DevOps Project\_02: Jenkins (CI/CD) , Using GitHub, Maven, SonarQube, Docker, Dependency check, Nexus , Tomcat For Pet shop

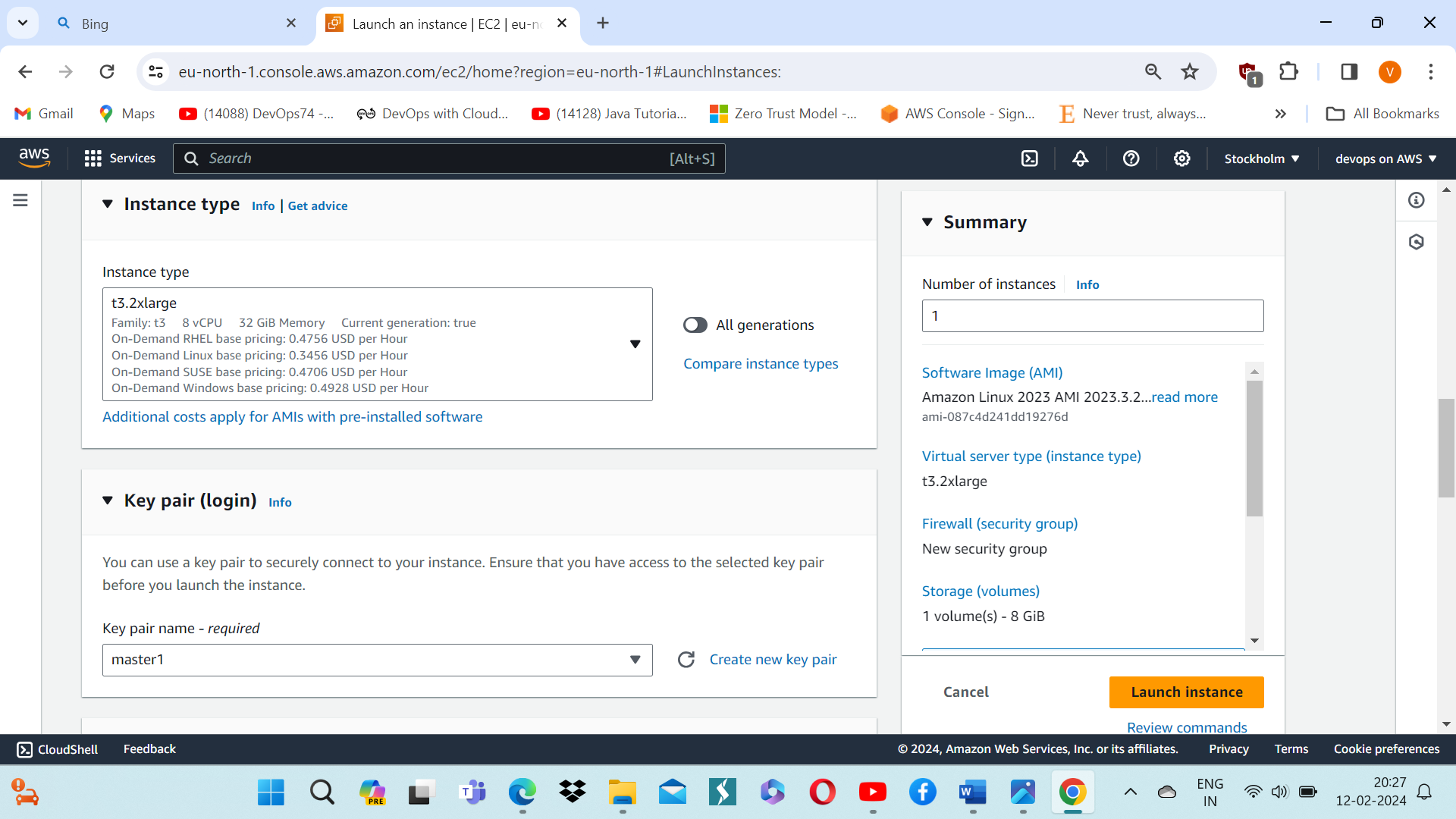
NAME: Y. VENKATESWARLU

BATCH NO:118

TIMINGS:4PM

TRAINER: Mr. MADHUKAR

STEP 1: Create an Ubuntu (22.04) t2 large instance in AWS Console.

* First we have to login aws management console and then launch new Instance.
* Select Ubuntu and click on instance type t3 large.
* Go to Security 🡪 click on Edit inbound Rules 🡪 Add rule 🡪Select All Traffic and then save rules.

STEP 2: Connect to Server and Install both Jenkins and Docker

* We have convert to root user and then enter sudo -i.
* We have to enter this commands and 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**](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**](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**](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/**](https://pkg.jenkins.io/debian-stable%20binary/) **| sudo tee \**

