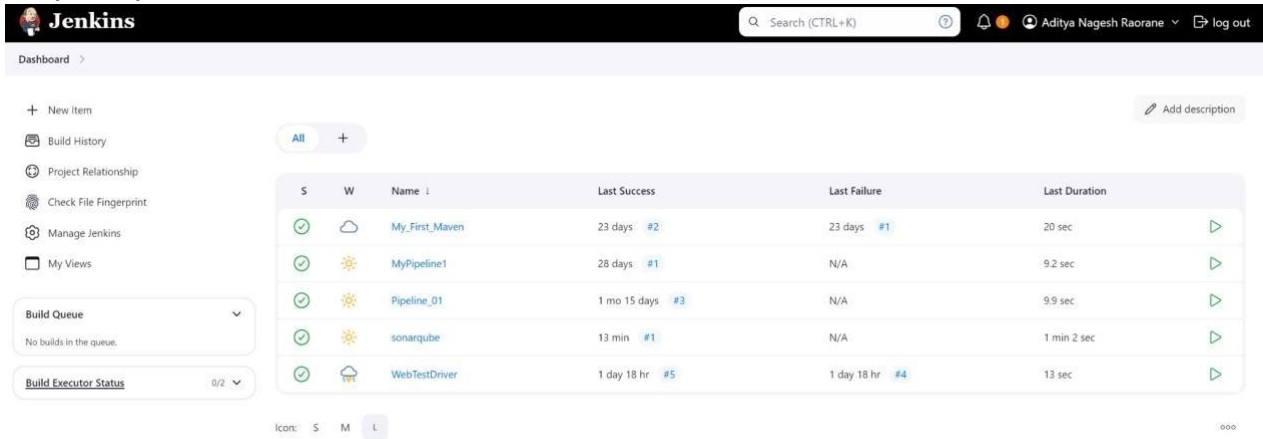


Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

1. Open up Jenkins Dashboard on localhost:8080.

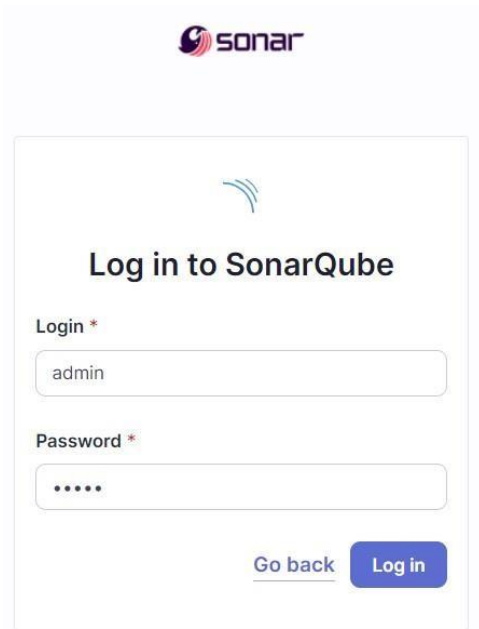


2. Run SonarQube in a Docker container using this command: a] `docker -v` b] `docker pull sonarqube` c] `docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest`

```
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 heading "Log in to SonarQube". Underneath the heading are two input fields: "Login *" with