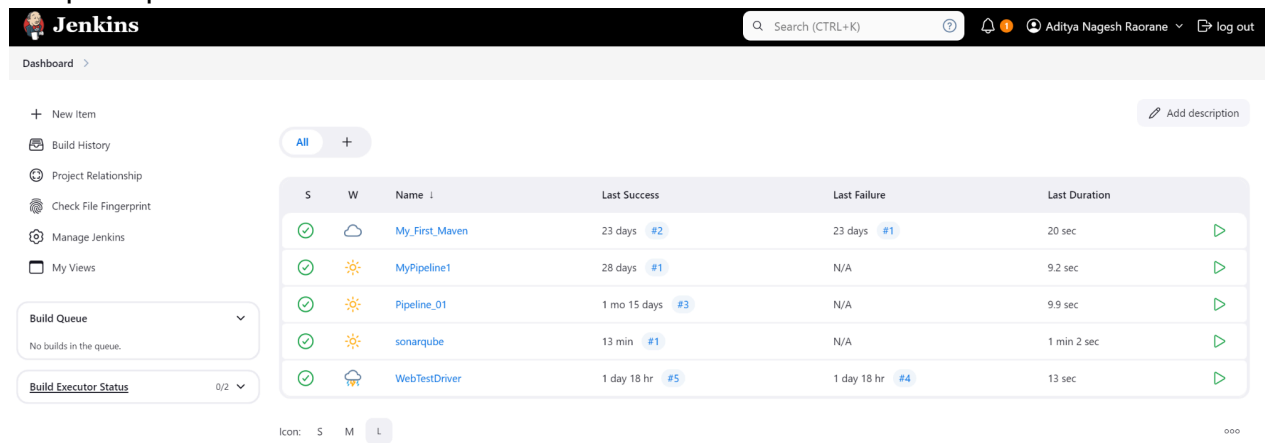


Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

1. Open up Jenkins Dashboard on localhost:8080.



The screenshot shows the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: New Item, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, and My Views. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '0/2'). The main area displays a table of builds with columns: S (Status), W (Icon), Name, Last Success, Last Failure, and Last Duration. The table lists five builds: 'My_First_Maven', 'MyPipeline1', 'Pipeline_01', 'sonarqube', and 'WebTestDriver'. Each build has a status icon (green checkmark), a build icon (cloud, sun, or gear), and a duration. The 'sonarqube' build is highlighted in blue.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	My_First_Maven	23 days #2	23 days #1	20 sec
✓	☀	MyPipeline1	28 days #1	N/A	9.2 sec
✓	☀	Pipeline_01	1 mo 15 days #3	N/A	9.9 sec
✓	☀	sonarqube	13 min #1	N/A	1 min 2 sec
✓	⚙	WebTestDriver	1 day 18 hr #5	1 day 18 hr #4	13 sec

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

a) `docker -v`

b) `docker pull sonarqube`

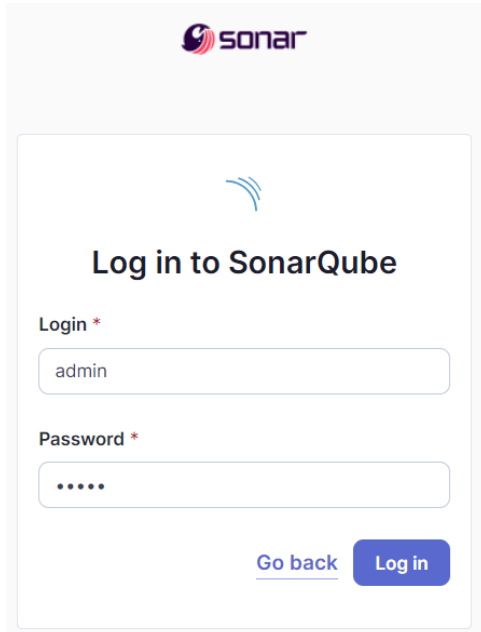
c) `docker run -d --name sonarqube -e`

`SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest`

```
C:\Users\adity>docker -v
Docker version 27.0.3, build 7d4bcd8

C:\Users\adity>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
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
4a6e73f4472de892b1ddead1abe77372a85a7b09408cce3a0abd37c5ab6b49a4
```

3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is “**aditya**”.