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

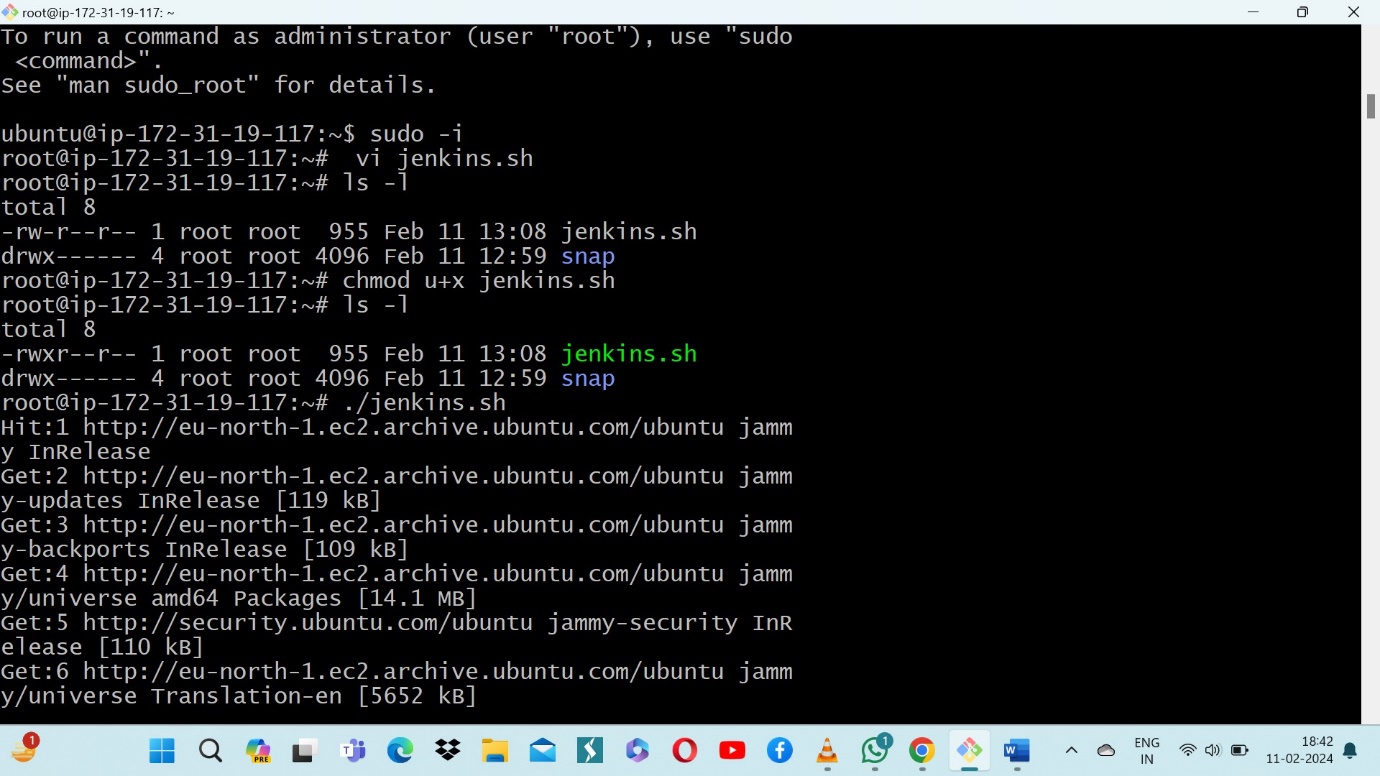
**sudo apt-get update -y**

**sudo apt-get install jenkins -y**

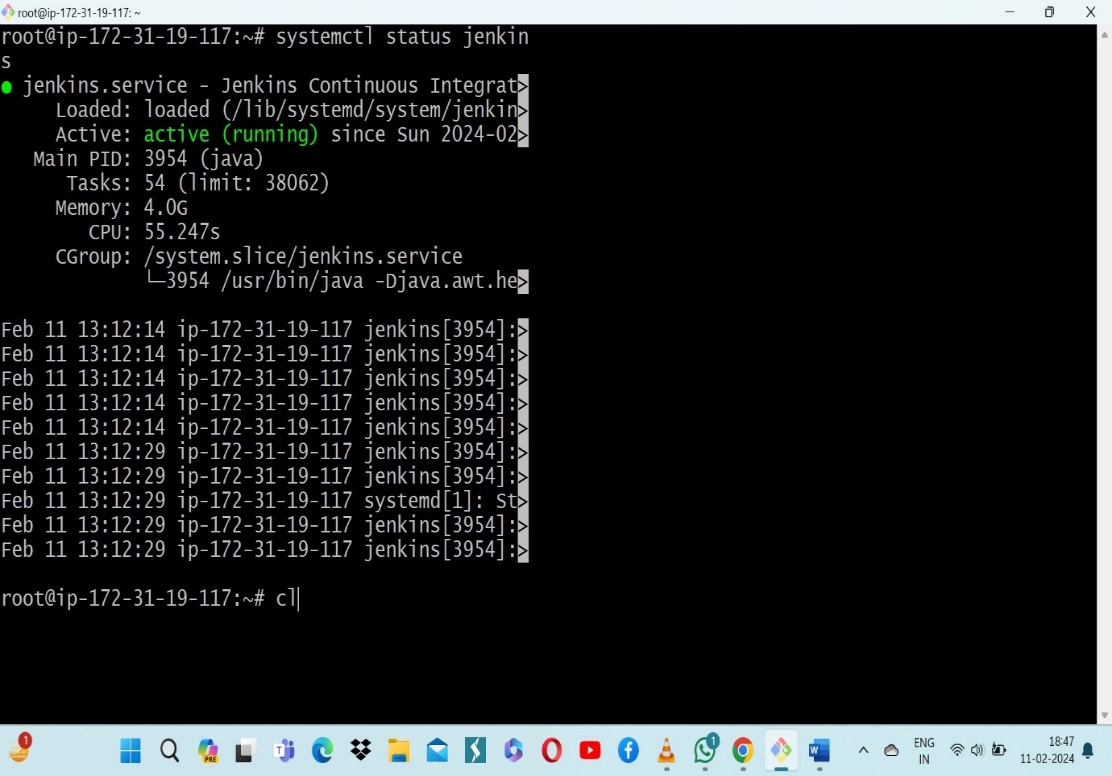
**sudo systemctl start jenkins**

**sudo systemctl status Jenkins**

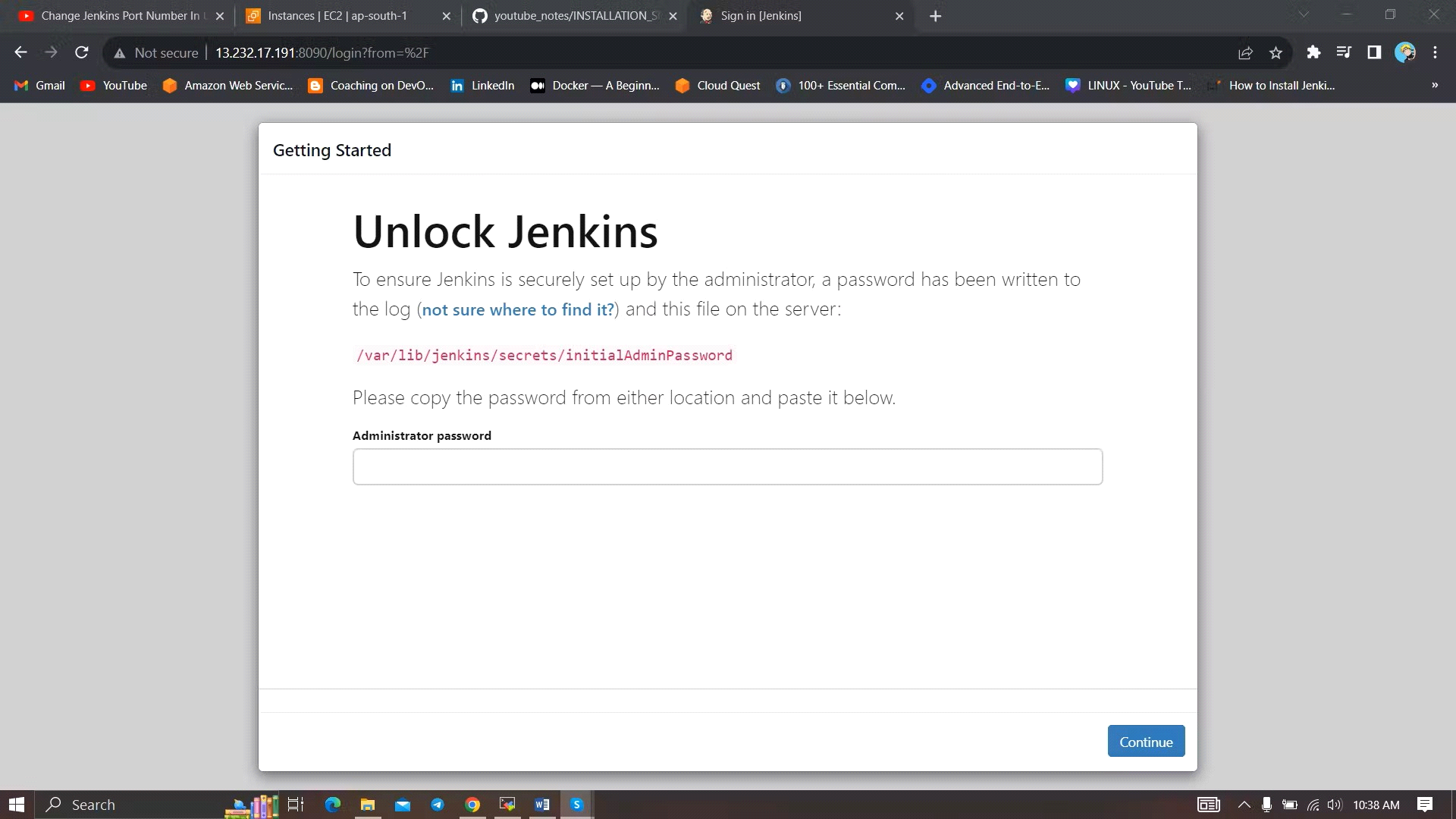
* Enter **esc + shift + : wq**
* Chmod u+x Jenkins.sh
* ./Jenkins.sh



* Status of Jenkins

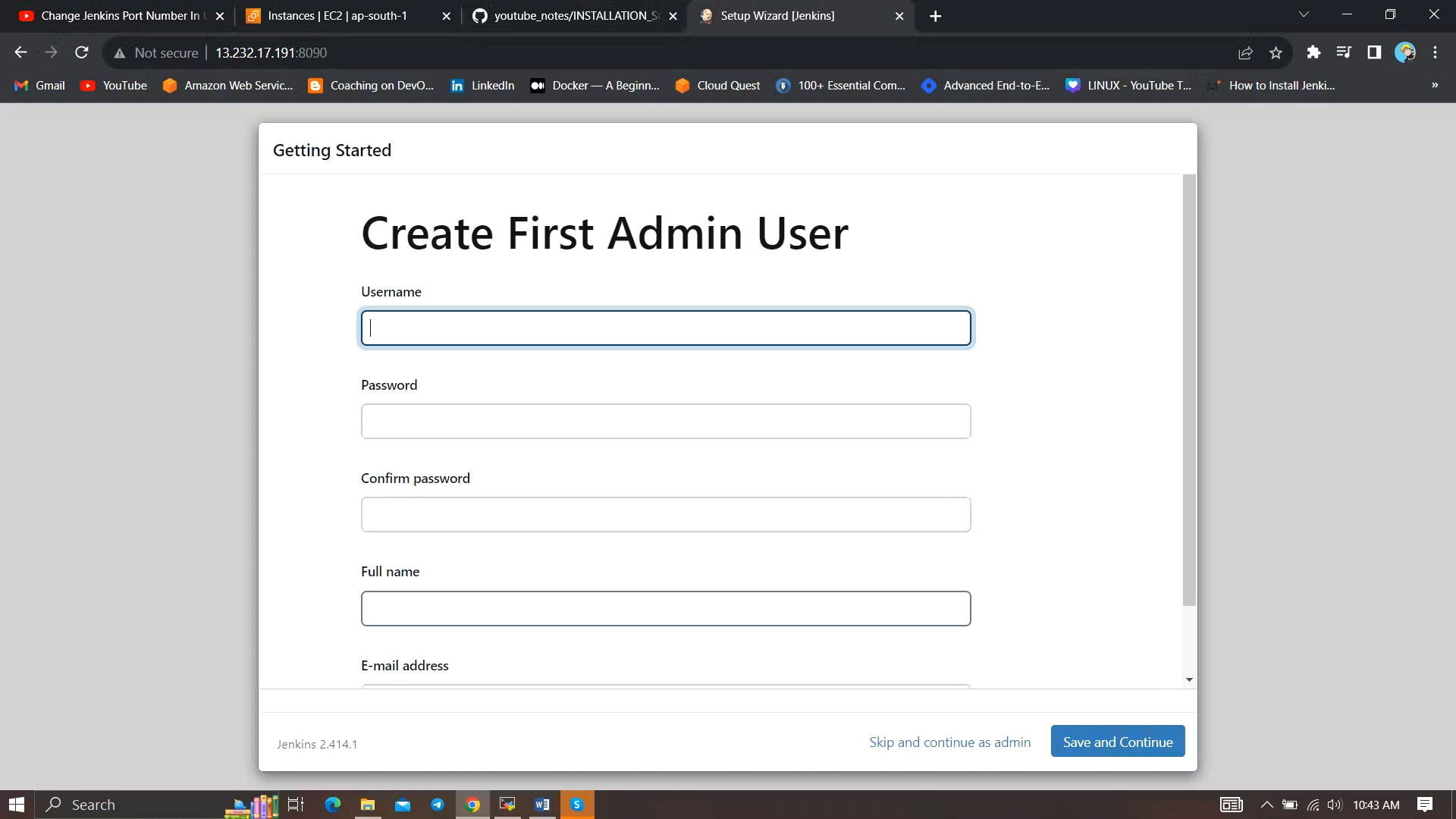


* **<EC2 Public IP Address:8080>**
* **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**



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.



* Install Docker

sudo apt-get update

sudo apt-get install docker.io -y

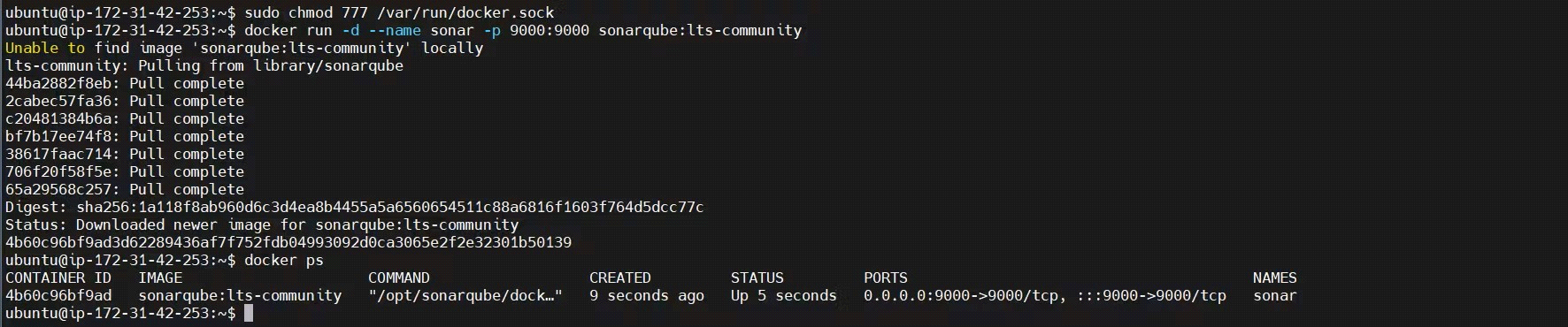
sudo usermod -aG docker $USER #my case is ubuntu