the text "admin" and "Password *" with five dots. At the bottom right 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.

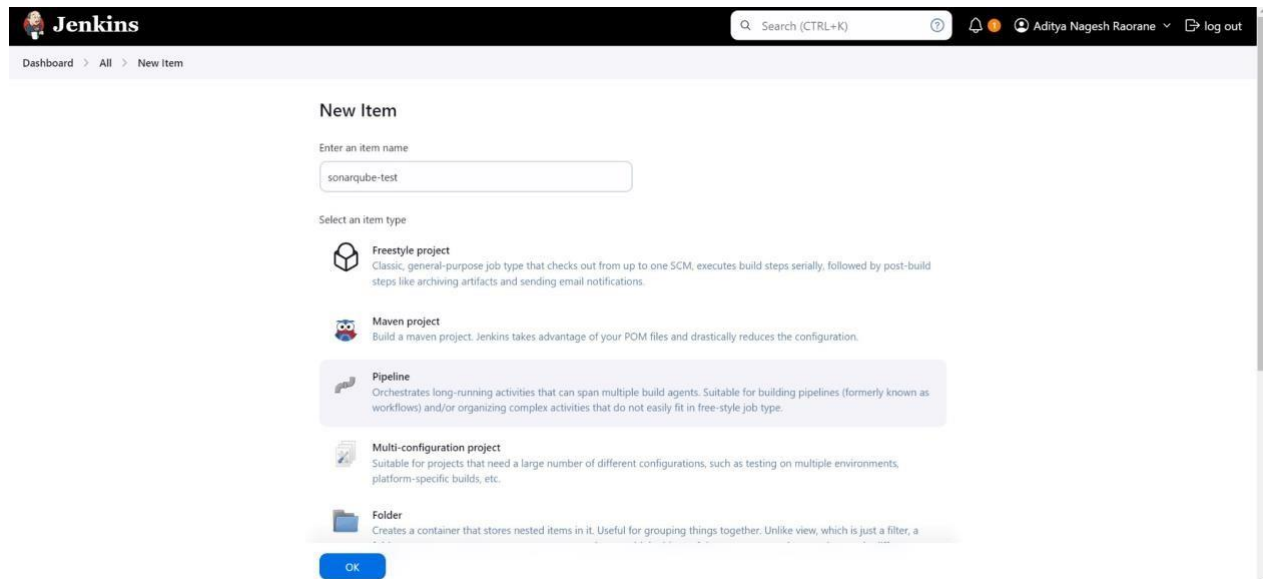
☐ Reference branch

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

Back Create project

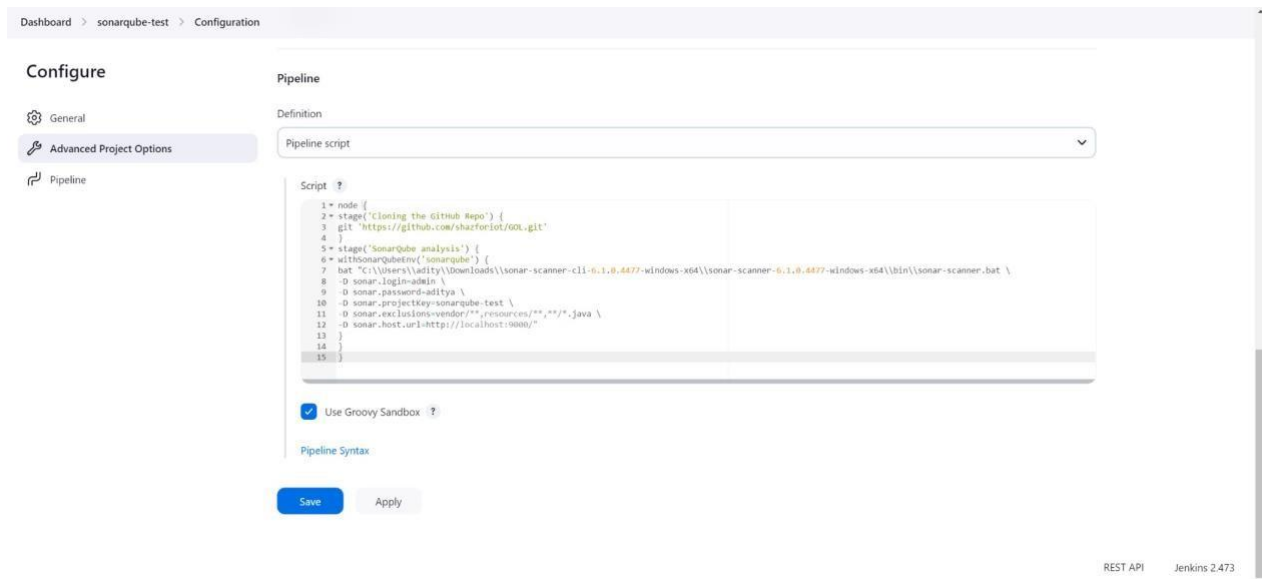
Setup the project and come back to Jenkins Dashboard.

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



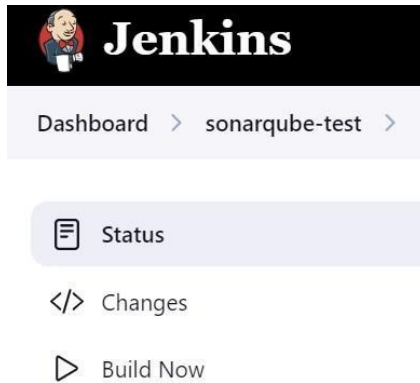
7. Under **Pipeline Script**, enter the following -