The image shows the SonarQube login interface. At the top is the Sonar logo. Below it is a large white box with a blue Sonar icon and the text "Log in to SonarQube". There are two input fields: "Login *" with the value "admin" and "Password *" with masked characters. At the bottom of the box are two buttons: "Go back" (a link) and "Log in" (a blue button).

4. Create a local 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

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

☐ Reference branch

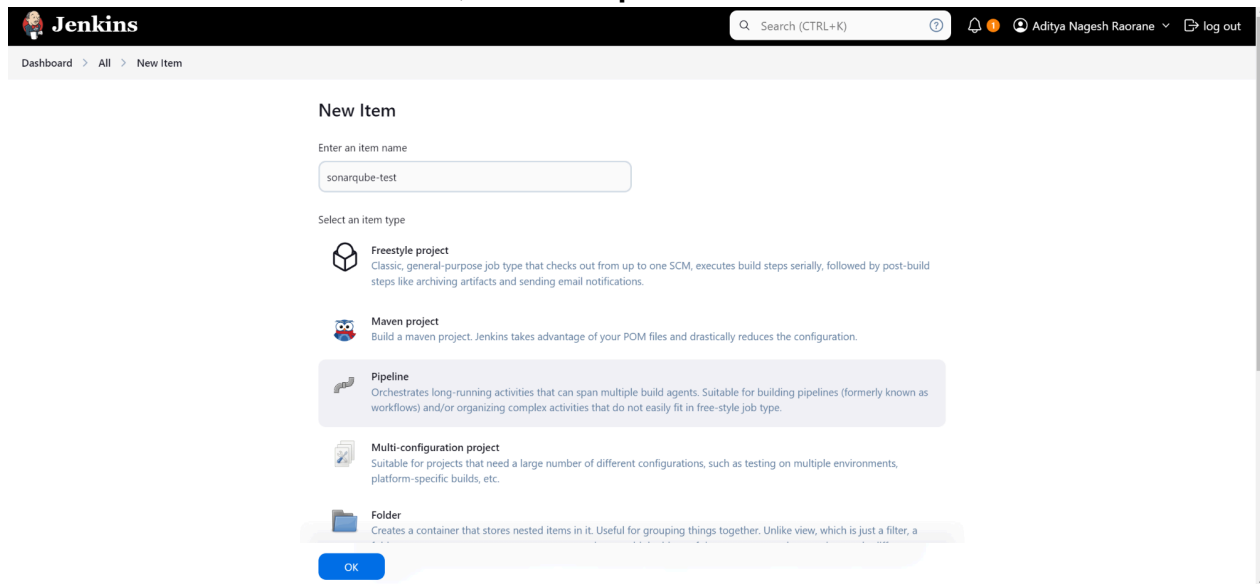
Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

Back

Create project

Setup the project and come back to Jenkins Dashboard.

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



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') {
            bat
            "C:\\Users\\adity\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-s
            canner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat \
            -D sonar.login=<YOUR ID> \
            -D sonar.password=<YOUR PASSWORD> \
            -D sonar.projectKey=<YOUR PROJECT KEY> \
            -D sonar.exclusions=vendor/**,resources/**,**/*.java \
            -D sonar.host.url=http://localhost:9000/"
        }
    }
}
```

The screenshot shows the Jenkins 'Configure' page for a pipeline named 'sonarqube-test'. The 'Pipeline' tab is selected, and the 'Definition' is set to 'Pipeline script'. The script is a Groovy pipeline with the following content:

