

Expt No. 08 Advanced DevOps Lab

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, optimizing 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

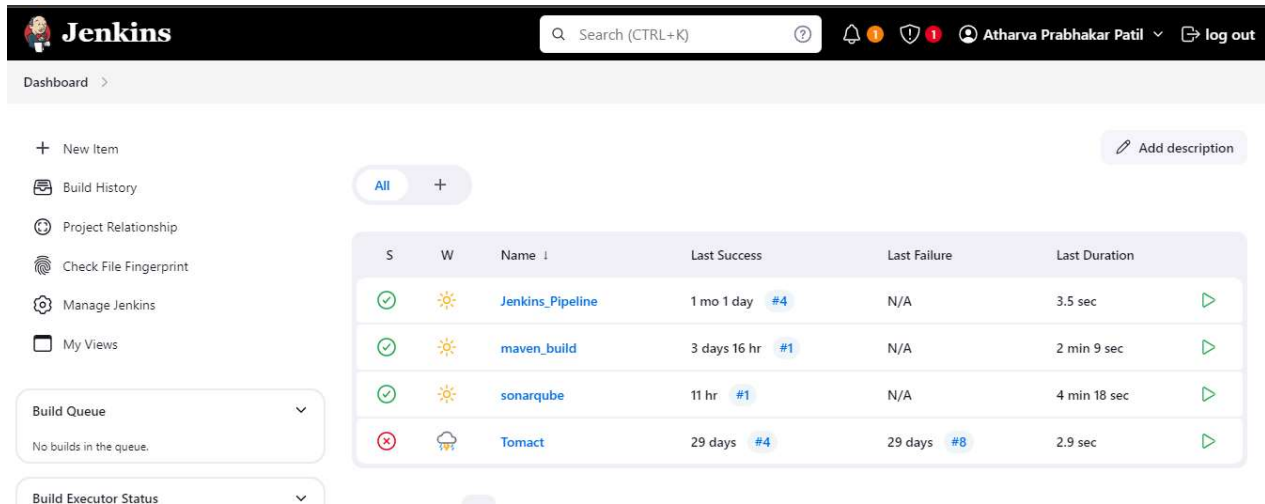
Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

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

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



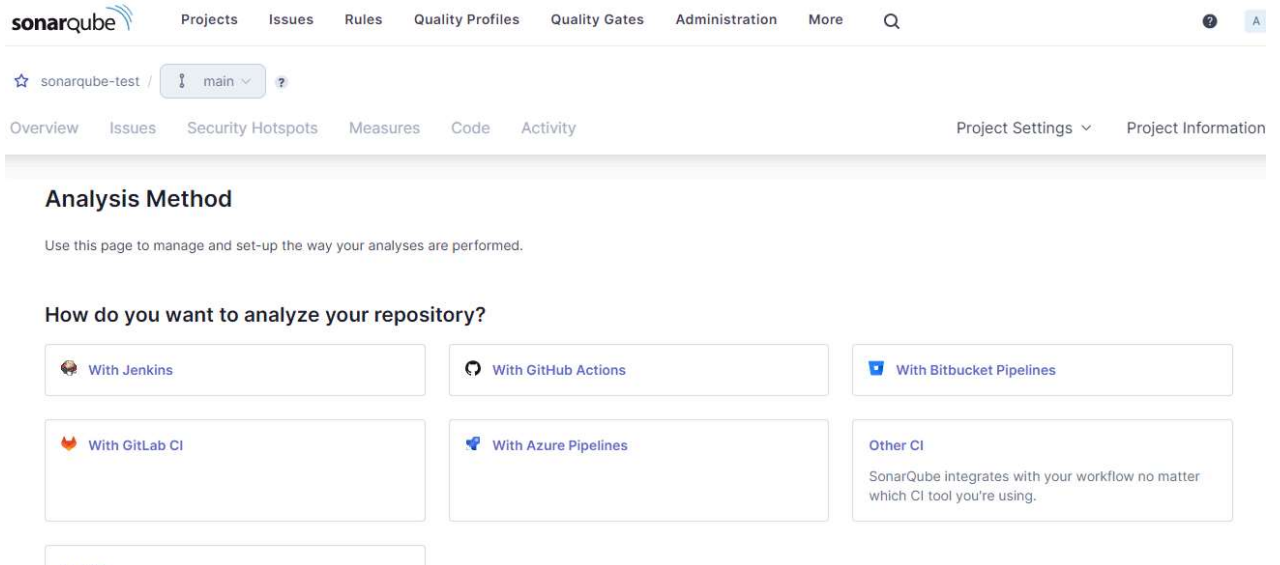
The screenshot shows the Jenkins Dashboard with a search bar and user profile. The main content area displays a table of build history. The table has columns for Status (S), Weather icon (W), Name, Last Success, Last Failure, and Last Duration. The builds listed are Jenkins_Pipeline, maven_build, sonarqube, and Tomact.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	Jenkins_Pipeline	1 mo 1 day #4	N/A	3.5 sec
✓	☀	maven_build	3 days 16 hr #1	N/A	2 min 9 sec
✓	☀	sonarqube	11 hr #1	N/A	4 min 18 sec
✗	☁	Tomact	29 days #4	29 days #8	2.9 sec

