PROJECT-2

Course: DevOps

Name: Vanguri Raja Vamsy

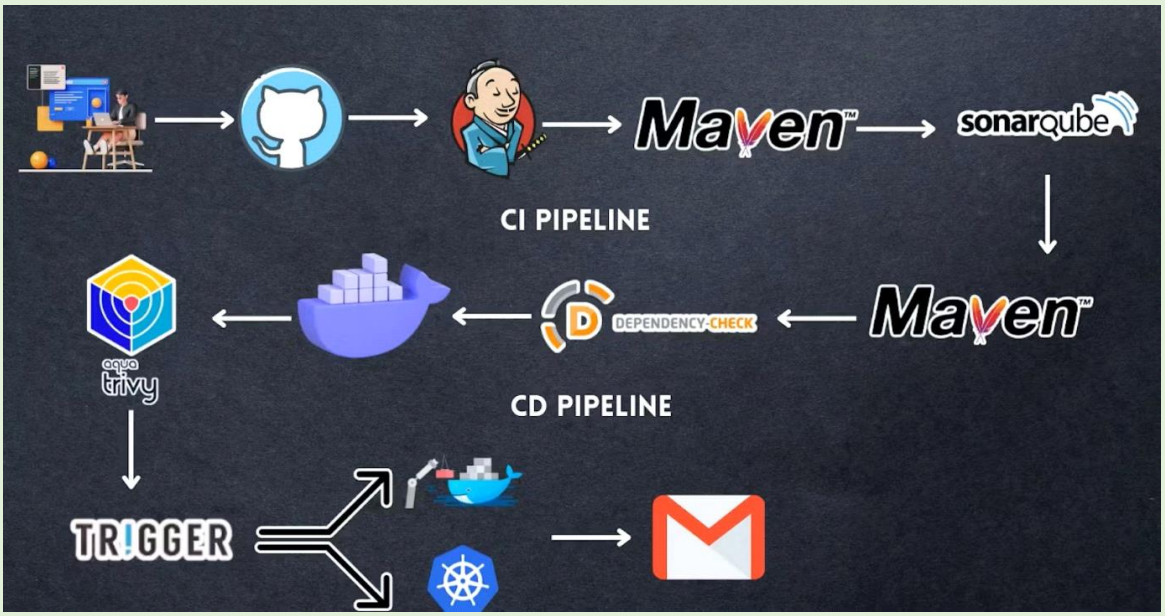
Batch No: 115 (9am-10am)

Mail id:rajavamsyvanguri@gmail.com

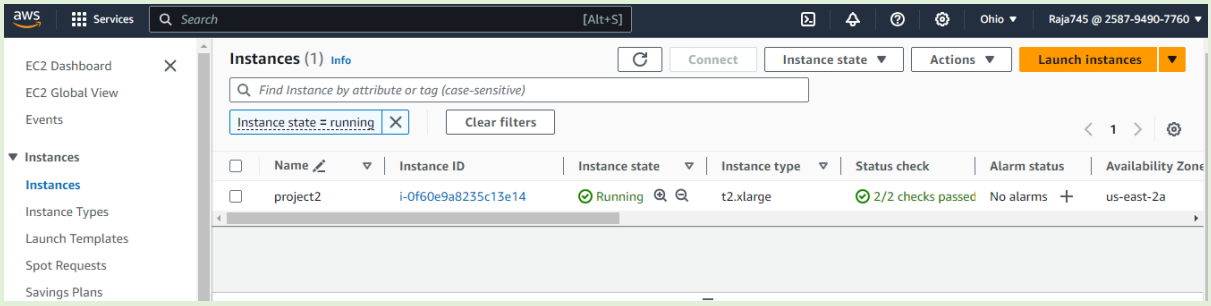
Trainer Name: Mr. Madhukar Reddy

Date: 22-12-23

Topic: DEVSECOPS Project: Complete CI-CD (3 tier app)-Dockerjenkins.

