Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

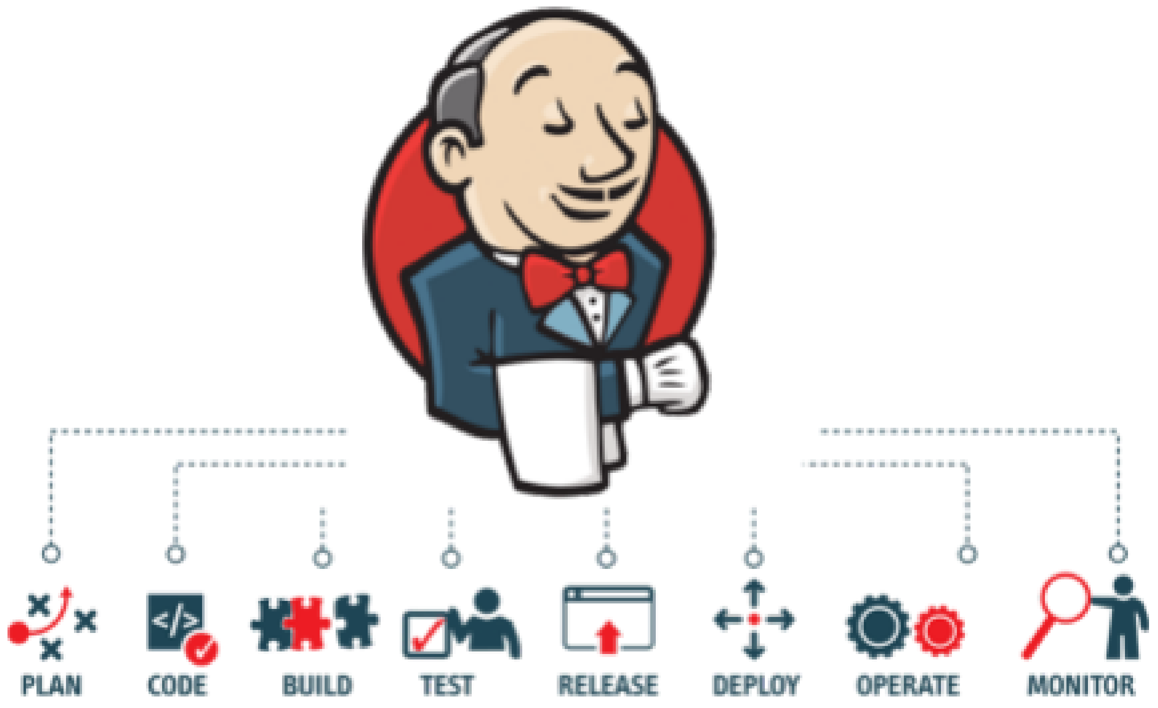
Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

What is a CI/CD Pipeline?

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.



Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code

smells, vulnerabilities, code duplications.
It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

● **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.

● **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

● **Quality code -** Code quality control is an inseparable part of the process of software development.

● **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.

● **Increase consistency -** Determines where the code criteria are breached and enhances the quality

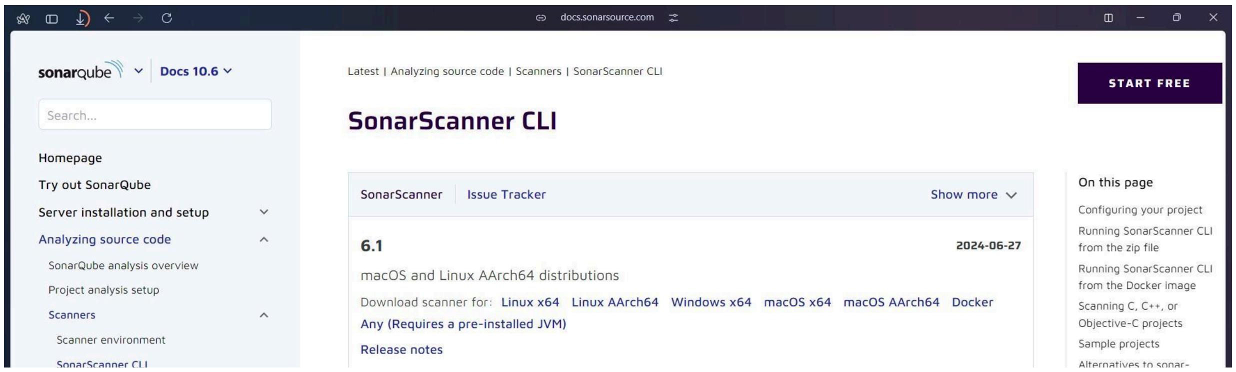
● **Business scaling -** No restriction on the number of projects to be evaluated

● **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

Step 1: Download sonar scanner

https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/ Visit this link and download the sonarqube scanner CLI.