2. Run SonarQube in a Docker container using this command -

```
PS C:\Users\sushmita> docker rm -f sonarqube
sonarqube
PS C:\Users\sushmita> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
365e3fcbba859edb271a2d9d7489e503571b2b822c64ec235a36a0aabf175c59
PS C:\Users\sushmita>
```

3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



The screenshot shows the SonarQube web interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main content area is titled "Analysis Method" and contains a section "How do you want to analyze your repository?". This section offers several options: With Jenkins, With GitHub Actions, With Bitbucket Pipelines, With GitLab CI, With Azure Pipelines, and Other CI. The "Other CI" option includes a note: "SonarQube integrates with your workflow no matter which CI tool you're using."

4. Login to SonarQube using username *admin* and password *admin*.

5. Create a local project in SonarQube with the name **sonarqube-test**

1 of 2

Create a local project

Project display name *

Project key *

Main branch name *

The name of your project's default branch [Learn More](#) 

Setup the project and come back to Jenkins Dashboard.

Download the sonar scanner and place it in a directory using

mkdir Downloads\sonarqube

cd Downloads\sonarqube

Download by pasting the link in browser:

<https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.2.0.1873-windows.zip>

Paste the following command in Docker terminal

unzip sonar-scanner-cli-4.2.0.1873-windows.zip

Also download the latest java version

6. Create a New Item in Jenkins, choose **Pipeline**.

New Item

Enter an item name

» This field cannot be empty, please enter a valid name

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



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.

OK

7. Under Pipeline Script, enter the following -

```
node {
  stage('Cloning the GitHub Repo')
  {
    git 'https://github.com/shazforiot/GOL.git'
  }
  stage('SonarQube analysis') {
    withSonarQubeEnv('sonarqube-test') {
      bat """
C:\\Users\\sushmita\\Downloads\\sonar-scanner-6.2.0.4584-windows-x6
4\\bin\\sonar-scanner.bat ^
-D sonar.login=admin ^
-D sonar.password=atharva ^
-D sonar.projectKey=sonarqube-test ^
-D sonar.exclusions=vendor/*,resources/,./java ^
-D sonar.host.url=http://localhost:9000/
      """
    }
  }
}
```