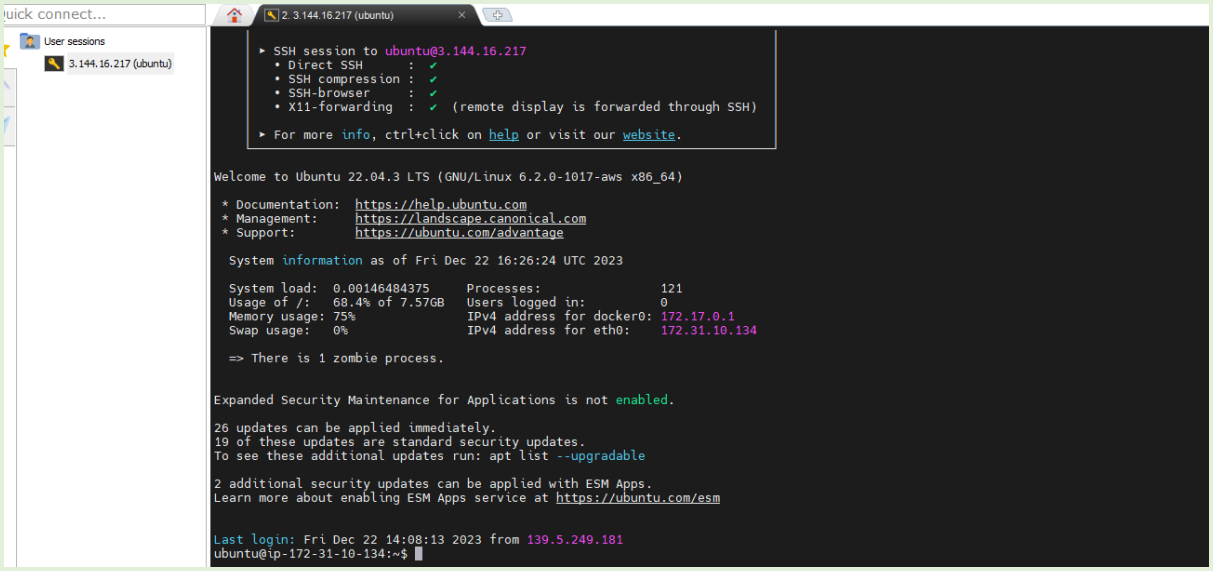


AWS:



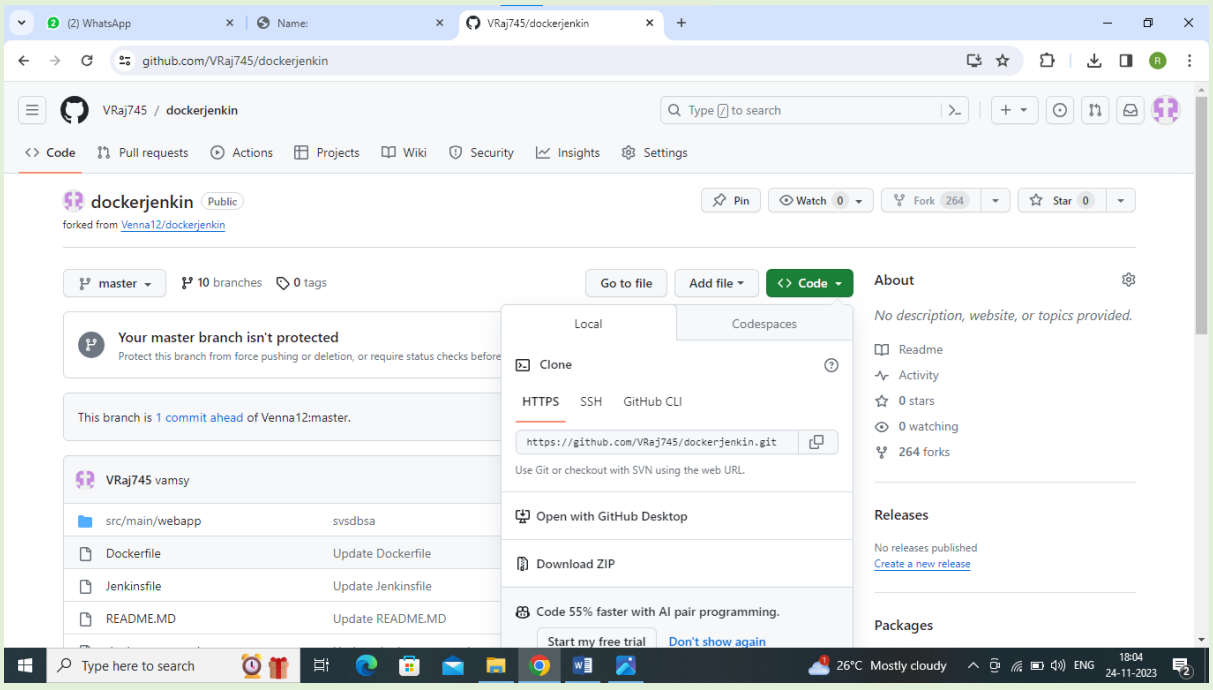
By using the AWS EC2, instance is created and installed the Jenkins, SonarQube, Nexus, and Tomcat and Docker in single server.

