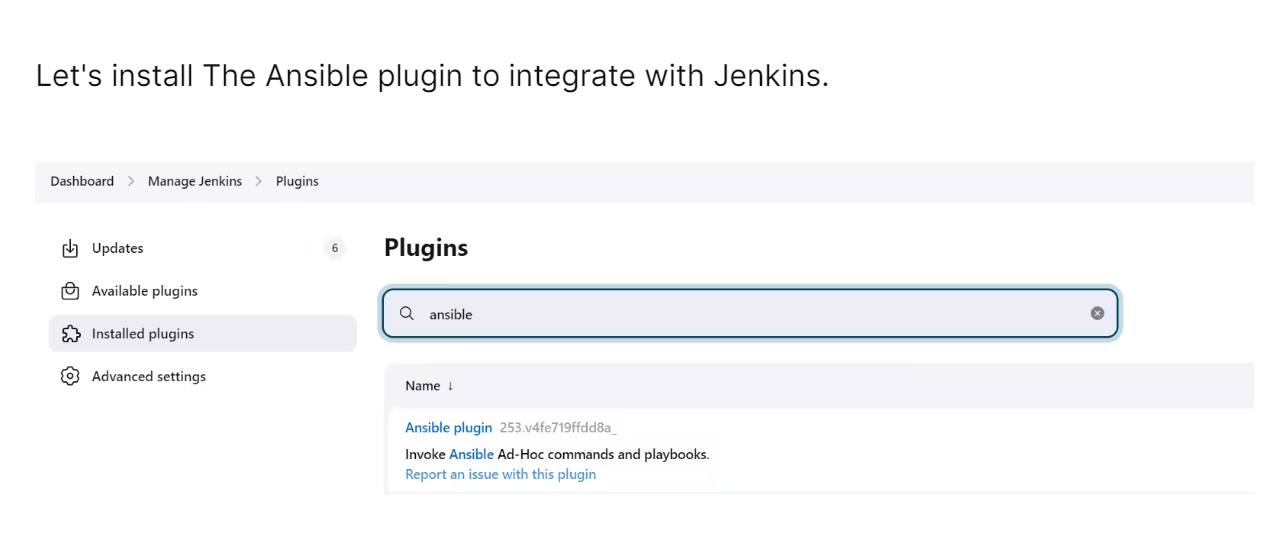
sudo vi hosts

You can create a group and paste ip address below:

