

Experiment - 8

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.

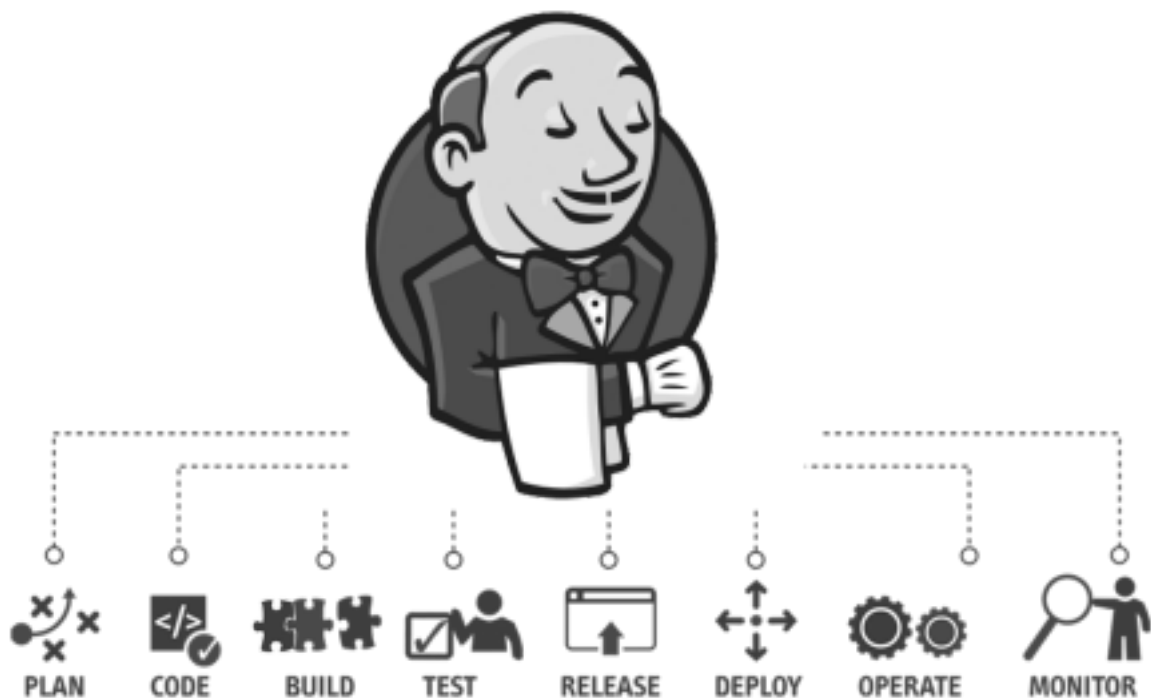
Why is SAST important?

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:

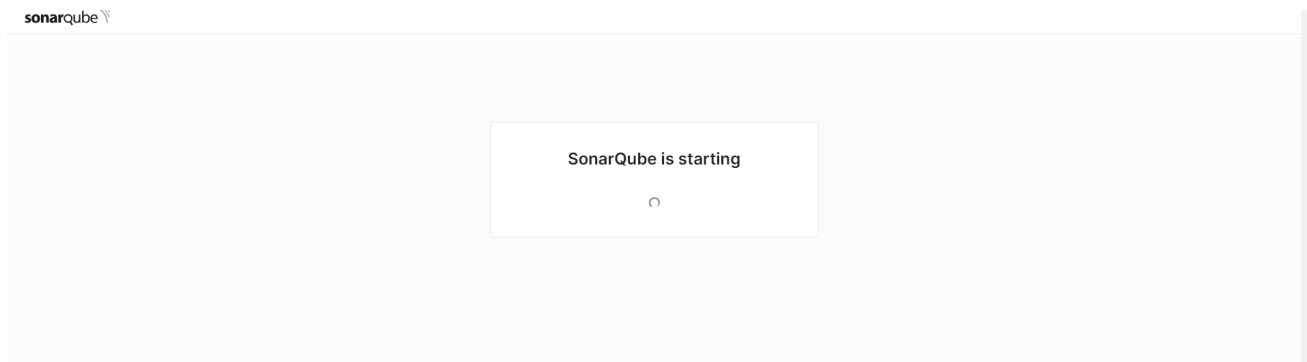
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

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

```
C:\Users\ADMIN>docker run -d --name sonarqube2 -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9001:9000 sonarqube:latest  
fda86b00e3989f3eb5aca8396b29b2a0adc95bcfe0fc5d85cf1237491e7678b9
```

3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.
4. Login to SonarQube using username *admin* and password *admin*.



