Academic Year: 2024-25 **Name: Mustqeem Masuldar**
Class / Branch: TE IT **Semester: V**
Subject: Advanced Devops Lab (ADL) **Moodle ID: 22104024**
Subject Lab Incharge: Prof. Manjusha K.

EXPERIMENT NO. 08

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perfo analysis of the code to detect bugs, code smells, and security vulnerabilities on application.

Theory:

Integrating Jenkins with SonarQube provides you with an automated platform for performing continuous inspection of code for quality and security assurance.

Everyday enhancement simplifies developers' tasks. Let's assume a situation where Developers have committed their codes to the repository, and then they want to know the project source code quality, code smells, any bugs, vulnerabilities, code analysis, etc. So it is extremely challenging for them to know all this information. So what if they want all these source codes and information beforehand? For such cases, Jenkins is the best fit. If a Software developer starts to build any new project, then the source code is automatically or manually saved while using Jenkins and their daily commit operation is not needed every time.

For this purpose, we can go for CI/CD i.e. Continuous Integration &Continuous Deployment of the code using SonarQube-Jenkins Integration.

SonarQube :

SonarQube is an open-source platform, which is used for continuous analysis of source code quality by performing analysis on your code to detect duplications, bugs, security vulnerabilities and code smells on programming languages.

Jenkins :

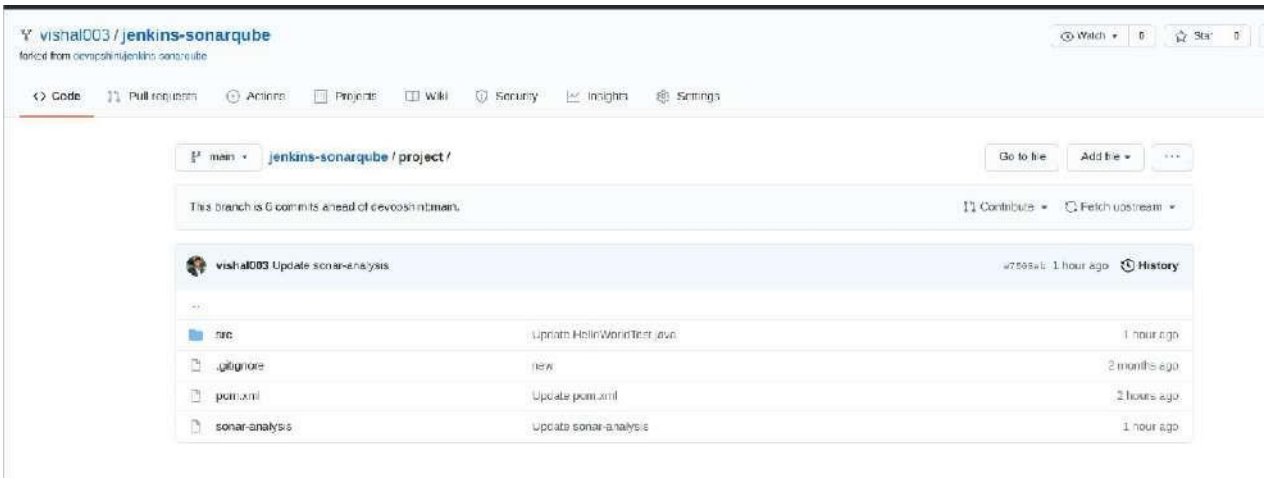
Jenkins an open-source automation tool is created using Java programming language. For the initial setup, it facilitates users with CI/CD(continuous integration (CI) or continuous delivery) technique that simplifies the use and management of processes. It is fundamentally focused on continuously building and testing software projects for developers and to implement changes in real-time. In addition, it also allows users to plan a new build whenever the need arises.