newgrp docker

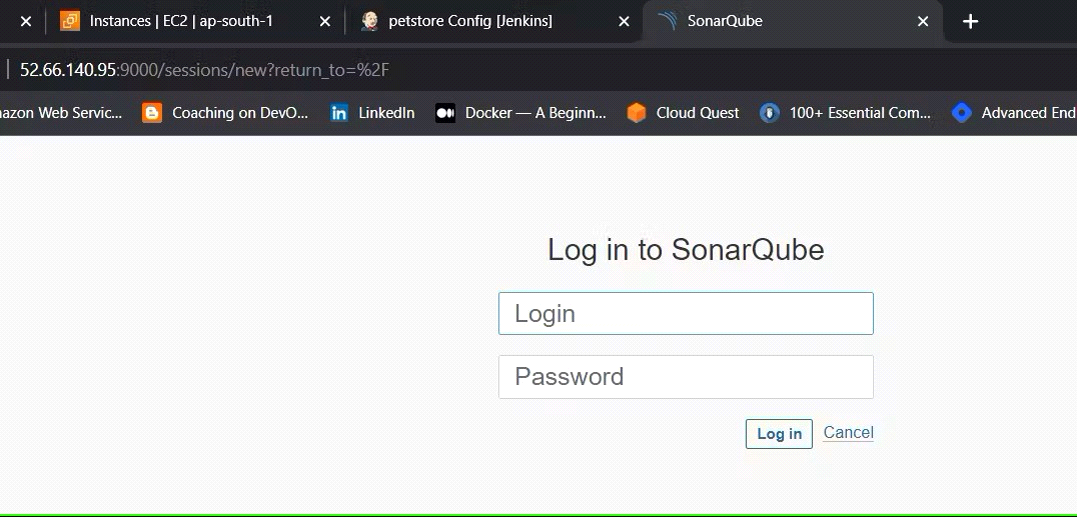
sudo chmod 777 /var/run/docker.sock

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



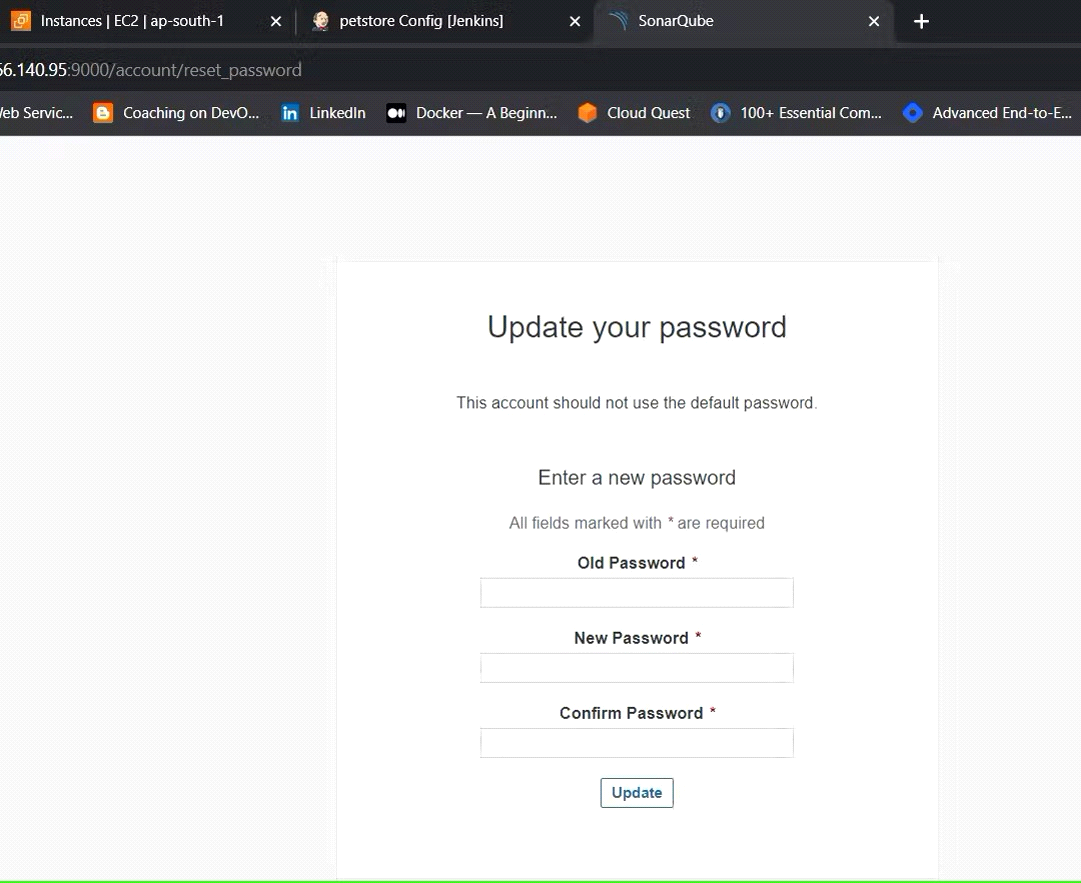
Now our SonarQube is up and running



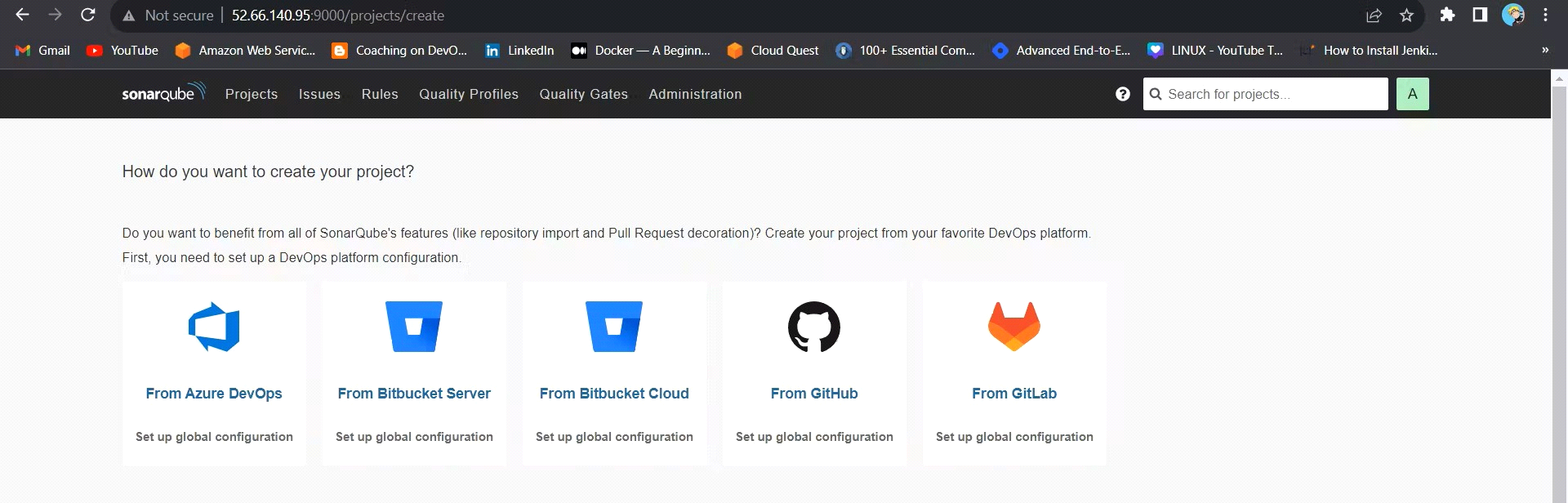
Enter username and password, click on login and change password

***Username: admin***

***password: admin***



Update New password, This is Sonar Dashboard.



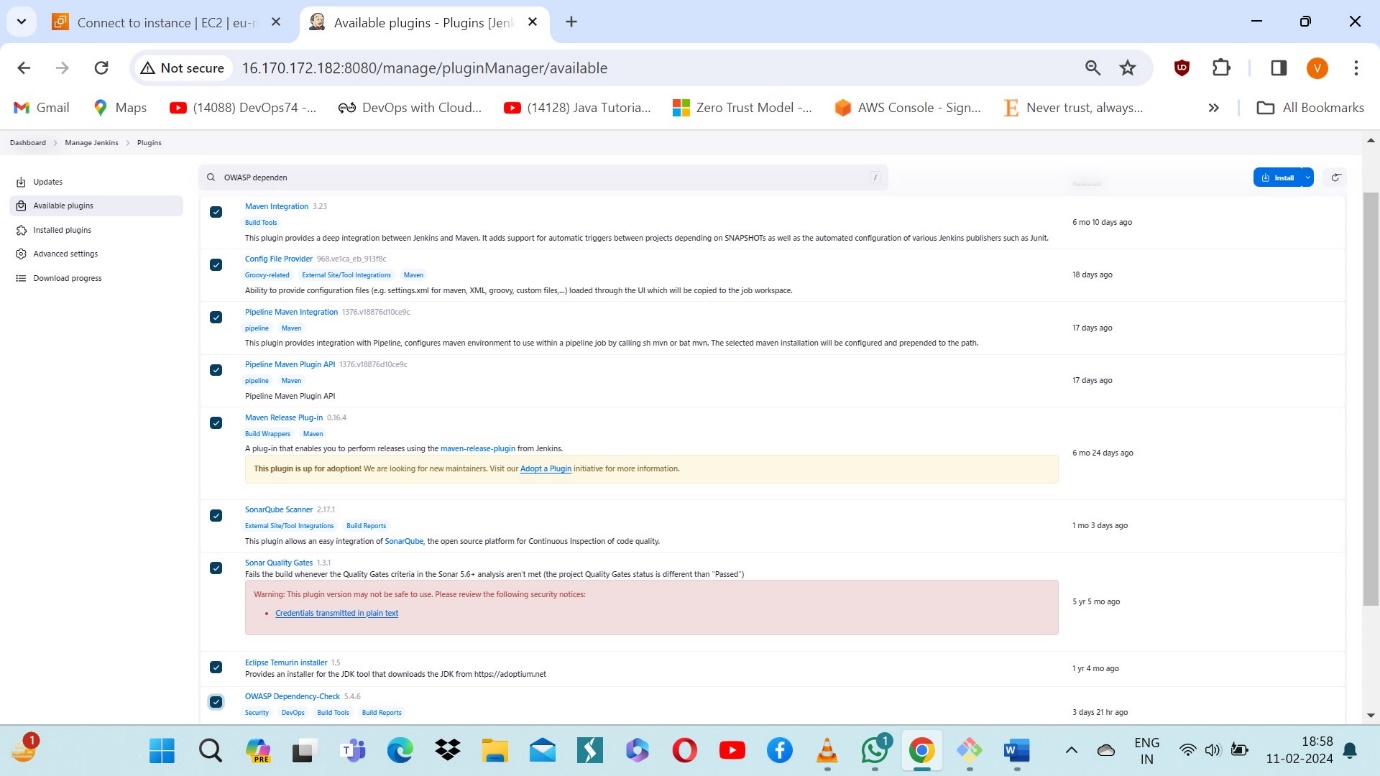
Step 3 : Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check in Jenkins

Goto Manage Jenkins →Plugins → Available Plugins →

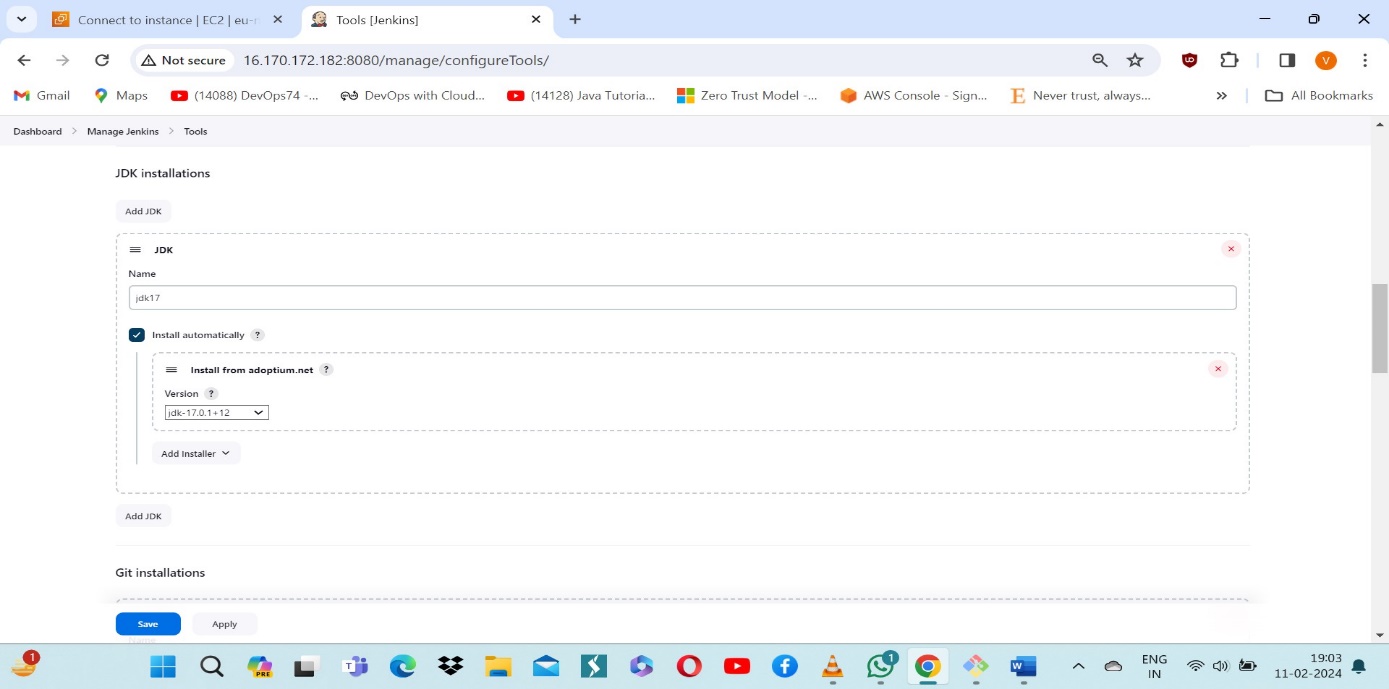
Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

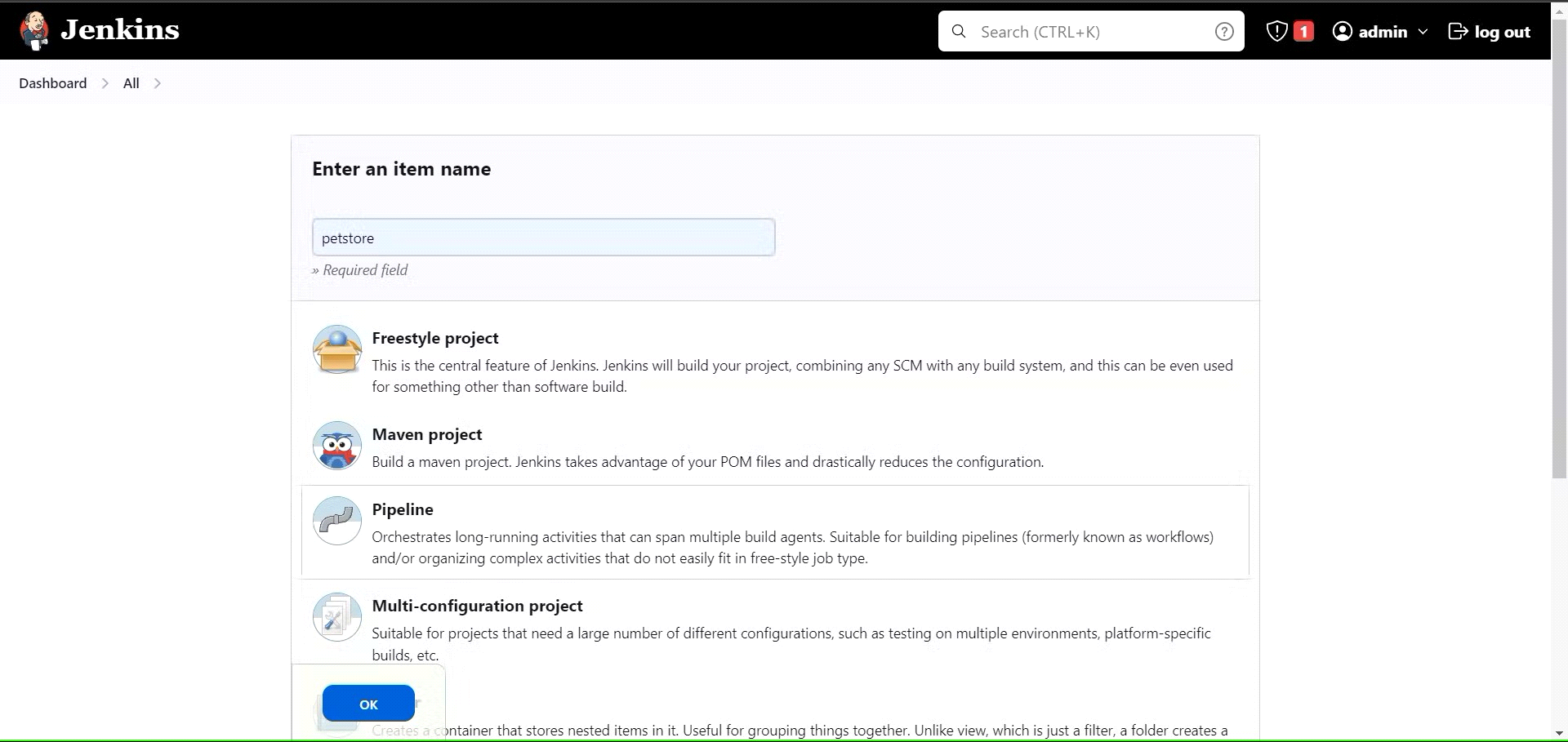


* Configure Java and Maven in Global Tool Configuration

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



* 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'