Latest | Analyzing source code | Scanners | SonarScanner CLI

SonarScanner CLI

6.1 2024-06-27

macOS and Linux AArch64 distributions

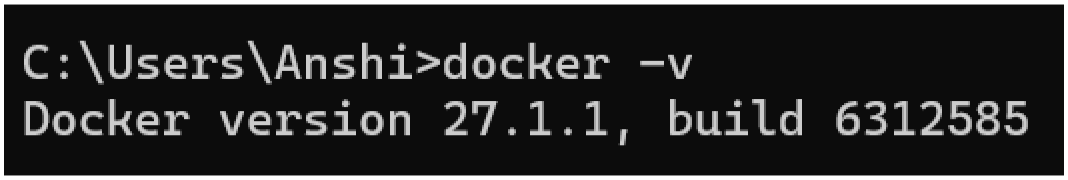
Download scanner for: Linux x64 Linux AArch64 Windows x64 macOS x64 macOS AArch64 Docker
Any (Requires a pre-installed JVM)

Release notes

Extract the downloaded zip file in a folder.

1) Docker

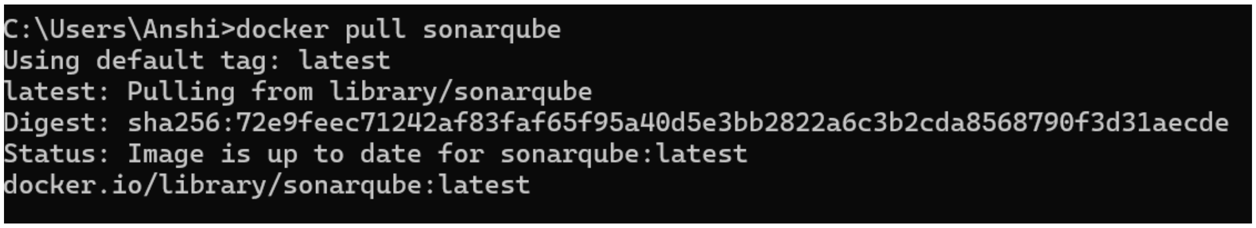
Run docker -v command.



```
C:\Users\Anshi>docker -v
Docker version 27.1.1, build 6312585
```

2) Install sonarqube image

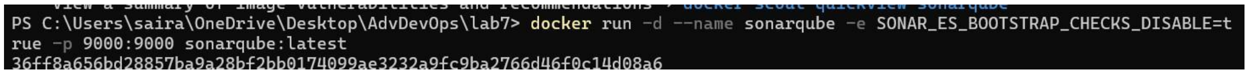
Command: docker pull sonarqube



```
C:\Users\Anshi>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Image is up to date for sonarqube:latest
docker.io/library/sonarqube:latest
```