```
1 * node {
2   stage('Cloning the Github Repo') {
3     git 'https://github.com/shazforint/GOL.git'
4   }
5   stage('SonarQube analysis') {
6     withSonarQubeEnv('sonarqube') {
7       bat "C:\Users\aditya\Downloads\sonar-scanner-cli-6.1.0.4477-windows-x64\sonar-scanner-6.1.0.4477-windows-x64\bin\sonar-scanner.bat \
8         -D sonar.login=admin \
9         -D sonar.password=aditya \
10        -D sonar.projectKey=sonarqube-test \
11        -D sonar.exclusions=vendor/**,resources/**,*/*.java \
12        -D sonar.host.url=http://localhost:9000/"
13     }
14   }
15 }
```

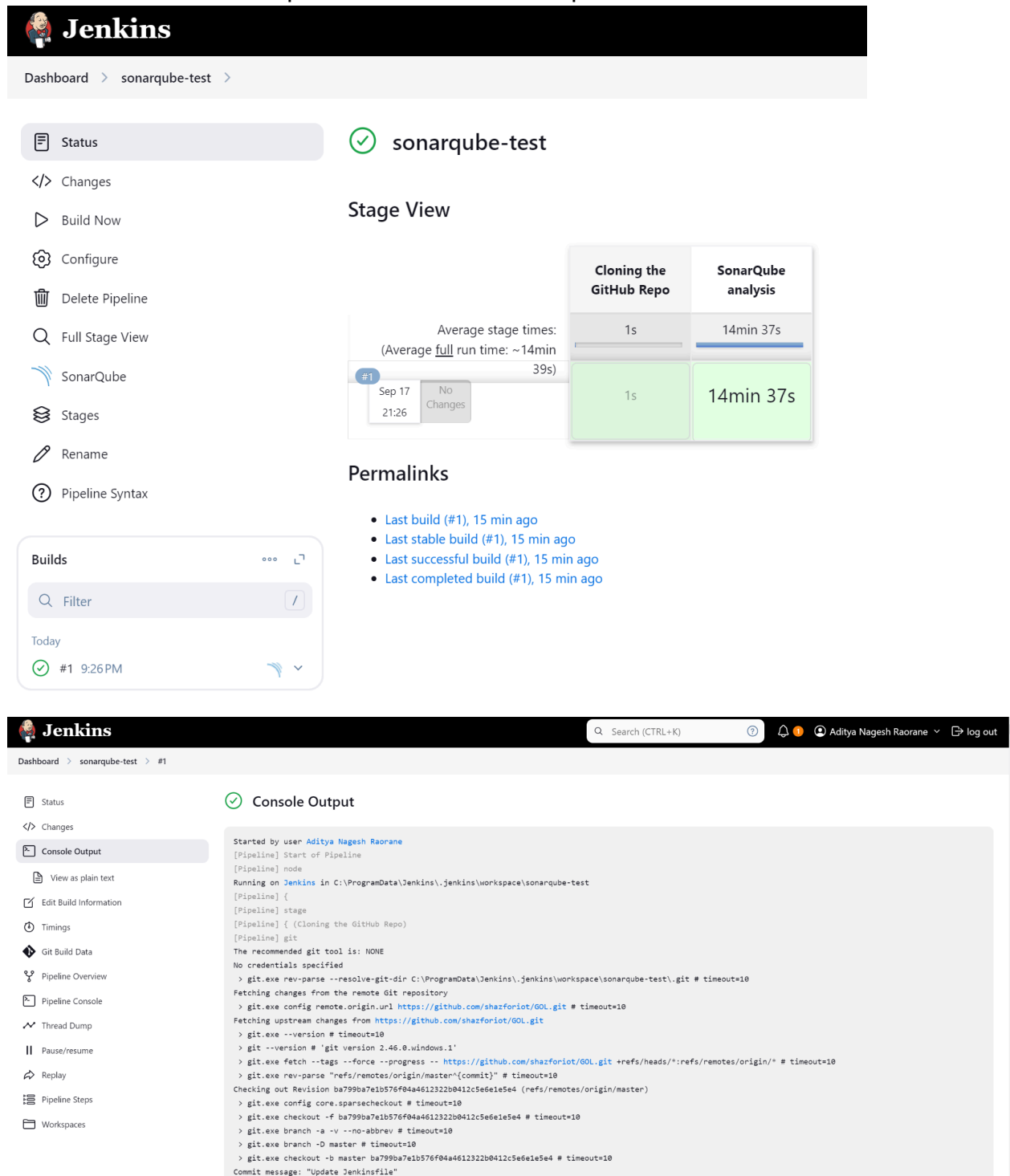
Below the script, the 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons. The footer of the page shows 'REST API' and 'Jenkins 2.473'.

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

8. Run The Build.

The screenshot shows the Jenkins interface for the 'sonarqube-test' pipeline. At the top is the Jenkins logo. Below it is a breadcrumb trail: 'Dashboard > sonarqube-test >'. The main content area has three buttons: 'Status' (with a document icon), 'Changes' (with a code icon), and 'Build Now' (with a play icon).

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



The image shows two screenshots of the Jenkins web interface. The top screenshot displays the 'sonarqube-test' pipeline view, and the bottom screenshot shows the 'Console Output' for the same pipeline.

Jenkins Pipeline View

Dashboard > **sonarqube-test**

Status ✓ **sonarqube-test**

Stage View

Stage	Cloning the GitHub Repo	SonarQube analysis
Average stage times:	1s	14min 37s
(Average full run time: ~14min 39s)	1s	14min 37s

Builds

Filter: /

Today

- #1 9:26 PM

Permalinks

- Last build (#1), 15 min ago
- Last stable build (#1), 15 min ago
- Last successful build (#1), 15 min ago
- Last completed build (#1), 15 min ago

Jenkins Console Output

Started by user Aditya Nagesh Raorane