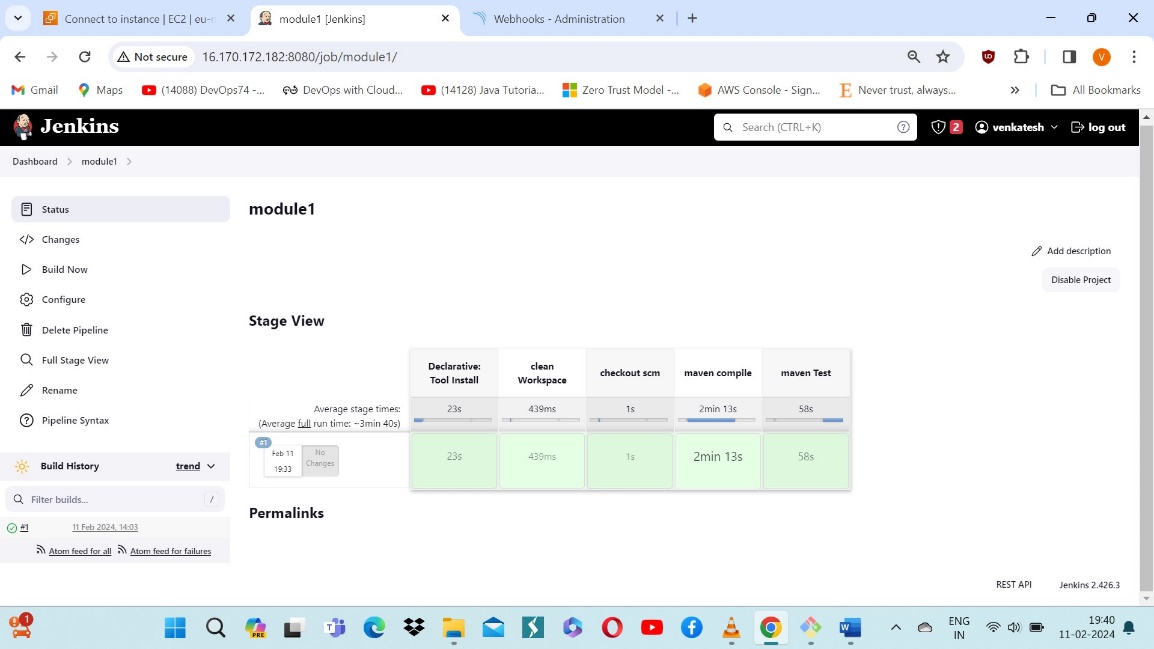
}

}

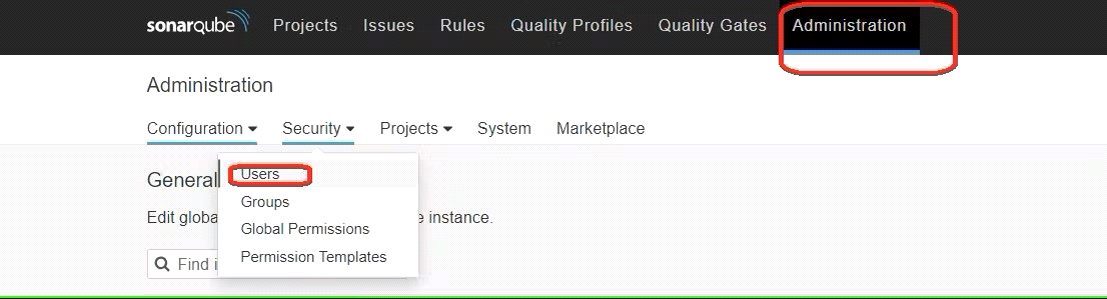
}

}

The stage view would look like this,

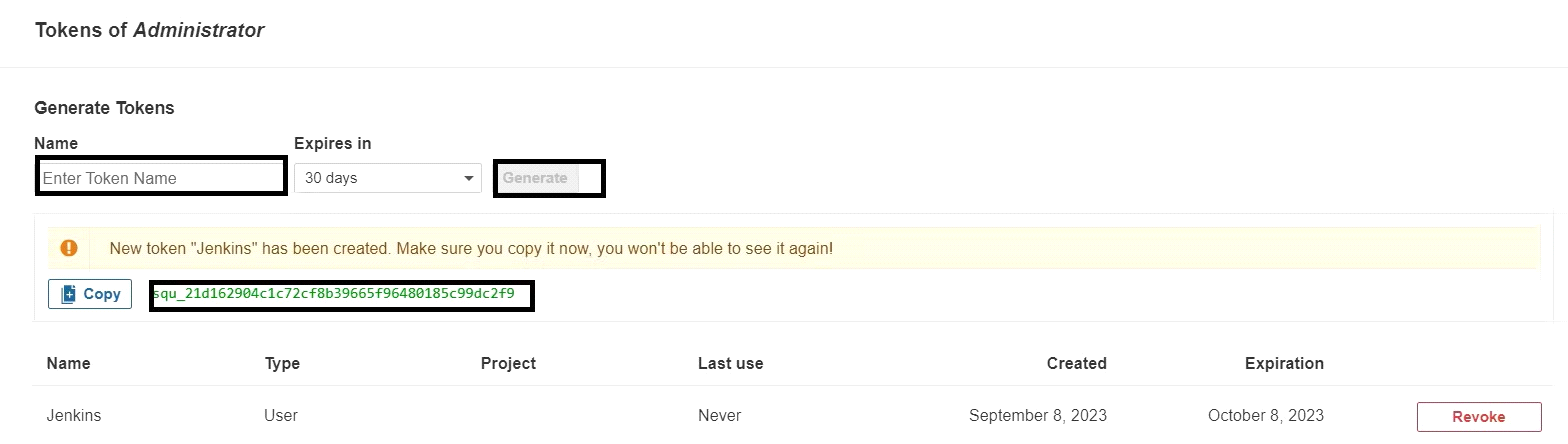


Step 4: Configure Sonar Server in Manage Jenkins





Create a token with a name and generate

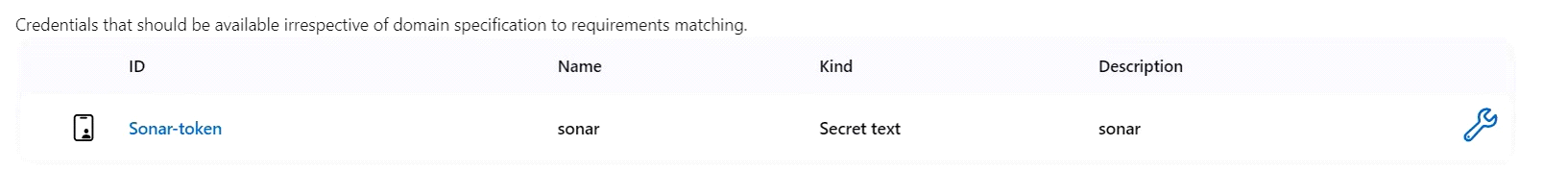


copy Token

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.

