



Academic Year: 2024-25

Class / Branch: TE IT

Subject: Advanced Devops Lab (ADL)

Subject Lab Incharge: Prof. Manjusha K.

Name: Mustqeen Masuldar

Semester: V

Moodle ID: 22104024

EXPERIMENT NO. 08

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform 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 CI/CD 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-requisite 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.


```
<!-- Optional URL to server. Default value is http://localhost:9000 -->
<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>
  <activation>
```

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.




Enter an item name


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Bitbucket Team/Project**

Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

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

☒ **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 ?

```
1 node
2 {
3     stage('clonning from GIT'){
4         git branch: 'main', credentialsId: 'GIT_REPO', url: 'https://github.com/vishal003/jenkins-sonarqube.git'
5     }
6
7     stage('SonarQube Analysis') {
8         def scannerHome = tool 'SonarQube'
9         withSonarQubeEnv('SonarQube') {
10             sh """/var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/SonarQube/bin/sonar-scanner \
11 -D sonar.projectVersion=1.0-SNAPSHOT \
12 -D sonar.login=admin \
13 -D sonar.password=India@11 \
14 -D sonar.projectBaseDir=/var/lib/jenkins/workspace/sonarqube \
15 -D sonar.projectKey=my-app1 \
16 -D sonar.sourceEncoding=UTF-8 \
17 -D sonar.language=java \
18 -D sonar.sources=project/src/main/java \
19 -D sonar.tests=project/src/test/java \
20 -D sonar.host.url=http://127.0.0.1:9000/"""
21         }
22     }
23 }
```