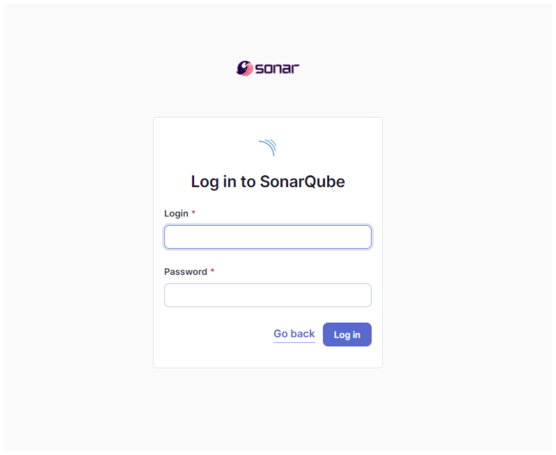
3) Run sonarqube image

docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

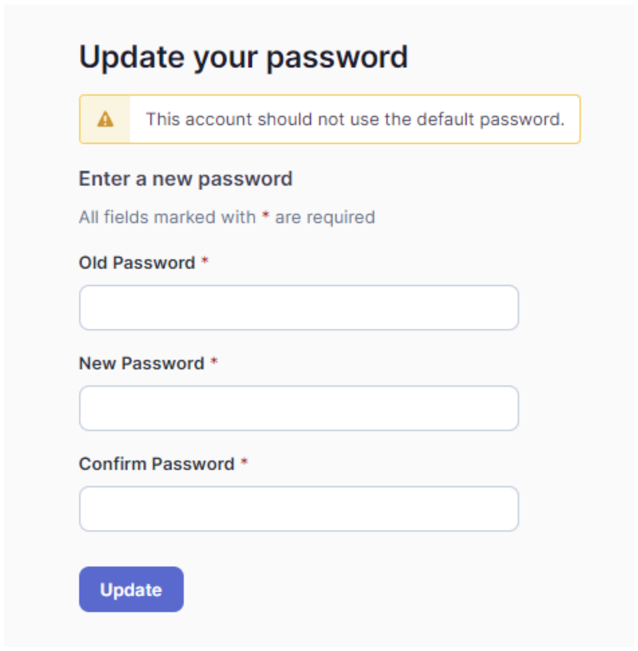


```
   View a summary of image vulnerabilities and recommendations → docker scout quickview sonarqube
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab7> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=t
rue -p 9000:9000 sonarqube:latest
36ff8a656bd28857ba9a28bf2bb0174099ae3232a9fc9ba2766d46f0c14d08a6
```

4) Run localhost:9000



5) Login using username=”admin”, password=”admin”. It will promt you to set a new password.



Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

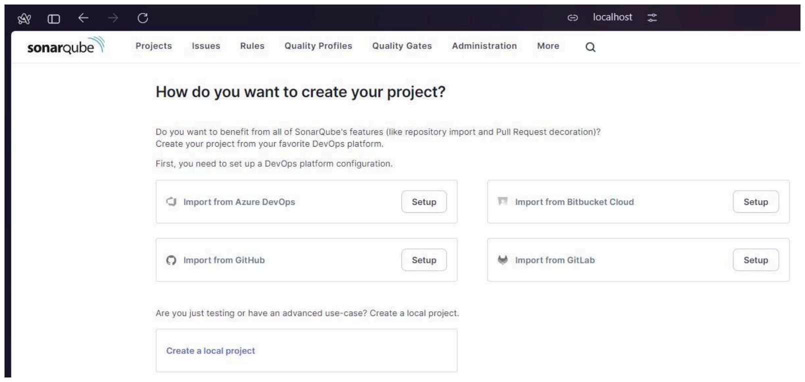
Old Password *

New Password *

Confirm Password *

Update

6) This is the interface. Create a local project with the name sonarqube.



How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)?
Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

Import from Azure DevOps Setup

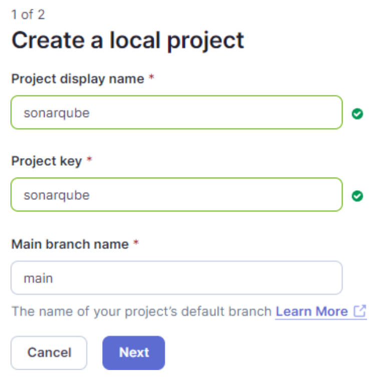
Import from Bitbucket Cloud Setup

Import from GitHub Setup

Import from GitLab Setup

Are you just testing or have an advanced use-case? Create a local project.

Create a local project



1 of 2

Create a local project

Project display name *

sonarqube

Project key *

sonarqube

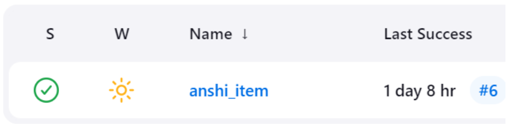
Main branch name *

main

The name of your project's default branch Learn More

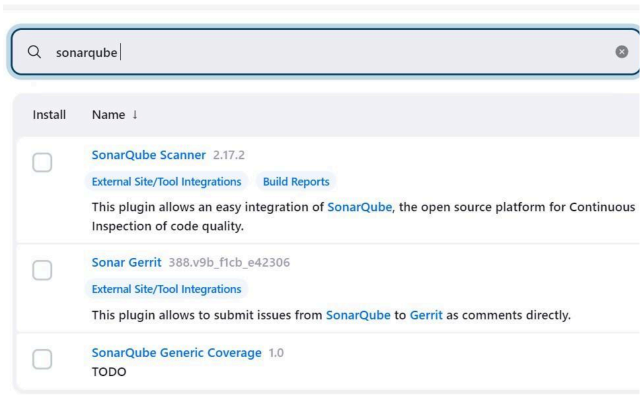
Cancel Next

7) Open jenkins dashboard using localhost on whichever port it is hosted.