```
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/"
    }
}
}
```




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.


Jenkins

Dashboard > sonarqube-test >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline


🔍 Full Stage View

🌊 SonarQube

📁 Stages

✏️ Rename

❓ Pipeline Syntax


sonarqube-test

Stage View

	Cloning the GitHub Repo	SonarQube analysis
Average stage times:	1s	14min 37s
(Average <u>full</u> run time: ~14min 39s)		
#1 Sep 17 21:26 No Changes	1s	14min 37s


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

Builds

Filter

Today

 #1 9:26 PM


Jenkins

Search (CTRL+K)

THADANI NIKITA

log out

Dashboard > MyPipeline1 > Configuration

Status

</> Changes

Console Output

View as plain text

Edit Build Information

⌚ Timings

🔍 Git Build Data

🌐 Pipeline Overview

📄 Pipeline Console


🧵 Thread Dump

⏸️ Pause/resume

🔄 Replay

📋 Pipeline Steps

📁 Workspaces


Console Output

Started by user THADANI NIKITA

```
[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 SCH revision ID 'ba799ba7e1b576f04a612322b0412c5e6e4e5e4'
21:40:14.276 INFO Analysis report generated in 5151ms, dir size=127.2 MB
21:40:35.678 INFO Analysis report compressed in 21388ms, 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] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

REST API Jenkins 2.473

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 main content area shows a list of projects, with 'sonarqube-test' highlighted. The project details for 'sonarqube-test' are shown below, indicating a 'Passed' quality gate status. The project has 683k Lines of Code (HTML, XML, ...). The main branch is empty. The project is public and has a last analysis 16 minutes ago. The project is also shown in the 'My Favorites' list.

