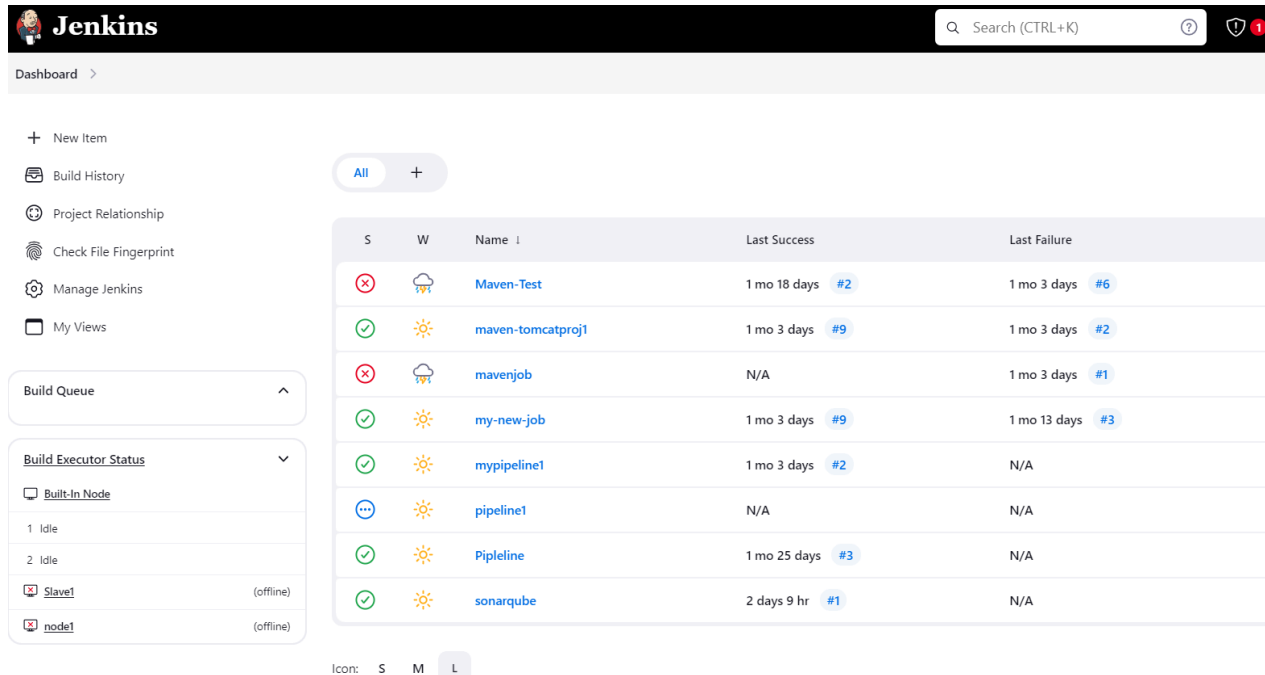


## EXPERIMENT 8

**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. dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

1. Open Jenkin dashboard.



The screenshot shows the Jenkins Dashboard. 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 (empty) and Build Executor Status (showing 2 idle nodes: Slave1 and node1, both offline). The main area displays a table of builds with columns: S (Status), W (Icon), Name, Last Success, and Last Failure. The table lists several builds, including Maven-Test, maven-tomcatproj1, mavenjob, my-new-job, mypipeline1, pipeline1, Pipeline, and sonarqube. The sonarqube build is the most recent, showing a success status and a last success time of 2 days 9 hr.