SERVER CONECTIONS:



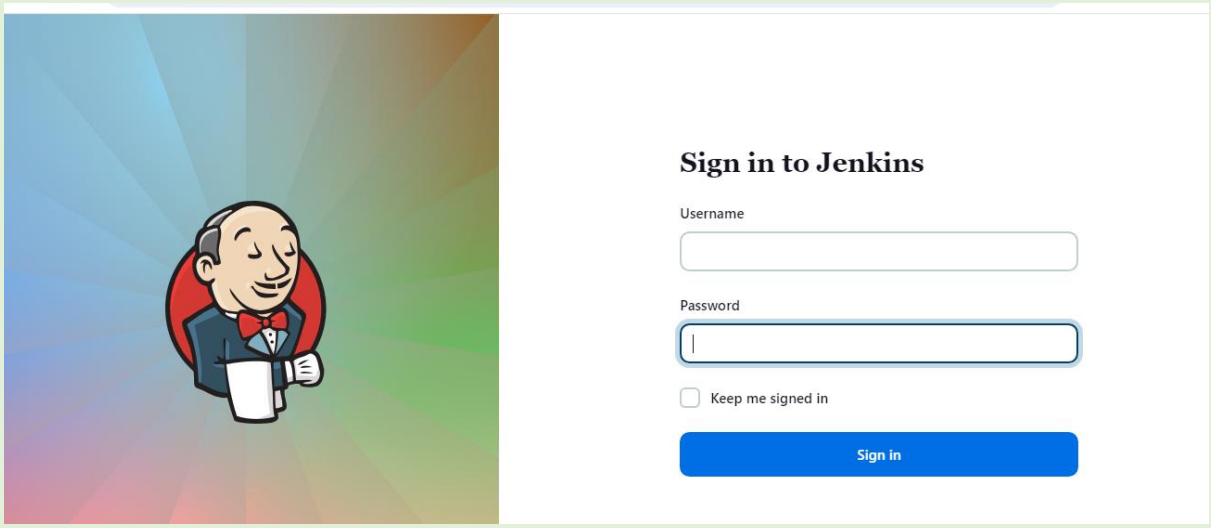
Above fig shows that the connection of instance in mobaxterm, which is used to install the jenkins, tomcat and required packages and more.

GITHUB:

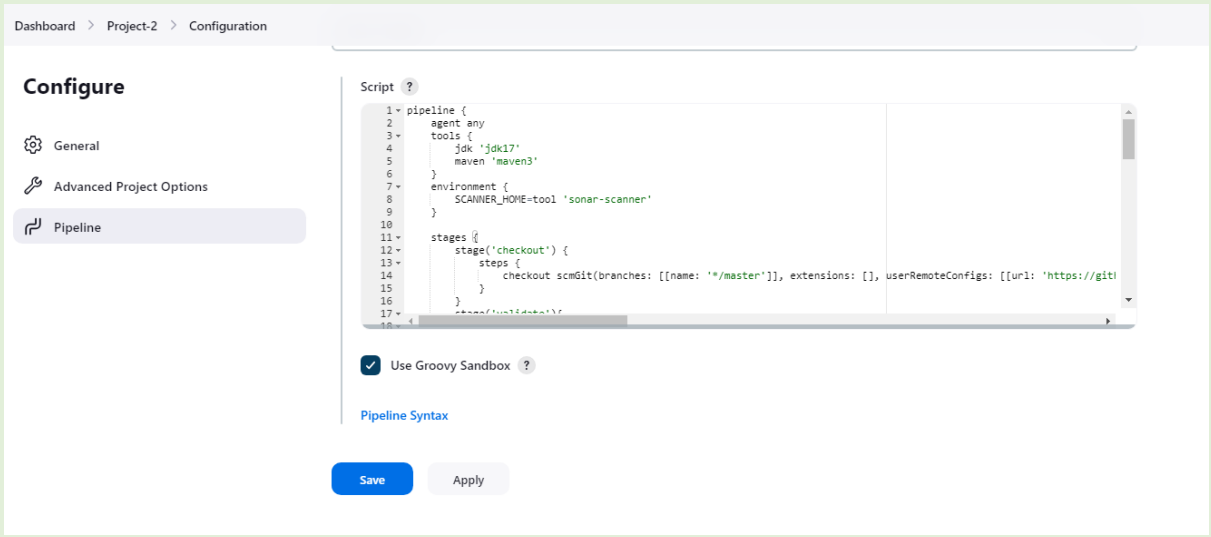


Above fig shows that the git repository that which is used to deploy the webapp in the tomcat server. It is forked by the https://github.com/Venna12/dockerjenkin.git.

JENKINS:



This is the Sign page for Jenkins.



Above shows the pipeline job creation in the jenkins.