The screenshot shows the Jenkins Pipeline Script editor. The script is as follows:

```
1 node {
2   stage('Cloning the GitHub Repo')
3   {
4     git 'https://github.com/shazforiot/GOL.git'
5   }
6   stage('SonarQube analysis') {
7     withSonarQubeEnv('sonarqube-test') {
8       bat """
9       C:\\Users\\sushmita\\Downloads\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat ^
10      -D sonar.login=admin ^
11      -D sonar.password=atharva ^
12      -D sonar.projectKey=sonarqube-test ^
13      -D sonar.exclusions=vendor/*,resources/,./java ^
14      -D sonar.host.url=http://localhost:9000/
15      """
16    }
17  }
18 }
```

Below the script editor, there is a checkbox labeled "Use Groovy Sandbox" which is checked. There is also a link for "Pipeline Syntax". At the bottom, there are two buttons: "Save" and "Apply".

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

8. Run The Build.

Dashboard >

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

Build Queue ▾
No builds in the queue.

Build Executor Status ▾

Stage View

	Cloning the GitHub Repo	SonarQube analysis
Average stage times:	23s	34s
#11 Sep 26 19:21 No Changes	9s	9s failed
#10 Sep 26 19:02 No Changes	6s	6s failed
#9 Sep 26 18:52 No Changes	53s	1min 26s failed
#8 Sep 26 18:15 No Changes		
#7 Sep 26 18:13 No Changes		