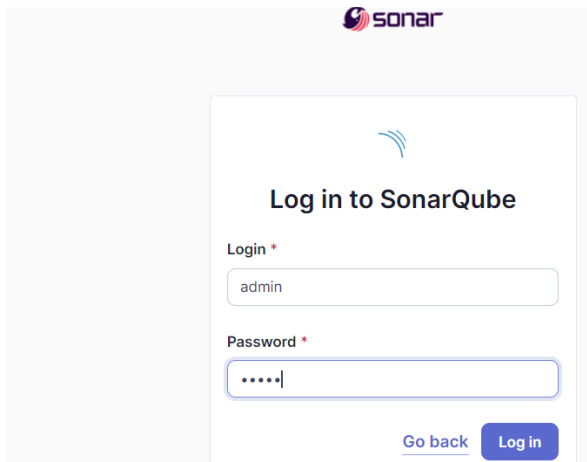
S	W	Name	Last Success	Last Failure
❌	☁️	Maven-Test	1 mo 18 days #2	1 mo 3 days #6
✅	☀️	maven-tomcatproj1	1 mo 3 days #9	1 mo 3 days #2
❌	☁️	mavenjob	N/A	1 mo 3 days #1
✅	☀️	my-new-job	1 mo 3 days #9	1 mo 13 days #3
✅	☀️	mypipeline1	1 mo 3 days #2	N/A
⋮	☀️	pipeline1	N/A	N/A
✅	☀️	Pipeline	1 mo 25 days #3	N/A
✅	☀️	sonarqube	2 days 9 hr #1	N/A

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

```
C:\Users\91900>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest d23accacd96c274f5f87912674ecf2d9adffff185a940c24740f44b29534485
```

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*.



The image shows the SonarQube login interface. At the top is the Sonar logo. Below it is a 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" (blue text) and "Log in" (blue button).