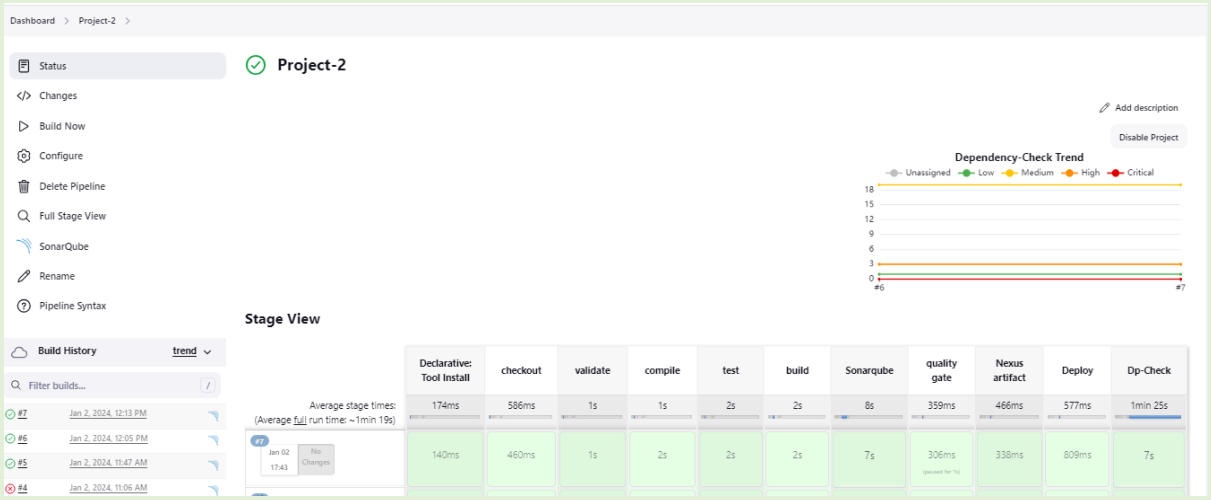
Pipeline code:

```
pipeline {
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
```

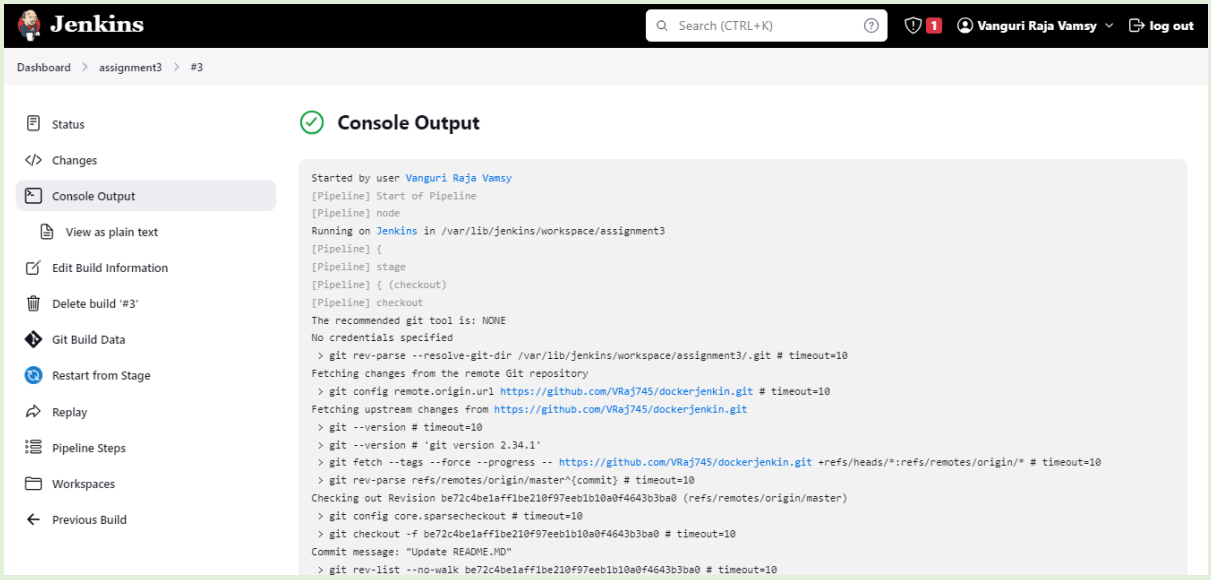
```
        SCANNER_HOME=tool 'sonar-scanner'
    }

    stages {
        stage('checkout') {
            steps {
                checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/VRaj745/dockerjenkin.git']])
            }
        }
        stage('validate'){
            steps{
                sh 'mvn validate '
            }
        }
        stage('compile'){
            steps{
                sh 'mvn compile'
            }
        }
        stage('test'){
            steps{
                sh 'mvn test'
            }
        }
        stage('build'){
            steps{
                sh 'mvn package'
            }
        }
        stage('Sonarqube'){
            steps{
```

```
            withSonarQubeEnv('sonar-server') {
            sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=dockerjenkins \
            -Dsonar.java.binaries=. \
            -Dsonar.projectKey=dockerjenkins '''}
          }
       }
       stage('quality gate'){
          steps {
            script {
              waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
            }
          }
       }
       stage('Nexus artifact'){
          steps{
            nexusArtifactUploader artifacts: [[artifactId: 'java-tomcat-maven-example', classifier: '', file: '/var/lib/jenkins/workspace/Project-2/target/java-tomcat-maven-example.war', type: 'war']], credentialsId: 'nexus123', groupId: 'com.example', nexusUrl: '13.59.143.13:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots', version: '1.0-SNAPSHOT'
          }
       }
       stage('Deploy'){
          steps{
            deploy adapters: [tomcat9(credentialsId: '52c24633-b9e7-4df0-b8aa-0fb09b4aa831', path: '', url: 'http://13.59.143.13:8086/')], contextPath: 'PROJECT-3', war: '**/*.war'
          }
       }
       stage('Dp-Check'){
          steps{
```