9. Check the console output once the build is complete.



Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Atharva Prabhakar Patil
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git --version # timeout=10
> git --version # 'git version 2.43.0.windows.1'
```

10. After that, check the project in SonarQube.

The screenshot displays the SonarQube web interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The left sidebar shows filters for Quality Gate (Passed: 2, Failed: 0) and Reliability (A: 1, B: 0, C: 1, D: 0, E: 0). The main content area lists two projects:

- sonarqube PUBLIC**: Last analysis: 1 hour ago. The main branch of this project is empty. Status: Passed.
- sonarqube-test PUBLIC**: Last analysis: 16 minutes ago - 683k Lines of Code - HTML, XML, ... Status: Passed.

Below the project list, a summary bar for 'sonarqube-test' shows the following metrics:

Metric	Value
Security	0
Reliability	68k
Maintainability	164k
Hotspots Reviewed	0.0%
Coverage	50.6%
Duplications	50.6%

At the bottom right, it indicates '2 of 2 shown'.

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

11. Code Problems -

Open Issues

The screenshot displays the SonarQube web interface for a project named 'sonarqube-test'. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The left sidebar shows a list of quality metrics: Security Review, Duplications, Size, Complexity, and Issues. The 'Issues' category is expanded, showing 'Open Issues' with a count of 210,549, and 'Confirmed Issues', 'Accepted Issues', and 'False Positive Issues' all with counts of 0. The main panel shows a tree view of the project structure, including folders like 'gameoflife-acceptance-tests', 'gameoflife-build', 'gameoflife-core', 'gameoflife-deploy', and 'gameoflife-web', and a file 'pom.xml'. The 'Open Issues' count is 210,549, and there is a link to 'See history'.

Consistency

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

sonarqube-test / main

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project I

My Issues All

Filters Clear All Filters

Issues in new code

Clean Code Attribute 1 x

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 user-experience L1 - 5min effort - 4 years ago - Bug - Major

Remove this deprecated `"width"` attribute. Consistency html5 obsolete L9 - 5min effort - 4 years ago - Code Smell - Major

Remove this deprecated `"align"` attribute. Consistency html5 obsolete L11 - 5min effort - 4 years ago - Code Smell - Major

Embedded database should be used for evaluation purposes only

Intentionality

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

sonarqube-test / main

Overview Issues Security Hotspots Measures Code Activity

Project Settings

My Issues All

Filters Clear All Filters

Issues in new code

Clean Code Attribute 1 x

Consistency 197k

Intentionality 14k

Adaptability 0

Responsibility 0

Add to selection Ctrl + click

Software Quality

Bulk Change