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

1 of 2

## Create a local project

Project display name \*

sonarqube-2



Project key \*

sonarqube-2



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

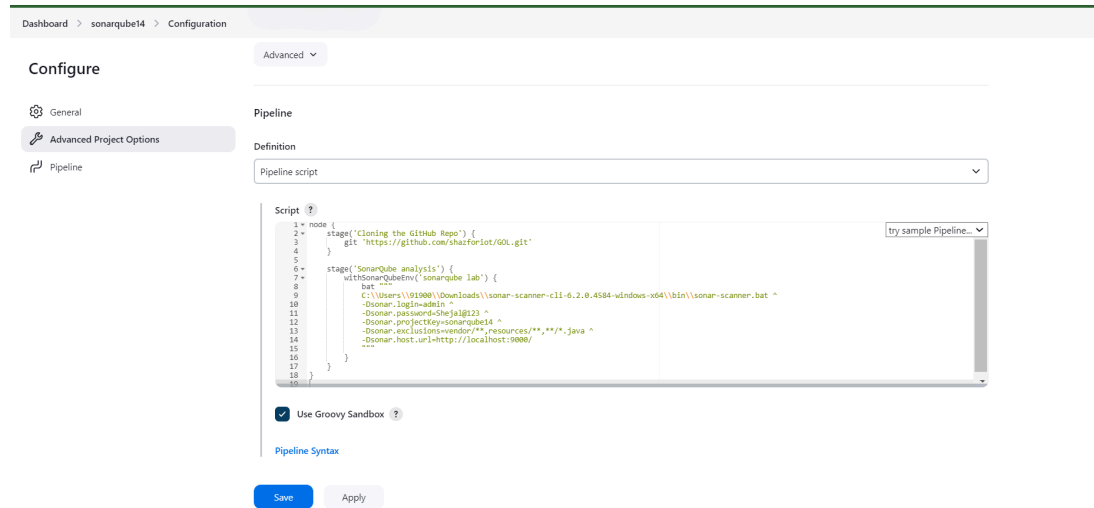
Select an item type

-  **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.
-  **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**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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') {
      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/"
    }
  }
}
```

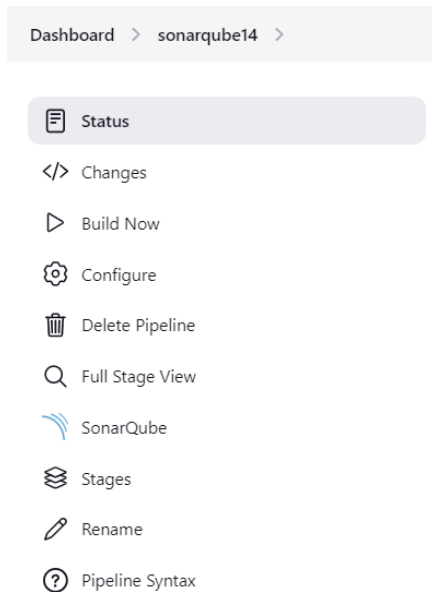


The screenshot shows the SonarQube Configuration page for a Pipeline script. The breadcrumb navigation at the top reads "Dashboard > sonarqube14 > Configuration". The left sidebar has three tabs: "General", "Advanced Project Options" (which is selected), and "Pipeline". The main content area is titled "Configure" and has a sub-tab "Advanced". Under the "Pipeline" section, there is a "Definition" dropdown menu set to "Pipeline script". Below this is a "Script" editor with a Groovy script. The script starts with a "node" stage for cloning a GitHub repository, followed by a "SonarQube analysis" stage that uses the "withSonarQubeEnv" function to set up the environment. The environment variables include the SonarQube URL, login, password, project key, and exclusions. The script ends with a "bat" stage. At the bottom of the script editor, there is a checkbox for "Use Groovy Sandbox" which is checked. A "Pipeline Syntax" link is also present. At the very bottom, there are "Save" and "Apply" buttons.


```
1 node {  
2   stage('Cloning the GitHub Repo') {  
3     git 'https://github.com/shuforiot/GOI.git'  
4   }  
5  
6   stage('SonarQube analysis') {  
7     withSonarQubeEnv('sonarqube14') {  
8       bat '''  
9         C:\Users\91000\Downloads\sonar-scanner-c11-6.2.0.4584-windows-x64\bin\sonar-scanner.bat ^  
10        -Dsonar.login=admin ^  
11        -Dsonar.password=Sheja@123 ^  
12        -Dsonar.projectkey=sonarqube14 ^  
13        -Dsonar.exclusions=vendor/**,resources/**,**.java ^  
14        -Dsonar.host.url=http://localhost:9000/  
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.



The screenshot shows the SonarQube Pipeline Actions menu. The breadcrumb navigation at the top reads "Dashboard > sonarqube14 >". The menu is titled "Status" and contains the following actions: "Changes", "Build Now", "Configure", "Delete Pipeline", "Full Stage View", "SonarQube", "Stages", "Rename", and "Pipeline Syntax".

**Jenkins**

Dashboard > sonarqube14 >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

🌊 SonarQube

📁 Stages

✎ Rename

❓ Pipeline Syntax

Stage View

Average stage times:  
(Average full run time: ~25min 42s)

	Cloning the GitHub Repo	SonarQube Analysis
#11 Sep 27 21:03 No Changes	3s	25min 37s
#10 Sep 27 21:00 No Changes	3s	1min 13s aborted
#9 Sep 27 20:54 No Changes	2s	1s failed
#8 Sep 27 20:52 No Changes	1s	83ms failed
#7 Sep 27 20:51 No Changes	3s	204ms failed

Build History

trend

Filter...