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

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

        }

    }

}
```



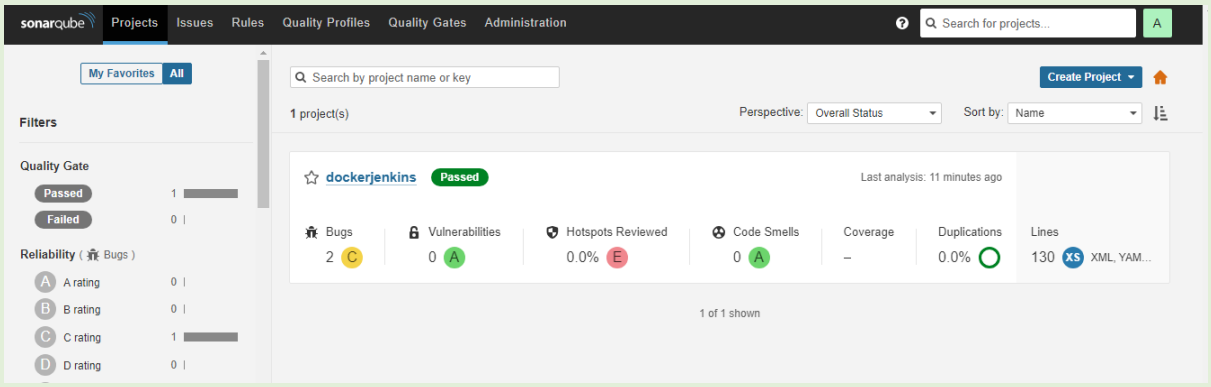
This shows the creation and building the pipeline job and it shows the stage view.

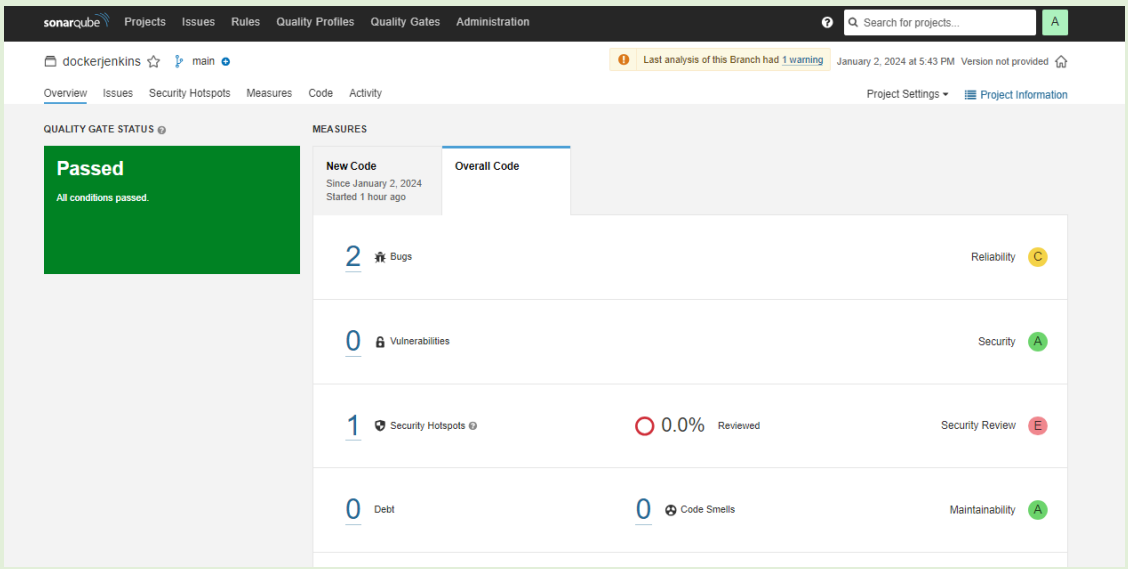


Above figures shoes the **console output** of the pipeline job.

SONARQUBE:

By using the sonarqube we done the code quality analysis. It says us