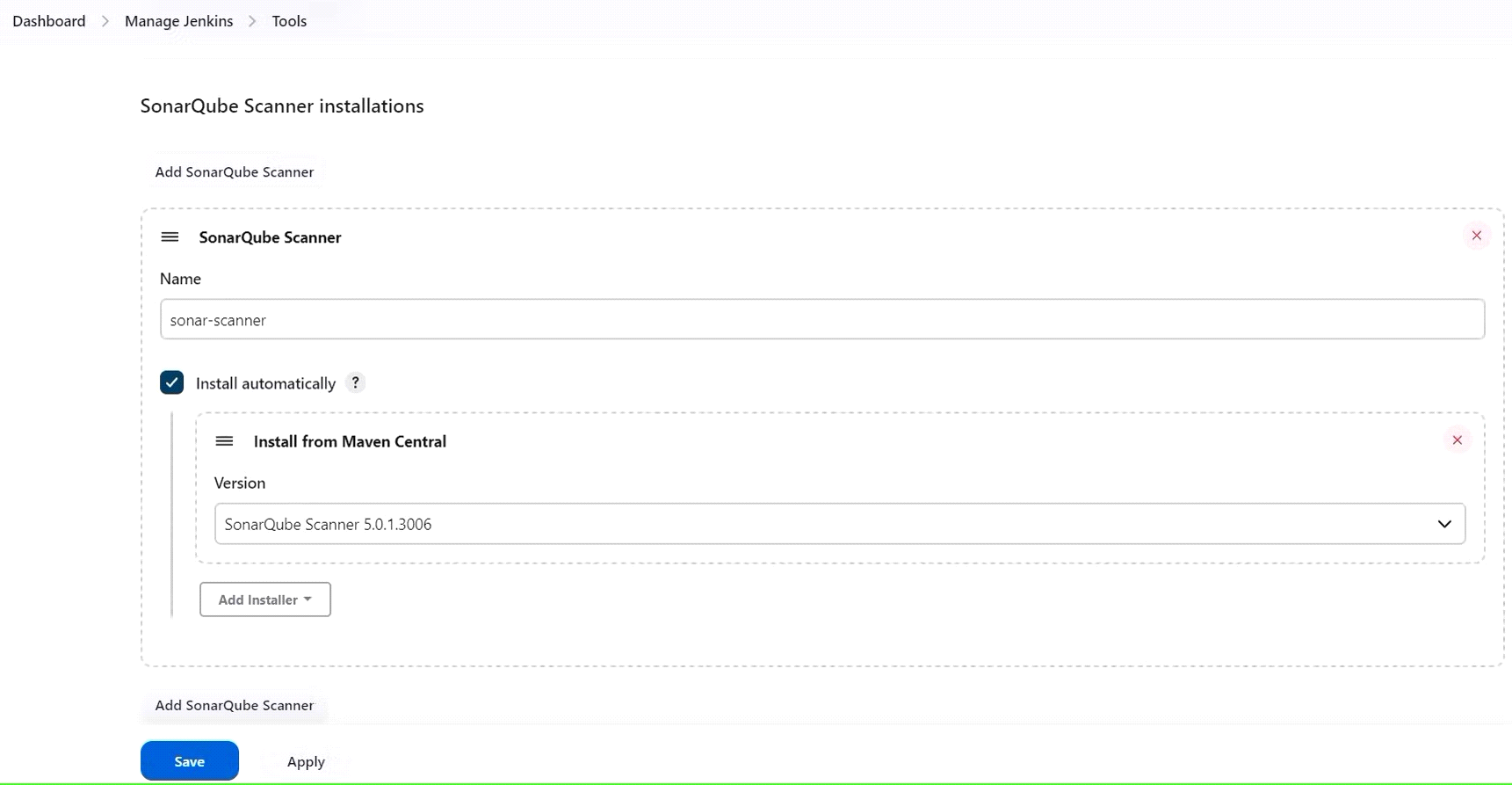


Click on Apply and Save

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

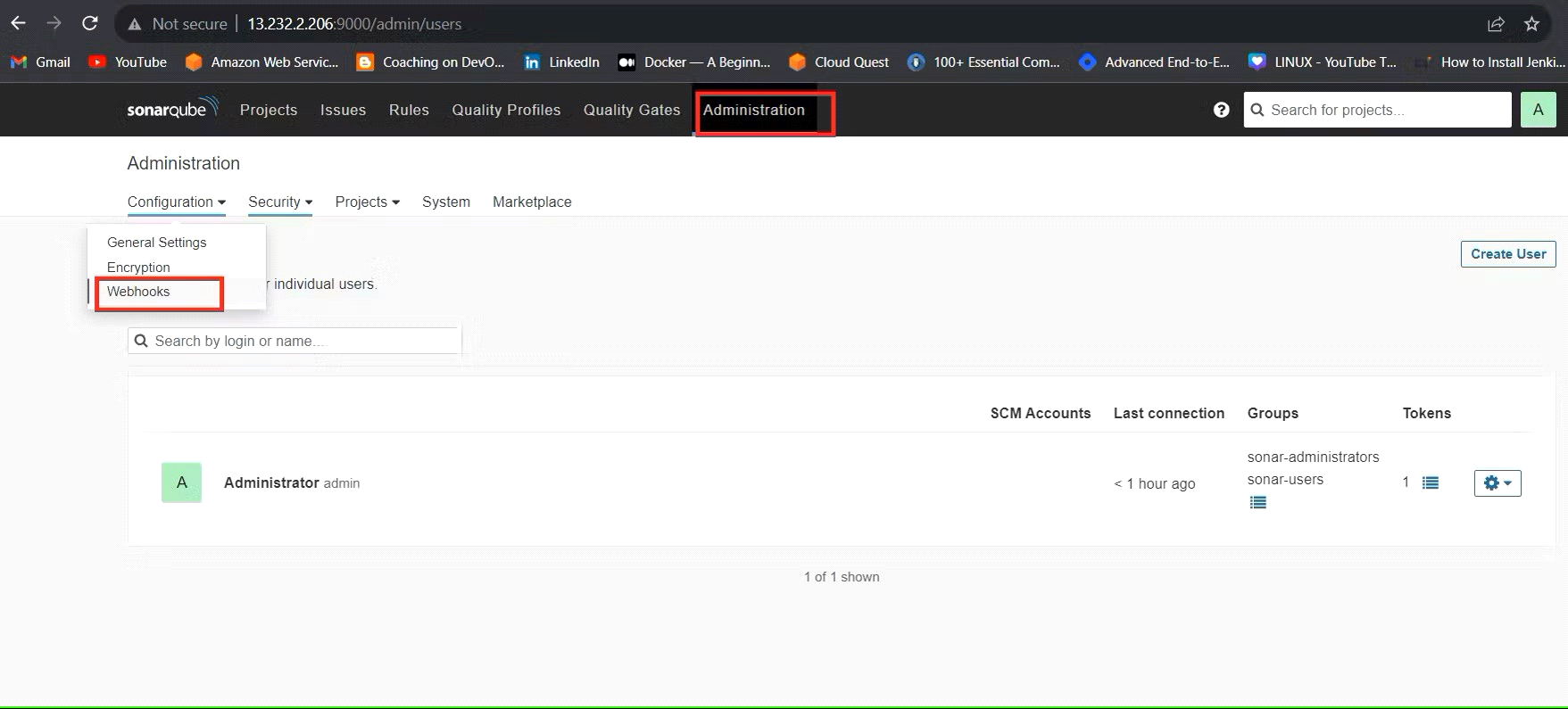
**Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

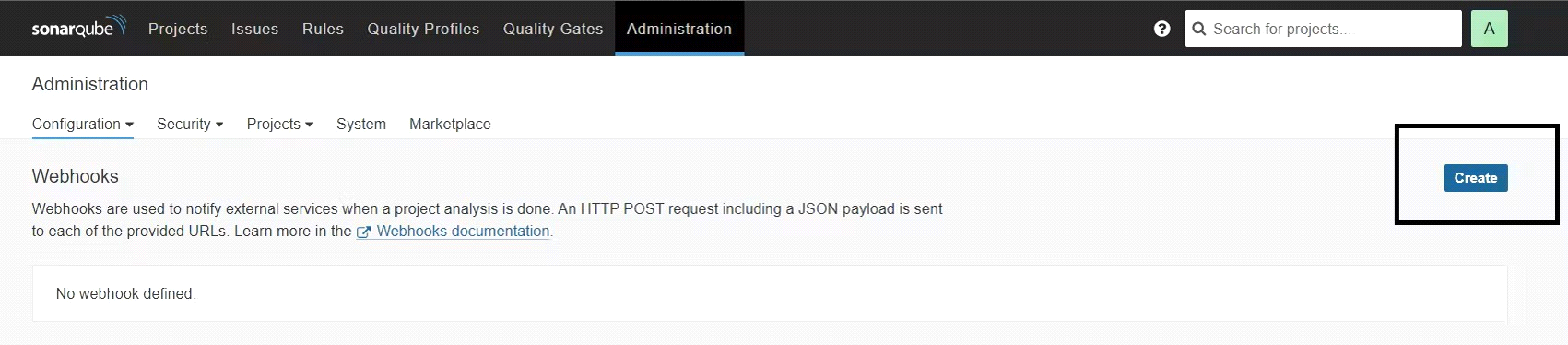


In the Sonarqube Dashboard add a quality gate also

Administration--> Configuration-->Webhooks



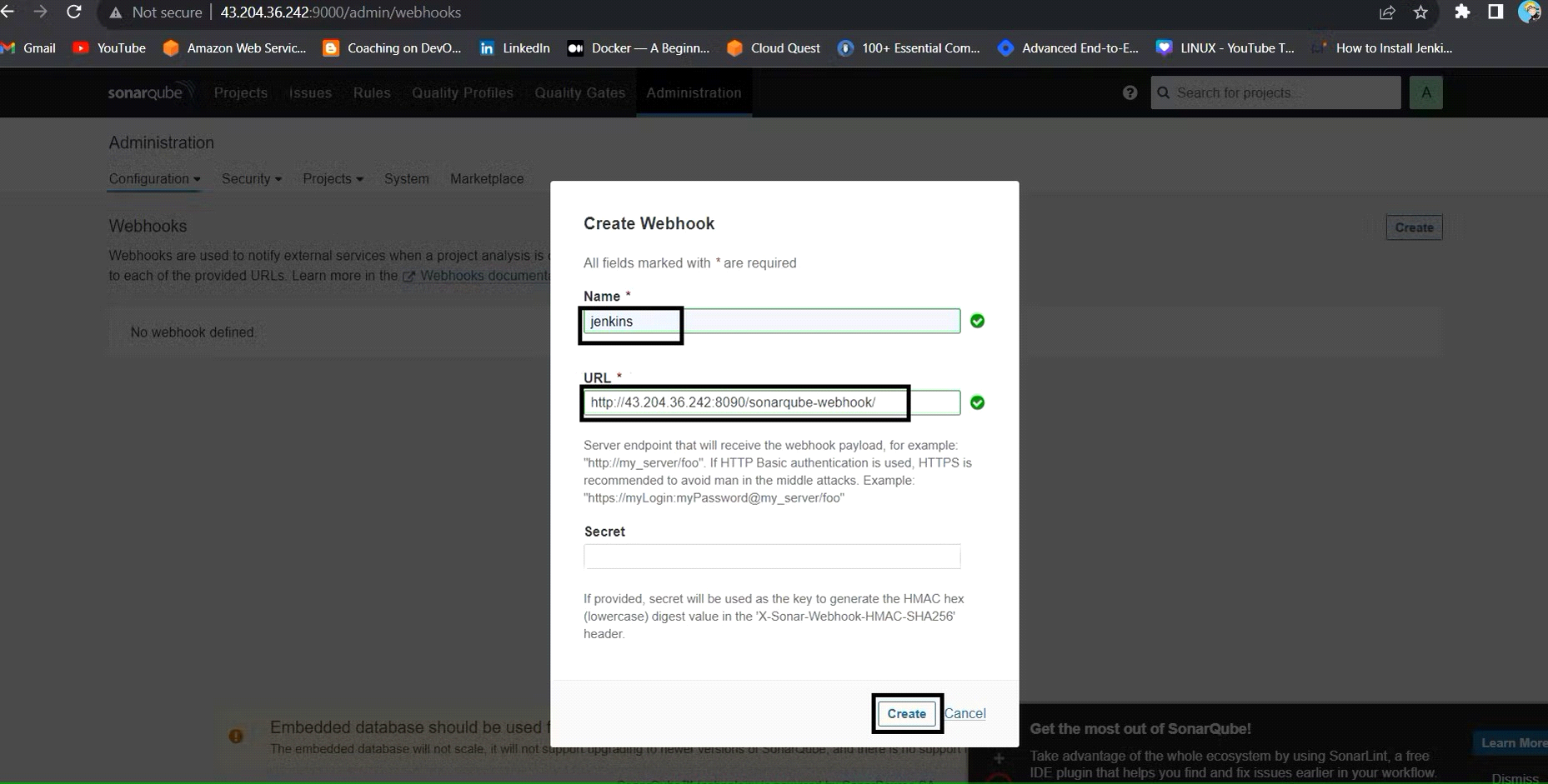
Click on Create



Add details

#in url section of quality gate

<<http://jenkins-public-ip:8080>>/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 {

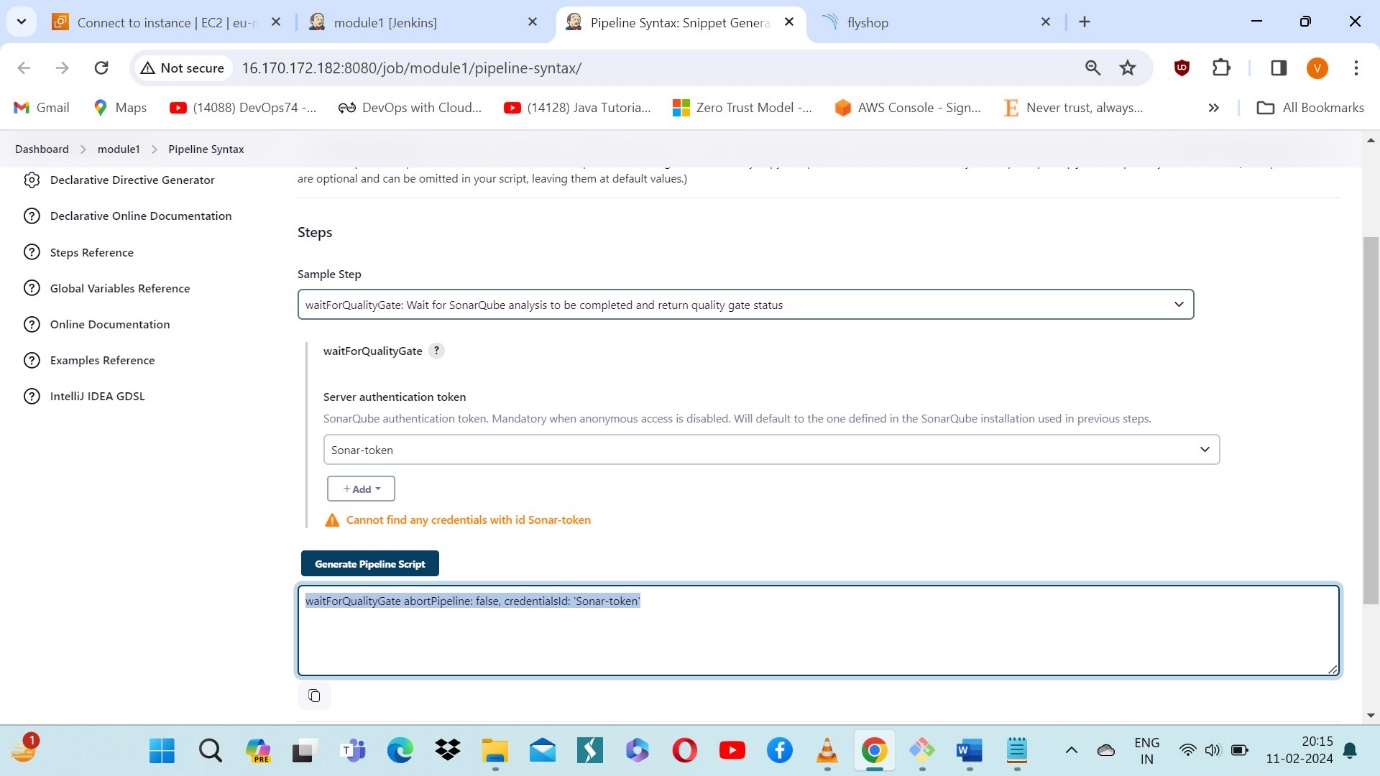
waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