5. Create a manual 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.

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

New Item

Enter an item name

SonarQube-8

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.



Multi-configuration project

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



Folder

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') {
      sh "<PATH_TO_SONARQUBE_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=http://127.0.0.1:9000/"
    }
  }
}
```

Pipeline

Definition

Pipeline script

Script ?

```
2 stage('Cloning the GitHub Repo') {
3   git 'https://github.com/shazforiot/GOL.git'
4 }
5
6 stage('SonarQube analysis') {
7   withSonarQubeEnv('sonarqube') {
8     bat """
9       C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner ^
10      -D sonar.login=admin ^
11      -D sonar.password=admin123 ^
12      -D sonar.projectKey=sonarqube-test ^
13      -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
14      -D sonar.host.url=http://127.0.0.1:9001/
15      """
16   }
17 }
18 }
19 }
```

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

8. Run The Build.

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

Status

</> Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

Build History

trend

Filter...

#4

Sep 26, 2024, 6:04 PM

#3

Sep 26, 2024, 5:13 PM

SonarQube-8

Stage View

Average stage times:
(Average full run time: ~12min 10s)

	Cloning the GitHub Repo	SonarQube analysis
#4	1s	12min 7s
#3	8s	50min 43s
#2		aborted
#1		

Dashboard > SonarQube-8 > #4

Status

</> Changes

Console Output

View as plain text

Edit Build Information

Delete build #4

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Replay

Pipeline Steps

Workspaces

Previous Build

Console Output

Skipping 4,248 KB. Full Log

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 798. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 810. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 823. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 844. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 865. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 776. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 778. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 530. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 648. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 798. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 546. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 546. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 634. Keep only the first 100 references.

18:13:34.790 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/gui/HTTPArgumentsPanel.html for block at line 778. Keep only the first 100 references.

```

references.
18:13:39.657 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 75. Keep only the first 100
references.
18:13:39.657 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 41. Keep only the first 100
references.
18:13:39.657 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 17. Keep only the first 100
references.
18:13:39.657 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 296. Keep only the first 100
references.
18:13:39.657 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/gui/util/TextAreaCellRenderer.html for block at line 75. Keep only the first 100
references.
18:13:39.657 INFO CPD Executor CPD calculation finished (done) | time=158971ms
18:13:39.674 INFO SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
18:15:49.696 INFO Analysis report generated in 5022ms, dir size=127.2 MB
18:16:08.759 INFO Analysis report compressed in 19048ms, zip size=29.6 MB
18:16:09.884 INFO Analysis report uploaded in 1125ms
18:16:09.887 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9001/dashboard?id=sonarqube-test
18:16:09.887 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
18:16:09.887 INFO More about the report processing at http://127.0.0.1:9001/api/ce/task?id=6f22c333-3777-4a21-b058-0ab4c049625c
18:16:22.970 INFO Analysis total time: 12:02.242 s
18:16:22.975 INFO SonarScanner Engine completed successfully
18:16:23.699 INFO EXECUTION SUCCESS
18:16:23.706 INFO Total time: 12:05.758s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

10. After that, check the project in SonarQube.

The screenshot shows the SonarQube web interface for a project named 'sonarqube-test'. The main branch is selected, and the Quality Gate is 'Passed'. The interface displays various metrics:

- Quality Gate:** Passed (Last analysis 15 minutes ago)
- New analysis in progress:** New analysis in progress
- Overall Code:**
 - Security:** 0 Open issues (A)
 - Reliability:** 68k Open issues (C)
 - Maintainability:** 164k Open issues (A)
 - Accepted issues:** 0 (Valid issues that were not fixed)
 - Coverage:** 50.6% (On 0 lines to cover)
 - Duplications:** 50.6% (On 759k lines)
 - Security Hotspots:** 3 (E)