about the details about code which shows the vulnerabilities, bugs etc.

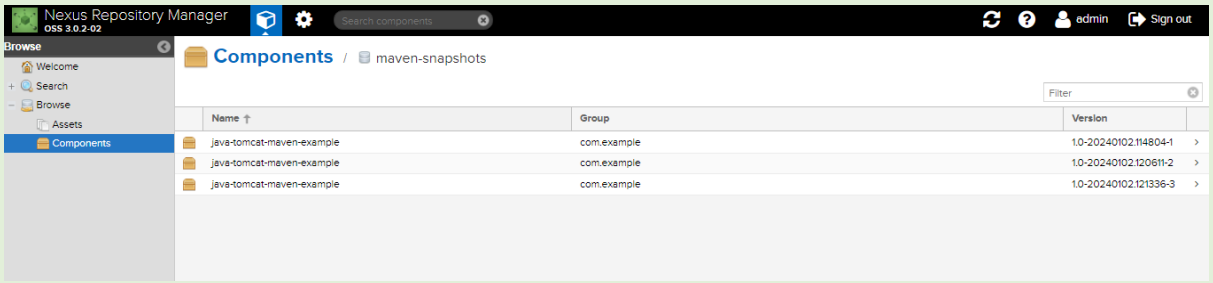


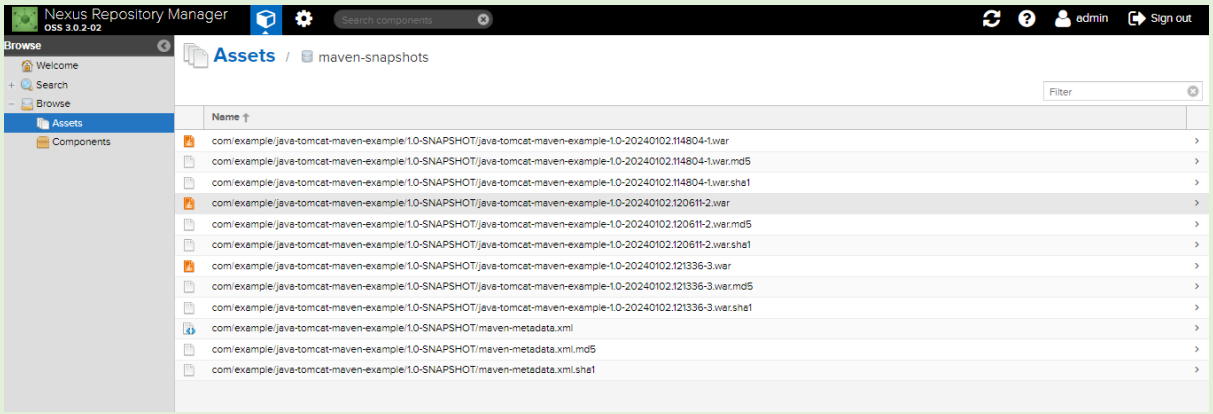


The above fig shows that the Analysis of code.

NEXUS:

Nexus is a Sonatype Artifactory repository manager [OSS]. It allows you to store, distribute, and retrieve build artifacts whenever it's required. Using Nexus, developers can easily access and deploy build artifacts in an organization from a single location.