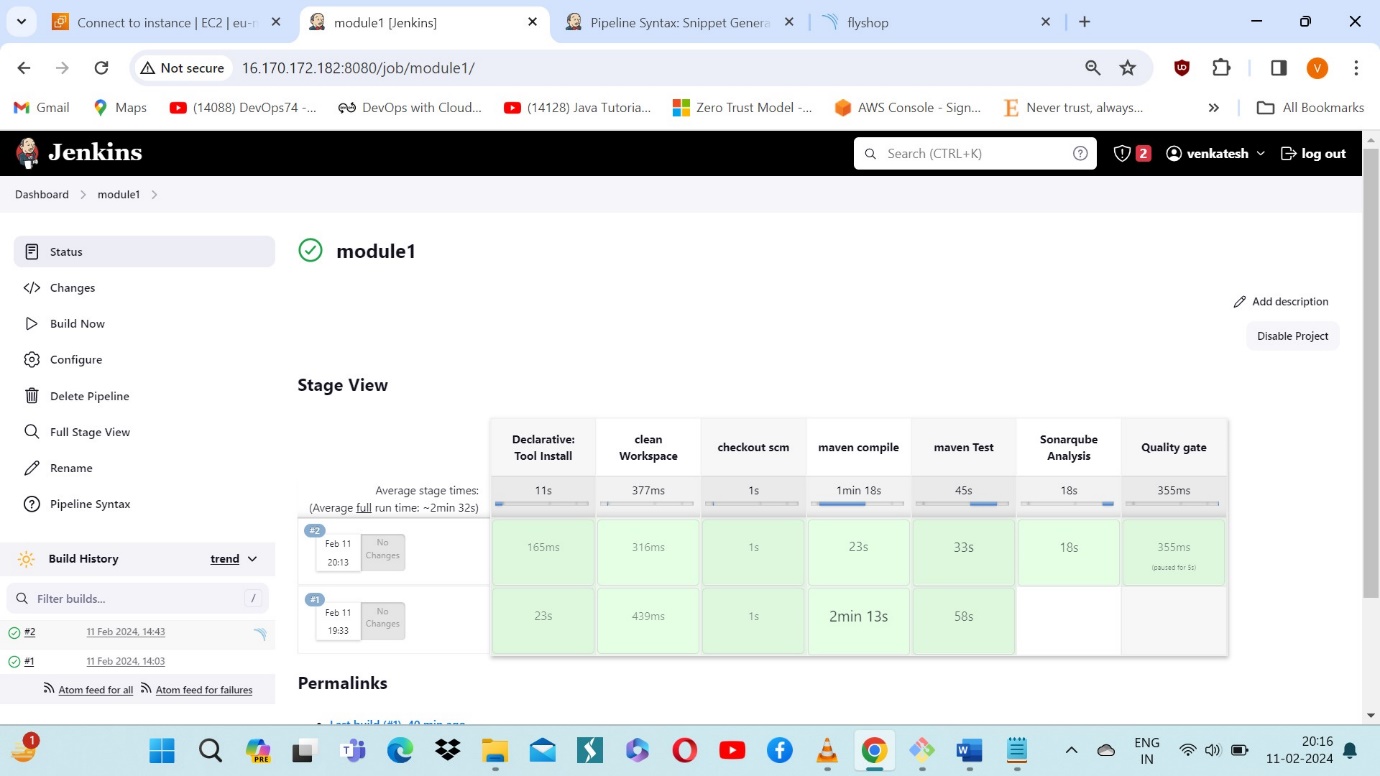
}

}

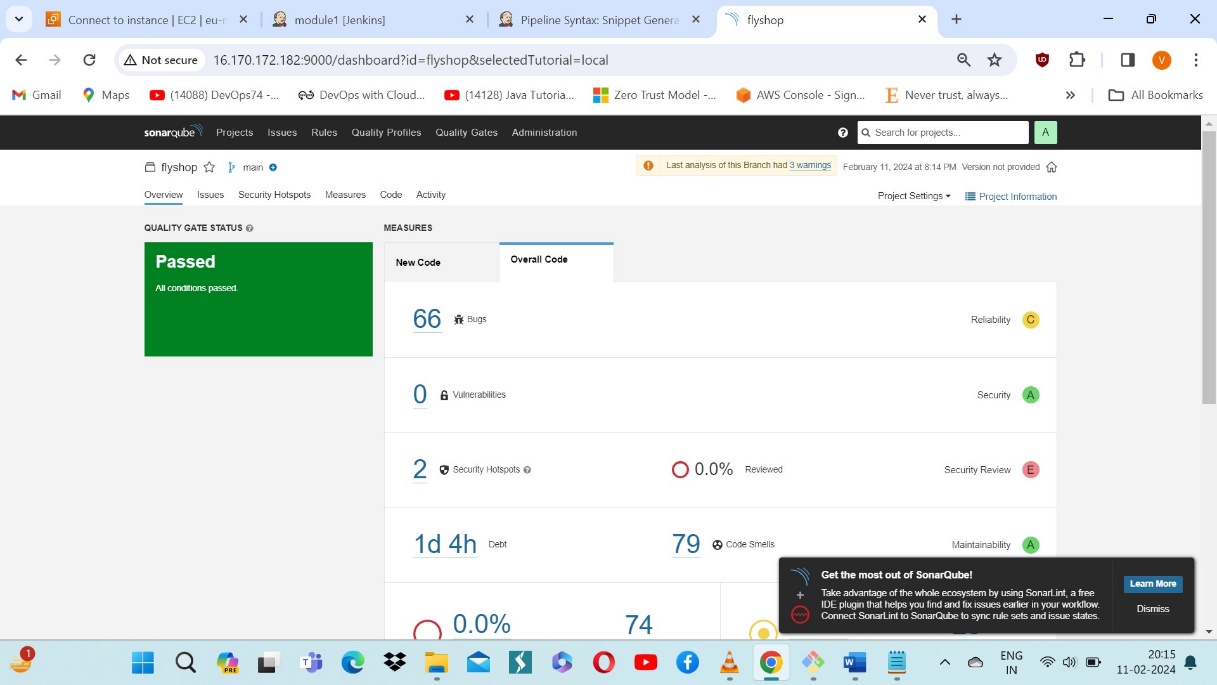
}

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

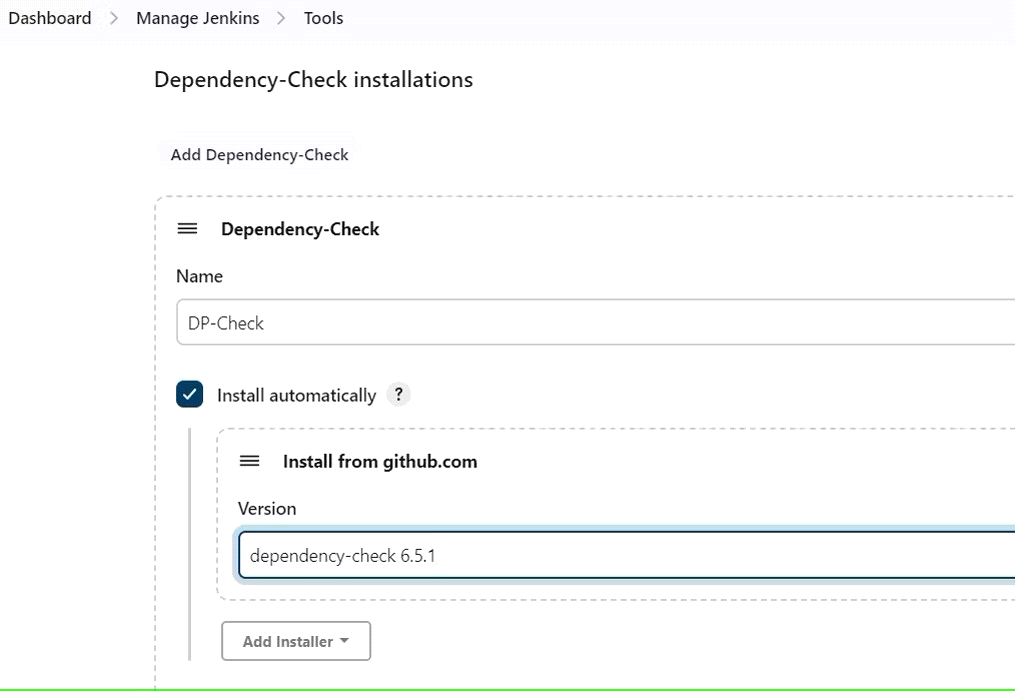




To see the report, you can go to Sonarqube Server and go to Projects.



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 ('package'){

steps{

sh 'mvn clean install -DskipTests=true'

}

}

stage("dependency Check"){

steps{

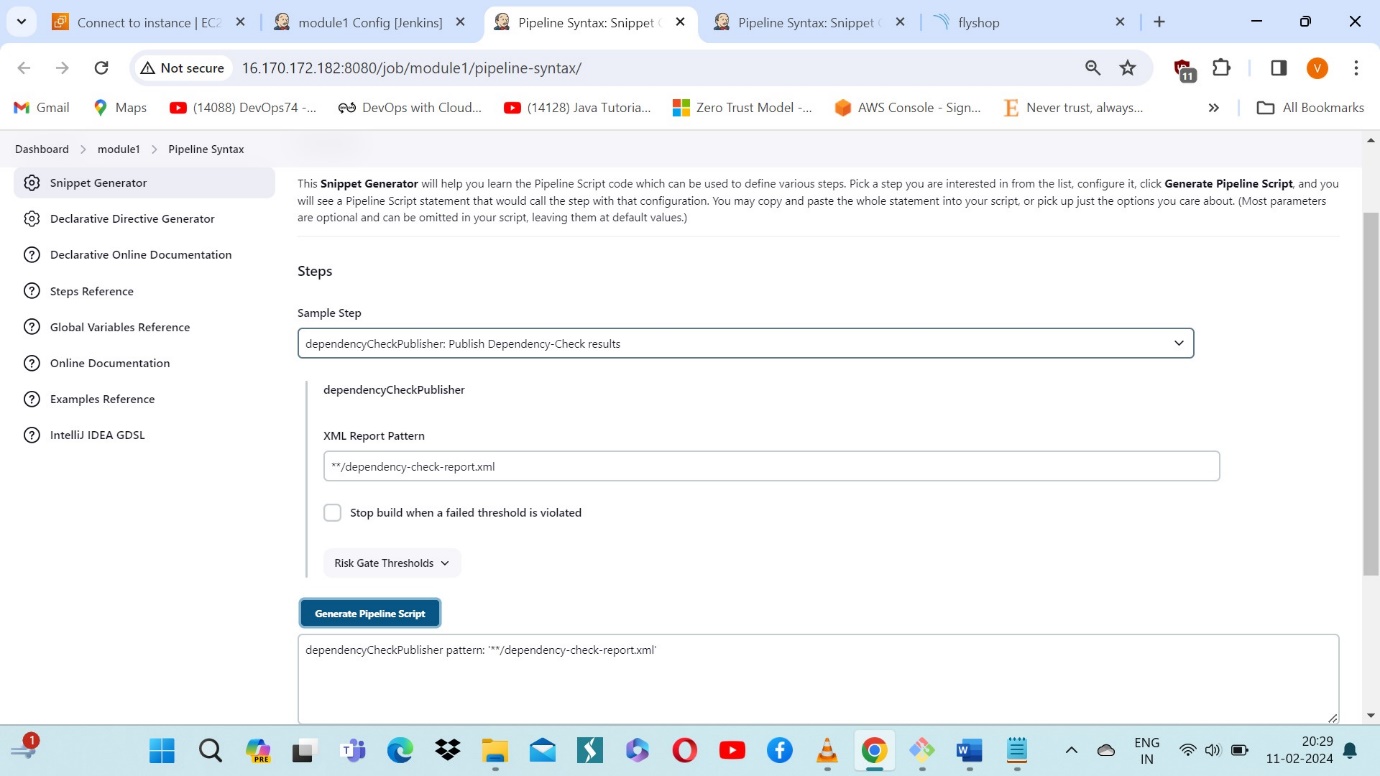
dependencyCheck additionalArguments: '--scan ./ --format XML ', odcInstallation: 'DP-Check'

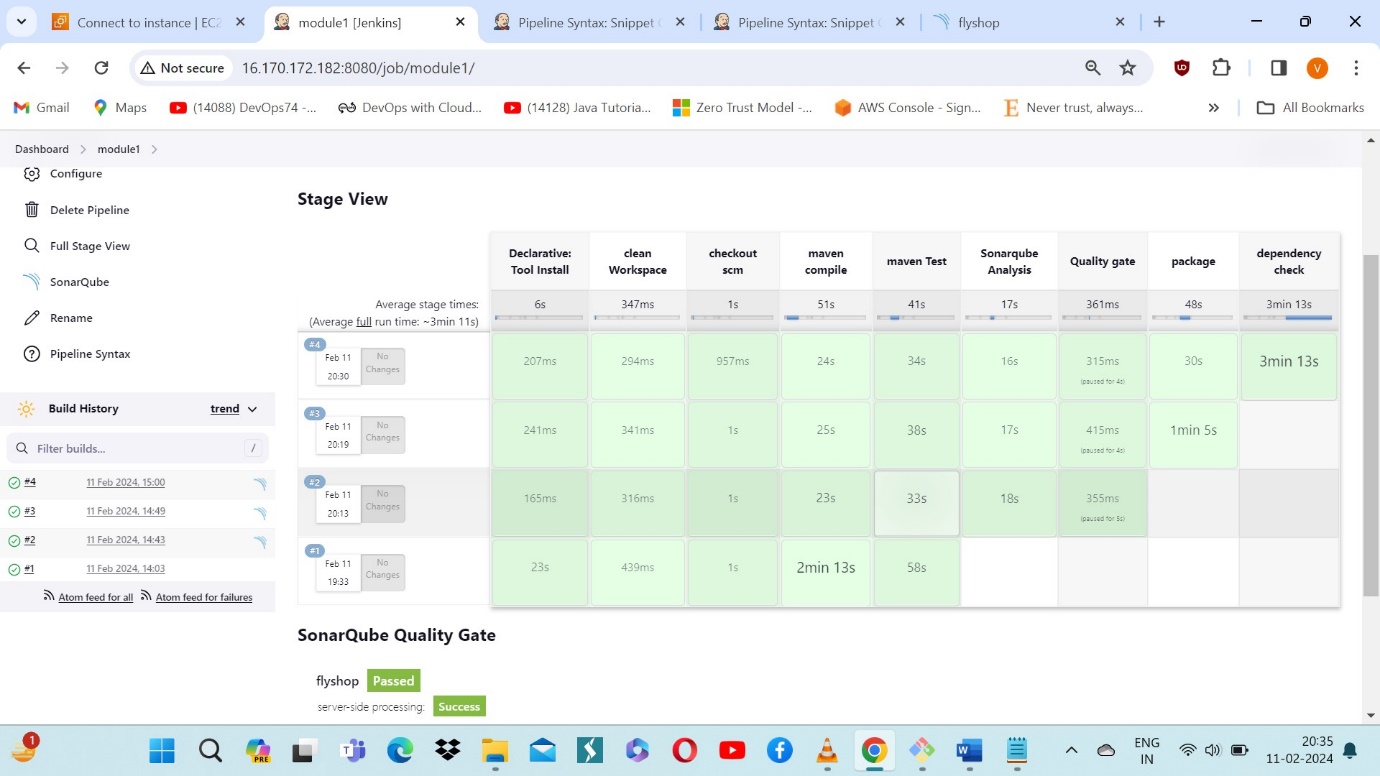
dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