```
[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.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git.exe --version # timeout=10
> git --version # git version 2.46.0.windows.1
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master:{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
Commit message: "Update Jenkinsfile"
```

Dashboard > sonarqube-test > #1

```

line 41. Keep only the first 100 references.
21:37:59.929 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.
21:37:59.929 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.
21:37:59.929 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.
21:37:59.930 INFO CPD Executor CPD calculation finished (done) | time=153336ms
21:37:59.955 INFO SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
21:40:14.276 INFO Analysis report generated in 5151ms, dir size=127.2 MB
21:40:35.678 INFO Analysis report compressed in 2138ms, zip size=29.6 MB
21:40:36.170 INFO Analysis report uploaded in 492ms
21:40:36.173 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
21:40:36.173 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:40:36.173 INFO More about the report processing at http://localhost:9000/api/ce/task?id=99fcd1e5-df4e-4f9f-8688-3169741e0856
21:40:53.466 INFO Analysis total time: 14:32.336 s
21:40:53.468 INFO SonarScanner Engine completed successfully
21:40:54.148 INFO EXECUTION SUCCESS
21:40:54.150 INFO Total time: 14:35.645s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

REST API Jenkins 2.473

10. After that, check the project in SonarQube.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

My Favorites All

Filters

Quality Gate

Passed 2

Failed 0

Reliability

A 1

B 0

C 1

D 0

E 0

Security

A 2

Create Project

Search for projects... Perspective Overall Status Sort by Name 2 project(s)

☆ sonarqube PUBLIC Passed

Last analysis: 1 hour ago

The main branch of this project is empty.

☆ sonarqube-test PUBLIC Passed

Last analysis: 16 minutes ago · 683k Lines of Code · HTML, XML, ...

A 0 B 68k C 164k D 0.0% E 50.6%

Security Reliability Maintainability Hotspots Reviewed Coverage Duplications

2 of 2 shown

The screenshot shows the SonarQube Overview page for the project 'sonarqube-test'. The main status is 'Passed' with a green checkmark. A warning message states: 'The last analysis has warnings. See details'. The page displays various quality metrics:

- Security:** 0 Open Issues (Grade A)
- Reliability:** 68k Open Issues (Grade C)
- Maintainability:** 164k Open Issues (Grade A)
- Accepted Issues:** 0 (Valid issues that were not fixed)
- Coverage:** On 0 lines to cover.
- Duplications:** 50.6% (On 759k lines)

The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The left sidebar shows the project 'sonarqube-test' and the 'main' branch. The right sidebar contains 'Project Settings' and 'Project Information'.

