

Adv.DevOps 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.

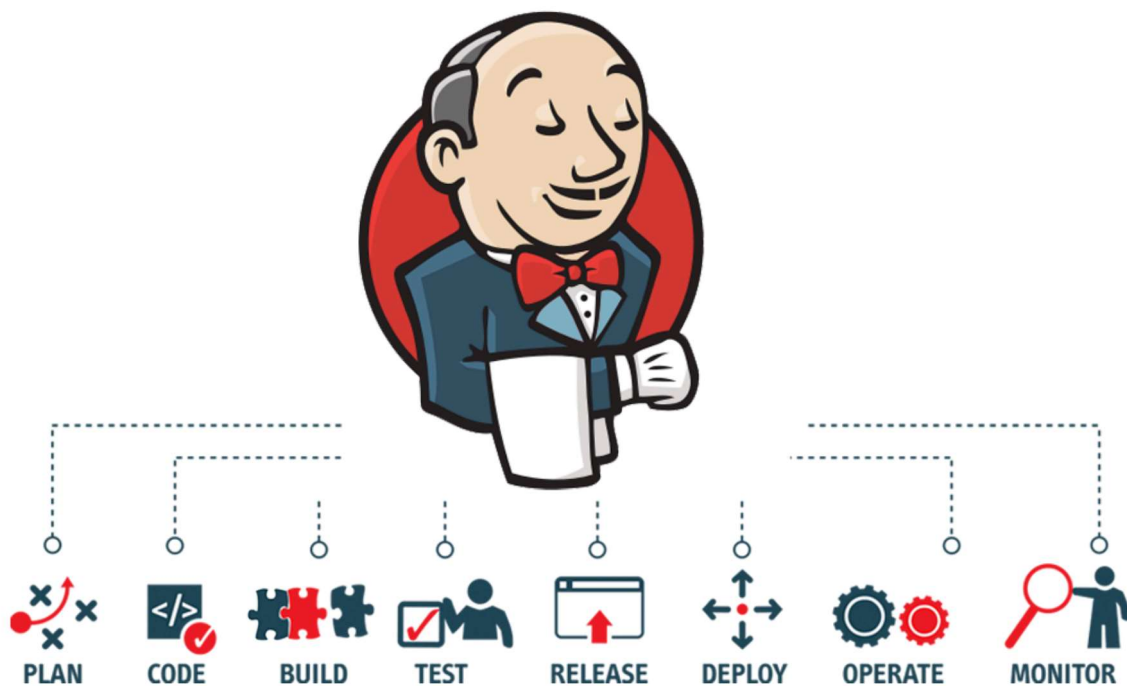
Why is SAST important?

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, 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

Integrating Jenkins with SonarQube:

Prerequisites:

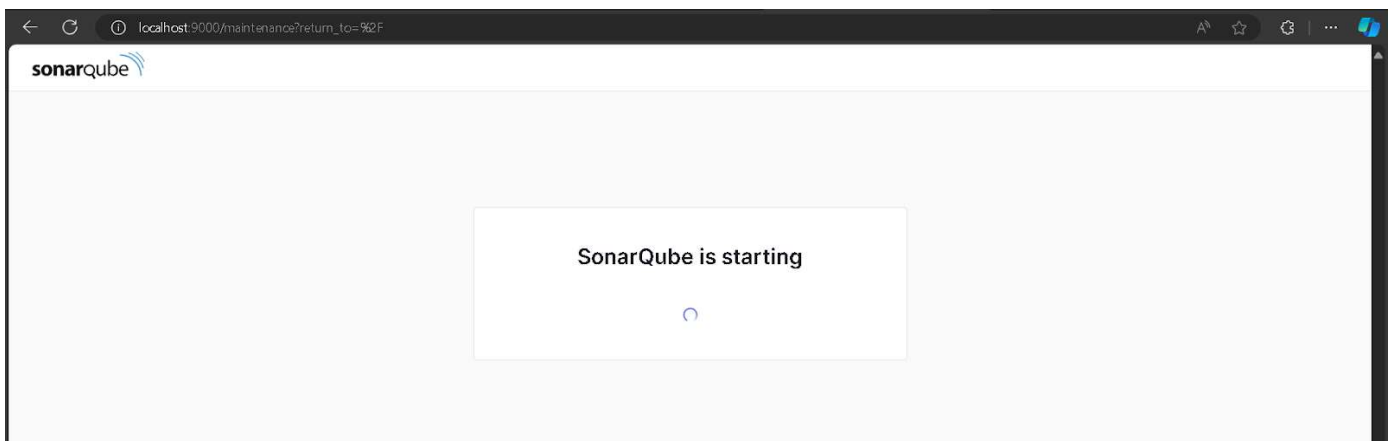
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