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

Bugs

SonarQube

- Projects
- Issues
- Rules
- Quality Profiles
- Quality Gates
- Administration
- More

sonarqube-test / main

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Software Quality

Security	0
Reliability	47k
Maintainability	0

Severity

Type	Count
Bug	47k
Vulnerability	0
Code Smell	164k

Add to selection Ctrl + click

Scope

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

☐ Add "lang" and/or "xml:lang" attributes to this "<html>" element
Intentionality Reliability wcag2-a L1 - 2min effort - 4 years ago - # Bug - @ Major

☐ Insert a <!DOCTYPE> declaration to before this <html> tag.
Consistency Reliability user-experience L1 - 5min effort - 4 years ago - # Bug - @ Major

☐ Add "<th>" headers to this "<table>".
Intentionality Reliability accessibility wcag2-a L9 - 2min effort - 4 years ago - # Bug - @ Major

gameoflife-core/build/reports/tests/allclasses-frame.html

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) ACTIVE LGPL v3 Community Documentation Plugins Web API

Code Smells

The screenshot displays the SonarQube web application interface. At the top, there's a navigation bar with tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. Below this is a breadcrumb trail: sonarqube-test / main. The left sidebar contains a tree view under 'Software Quality' with categories like Security, Reliability, and Maintainability. Under 'Severity', there are options for Bug and Vulnerability. Under 'Type', 'Code Smell' is selected. A message at the bottom of the sidebar states: 'Add to selection Ctrl + click'. The main area shows the configuration for the file 'gameoflife-acceptance-tests/Dockerfile'. It lists three issues, all with the same description: 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' Each issue has a severity level (L1 or L2), an effort estimate (5min), and a creation time (4 years ago). The first two issues are assigned to 'Code Smell' and have a 'Major' priority, while the third is assigned to 'Code Smell' and has a 'Major' priority. Each issue also has a 'Maintainability' icon and a dropdown menu for 'Open' and 'Not assigned'.

Intentional Issues

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

sonarqube-test / main

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

Filters

Clear All Filters

Issues in new code

Clean Code Attribute

Consistency164k

Intentionality268

Adaptability0

Responsibility0

Add to selectionCtrl + click

Software Quality

Security0

Reliability253

Maintainability15

Add to selectionCtrl + click

gameoflife-acceptance-tests/Dockerfile

Use a specific version tag for the image.

Maintainability

Intentionality

No tags

OpenNot assigned

L1 • 5min effort • 4 years ago • Code Smell • Major

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.

Maintainability

Intentionality

No tags

OpenNot assigned

L12 • 5min effort • 4 years ago • Code Smell • Major

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.

Maintainability

Intentionality

No tags

OpenNot assigned

L12 • 5min effort • 4 years ago • Code Smell • Major

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.

Maintainability

Intentionality

No tags

OpenNot assigned

L12 • 5min effort • 4 years ago • Code Smell • Major

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

Reliability Issue

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

sonarqube-test / main

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

My IssuesAll

Filters

Clear All Filters

Issues in new code

Clean Code Attribute

Consistency21k

Intentionality253

Adaptability0

Responsibility0

Add to selectionCtrl + click

Software Quality

Security0

Reliability21k

Maintainability164k

Add to selectionCtrl + click

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

Bulk Change

Select issues

Navigate to issue

20,856 issues217d effort

Anchors must have content and the content must be accessible by a screen reader.

MaintainabilityReliability

Consistency

accessibility

OpenNot assigned

L29 • 5min effort • 4 years ago • Code Smell • Minor

Anchors must have content and the content must be accessible by a screen reader.

MaintainabilityReliability

Consistency

accessibility

OpenNot assigned

L38 • 5min effort • 4 years ago • Code Smell • Minor

Anchors must have content and the content must be accessible by a screen reader.

MaintainabilityReliability

Consistency

accessibility

OpenNot assigned

L47 • 5min effort • 4 years ago • Code Smell • Minor

Anchors must have content and the content must be accessible by a screen reader.