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

11. Code Problems - Open Issues

The screenshot shows the SonarQube Measures page for the project 'sonarqube-test'. The 'Measures' tab is selected, and the 'Open Issues' measure is chosen from the left sidebar. The main area displays a tree view of the project structure with the number of open issues for each component:

Component	Open Issues
gameoflife-acceptance-tests	4
gameoflife-build	0
gameoflife-core	603
gameoflife-deploy	0
gameoflife-web	209,940
pom.xml	2

The left sidebar lists various measures: Security Review, Duplications, Size, Complexity, Issues, Overall Code, Open Issues (210,549), Confirmed Issues (0), Accepted Issues (0), and False Positive Issues (0). The top navigation bar and project information are consistent with the previous screenshot.

Consistency

The screenshot displays the SonarQube web interface for a project named 'sonarqube-test'. The 'Issues' tab is active, showing a list of consistency issues. The left sidebar contains filters and a 'Clean Code Attribute' section with a table of attributes: Consistency (197k), Intentionality (14k), Adaptability (0), and Responsibility (0). The main panel shows three issues related to HTML tags: 'Insert a <DOCTYPE> declaration to before this <html> tag.', 'Remove this deprecated "width" attribute.', and 'Remove this deprecated "align" attribute.'. Each issue is categorized by 'Reliability' or 'Maintainability' and includes a 'Consistency' tag. The bottom of the page features a yellow warning banner: 'Embedded database should be used for evaluation purposes only'.

Clean Code Attribute	Count
Consistency	197k
Intentionality	14k
Adaptability	0
Responsibility	0

Intentionality

The screenshot displays the SonarQube web interface for the same project, now showing 'Intentionality' issues. The left sidebar is identical to the previous screenshot. The main panel shows three issues related to Dockerfile tags: 'Use a specific version tag for the image.', 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.', and another instance of 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.'. Each issue is categorized by 'Maintainability' and includes an 'Intentionality' tag. The bottom of the page features the same yellow warning banner: 'Embedded database should be used for evaluation purposes only'.