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 -

```
PS C:\Users\devpg> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
60de6878d0614254500f608f43d81a3430585dc282e74225fe2a8fa237ee9d76
PS C:\Users\devpg> |
```

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 *

sonarqube-test ✓

Project key *

sonarqube-test ✓

Main branch name *

main

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

Cancel

Next

Setup the project and come back to Jenkins Dashboard.

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

New Item

Enter an item 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.

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 ?

```
1 node {
2   stage('Cloning the GitHub Repo') {
3     git 'https://github.com/shazforiot/GOL.git'
4   }
5   stage('SonarQube Analysis') {
6     withSonarQubeEnv('sonarqube') {
7       bat """
8         C:/ProgramData/Jenkins/.jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarqube/bin/sonar-scanner.bat ^
9         -Dsonar.login=admin ^
10        -Dsonar.password=helloworld ^
11        -Dsonar.projectKey=sonarqube-test ^
12        -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
13        -Dsonar.host.url=http://127.0.0.1:9000/
14        """
15     }
16   }
17 }
18 }
```

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

8. Run The Build.

 Status

 Changes

 Build Now

 Configure

 Delete Pipeline

 Stages

 Rename

 Pipeline Syntax

 < Build #2

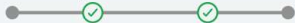
 Rebuild

 Console


 Configure

Pipeline

Start Cloning the Git... SonarQube Ana... End



Details

 Manually run by Dev Gaonkar

 Started 11 min ago

 Queued 9 ms

 Took 8 min 51 sec

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

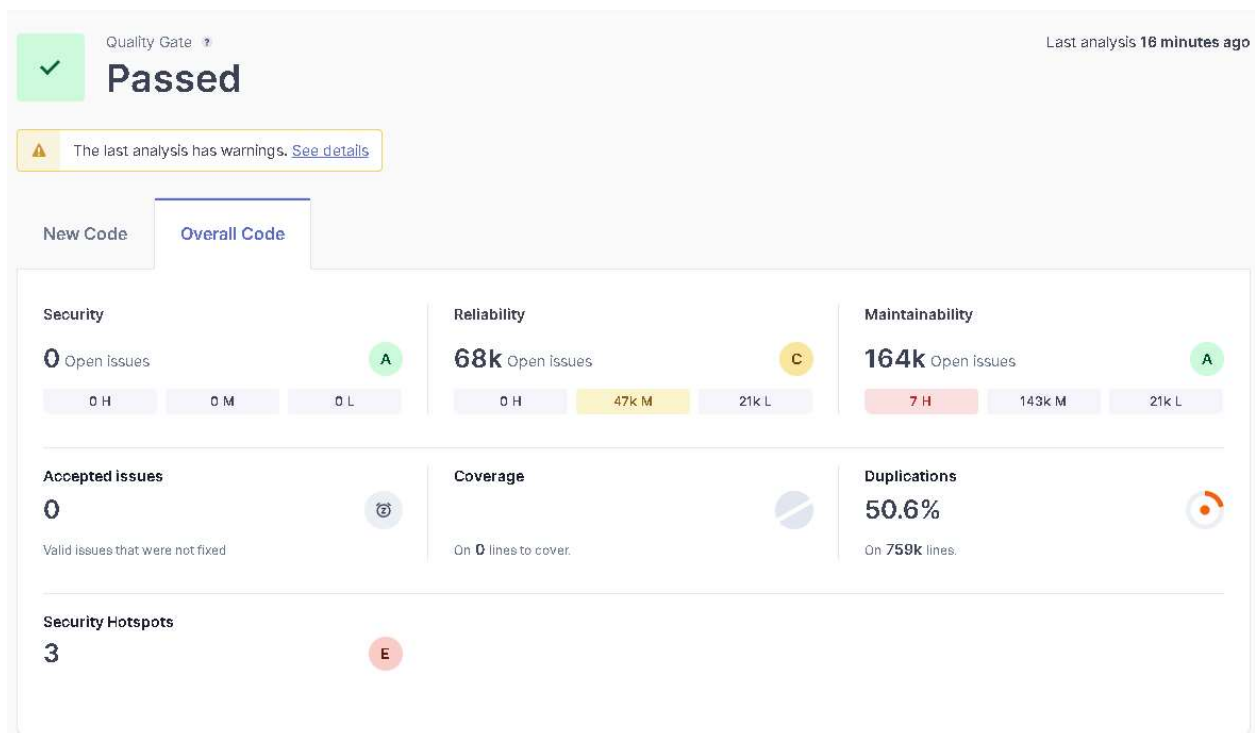
Console Output

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

```
Started by user Dev Gaonkar
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\devSonarQube
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\devSonarQube\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.42.0.windows.2'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
> C:\Program Files\Git\bin\git.exe branch -a -v --no-abbrev # timeout=10
> C:\Program Files\Git\bin\git.exe branch -D master # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -b master ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