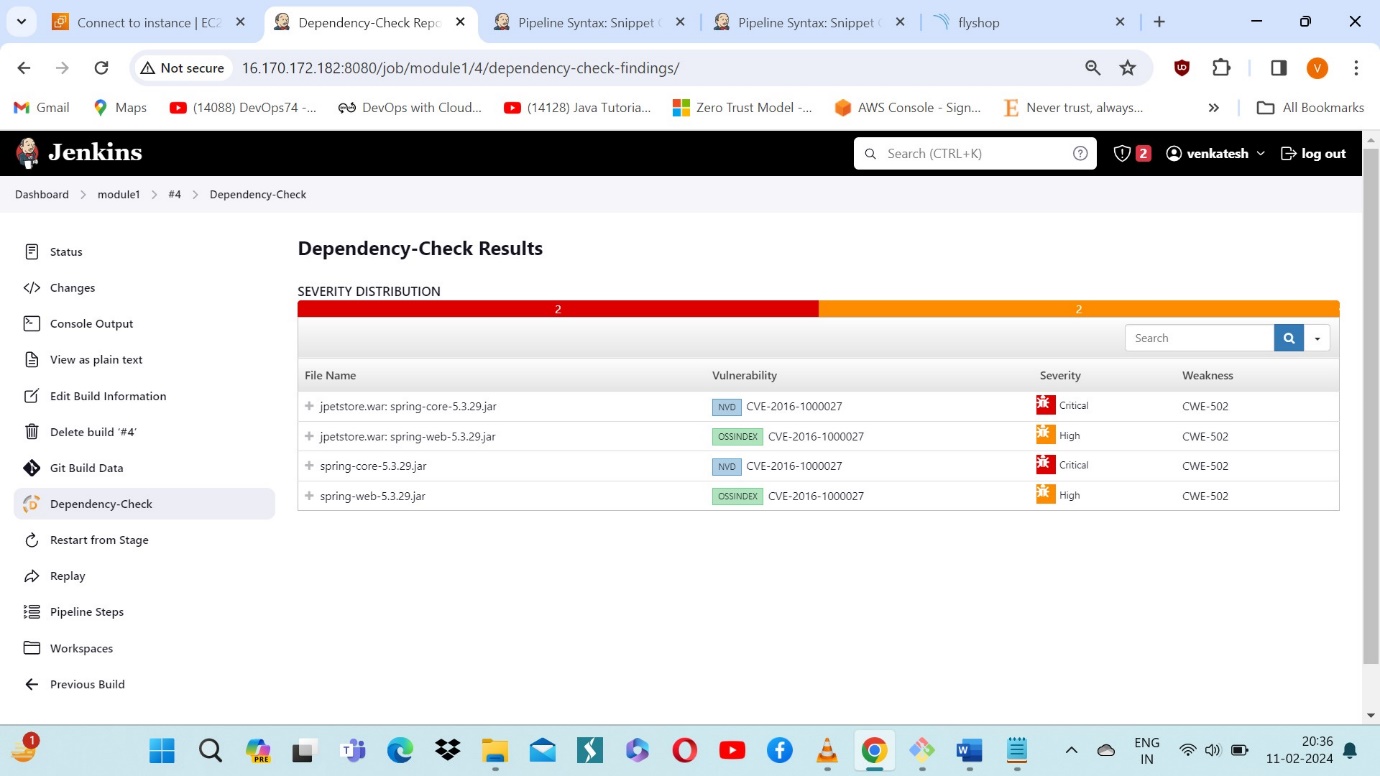
}

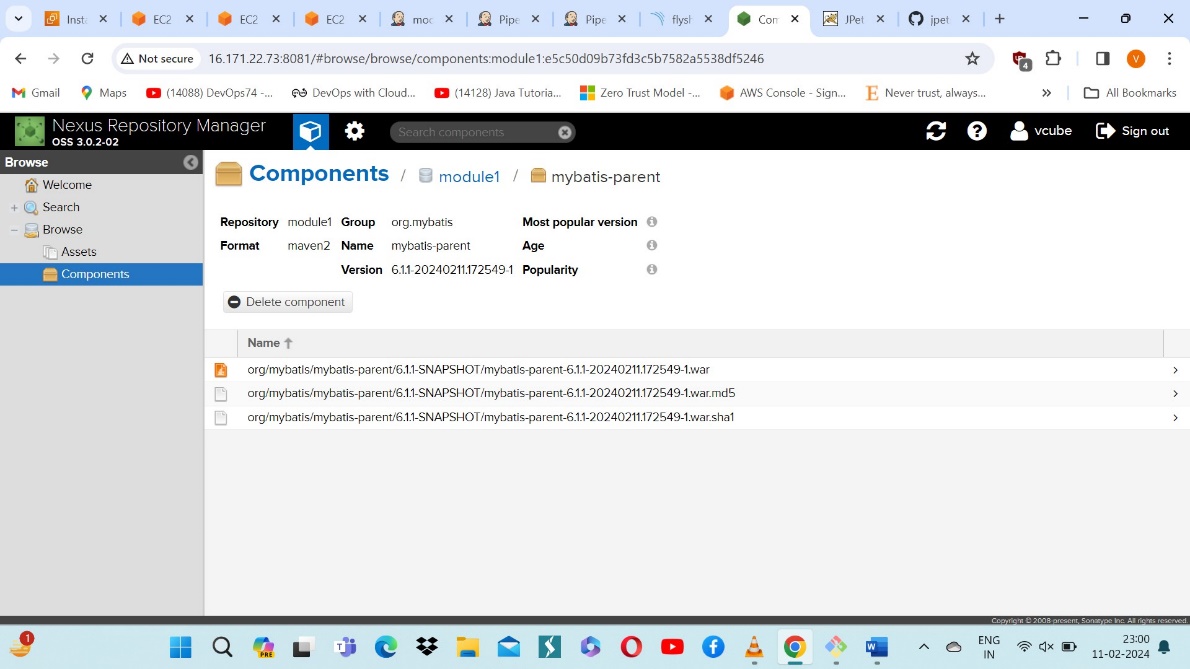
The stage view would look like this,

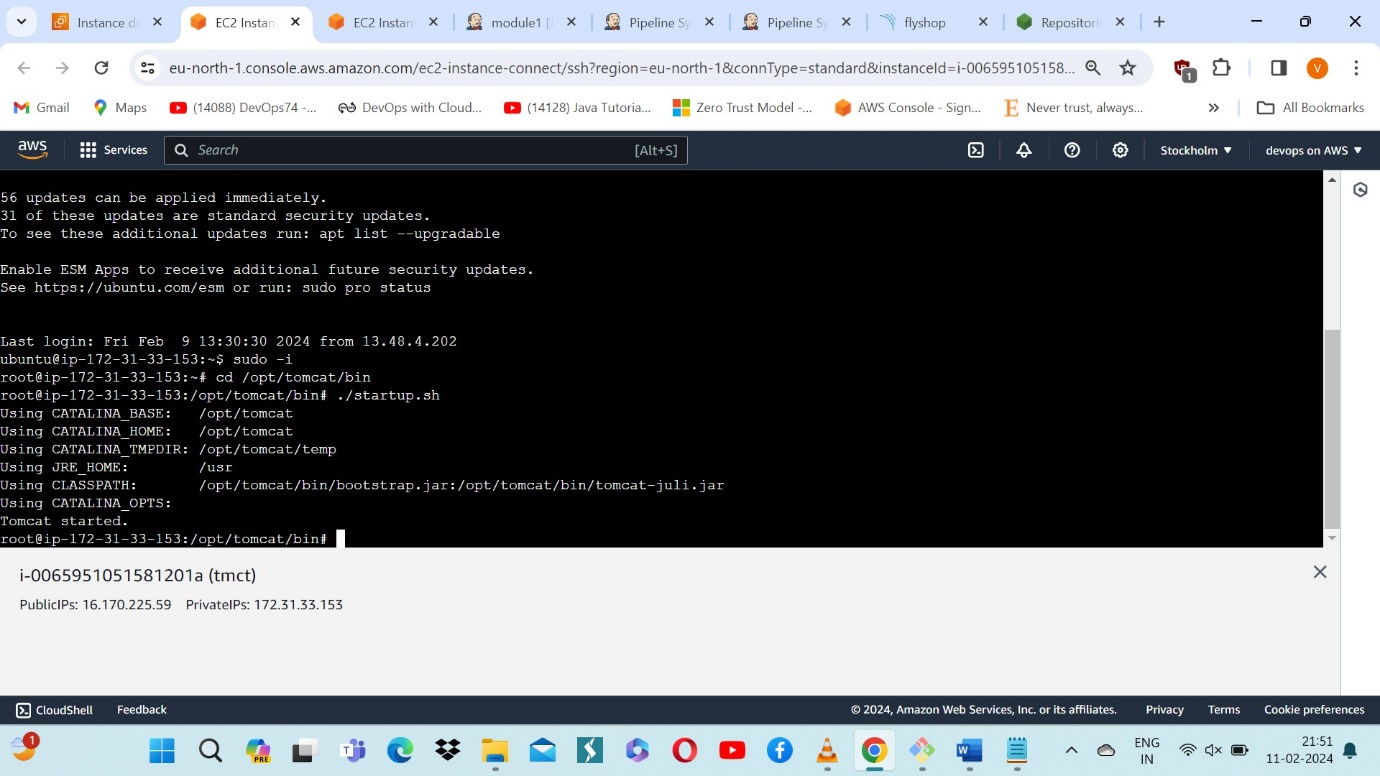




You will see that in status, a graph will also be generated and Vulnerabilities.