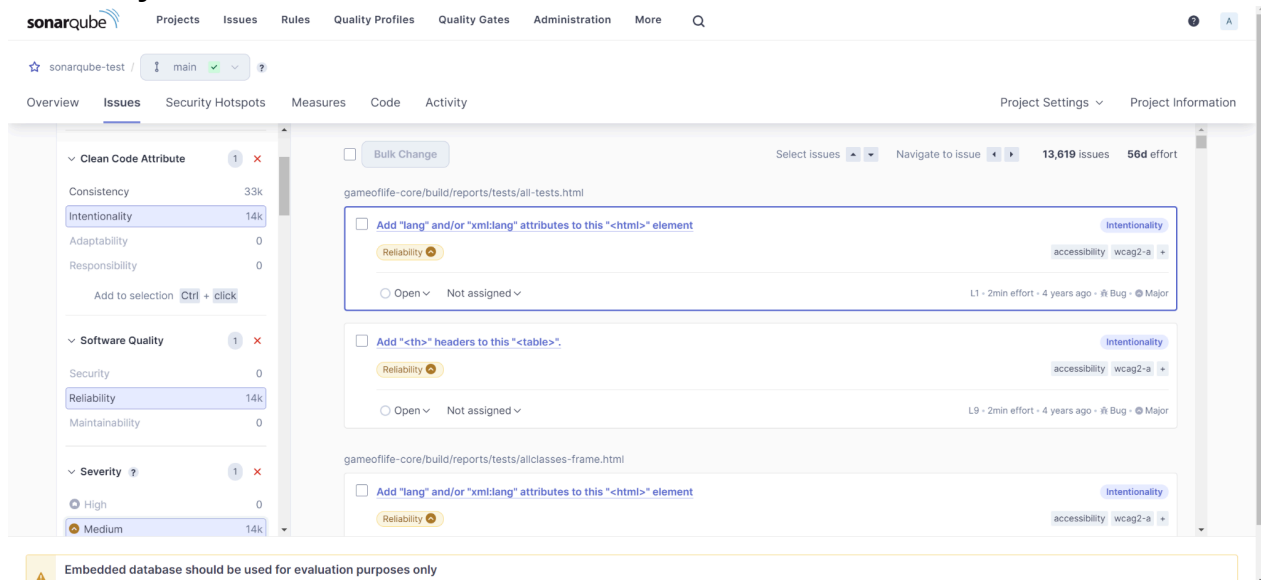
Code Smells

The screenshot shows the SonarQube interface with the 'Issues' tab selected. The left sidebar displays filters for Severity (High: 0, Medium: 0, Low: 253) and Type (Bug: 14k, Vulnerability: 0, Code Smell: 253). The main area shows three issues, all of type 'Code Smell' and severity 'Low'. Each issue is titled 'Add an "alt" attribute to this image.' and is located in the file 'gameoflife-web/tools/jmeter/printable_docs/building.html'. The issues are L29, L31, and L31 (repeated). The right sidebar shows '253 issues' and '2d 5h effort'. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

Bugs

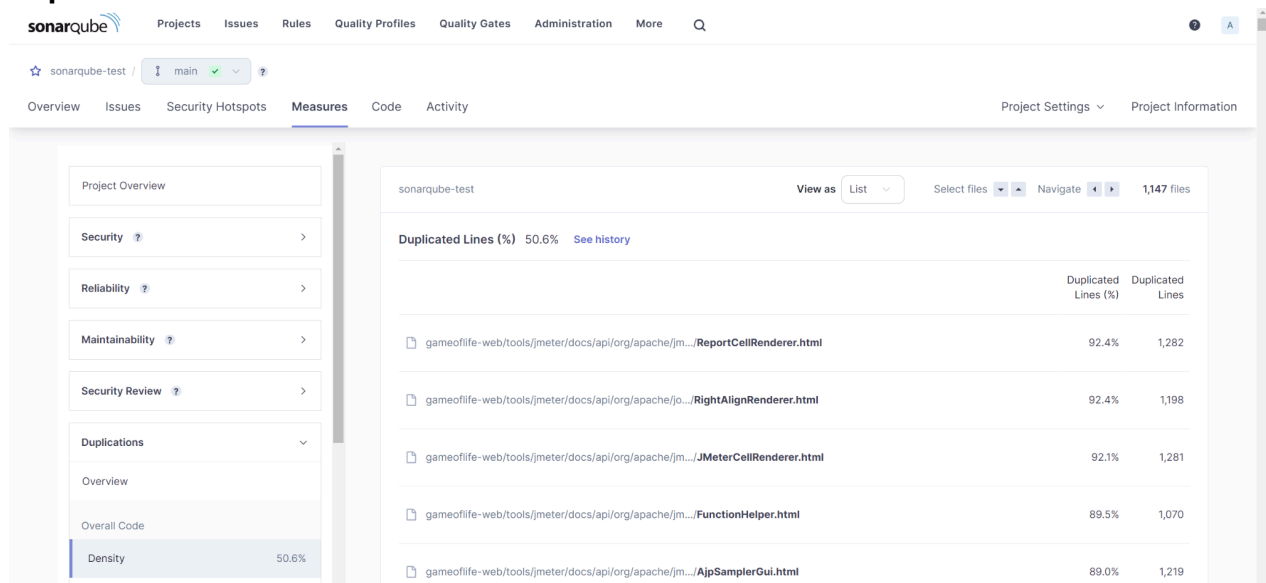
The screenshot shows the SonarQube interface with the 'Issues' tab selected. The left sidebar displays filters for Severity (High: 0, Medium: 14k, Low: 0) and Type (Bug: 14k, Vulnerability: 0, Code Smell: 253). The main area shows three issues, all of type 'Bug' and severity 'Major'. Each issue is titled 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' and is located in the file 'gameoflife-core/build/reports/tests/all-tests.html'. The issues are L1, L9, and L9 (repeated). The right sidebar shows '13,619 issues' and '56d effort'. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