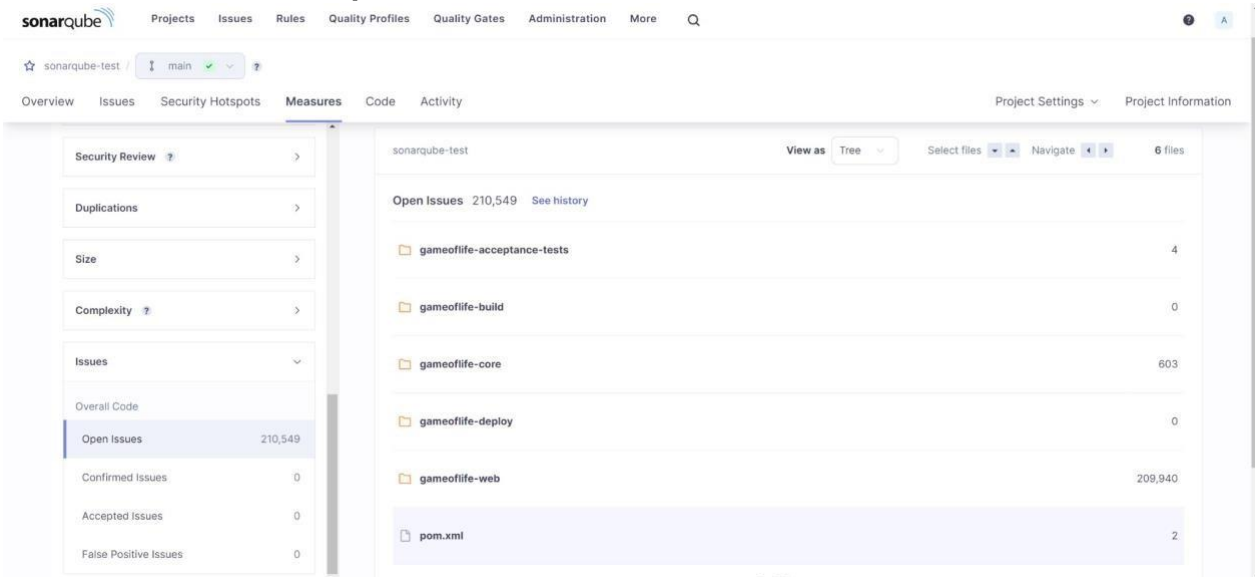
Below the project list, the 'sonarqube-test' project is selected, showing the 'main' branch. The project has 683k Lines of Code and a version not provided. The quality gate status is 'Passed' (green checkmark). The last analysis was 38 minutes ago. A warning message states: 'The last analysis has warnings. See details'.

The project details are organized into several sections:

- New Code** and **Overall Code** tabs are visible.
- Security**: 0 Open issues, 0 H, 0 M, 0 L.
- Reliability**: 68k Open issues, 0 H, 47k M, 21k L.
- Maintainability**: 164k Open issues, 7 H, 143k M, 21k L.
- Accepted issues**: 0 (Valid issues that were not fixed).
- Coverage**: On 0 lines to cover.
- Duplications**: 50.6% (On 759K lines).