* We have to Install Nexus and integrate the Jenkins pipeline.
* The following diagram reprents the nexus
* 

* Now install both tomcat and default jdk.
* 

Pipeline script:

stage('nexus artifact'){

steps{

nexusArtifactUploader artifacts: [[artifactId: 'mybatis-parent', classifier: '', file: '/var/lib/jenkins/workspace/module1/target/jpetstore.war', type: 'war']], credentialsId: 'vcube-nexus', groupId: 'org.mybatis', nexusUrl: '16.171.22.73:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'module1', version: '6.1.1-SNAPSHOT'

}

}

stage('deploy to tomcat'){

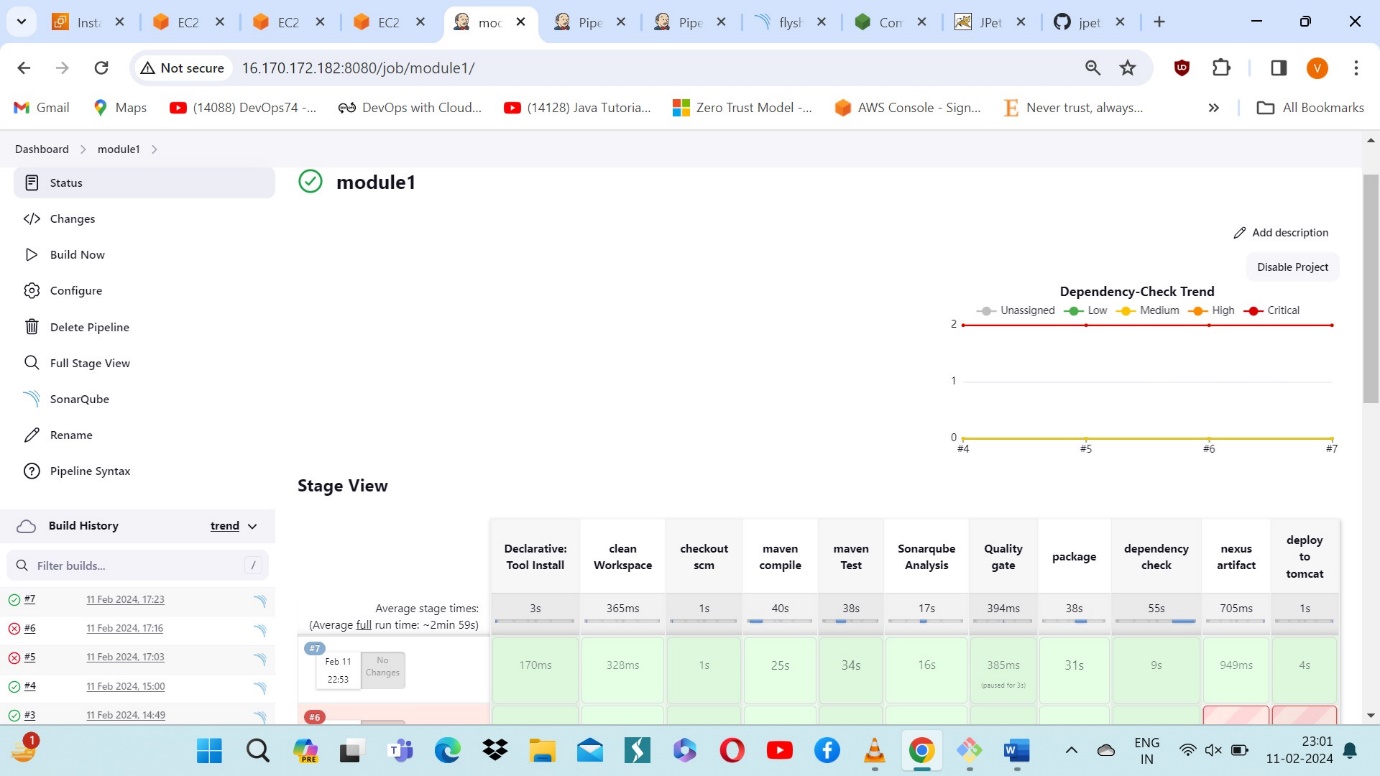
steps{

deploy adapters: [tomcat9(credentialsId: 'jenkins1', path: '', url: 'http://16.170.225.59:8080/')], contextPath: 'module01', war: '\*\*/\*.war'

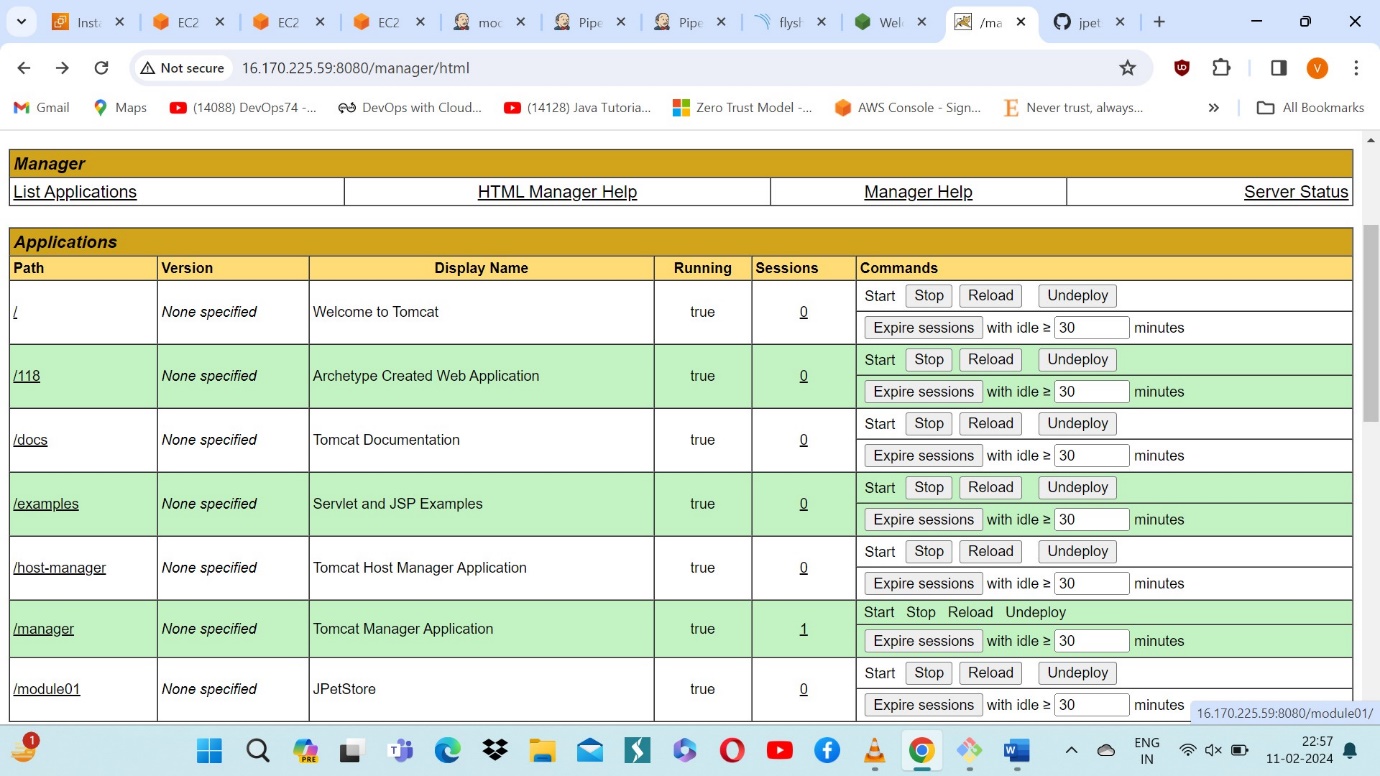
}

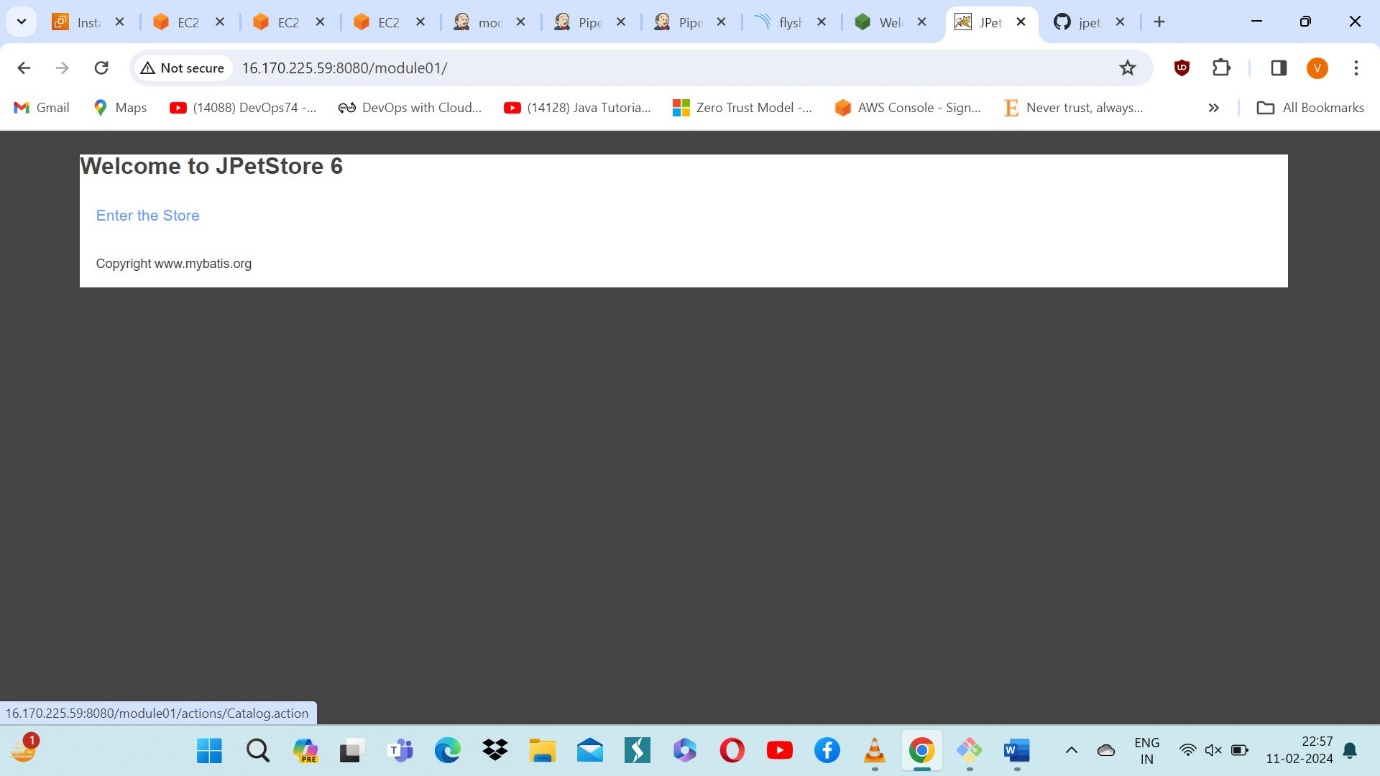
}

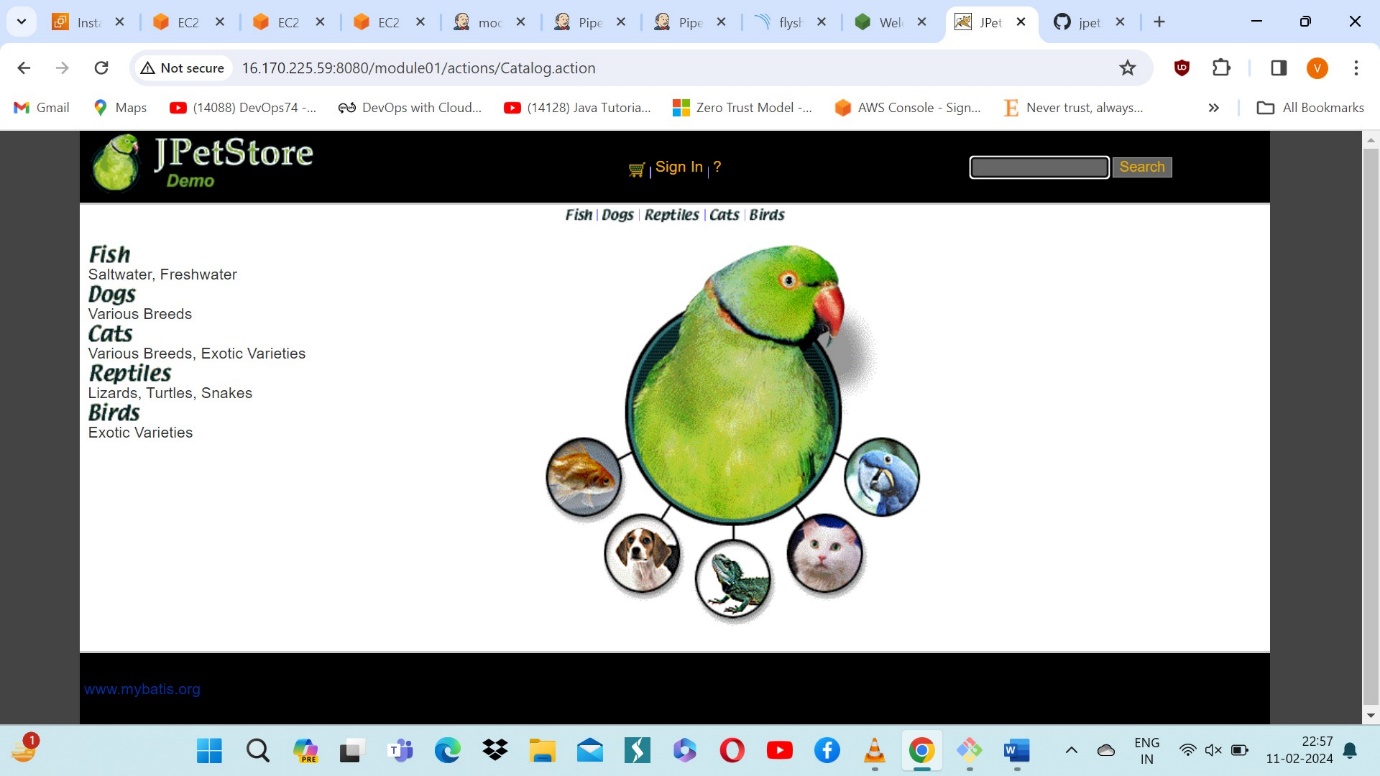
You will see the output below, with a dependency trend.



* The final output is :







*THANK YOU…….*