S	W	Name ↓	Last Success
✓	☀	anshi_item	1 day 8 hr #6

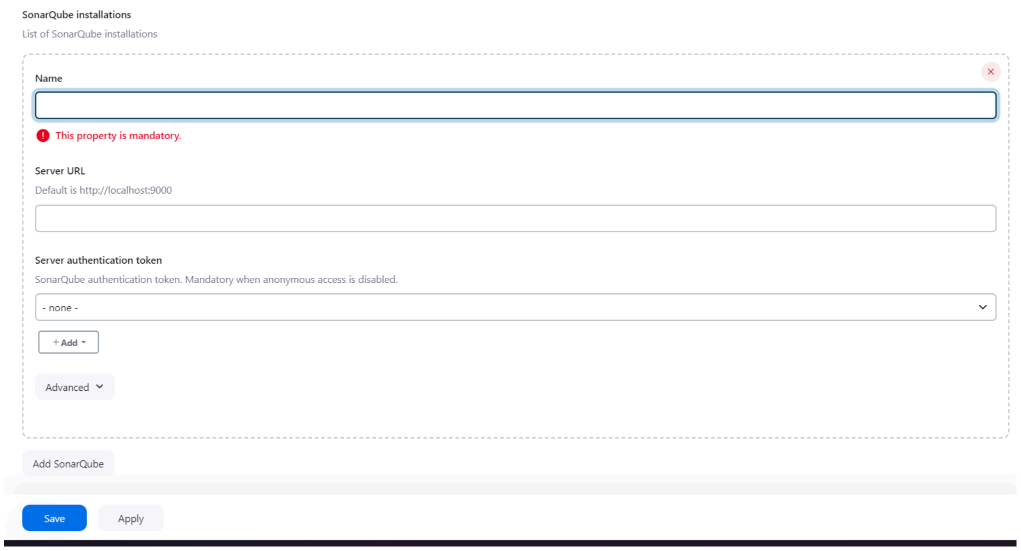
8) Go to manage jenkins → Search for Sonarqube Scanner for Jenkins and install it.



sonarqube

Install	Name ↓
☐	SonarQube Scanner 2.17.2 External Site/Tool Integrations · Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.
☐	Sonar Gerrit 388.v9b_f1cb_e42306 External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly.
☐	SonarQube Generic Coverage 1.0 TODO

9) Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.



SonarQube installations

List of SonarQube installations

Name

This property is mandatory.

Server URL

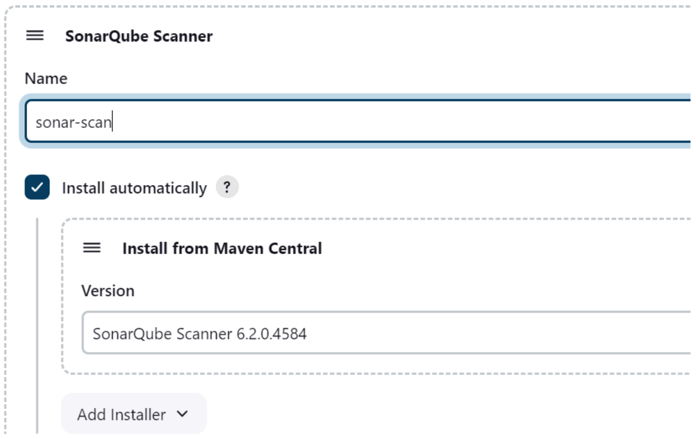
Default is http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

10) Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose install automatically.


SonarQube Scanner

Name

sonar-scan

✓ Install automatically

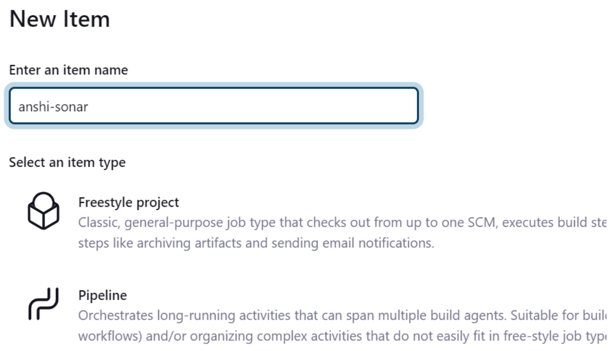
Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer

11) After configuration, create a New Item → choose a pipeline project.