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

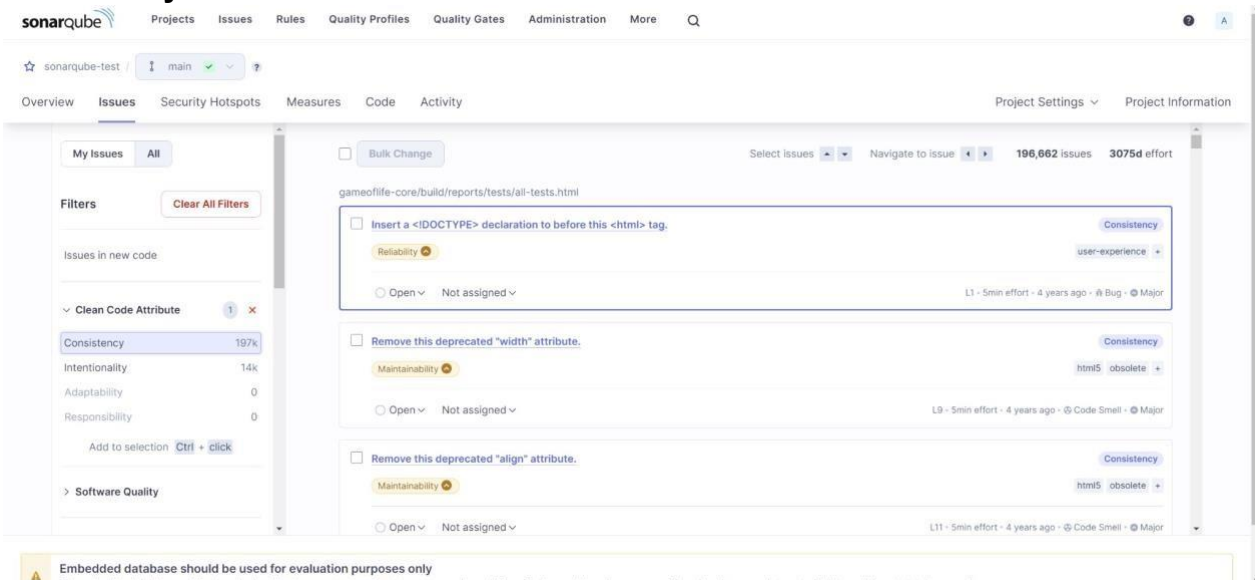
11. Code Problems Open Issues



The screenshot shows the SonarQube interface for a project named 'sonarqube-test'. The 'Measures' tab is selected, displaying a list of metrics on the left and a tree view of the project structure on the right. The 'Open Issues' metric is highlighted, showing 210,549 issues. The tree view shows the following structure:

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

Consistency



The screenshot shows the SonarQube interface for the same project, now with the 'Issues' tab selected. The 'Consistency' issue category is highlighted in the left sidebar, showing 197k issues. The main area displays a list of issues, including:

- Insert a <DOCTYPE> declaration to before this <html> tag.** (Reliability, L1 - 5min effort - 4 years ago - Bug - Major)
- Remove this deprecated "width" attribute.** (Maintainability, L9 - 5min effort - 4 years ago - Code Smell - Major)
- Remove this deprecated "align" attribute.** (Maintainability, L11 - 5min effort - 4 years ago - Code Smell - Major)

A warning message at the bottom states: "Embedded database should be used for evaluation purposes only".

Intentionality

The screenshot shows the SonarQube web interface for a project named 'sonarqube-test'. The 'Issues' tab is selected, and the 'Intentionality' filter is applied. The left sidebar shows a list of filters, including 'Clean Code Attribute' with 'Intentionality' selected, showing 14k issues. The main area displays a list of issues, including 'Use a specific version tag for the image.' and 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' The issues are categorized by 'Intentionality' and 'Maintainability'. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

Code Smells

The screenshot shows the SonarQube web interface for the same project 'sonarqube-test'. The 'Issues' tab is selected, and the 'Code Smell' filter is applied. The left sidebar shows a list of filters, including 'Severity' and 'Type', with 'Code Smell' selected, showing 253 issues. The main area displays a list of issues, including 'Add an "alt" attribute to this image.' and 'Add an "alt" attribute to this image.' The issues are categorized by 'Intentionality' and 'Reliability'. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