Steps:

1) **Install and configure a Jenkins and SonarQube CICD environment using containers.**
2) **Configure Jenkins with the SonarQube Scanner plugin for automated analysis.**
3) **Create and set up a Jenkins build pipeline using a Jenkinsfile stor GitHub repo.**
4) **Use the SonarQube web application to examine and review the ge analysis report.**
5) **Use the Blue Ocean Plugin to review Pipeline Steps.**

Note: From Step 1 and 2 we have already done in Expt. 7 as a Pre-requiste required for Integration settings of Jenkins SAST with SonarQube so in this Experiment we will continue from 3rd Step.

Check the contents of jenkins-sonarqube repository which we are using for Pipeline Project.

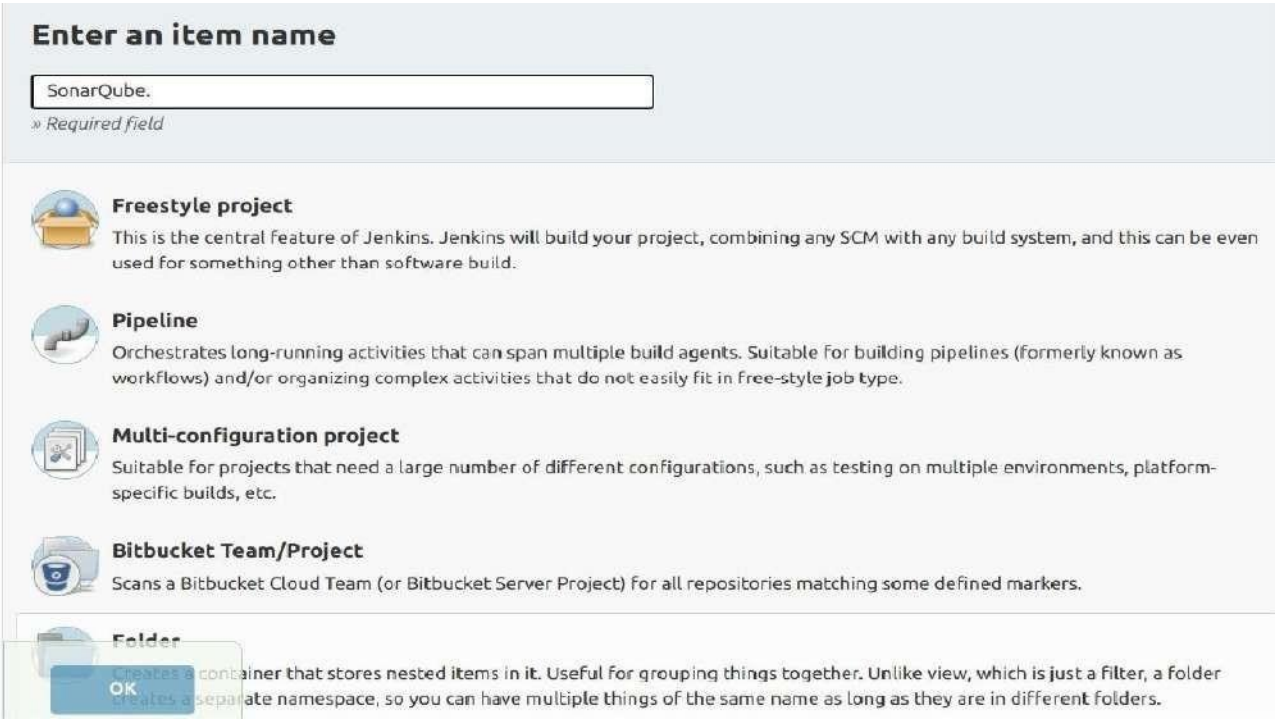


Check path for the Source and Test Java Programs from repository

Provide sonar host as http://127.0.0.1:9000 in POM.xml which is available in Github.

```
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
        <!-- Optional URL to server. Default value is http://localhost:9000 -->
        <sonar.host.url>
            http://127.0.0.1:9000/
        </sonar.host.url>
    </properties>
</profile>
<profile>
    <id>coverage</id>
```

To integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called SonarQube and inside that, I am adding the following settings and code.