New Item

Enter an item name

anshi-sonar

Select an item type

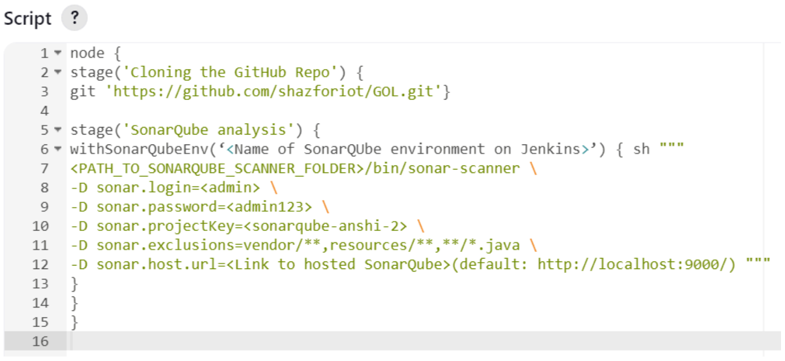
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build ste
steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for buil
workflows) and/or organizing complex activities that do not easily fit in free-style job typ

12) Under Pipeline script, enter the following:

```
node {
stage('Cloning the GitHub Repo') {
git 'https://github.com/shazforiot/GOL.git'}

stage('SonarQube analysis') {
withSonarQubeEnv('<Name of SonarQUbe environment on Jenkins>') { sh """
<PATH_TO_SONARQUBE_SCANNER_FOLDER>/bin/sonar-scanner \
-D sonar.login=<SonarQube_USERNAME> \
-D sonar.password=<SonarQube_PASSWORD> \
-D sonar.projectKey=<Project_KEY> \
-D sonar.exclusions=vendor/**,resources/**,**/*.java \
-D sonar.host.url=<Link to hosted SonarQube>(default: http://localhost:9000/) """
}
}
}
```

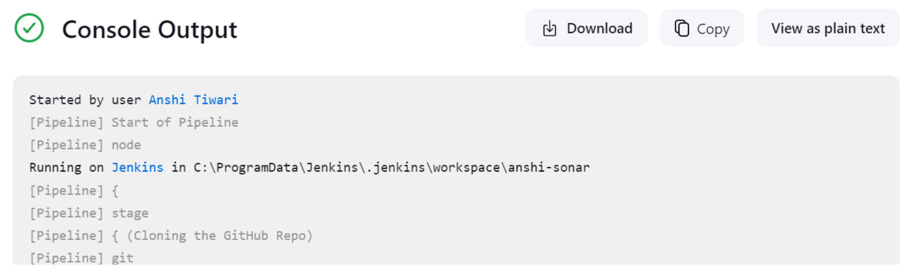
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.



Script ?