Commit message: "Update Jenkinsfile"
```

```
21:33:24.250 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.
21:33:24.253 INFO CPD Executor CPD calculation finished (done) | time=109702ms
21:33:24.381 INFO SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
21:34:52.072 INFO Analysis report generated in 3326ms, dir size=127.2 MB
21:35:00.718 INFO Analysis report compressed in 8633ms, zip size=29.6 MB
21:35:02.183 INFO Analysis report uploaded in 1462ms
21:35:02.188 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-test
21:35:02.188 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:35:02.188 INFO More about the report processing at http://127.0.0.1:9000/api/ce/task?id=4fcac741-ad4f-4465-99b6-bd81a49a94cd
21:35:11.422 INFO Analysis total time: 8:38.464 s
21:35:11.434 INFO SonarScanner Engine completed successfully
21:35:12.040 INFO EXECUTION SUCCESS
21:35:12.079 INFO Total time: 8:45.529s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

10. After that, check the project in SonarQube.



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

11. Code Problems -

Bugs:

The image shows the SonarQube interface for viewing code problems. On the left, there is a sidebar with filters:

- Clean Code Attribute:** Consistency (33k), Intentionality (14k), Adaptability (0), Responsibility (0).
- Software Quality:** Security (0), Reliability (47k), Maintainability (0).
- Severity:** (Expandable)
- Type:** Bug (47k), Vulnerability (0), Code Smell (164k). A note says "Add to selection Ctrl + click".

The main area displays a list of issues. At the top, it says "Bulk Change" and "Select Issues". Below this, the file path "gameoflife-core/build/reports/tests/all-tests.html" is shown. The issues listed are:

- Issue 1:** "Insert a <DOCTYPE> declaration to before this <html> tag." (Consistency). Reliability: C. Status: Open. Not assigned. L1 • 5min effort • 4 years ago • Bug • Major.
- Issue 2:** "Add 'lang' and/or 'xml:lang' attributes to this '<html>' element" (Intentionality). Reliability: C. Status: Open. Not assigned. L1 • 2min effort • 4 years ago • Bug • Major.
- Issue 3:** "Add '<th>' headers to this '<table>'." (Intentionality). Reliability: C. Status: Open. Not assigned. L9 • 2min effort • 4 years ago • Bug • Major.