Github Repository Configuration in Jenkins Pipeline Project

General | Build Triggers | Advanced Project Options | Pipeline

Description

Hello Pipeline job

[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

Project url

https://github.com/vishal003/jenkins-sonarqube/

Pipeline Script where stages are written along with scanner tool, repository pat and test Java sample program SonarQube Credential for integration, Applicatio sonarqube, code language,etc.

Pipeline

Definition

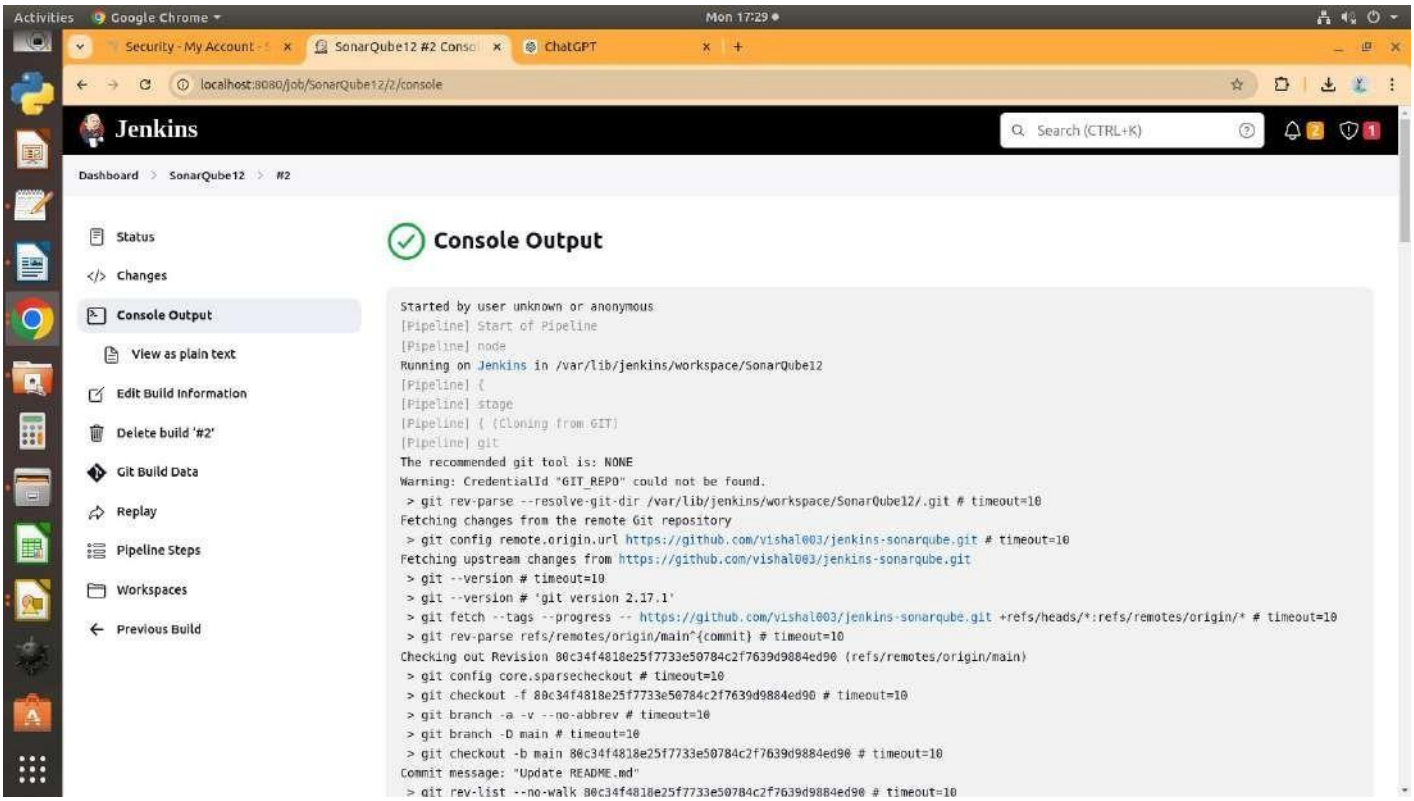
Pipeline script

Script

```
node
{
    stage('clonning from GIT'){
git branch: 'main', credentialsId: 'GIT_REPO', url: 'https://github.com/vishal003/jenkins-sonarqube.git'
    }

stage('SonarQube Analysis') {
    def scannerHome = tool 'SonarQube'