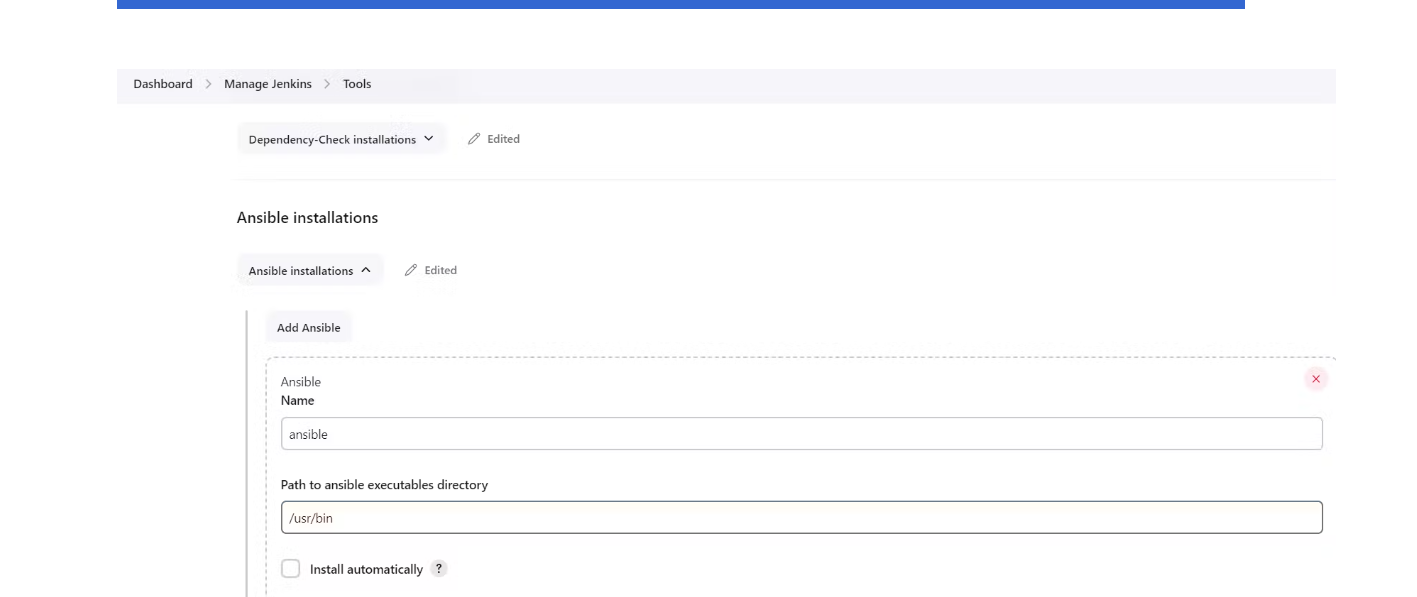
[local]#any name you want

Ip of Jenkins



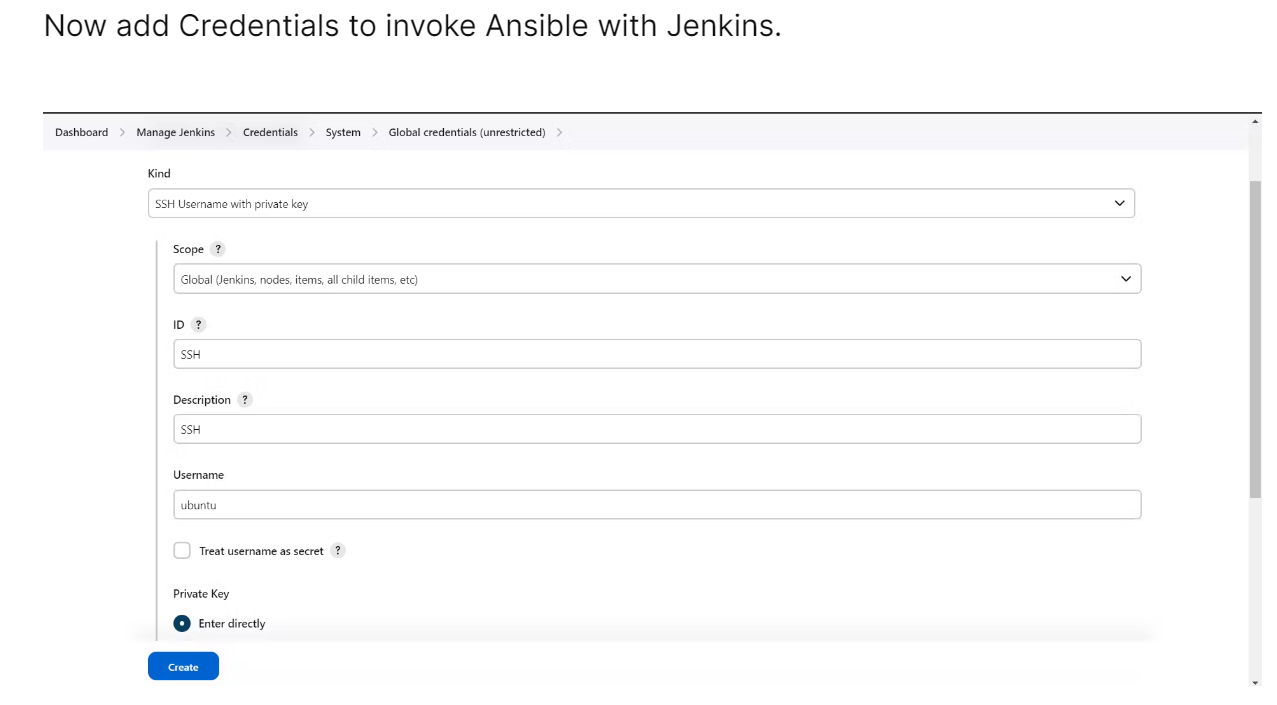


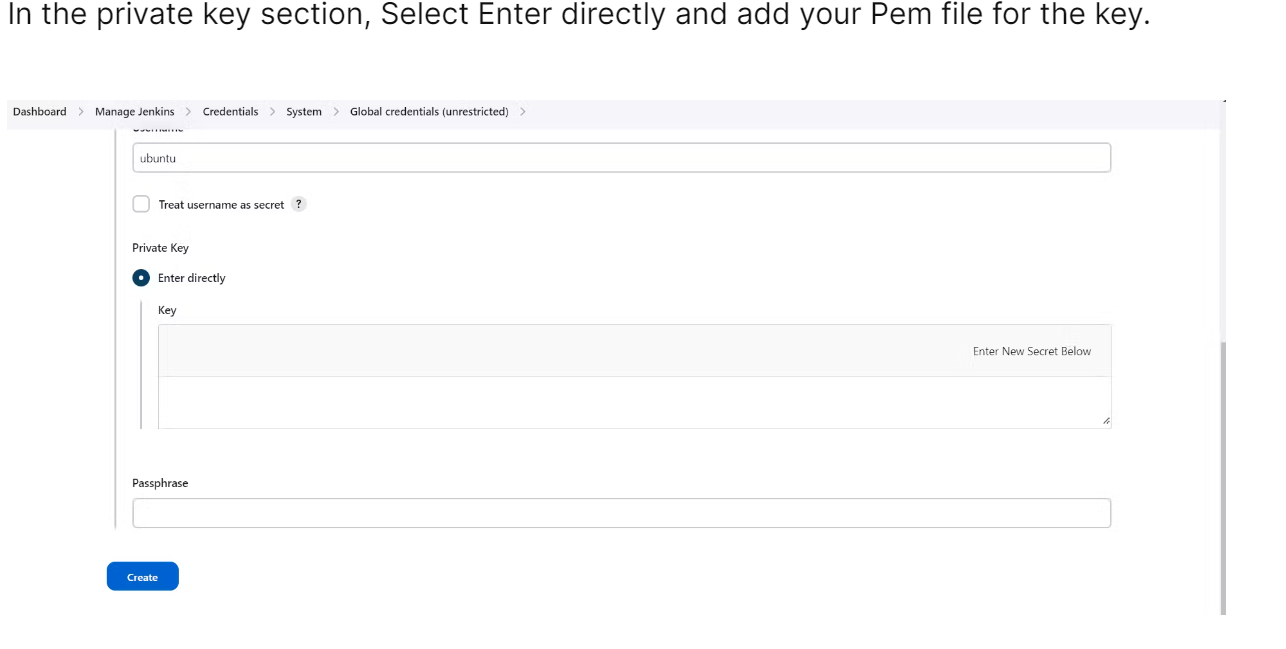
Copy that path and add it to the tools section of Jenkins at ansible installations.



Apply and save

Credential 🡪





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/buvan30/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("OWASP Dependency Check"){

steps{

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

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

}

}

stage('Install Docker') {

steps {

dir('Ansible'){

script {

ansiblePlaybook credentialsId: 'ssh', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/', playbook: 'docker.yaml'