Select issues Navigate to issue 13,88

gameoflife-acceptance-tests/Dockerfile

Use a specific version tag for the image. Maintainability L1 - 5min effort - 4 years ago -

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Maintainability L12 - 5min effort - 4 years ago -

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Maintainability L12 - 5min effort - 4 years ago -

Embedded database should be used for evaluation purposes only

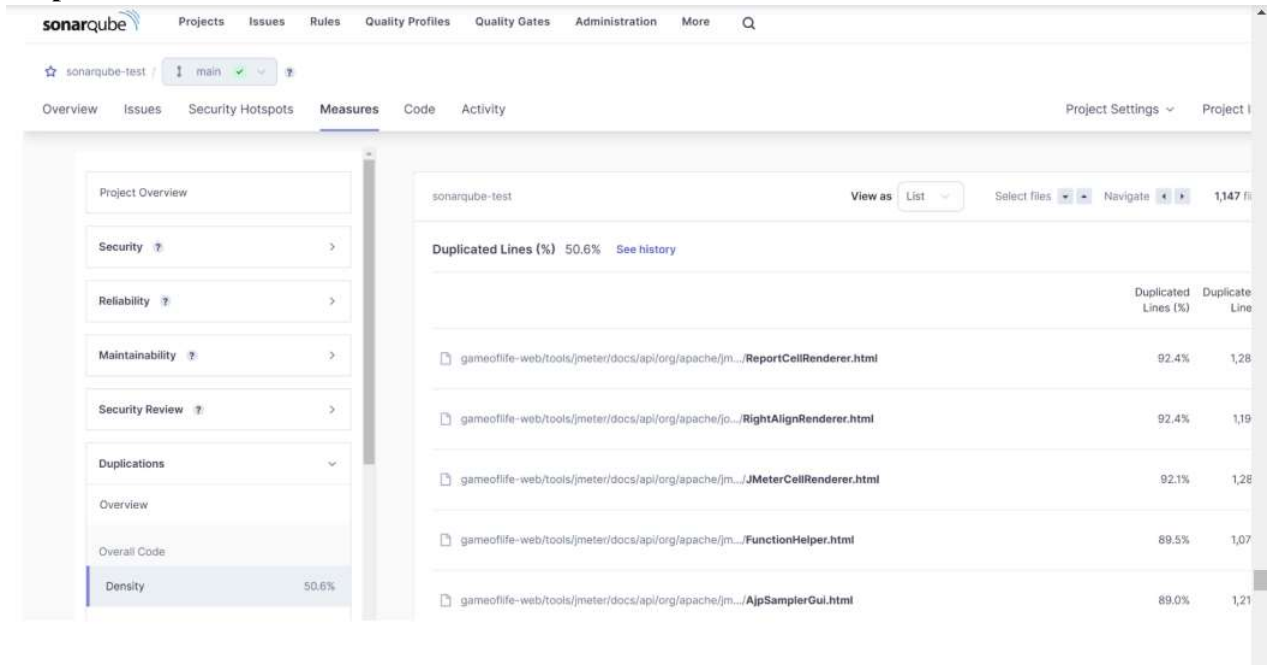
Bugs and Code Smells

The screenshot shows the SonarQube interface for the 'main' branch. The left sidebar displays the 'Issues' tab with a filter for 'Code Smell' (253 issues). The main panel shows a list of issues, including 'Add an "alt" attribute to this image.' (Intentionality, Reliability) and 'Add an "alt" attribute to this image.' (Intentionality, Reliability). The bottom of the page displays a warning: 'Embedded database should be used for evaluation purposes only'.

Reliability

The screenshot shows the SonarQube interface for the 'main' branch. The left sidebar displays the 'Issues' tab with a filter for 'Reliability' (14k issues). The main panel shows a list of issues, including 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' (Intentionality, Reliability) and 'Add "<th>" headers to this "<table>:"' (Intentionality, Reliability). The bottom of the page displays a warning: 'Embedded database should be used for evaluation purposes only'.

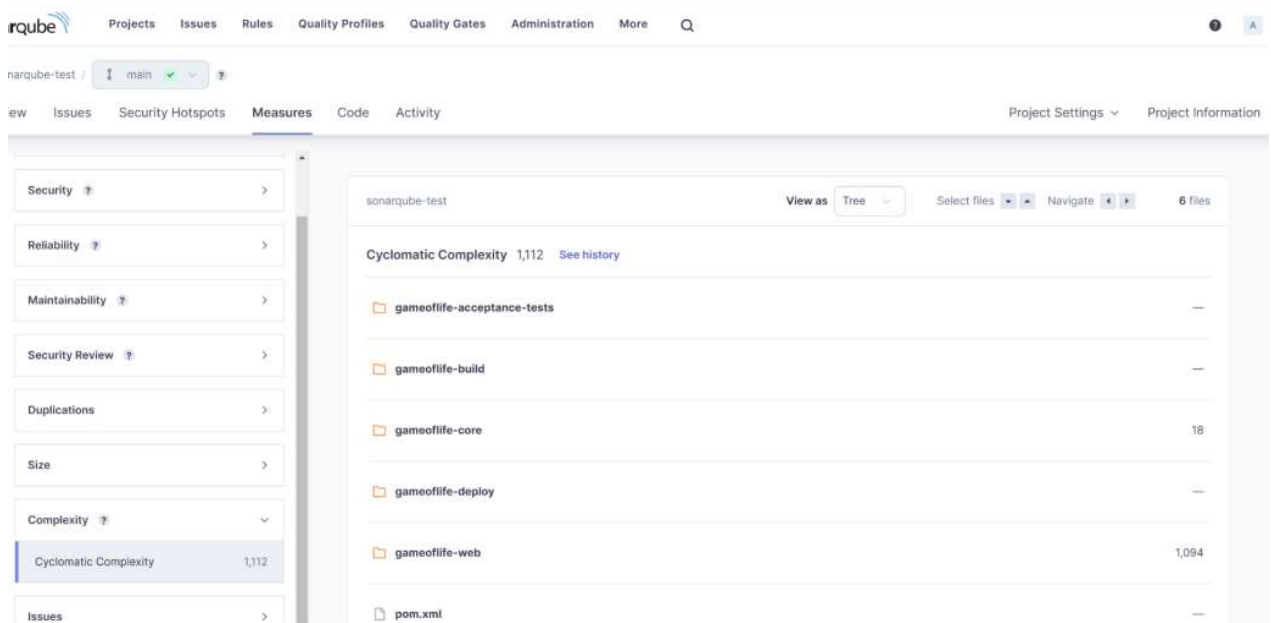
Duplicates



The screenshot shows the SonarQube interface for the 'sonarqube-test' project. The 'Measures' tab is selected, and the 'Duplications' section is expanded. The 'Density' measure is highlighted, showing a value of 50.6%. The main content area displays a table of duplicated lines across various files.

File	Duplicated Lines (%)	Duplicate Line
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../ReportCellRendererer.html	92.4%	1,28
gameoflife-web/tools/jmeter/docs/api/org/apache/jo.../RightAlignRendererer.html	92.4%	1,19
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../JMeterCellRendererer.html	92.1%	1,28
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../FunctionHelper.html	89.5%	1,07
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../AppSamplerGui.html	89.0%	1,21

Cyclomatic Complexities



The screenshot shows the SonarQube interface for the 'sonarqube-test' project. The 'Measures' tab is selected, and the 'Complexity' section is expanded. The 'Cyclomatic Complexity' measure is highlighted, showing a value of 1,112. The main content area displays a table of cyclomatic complexity across various files and directories.

File	Cyclomatic Complexity
gameoflife-acceptance-tests	—
gameoflife-build	—
gameoflife-core	18
gameoflife-deploy	—
gameoflife-web	1,094
pom.xml	—

In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

Conclusion:

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample Java application.