#11  
Sep 27, 2024, 9:03 PM

#10  
Sep 27, 2024, 9:00 PM

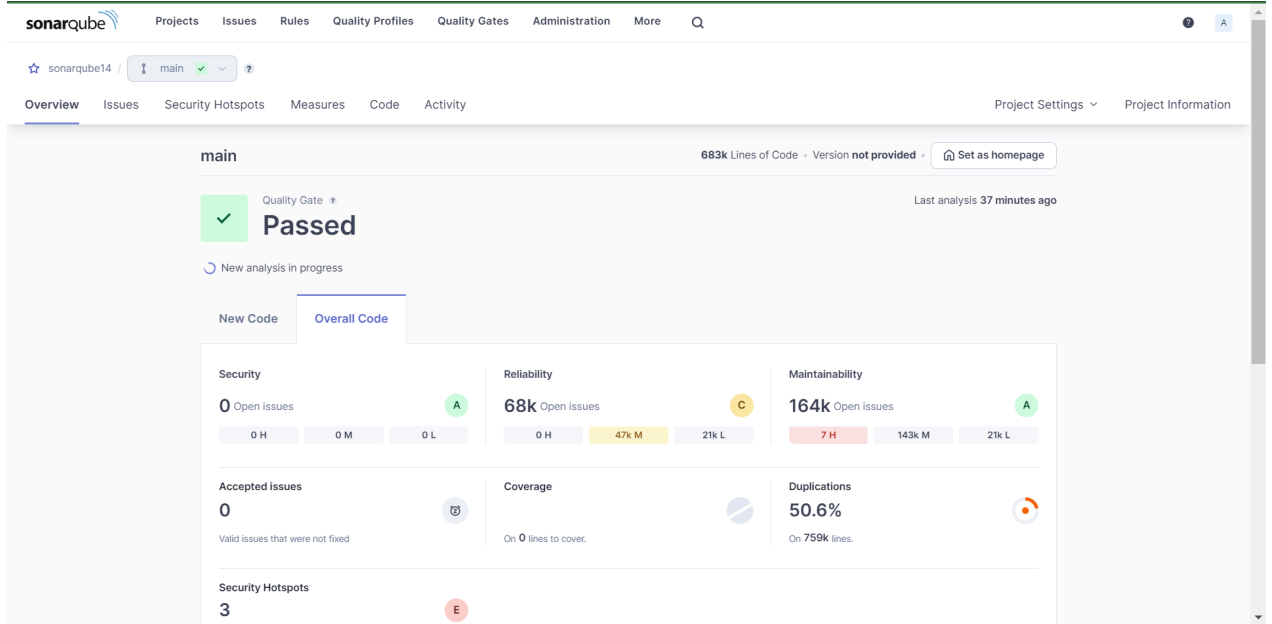
#9  
Sep 27, 2024, 8:54 PM

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

```
Dashboard > sonarqube14 > #11

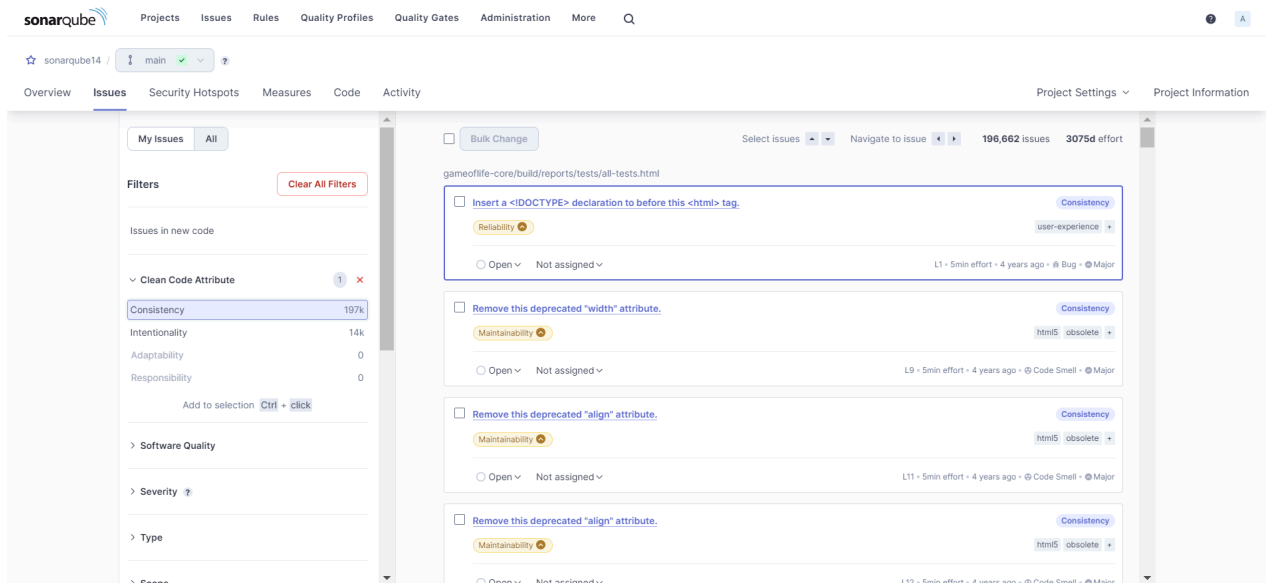
21:22:54.863 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 75. Keep only the first 100 references.
21:22:54.863 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 41. Keep only the first 100 references.
21:22:54.864 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 17. Keep only the first 100 references.
21:22:54.864 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 185. Keep only the first 100 references.
21:22:54.864 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 185. Keep only the first 100 references.
21:22:54.864 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 550. Keep only the first 100 references.
21:22:54.864 WARN Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/BSFListener.html for block at line 75. Keep only the first 100 references.
21:22:54.868 INFO CPD Executor CPD calculation finished (done) | time=443308ms
21:22:55.007 INFO SCH revision ID "ba799ba7e1b576f0ba461232b6412c6e6e1e5e4"
21:27:00.007 INFO Analysis report generated in 5834ms, dir-size=127.2 MB
21:27:38.158 INFO Analysis report compressed in 30044ms, zip size=29.6 MB
21:27:52.892 INFO Analysis report uploaded in 14680ms
21:27:52.947 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube14
21:27:52.947 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:27:52.947 INFO More about the report processing at http://127.0.0.1:9000/api/cv/task?id=d45a94f-fd29-48d5-bfcb-78c329f095f9
21:28:36.635 INFO Analysis total time: 25:29.835 s
21:28:36.702 INFO SonarScanner Engine completed successfully