}

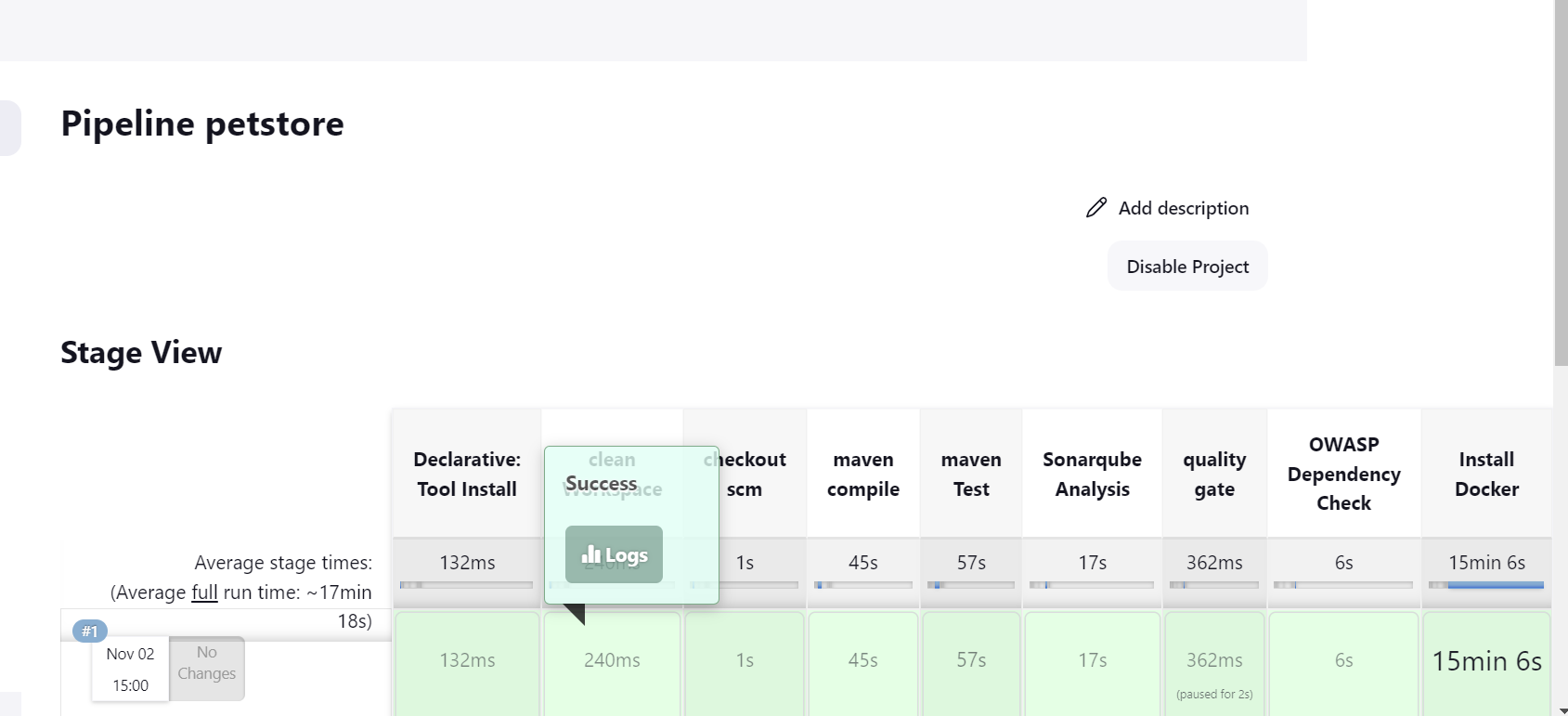
}

}

}

}

}



<jenkins-ip:8081>/jpetstore

vi kubectl.sh

sudo apt update

sudo apt install curl -y

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

./kubectl.sh

sudo chmod 777 kubectl.sh

**launch 2 instances:**

**ubuntu:20.04**

**instance type: t2.medium**

**size: 15**

**launch**

**master machine**

**sudo hostnamectl set-hostname k8s-master**

**exec bash**

**worker machine**

**sudo hostnamectl set-hostname k8s-worker**

**exec bash**

sudo apt update

**master machine and worker**

**sudo apt install docker.io -y**

**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**

**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>

sudo kubeadm join 172.31.47.180:6443 --token rnq1ha.p3axd3315iwl61u1 \

--discovery-token-ca-cert-hash sha256:46880e38498e14adff6dccded80acdadd0f4606c30967e587569621c759f071f