Reliability



The screenshot shows the SonarQube interface for the 'sonarqube-test' project, specifically the 'Issues' tab under the 'Reliability' category. The left sidebar displays a hierarchy of quality attributes: Clean Code Attribute (1 issue), Software Quality (1 issue), and Severity (1 issue). Under Clean Code Attribute, 'Intentionality' is highlighted with 14k issues. Under Software Quality, 'Reliability' is highlighted with 14k issues. The main area shows a list of issues for the file 'gameoflife-core/build/reports/tests/all-tests.html'. Two issues are visible, both with a 'Reliability' severity and an 'Intentionality' category. The first issue is 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' with a 'wcag2-a' accessibility rule. The second issue is 'Add "<th>" headers to this "<table>"' with a 'wcag2-a' accessibility rule. A warning banner at the bottom states: 'Embedded database should be used for evaluation purposes only'.

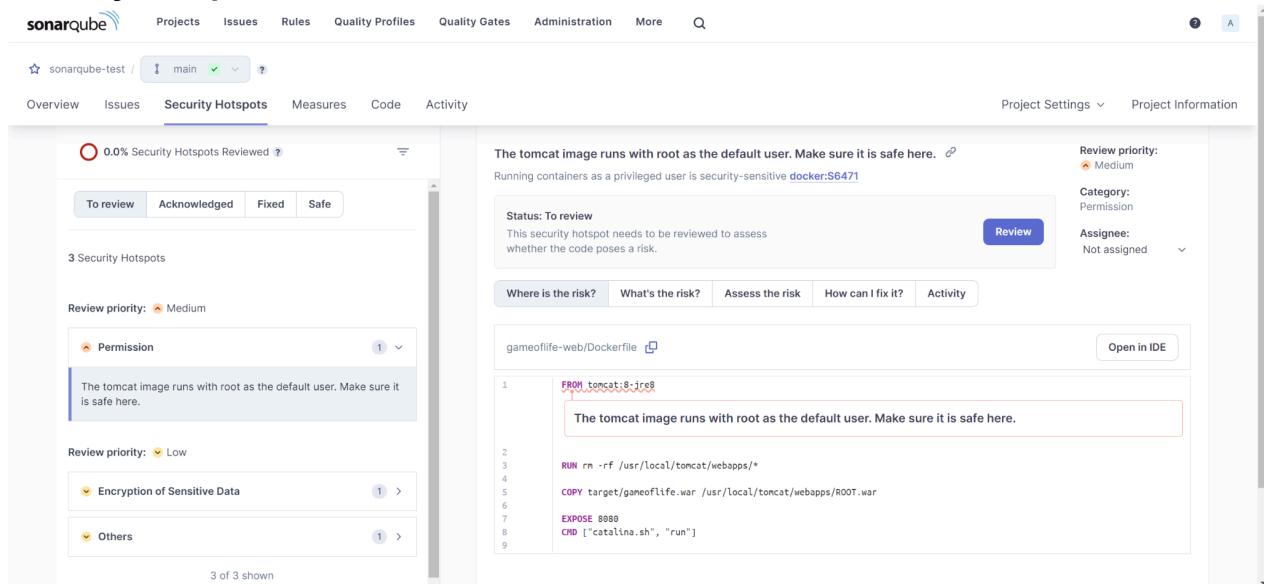
Duplicates



The screenshot shows the SonarQube interface for the 'sonarqube-test' project, specifically the 'Measures' tab under the 'Duplicates' category. The left sidebar displays a hierarchy of quality attributes: Project Overview, Security, Reliability, Maintainability, Security Review, and Duplications. Under Duplications, 'Density' is highlighted with 50.6%. The main area shows a table of duplicated lines for the project 'sonarqube-test'. The table has columns for 'Duplicated Lines (%)' and 'Duplicated Lines'. The data is as follows:

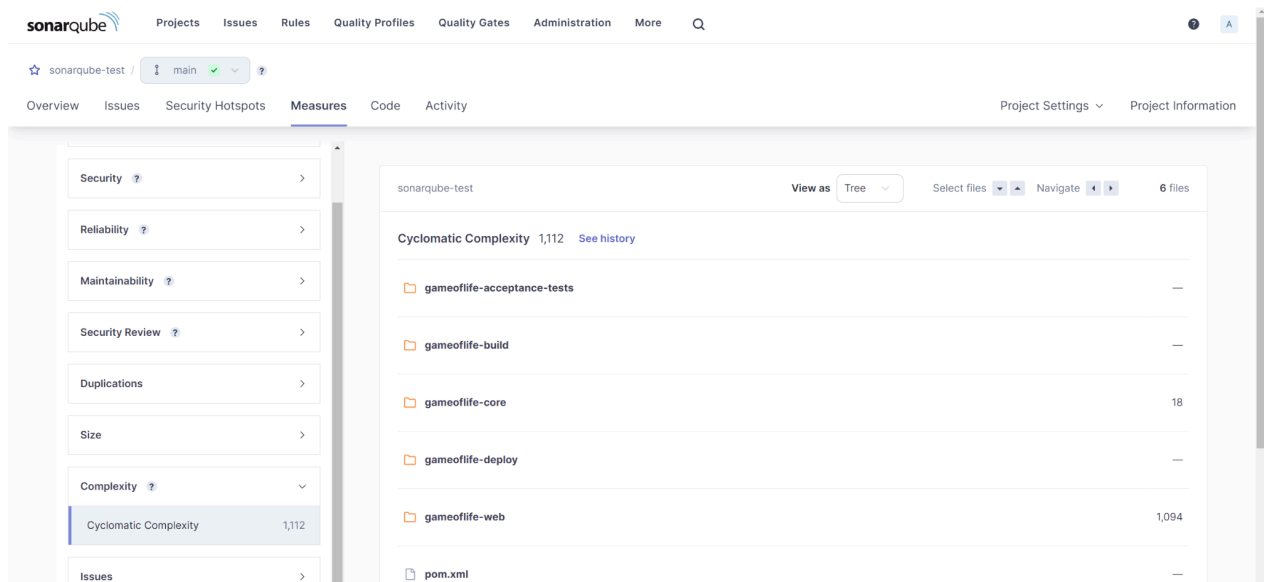
File	Duplicated Lines (%)	Duplicated Lines
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../ReportCellRenderer.html	92.4%	1,282
gameoflife-web/tools/jmeter/docs/api/org/apache/jo.../RightAlignRenderer.html	92.4%	1,198
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../JMeterCellRenderer.html	92.1%	1,281
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../FunctionHelper.html	89.5%	1,070
gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../AjpSamplerGui.html	89.0%	1,219

Security Hotspot



The screenshot shows the SonarQube interface for a project named 'sonarqube-test'. The 'Security Hotspots' tab is active, displaying a summary of 3 security hotspots. The first hotspot is titled 'Permission' with a 'Medium' review priority. The description states: 'The tomcat image runs with root as the default user. Make sure it is safe here.' The second hotspot is titled 'Encryption of Sensitive Data' with a 'Low' review priority. The third hotspot is titled 'Others' with a 'Low' review priority. The interface also shows a list of hotspots with their status (To review, Acknowledged, Fixed, Safe) and a 'Review' button. The 'Where is the risk?' tab is selected, showing the location of the risk in the code (gameoflife-web/Dockerfile). The code snippet shows the 'FROM' line for the tomcat image and the 'CMD' line for the container command.

Cyclomatic Complexity



The screenshot shows the SonarQube interface for a project named 'sonarqube-test'. The 'Measures' tab is active, displaying a list of measures. The 'Cyclomatic Complexity' measure is selected, showing a value of 1,112. The interface also shows a list of measures with their status (Security, Reliability, Maintainability, Security Review, Duplications, Size, Complexity, Issues) and a 'View as' dropdown menu. The 'View as' dropdown is set to 'Tree', showing a tree view of the project structure. The tree view shows the following structure:

- sonarqube-test
 - gameoflife-acceptance-tests
 - gameoflife-build
 - gameoflife-core
 - gameoflife-deploy
 - gameoflife-web
 - 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.