21:28:37.637 INFO EXECUTION SUCCESS
21:28:38.003 INFO Total time: 25:29.478s
[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 - Consistency



## Intentionality

The screenshot displays the SonarQube web interface for project 'sonarqube14'. The 'Issues' tab is active, showing a list of 13,887 issues with a total effort of 59d. The left sidebar contains filters for 'Clean Code Attribute' (Consistency: 197k, Intentionality: 14k, Adaptability: 0, Responsibility: 0) and 'Severity' (High: 7, Medium: 189k, Low: 21k). The main panel shows a list of issues under the 'Intentionality' category. The first issue is 'Use a specific version tag for the image.' with a 'Maintainability' rating and 'No tags'. The second issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' rating and 'No tags'. The third issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' rating and 'No tags'. The fourth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' rating and 'No tags'.

## Severity

The screenshot displays the SonarQube web interface for project 'sonarqube14'. The 'Issues' tab is active, showing a list of 7 issues with a total effort of 14min. The left sidebar contains filters for 'Severity' (High: 7, Medium: 189k, Low: 21k) and 'Type' (Type: 7, Scope: 189k, Status: 21k, Security Category: 189k, Creation Date: 21k, Language: 189k). The main panel shows a list of issues under the 'Severity' category. The first issue is 'Add the "let", "const" or "var" keyword to this declaration of "prop" to make it explicit.' with a 'Maintainability' rating and 'No tags'. The second issue is 'Add the "let", "const" or "var" keyword to this declaration of "prop" to make it explicit.' with a 'Maintainability' rating and 'No tags'. The third issue is 'Add the "let", "const" or "var" keyword to this declaration of "prop" to make it explicit.' with a 'Maintainability' rating and 'No tags'. The fourth issue is 'Add the "let", "const" or "var" keyword to this declaration of "prop" to make it explicit.' with a 'Maintainability' rating and 'No tags'.

## Bugs and Code Smells

The screenshot shows the SonarQube interface for project 'sonarqube14' on the 'main' branch. The 'Issues' tab is active, displaying a list of issues on the left and a detailed view of three issues on the right.

**Left Panel (Issues List):**

- Medium: 47k
- Low: 3
- Type: 1 (Bug icon)
- Bug: 47k
- Vulnerability: 0
- Code Smell: 164k
- Scope: Add to selection Ctrl + click
- Status: Add to selection Ctrl + click
- Security Category: Add to selection Ctrl + click

**Right Panel (Issue Details):**

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

- ☐ Bulk Change
- Select issues: [dropdown]
- Navigate to issue: [dropdown]
- 46,515 issues 1426d effort

**Issue 1:** Add "lang" and/or "xml:lang" attributes to this "<html>" element. Intentionality. Reliability. accessibility wcag2-a. L1 - 2min effort - 4 years ago - Bug - Major.

**Issue 2:** Insert a <!DOCTYPE> declaration to before this <html> tag. Consistency. Reliability. user-experience. L1 - 5min effort - 4 years ago - Bug - Major.

**Issue 3:** Add "<th>" headers to this "<table>". Intentionality.

The screenshot shows the SonarQube interface for project 'sonarqube14' on the 'main' branch. The 'Issues' tab is active, displaying a list of issues on the left and a detailed view of three issues on the right.

**Left Panel (Issues List):**

- Medium: 143k
- Low: 21k
- Type: 1 (Bug icon)
- Bug: 47k
- Vulnerability: 0
- Code Smell: 164k
- Scope: Add to selection Ctrl + click
- Status: Add to selection Ctrl + click