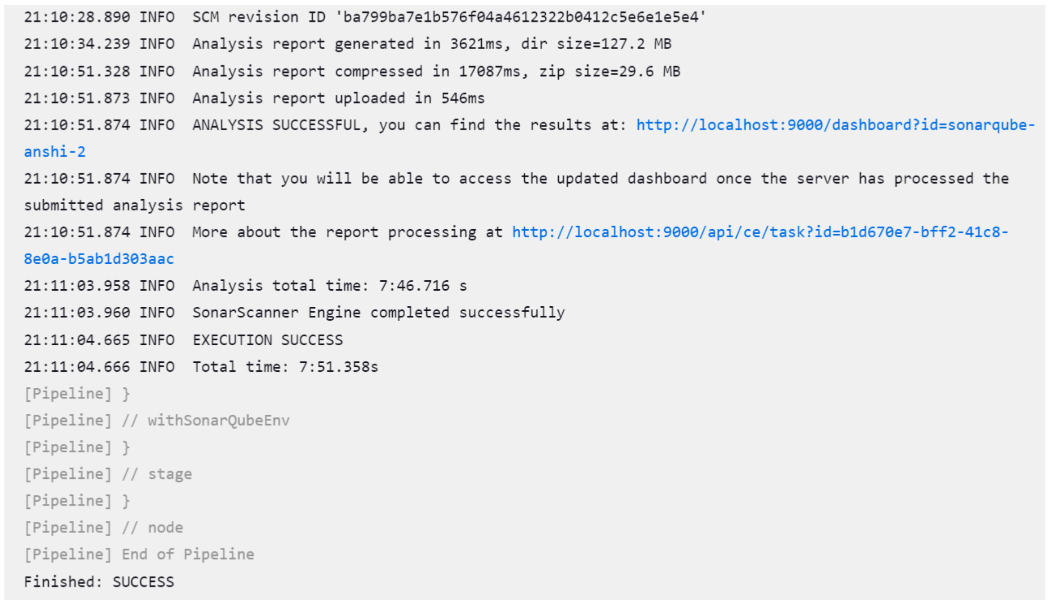
```
node {
stage('Cloning the GitHub Repo') {
git 'https://github.com/shazforiot/GOL.git'}

stage('SonarQube analysis') {
withSonarQubeEnv('<Name of SonarQUbe environment on Jenkins>') { sh """
<PATH_TO_SONARQUBE_SCANNER_FOLDER>/bin/sonar-scanner \
-D sonar.login=<admin> \
-D sonar.password=<admin123> \
-D sonar.projectKey=<sonarqube-anshi-2> \
-D sonar.exclusions=vendor/**,resources/**,**/*.java \
-D sonar.host.url=<Link to hosted SonarQube>(default: http://localhost:9000/) """
}
}
}
```