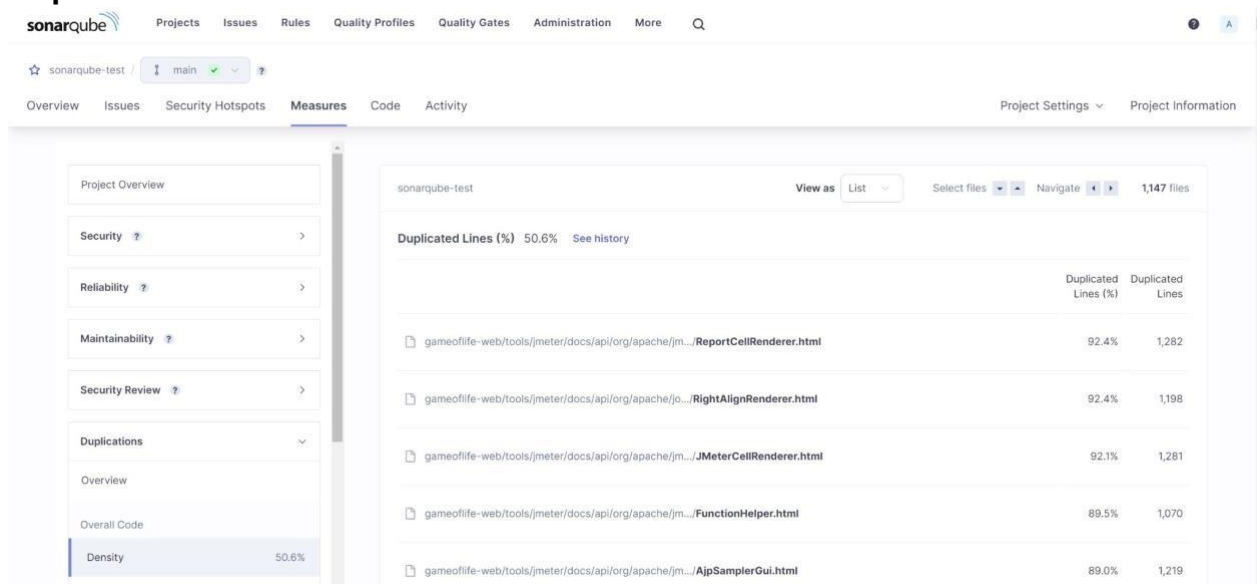
Bugs

The screenshot shows the SonarQube web interface with the 'Issues' tab selected. The left sidebar displays a filter for 'Bug' type issues, showing 14k results. The main panel shows a list of issues, including 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' and 'Add "<th>" headers to this "<table>"'. Each issue is categorized by 'Intentionality' (Reliability) and 'accessibility' (wcag2-a). The interface also shows a 'Bulk Change' button and a 'Select issues' dropdown. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

Reliability

The screenshot shows the SonarQube web interface with the 'Issues' tab selected. The left sidebar displays a filter for 'Reliability' type issues, showing 14k results. The main panel shows a list of issues, including 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' and 'Add "<th>" headers to this "<table>"'. Each issue is categorized by 'Intentionality' (Reliability) and 'accessibility' (wcag2-a). The interface also shows a 'Bulk Change' button and a 'Select issues' dropdown. A warning message at the bottom states: 'Embedded database should be used for evaluation purposes only'.

Duplicates

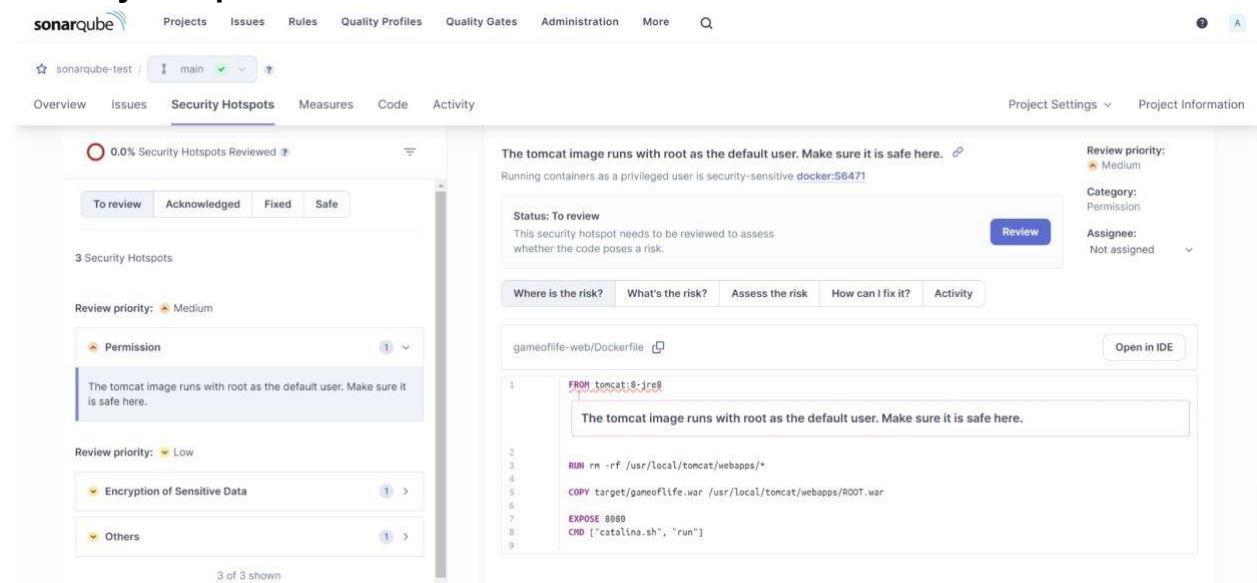


The screenshot shows the SonarQube interface for the 'sonarqube-test' project. The 'Measures' tab is selected, displaying a table of duplicated lines. The table has columns for 'Duplicated Lines (%)' and 'Duplicated Lines'. The data is as follows:

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/jm.../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

On the left sidebar, the 'Duplications' section shows an 'Overview' and 'Overall Code' with a 'Density' of 50.6%.