- Security Category: Add to selection Ctrl + click

**Right Panel (Issue Details):**

gameoflife-acceptance-tests/Dockerfile

- ☐ Bulk Change
- Select issues: [dropdown]
- Navigate to issue: [dropdown]
- 164,034 issues 1708d effort

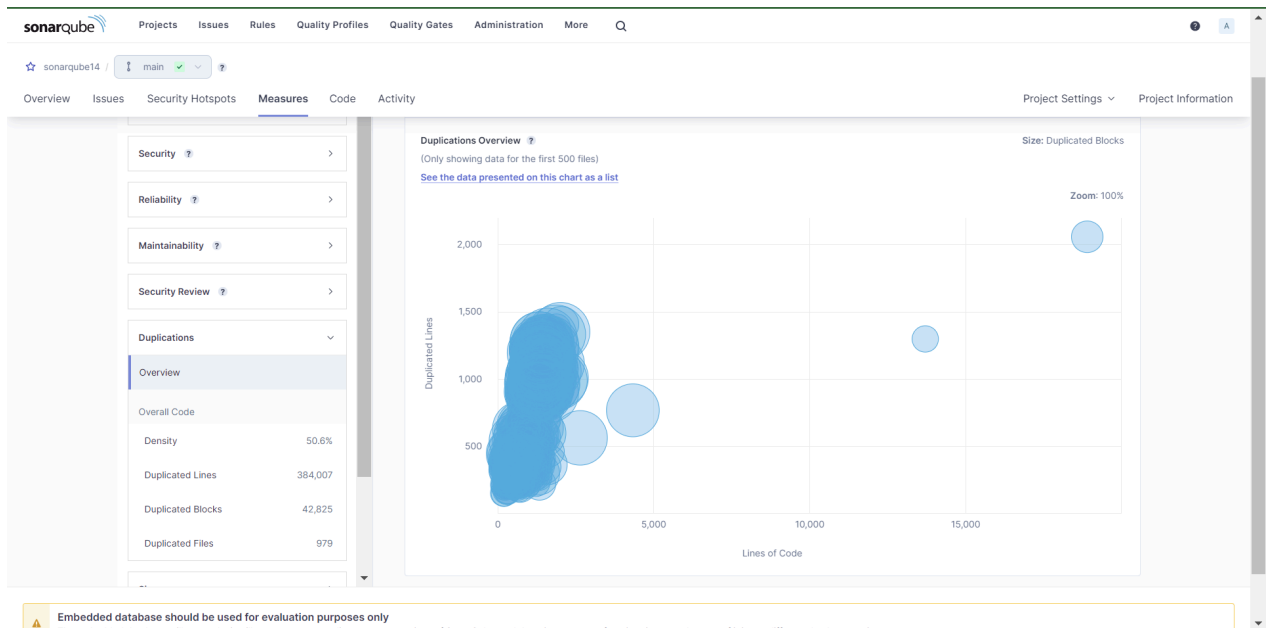
**Issue 1:** Use a specific version tag for the image. Intentionality. Maintainability. No tags. L1 - 5min effort - 4 years ago - Code Smell - Major.

**Issue 2:** Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality. Maintainability. No tags. L12 - 5min effort - 4 years ago - Code Smell - Major.

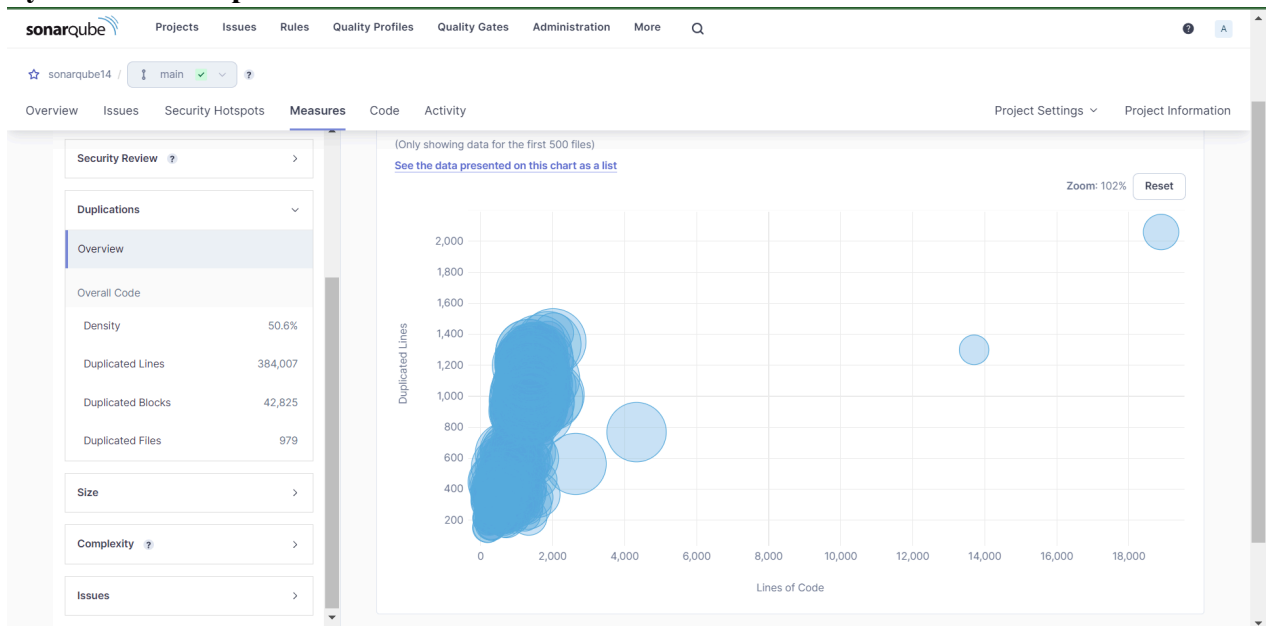
**Issue 3:** Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality.



## Duplicates



## Cyclomatic Complexities



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 successfully cloned a GitHub repository and integrated it with SonarQube for code analysis. During the analysis, SonarQube highlighted various types of program issues, including:

- **Consistency:** Code standards and formatting issues.
- **Intentionality:** Potential logical or structural errors in code.
- **Severity:** Classification of errors by criticality.
- **Duplicates:** Identification of duplicate code sections.
- **Cyclomatic Complexity:** Measurement of code complexity based on control flow.

These insights provided a comprehensive understanding of the code quality and highlighted areas that needed improvement.

### **Issues Faced:**

1. **SonarQube Scanner Path Error:** The Jenkins pipeline script was initially unable to run correctly due to Jenkins not detecting the correct path for the SonarQube Scanner. This required manual intervention to modify the script and set the proper path for the scanner bash file, allowing the pipeline to function as expected.