13) Check the console output once



Console Output

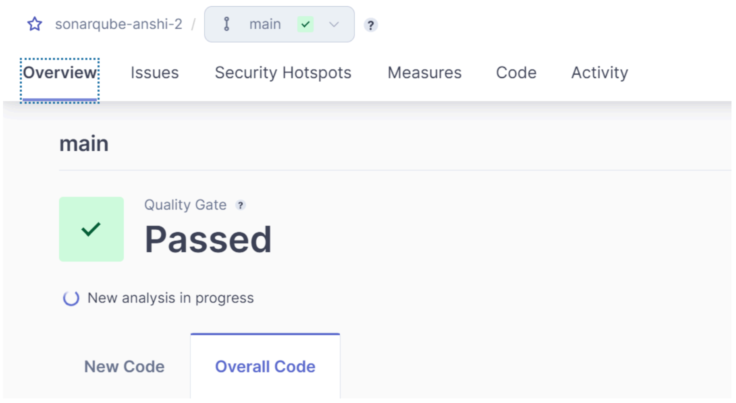
```
Started by user Anshi Tiwari
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\anshi-sonar
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
```



```
21:10:28.890 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
21:10:34.239 INFO  Analysis report generated in 3621ms, dir size=127.2 MB
21:10:51.328 INFO  Analysis report compressed in 17087ms, zip size=29.6 MB
21:10:51.873 INFO  Analysis report uploaded in 546ms
21:10:51.874 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-
anshi-2
21:10:51.874 INFO  Note that you will be able to access the updated dashboard once the server has processed the
submitted analysis report
21:10:51.874 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=b1d670e7-bff2-41c8-
8e0a-b5ab1d303aac
21:11:03.958 INFO  Analysis total time: 7:46.716 s
21:11:03.960 INFO  SonarScanner Engine completed successfully
21:11:04.665 INFO  EXECUTION SUCCESS
21:11:04.666 INFO  Total time: 7:51.358s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

14) Now, check the project in SonarQube

Under different tabs, check all the issues with the code.


sonarqube-anshi-2 / main

Overview Issues Security Hotspots Measures Code Activity

main

Quality Gate

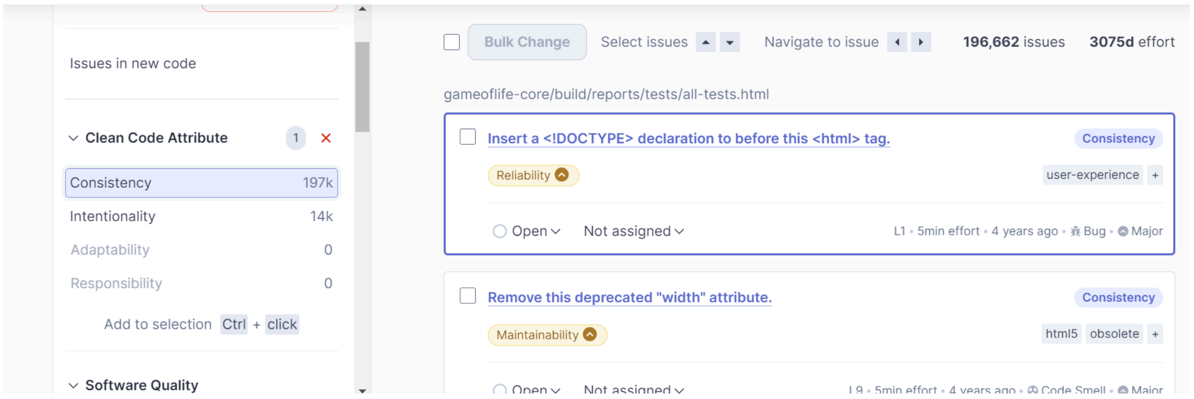
Passed

New analysis in progress

New Code Overall Code

15) Code Problems

- Consistency


Issues in new code

Clean Code Attribute 1

Consistency	197k
Intentionality	14k
Adaptability	0
Responsibility	0

Add to selection Ctrl + click

Software Quality

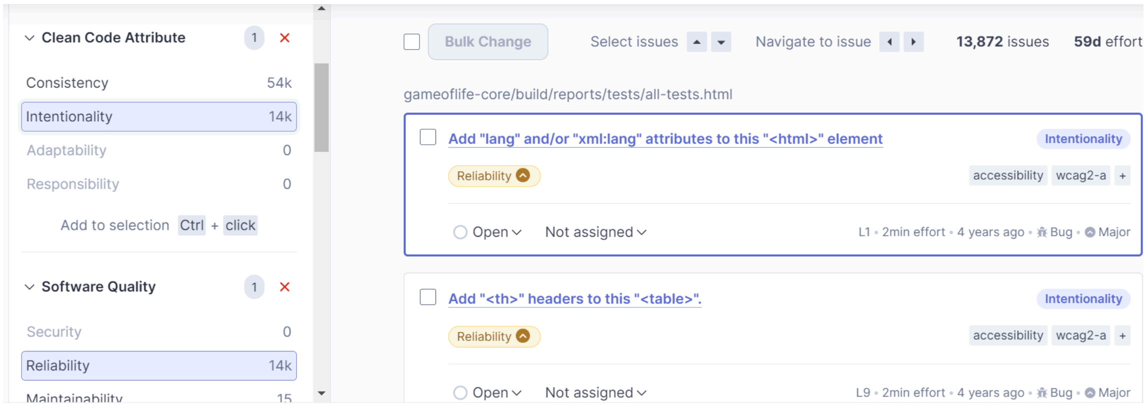
Bulk Change Select issues Navigate to issue **196,662** issues **3075d** effort

gameoflife-core/build/reports/tests/all-tests.html

Insert a <!DOCTYPE> declaration to before this <html> tag. Consistency
Reliability user-experience
Open Not assigned L1 • 5min effort • 4 years ago • Bug • Major

Remove this deprecated "width" attribute. Consistency
Maintainability html5 obsolete
Open Not assigned L9 • 5min effort • 4 years ago • Code Smell • Major

- Intentionality


Clean Code Attribute 1

Consistency	54k
Intentionality	14k
Adaptability	0
Responsibility	0

Add to selection Ctrl + click

Software Quality 1

Security	0
Reliability	14k
Maintainability	15

Bulk Change Select issues Navigate to issue **13,872** issues **59d** effort

gameoflife-core/build/reports/tests/all-tests.html

Add "lang" and/or "xml:lang" attributes to this "<html>" element Intentionality
Reliability accessibility wcag2-a
Open Not assigned L1 • 2min effort • 4 years ago • Bug • Major

Add "<th>" headers to this "<table>". Intentionality
Reliability accessibility wcag2-a
Open Not assigned L9 • 2min effort • 4 years ago • Bug • Major

Conclusion

The experiment successfully integrated SonarQube analysis into a Jenkins pipeline for a GitHub project, allowing for automated code quality checks. While the cloning of the repository worked well, several challenges arose, particularly with configuring the SonarScanner path and handling command execution errors in the Windows environment. Issues with authentication and the need for a user token added complexity to the setup. Despite these hurdles, the integration improved the project's continuous integration process, and lessons learned can inform future enhancements, such as better error handling and notifications for analysis results.