      withSonarQubeEnv('SonarQube') {
      sh """/var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/SonarQube/bin/sonar-scanner \
     -D sonar.projectVersion=1.0-SNAPSHOT \
       -D sonar.login=admin \
      -D sonar.password=India@11 \
      -D sonar.projectBaseDir=/var/lib/jenkins/workspace/sonarqube \
        -D sonar.projectKey=my-app1 \
        -D sonar.sourceEncoding=UTF-8 \
        -D sonar.language=java \
        -D sonar.sources=project/src/main/java \
        -D sonar.tests=project/src/test/java \
        -D sonar.host.url=http://127.0.0.1:9000/"""
        }
```

After creating a Pipeline Script Build it in Jenkins , Click on save and then Click o Now

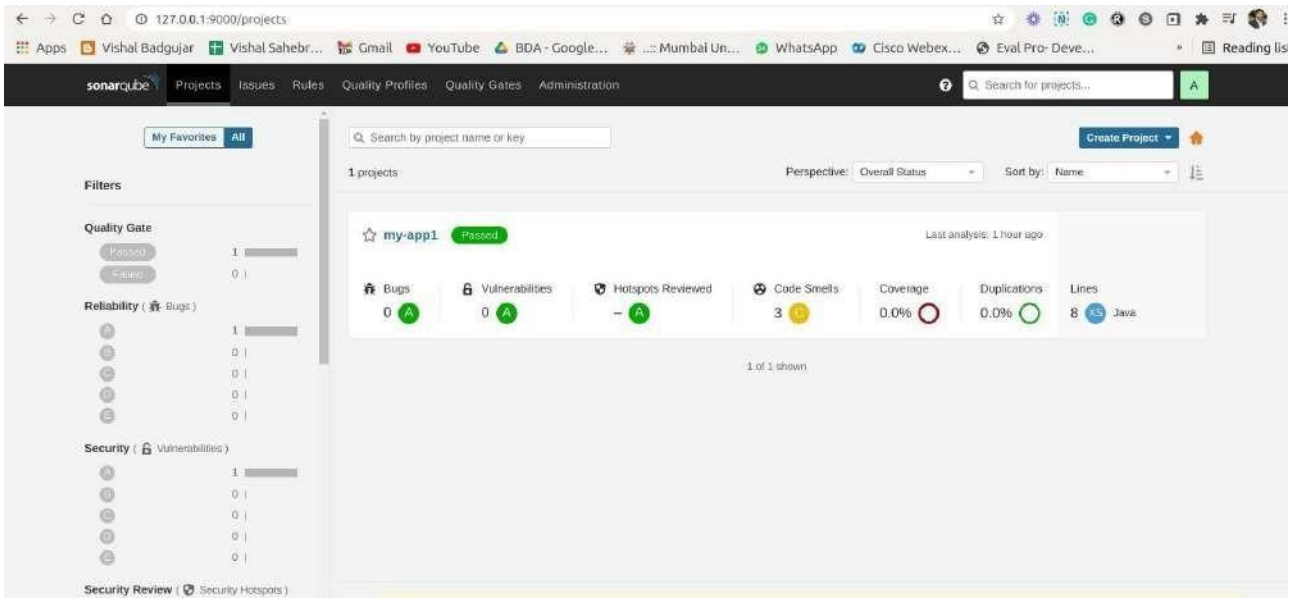


Click on Console Output to check output whether build is successful or not.