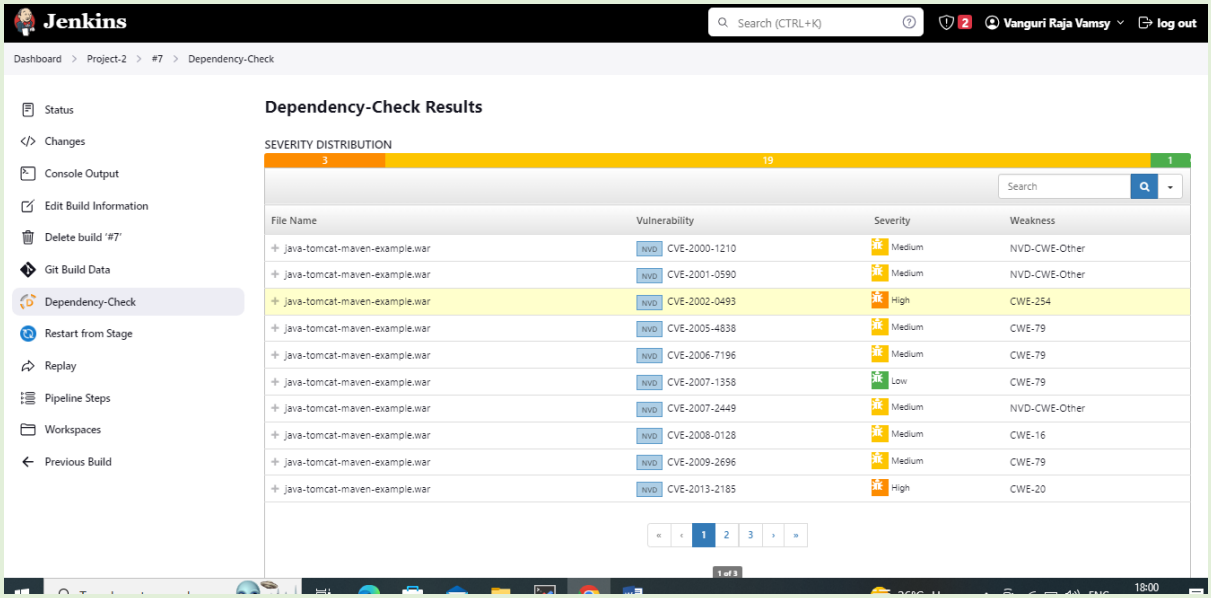




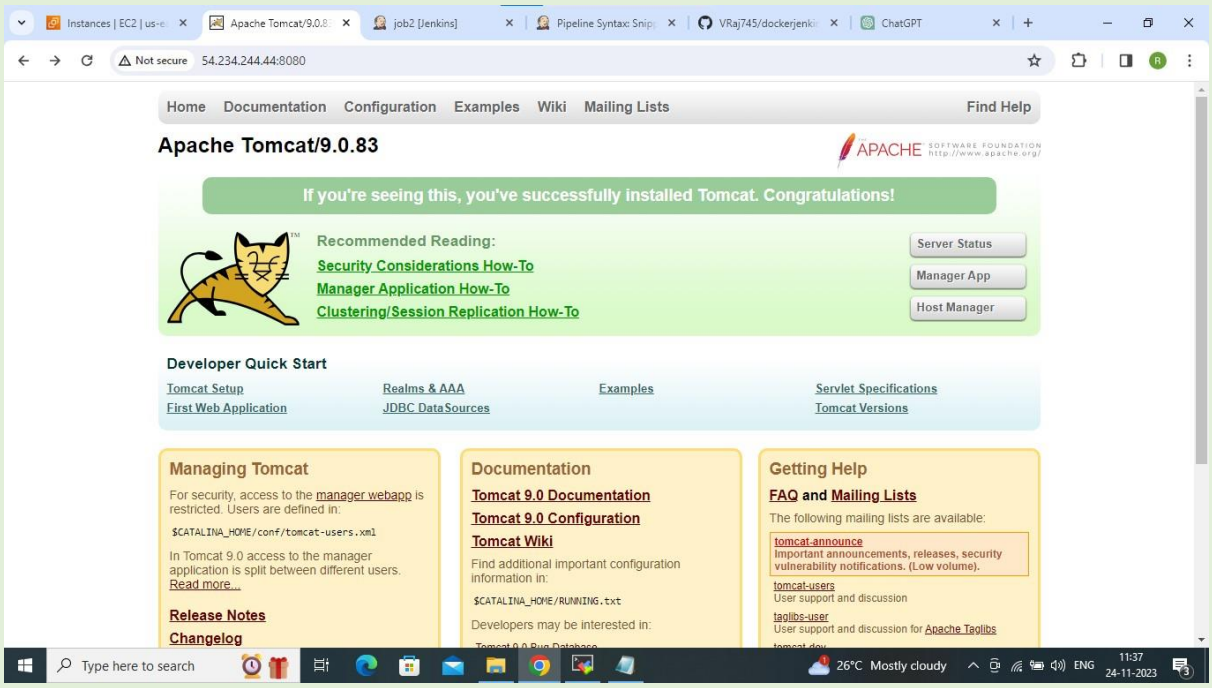
Above fig shows the nexus artifacts.

DP-CHECK:

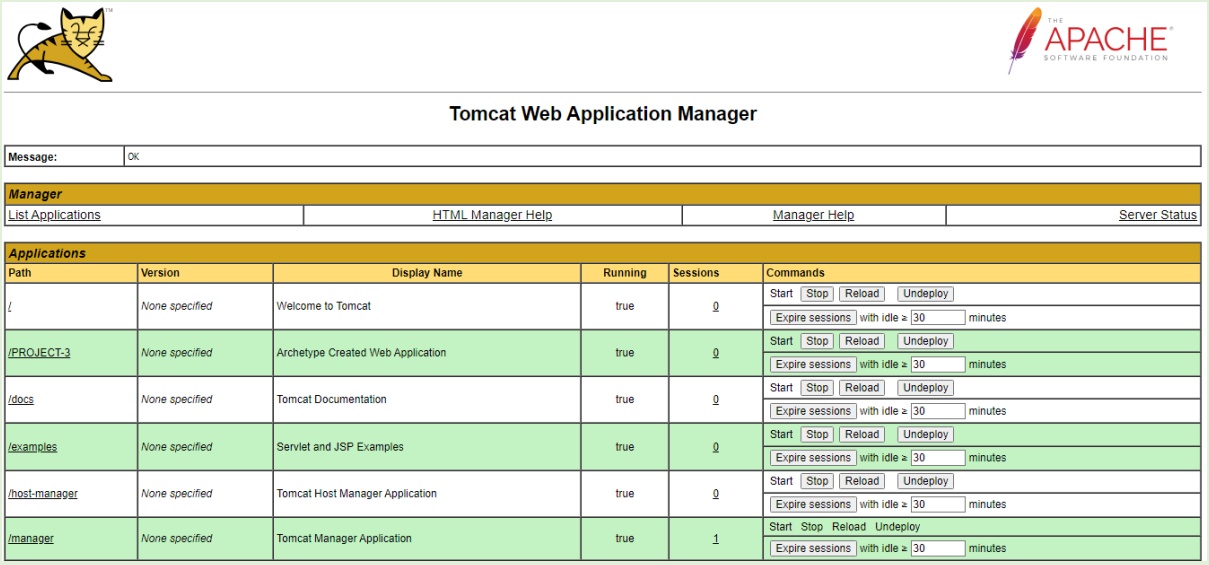
Dependency-check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities. This tool can be part of the solution to the OWASP top 10 2017: a9 - using components with known vulnerabilities.



TOMCAT:



Above fig shows that the open page of the TOMCAT server. By clecking on the manager app button, it takes us into the below view of the web page.



Tomcat Web Application Manager

Message:	OK

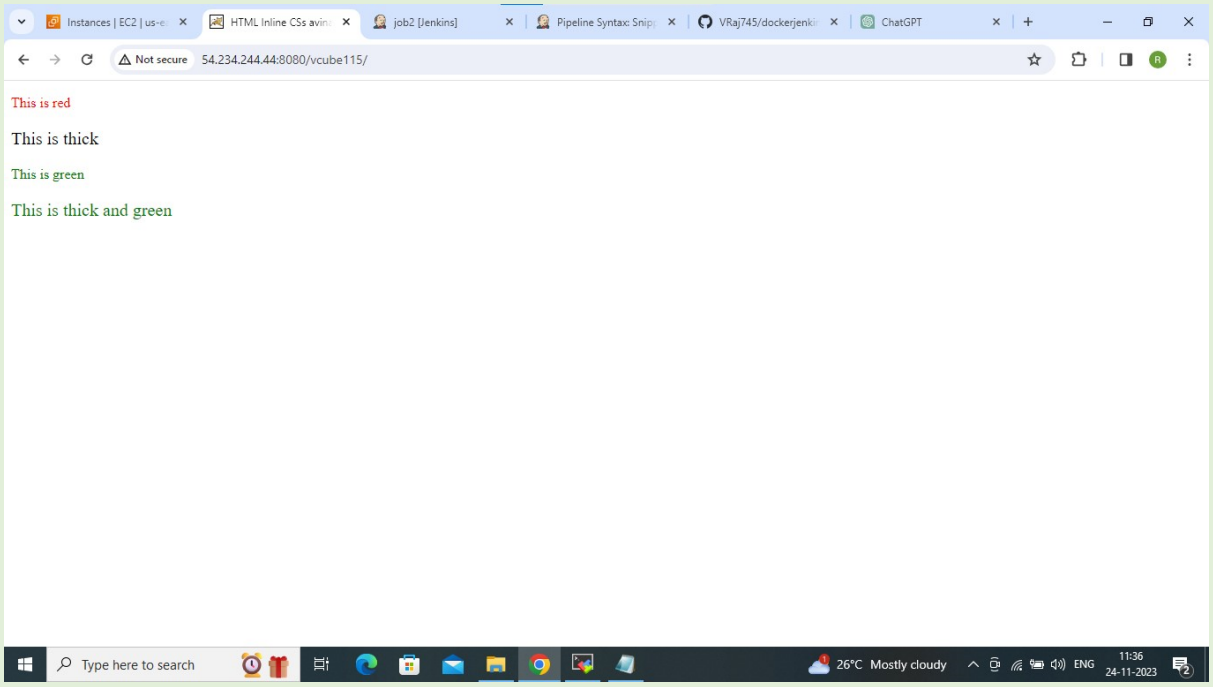
Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/PROJECT-3	None specified	Archetype Created Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

In this page we can see the deployed webapp in the list as ‘assignment’.



This is the Final output of the assignment.