Below these, the file path "gameoflife-core/build/reports/tests/allclasses-frame.html" is shown, followed by another issue:

- Issue 4:** "Add 'lang' and/or 'xml:lang' attributes to this '<html>' element" (Intentionality).

Code Smells:

Clean Code Attribute

Consistency184k

Intentionality268

Adaptability0

Responsibility0

Software Quality

Security0

Reliability21k

Maintainability164k

Severity ?

Type

Bug47K

Vulnerability0

Code Smell164k

Add to selectionCtrl + click

Bulk Change

Select Issues

Navigate to Issue

164,034 issues

1708d effort

gameoflife-acceptance-tests/Dockerfile

Use a specific version tag for the image.

Intentionality

Maintainability

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.

Intentionality

Maintainability

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.

Intentionality

Maintainability

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.

Intentionality

Maintainability

No tags

OpenNot assigned

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

Consistency Issues:

My IssuesAll

Filters

Clear All Filters

Issues in new code

Clean Code Attribute

Consistency197k

Intentionality14k

Adaptability0

Responsibility0

Add to selectionCtrl + click

Software Quality

Security0

Reliability54k

Maintainability164k

Severity ?

Bulk Change

Select Issues

Navigate to Issue

198,862 issues

3075d effort

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

Insert a <!DOCTYPE> declaration to before this <html> tag.

Consistency

Reliability

user-experience

OpenNot assigned

L1 • 5min effort • 4 years ago • Bug • Major

Remove this deprecated "width" attribute.

Consistency

Maintainability

html5obsolete

OpenNot assigned

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

Remove this deprecated "align" attribute.

Consistency

Maintainability

html5obsolete

OpenNot assigned

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

Remove this deprecated "align" attribute.

Consistency

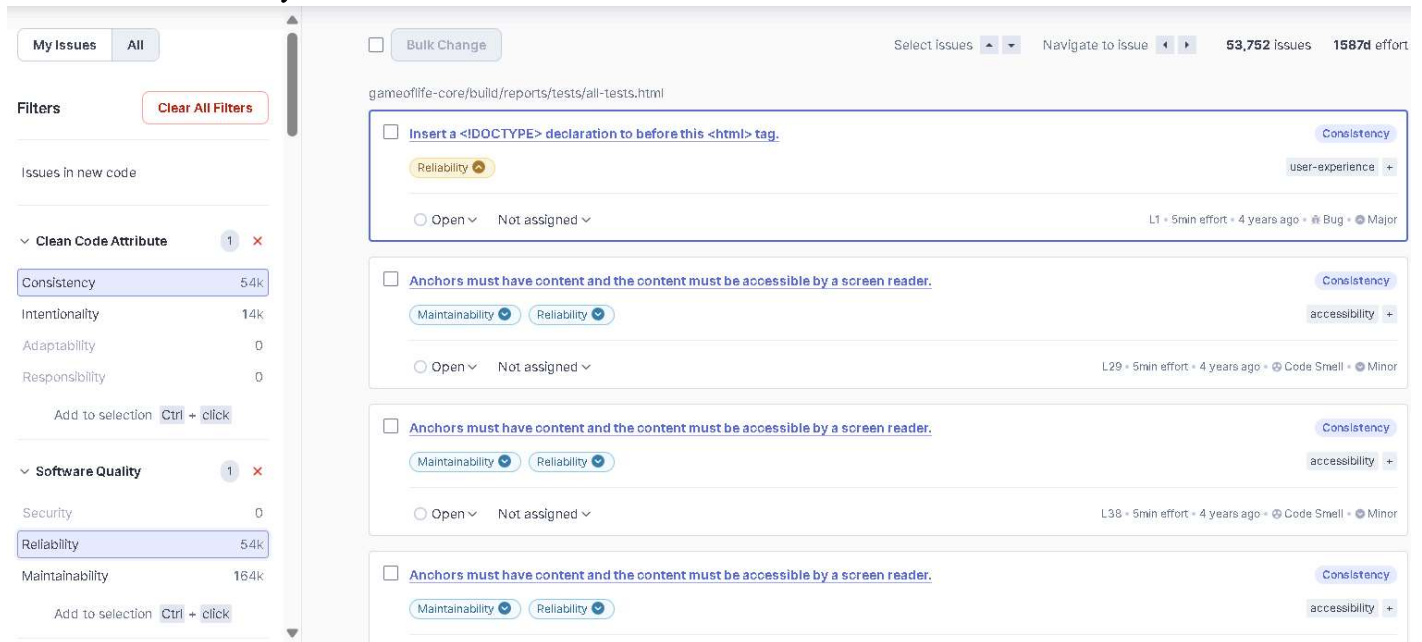
Maintainability

html5obsolete

OpenNot assigned

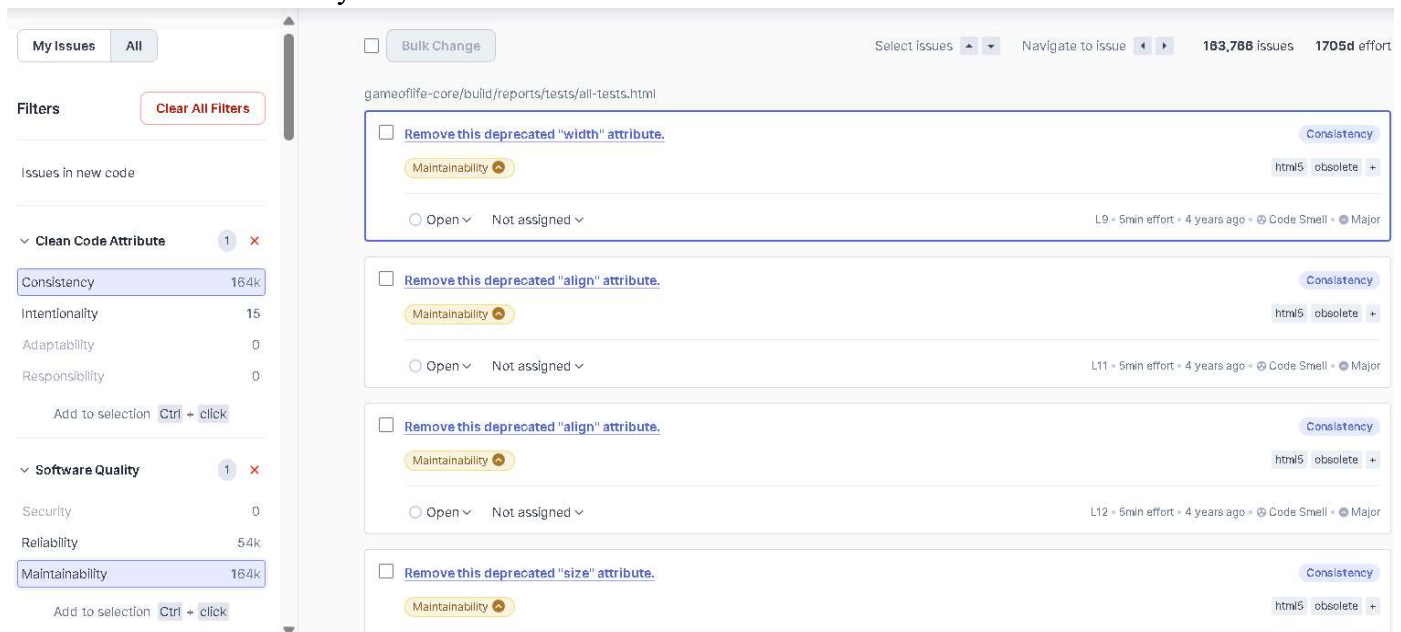
L11 • 5min effort • 4 years ago • Code Smell • Major

Reliability Issues:



The screenshot shows the SonarQube interface for Reliability Issues. On the left, a sidebar contains filters for 'Clean Code Attribute' (Consistency: 54k, Intentionality: 14k, Adaptability: 0, Responsibility: 0) and 'Software Quality' (Security: 0, Reliability: 54k, Maintainability: 164k). The main panel displays a list of issues under the path 'gameoflife-core/build/reports/tests/all-tests.html'. The first issue is 'Insert a <!DOCTYPE> declaration to before this <html> tag.' with a 'Reliability' severity and 'user-experience' category. The second and third issues are 'Anchors must have content and the content must be accessible by a screen reader.' with 'Maintainability' and 'Reliability' severities and 'accessibility' category. Each issue entry includes a checkbox, a link to the issue, a severity tag, a category tag, and a status dropdown (Open/Not assigned). The top right of the main panel shows '53,752 issues' and '1587d effort'.

Maintainability Issues:



The screenshot shows the SonarQube interface for Maintainability Issues. The sidebar filters are identical to the Reliability Issues page. The main panel displays a list of issues under the same path. The first issue is 'Remove this deprecated "width" attribute.' with a 'Maintainability' severity and 'html5 obsolete' category. The second and third issues are 'Remove this deprecated "align" attribute.' with a 'Maintainability' severity and 'html5 obsolete' category. The fourth issue is 'Remove this deprecated "size" attribute.' with a 'Maintainability' severity and 'html5 obsolete' category. Each issue entry includes a checkbox, a link to the issue, a severity tag, a category tag, and a status dropdown (Open/Not assigned). The top right of the main panel shows '183,788 issues' and '1705d effort'.

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 conclusion, integrating Jenkins with SonarQube through a CI/CD pipeline enhances code quality by automating static code analysis. By setting up SonarQube in a Docker container and configuring Jenkins to run SonarQube scans, we can efficiently detect and address bugs, code smells, and security vulnerabilities early in the development cycle. This approach ensures comprehensive code inspections and provides actionable feedback, leading to more secure and maintainable code. Ultimately, this integration streamlines the development process, improves software quality, and supports continuous improvement in coding practices.