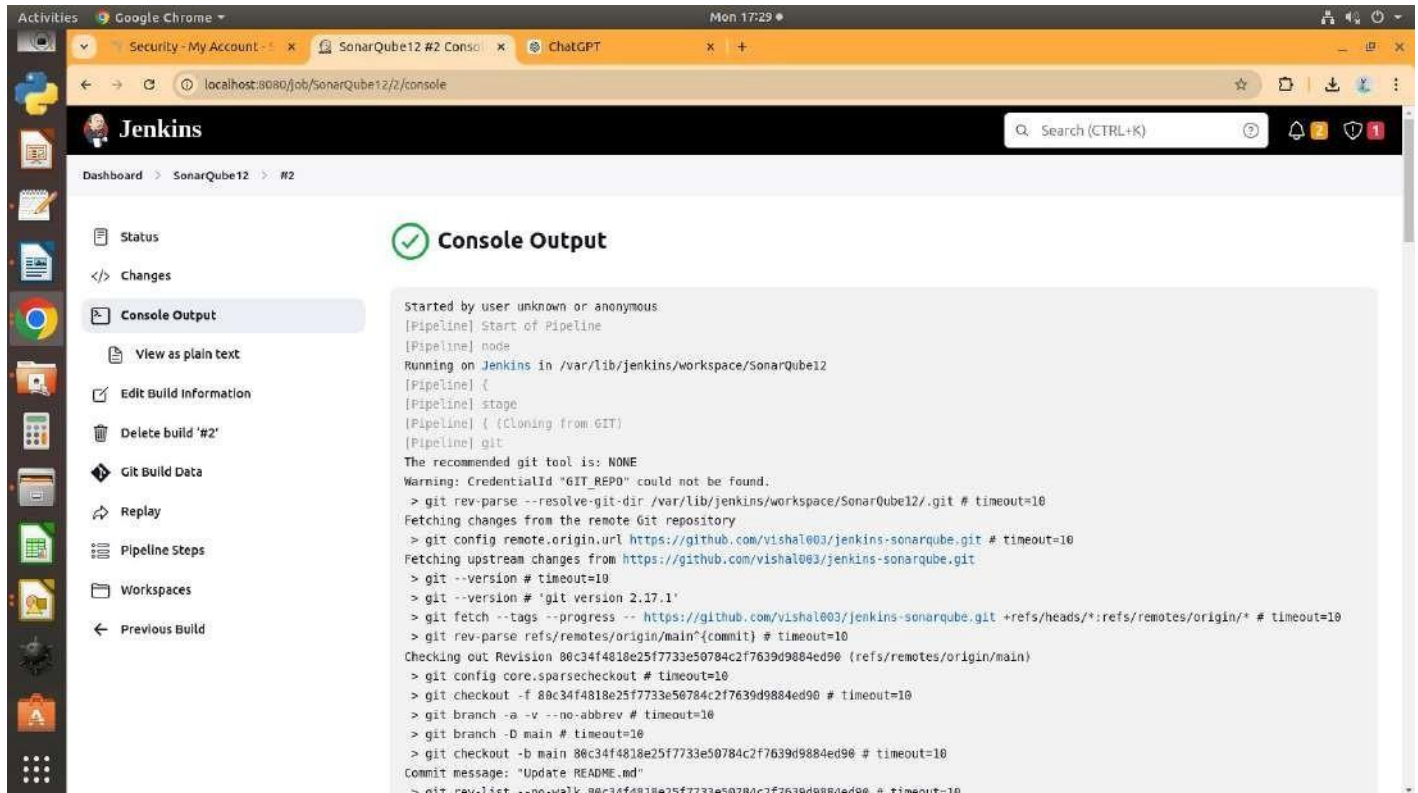


PARSHVANATH CHARITABLE TRUST'S A. P. SHAH INSTITUTE OF TECHNOLOGY

(All Branches NBA Accredited)



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



```
Started by user unknown or anonymous.
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/SonarQube12
[Pipeline] {
[Pipeline] stage
[Pipeline] (Cloning from GIT)
[Pipeline] git
The recommended git tool is: NONE
Warning: CredentialId "GIT_REPO" could not be found.
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/SonarQube12/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/vishal003/jenkins-sonarqube.git # timeout=10
Fetching upstream changes from https://github.com/vishal003/jenkins-sonarqube.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
> git fetch --tags --progress -- https://github.com/vishal003/jenkins-sonarqube.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 80c34f4818e25f7733e50784c2f7639d9884ed90 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 80c34f4818e25f7733e50784c2f7639d9884ed90 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main 80c34f4818e25f7733e50784c2f7639d9884ed90 # timeout=10
Commit message: "Update README.md"
> git rev-list --no-walk 80c34f4818e25f7733e50784c2f7639d9884ed90 # timeout=10
```

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



Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE
(All Programs Accredited by NBA)
Department of Information Technology



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

Dashboard	sonarqube	#7	Pipeline Steps
Back to Project			
Status			
Changes			
Console Output			
Edit Build Information			
Delete build '#7'			
Git Build Data			
Open Blue Ocean			
Replay			
Pipeline Steps			
Workspaces			
Previous Build			

Step	Arguments	Status
Start of Pipeline - (11 sec in block)		✓
Allocate node: Start - (11 sec in block)		✓
Allocate node: Body: Start - (11 sec in block)		✓
Stage: Start - (1.1 sec in block)	cloning from GIT	✓
cloning from GIT - (1 sec in block)		✓
Git - (1 sec in self)		✓
Stage: Start - (10 sec in block)	SonarQube Analysis	✓
SonarQube Analysis - (10 sec in block)		✓
Use a tool from a predefined Tool Installation - (0.18 sec in self)	SonarQube	✓
Prepare SonarQube Scanner environment: Start - (10 sec in block)	SonarQube	✓
General Build Wrapper: Body: Start - (9.9 sec in block)		✓
Shell Script - (9.9 sec in self)	<pre>/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=src/main/java -D sonar.test.sqproject=/var/lib/jenkins -D sonar.host.url=http://127.0.0.1:9000/</pre>	✓

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

127.0.0.1:8080/blue/organizations/jenkins/sonarqube/detail/sonarqube/7/pipeline/

✓ sonarqube < 7

Branch: — 12s No changes

Commit: — 32 minutes ago Replayed #6

Pipeline Changes Tests Artifacts

Start cloning from GIT SonarQube Analysis End

SonarQube Analysis - 10s

- ✓ > SonarQube — Use a tool from a predefined Tool Installation <1s
- ✓ > /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 ... — Shell Script 10s



Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE
(All Programs Accredited by NBA)
Department of Information Technology