**Master node:**

cd .kube/

cat config

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

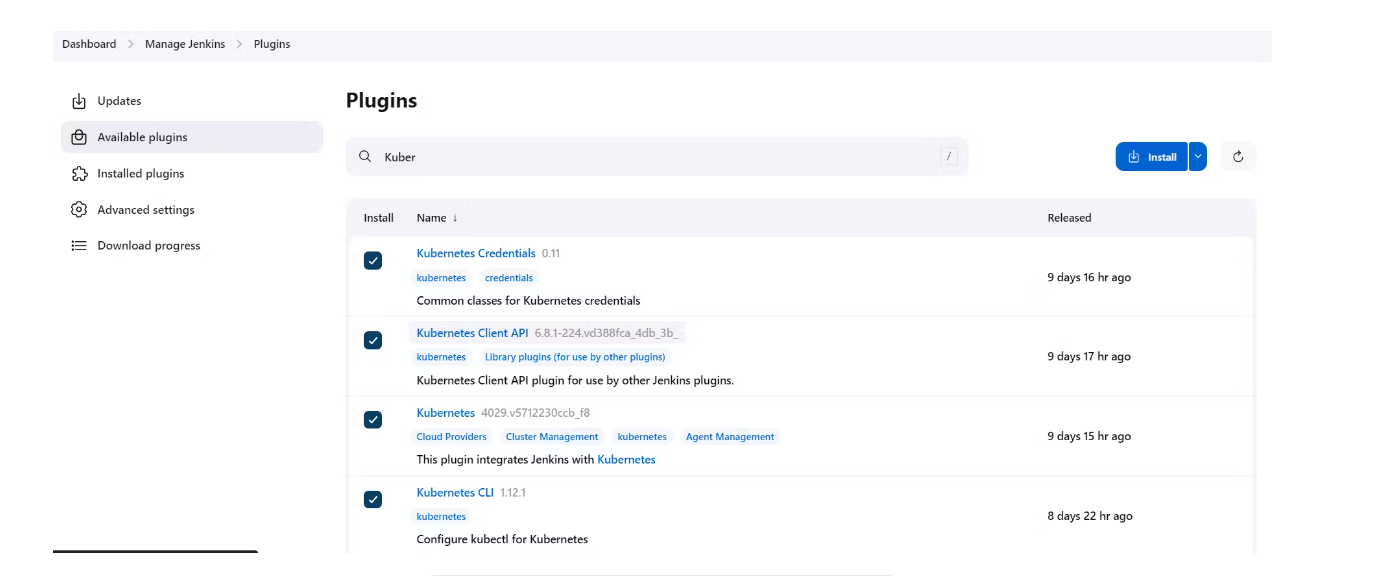
add credentials 🡪 kind: secret file

choose file

ID: k8s

Create

Install Kubernetes Plugin, Once it's installed successfully



**Jenkins machine:**

**Master-slave Setup for Ansible and Kubernetes**

ssh-keygen

cd id\_rsa.pub

copy

**master machine:**

cd .ssh

cat id\_rsa.pub #copy this public key

sudo vi authorized\_key

paste

By adding a public key from the master to the k8s machine we have now configured keyless access. To verify you can try to access the k8s master and use the command as mentioned in the below format.