Security Hotspot



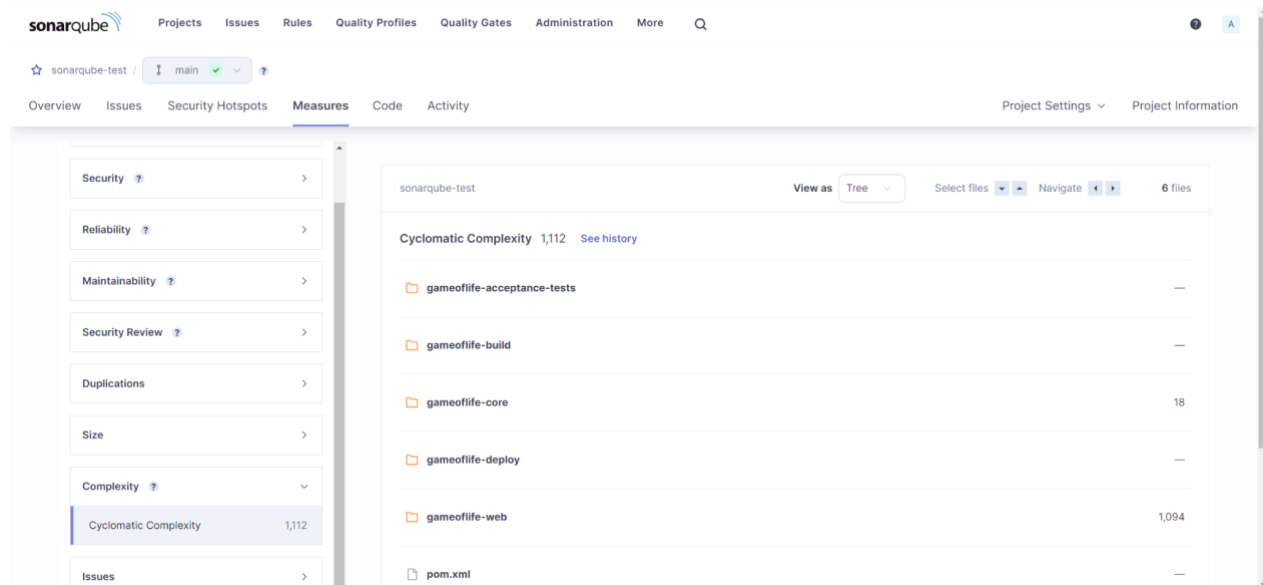
The screenshot shows the SonarQube interface for the 'sonarqube-test' project, specifically the 'Security Hotspots' tab. A security hotspot is displayed with the following details:

- Review priority:** Medium
- Category:** Permission
- Assignee:** Not assigned
- Status:** To review
- Description:** The tomcat image runs with root as the default user. Make sure it is safe here.
- Where is the risk?** gameoflife-web/Dockerfile
- Code Snippet:**

```
1 FROM tomcat:8-jre8
2
3 RUN rm -rf /usr/local/tomcat/webapps/*
4
5 COPY target/gameoflife.war /usr/local/tomcat/webapps/ROOT.war
6
7 EXPOSE 8080
8 CMD ["catalina.sh", "run"]
```

On the left sidebar, the 'Security Hotspots' section shows a list of 3 security hotspots, with the first one being 'Permission' with a 'Review priority' of Medium.

Cyclomatic Complexity



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.