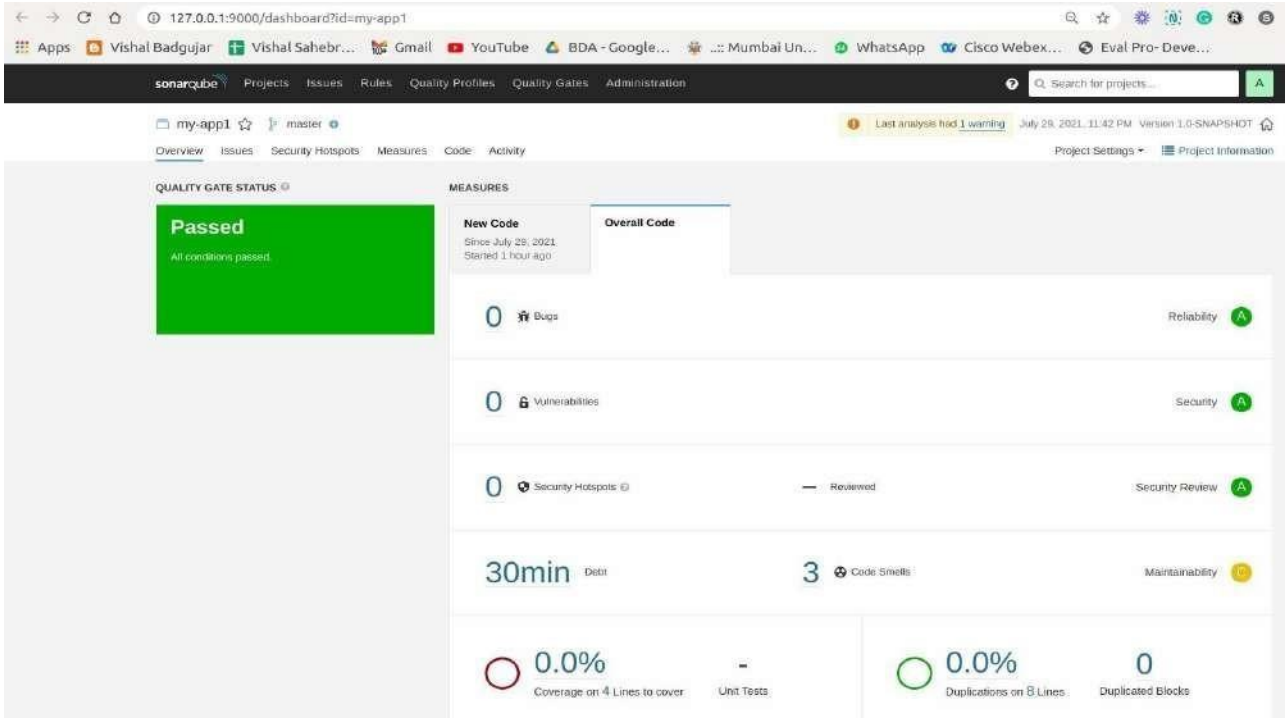
Click on Pipeline Steps to check Sequence of events during building of pipeline

Also you can use Blue Ocean to check Pipeline execution stage by stage and log too.

If you login to the SonarQube and visit the Dashboard, you will see the Analysis there.



For Detailed Report for code analysis you can go to application overview and ch Bugs, Vulnerabilities, code sm and all parameters as shown in below image.



Compiled By: Prof.Manjusha K. **Information Technology Department** Since we have both Jenkins and SonarQube in the Enterprise standard, we have a lot of features including the alert system. Where we can configure the Email, or Instance message Notification system for the findings in the SonarQube or Jenkins. In the best case, we can auto convert certain bugs or findings as ticket and assign to the respective developer as a one option.

Conclusion: Thus we executed po.xml file using jenkins.