**Jenkins machine:**

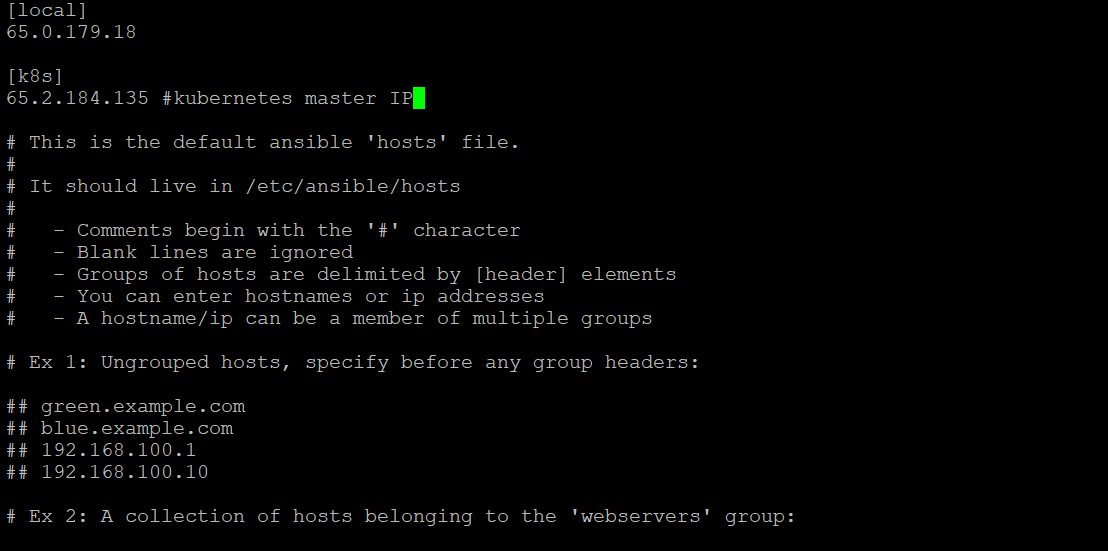
ssh ubuntu@<public-ip-k8s-master>

ssh ubuntu@65.2.184.135

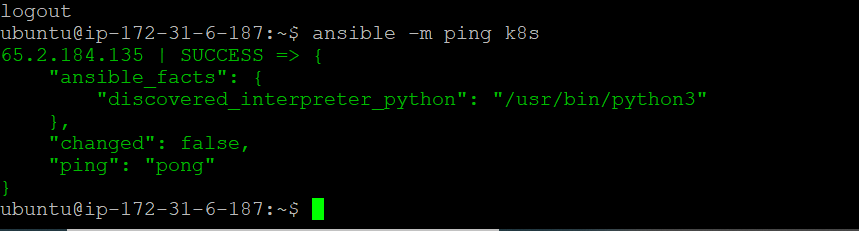
exit

cd /etc/ansible

sudo vi hosts



ansible -m ping k8s



ansible -m ping all#use this one

pipeline{

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

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'

}

}

}

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 ('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('Install Docker') {

steps {

dir('Ansible'){

script {

ansiblePlaybook credentialsId: 'ssh', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/', playbook: 'docker.yaml'

}

}

}

}

stage('k8s using ansible'){

steps{

dir('Ansible') {

script{

ansiblePlaybook credentialsId: 'ssh', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/', playbook: 'kube.yaml'

}

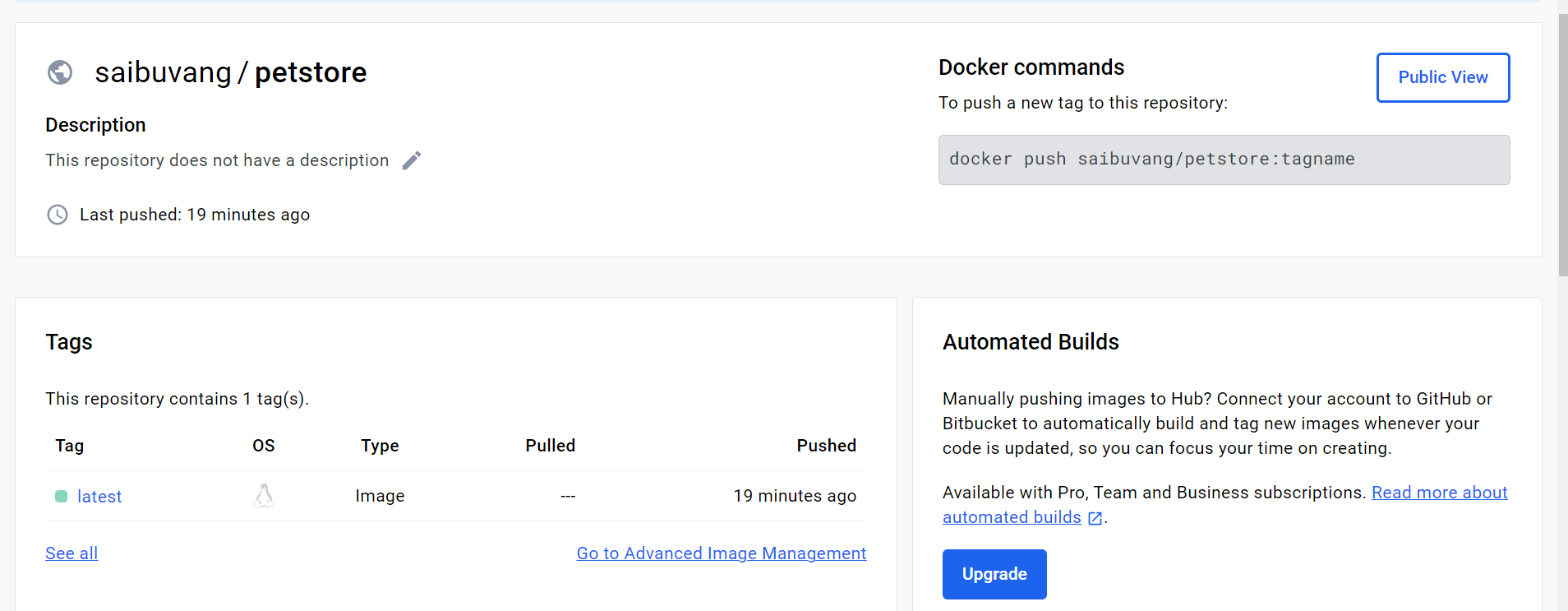
}

}

}

}

}



Build