MaintainabilityReliability

Consistency

accessibility

OpenNot assigned

L47 • 5min effort • 4 years ago • Code Smell • Minor

Embedded database should be used for evaluation purposes only

Maintainability Issue

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

sonarqube-test / main

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

My IssuesAll

Filters

Clear All Filters

Issues in new code

Clean Code Attribute

Consistency164k

Intentionality15

Adaptability0

Responsibility0

Add to selectionCtrl + click

Software Quality

Security0

Reliability21k

Maintainability164k

Add to selectionCtrl + click

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

Bulk Change

Select issues

Navigate to issue

163,766 issues1705d effort

Remove this deprecated "width" attribute.

Maintainability

Consistency

html5obsolete

OpenNot assigned

L9 • 5min effort • 4 years ago • Code Smell • Major

Remove this deprecated "align" attribute.

Maintainability

Consistency

html5obsolete

OpenNot assigned

L11 • 5min effort • 4 years ago • Code Smell • Major

Remove this deprecated "align" attribute.

Maintainability

Consistency

html5obsolete

OpenNot assigned

L12 • 5min effort • 4 years ago • Code Smell • Major

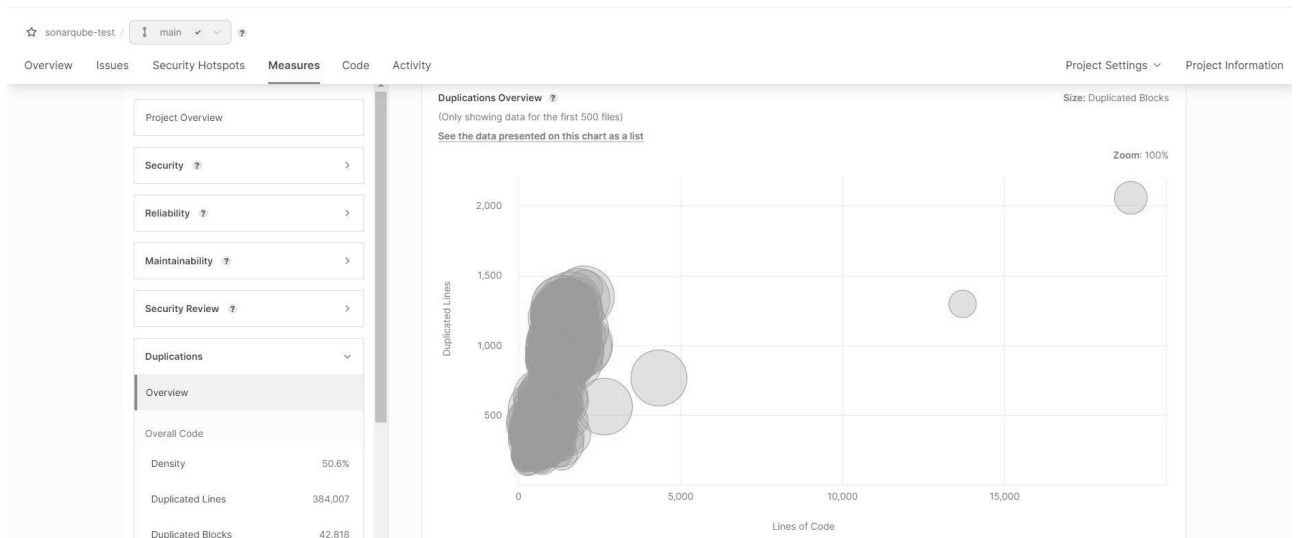
Remove this deprecated "size" attribute.

Maintainability

Consistency

OpenNot assigned

Duplicates



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, I successfully created a CI/CD pipeline using Jenkins integrated with SonarQube for static code analysis on a sample Java application. I set up SonarQube in a Docker container and configured Jenkins to clone the GitHub repository and perform the analysis. The pipeline detected various issues, including bugs, code smells, and security vulnerabilities, which I reviewed in SonarQube. This experience enhanced my skills in configuring CI/CD tools and highlighted the importance of maintaining code quality through automation. Overall, I gained valuable insights into integrating tools for effective software development practices.