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

The screenshot displays the SonarQube dashboard interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. A search bar for projects is located on the right. The main content area shows a list of projects, with 'my-app1' selected and its analysis details displayed. The analysis status is 'Passed' (green). The dashboard provides a summary of various metrics:

Metric	Value	Status
Bugs	0	Passed (Green)
Vulnerabilities	0	Passed (Green)
Hotspots Reviewed	1	Passed (Green)
Code Smells	3	Warning (Yellow)
Coverage	0.0%	Failed (Red)
Duplications	0.0%	Passed (Green)
Lines	8	Java

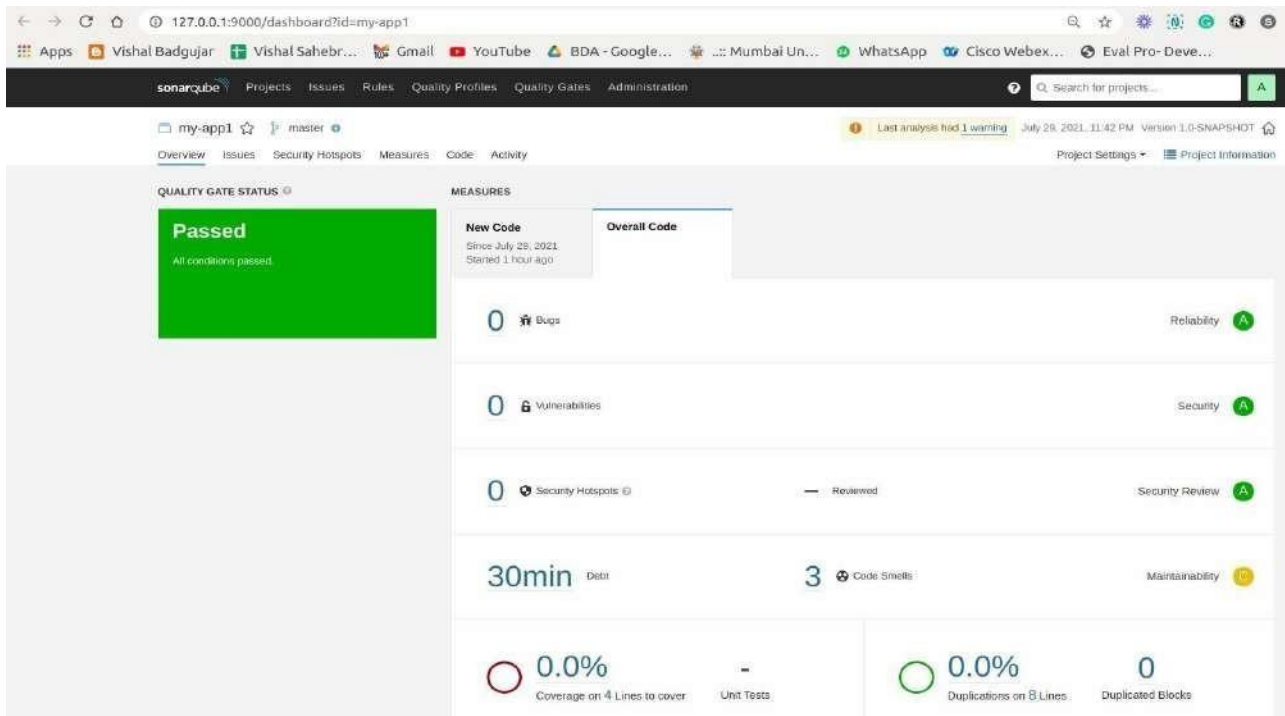
The left sidebar contains filters for Quality Gate, Reliability (Bugs), and Security (Vulnerabilities), each with a progress bar and a list of items. The bottom of the dashboard shows a 'Security Review' section with 'Security Hotspots'.



Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE
(All Programs Accredited by NBA)
Department of Information Technology



For Detailed Report for code analysis you can go to application overview and ch Bugs, Vulnerabilities, code smells 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.



Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE
(All Programs Accredited by NBA)
Department of Information Technology



Compiled By: Prof.Manjusha K.

Information Technology Department