

AdvanceDevops 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's 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 two idle built-in nodes and two offline slave nodes (Slave1 and node1). The main area displays a table of build history for various jobs.

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

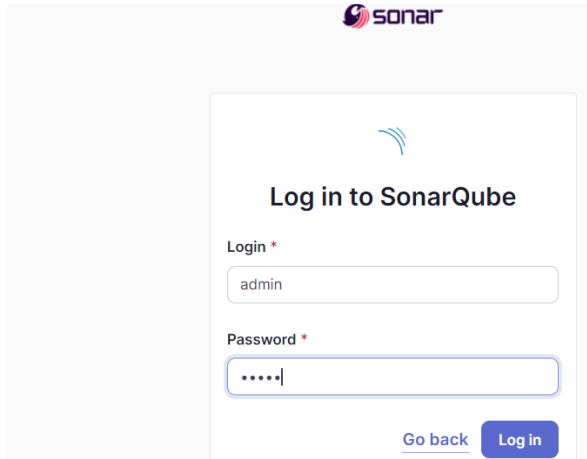
At the bottom, there's a filter for 'Icon' with options S, M, and L.

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

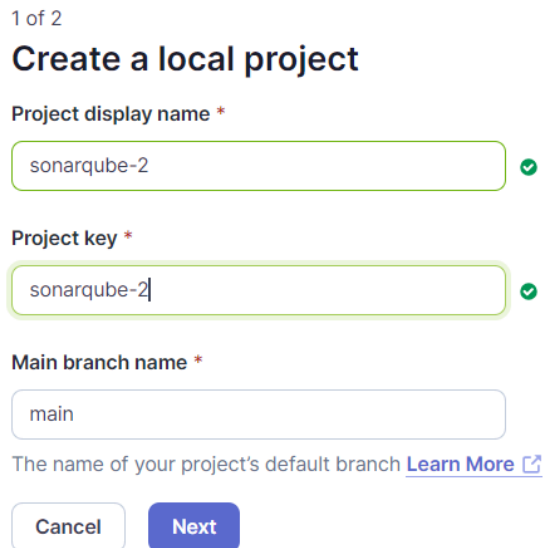
```
C:\Users\91900>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest d23accacd96c274f5f87912674ecf2d9adffff185a940c24740f44b29534485
```

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, there is a Sonar logo. Below it, the text "Log in to SonarQube" is displayed. There are two input fields: "Login *" with the value "admin" and "Password *" with masked characters "*****". At the bottom, there are two buttons: "Go back" (a link) and "Log in" (a button).

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

The image shows the "Create a local project" form in SonarQube. It is labeled "1 of 2". The form has three main sections: "Project display name *" with the value "sonarqube-2" and a green checkmark; "Project key *" with the value "sonarqube-2" and a green checkmark; and "Main branch name *" with the value "main". Below the last section, there is a note: "The name of your project's default branch [Learn More](#)". At the bottom, there are two buttons: "Cancel" and "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

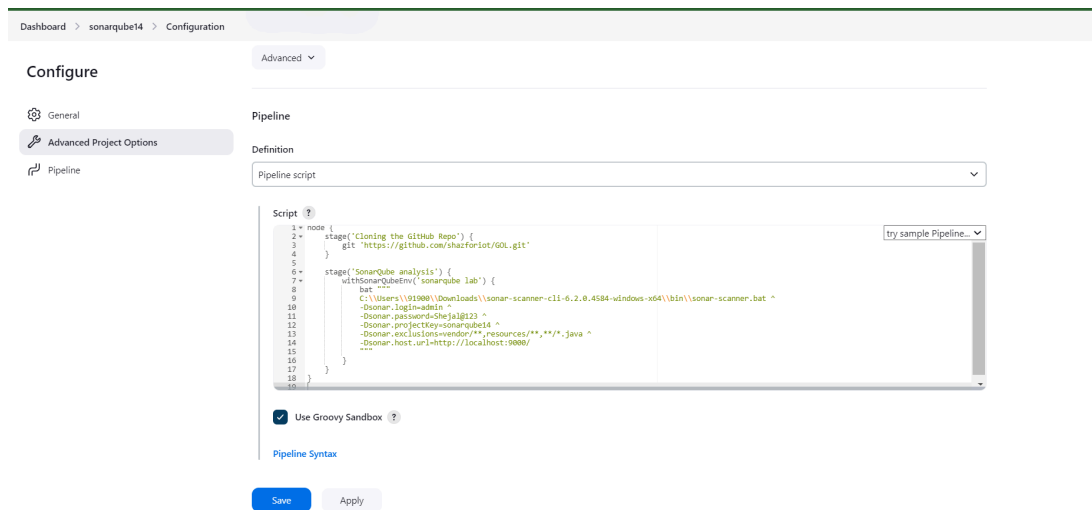
Select an item type

-  **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

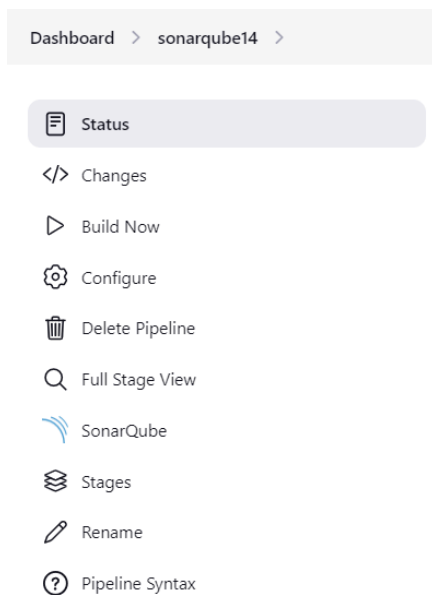
7. Under Pipeline Script, enter the following -


```
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/"
    }
  }
}
```



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

8. Run The Build.



**Jenkins**

Dashboard > sonarqube14 >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

🌊 SonarQube

📁 Stages

✎ Rename

❓ Pipeline Syntax

Stage View

	Cloning the GitHub Repo	SonarQube Analysis
Average stage times: (Average full run time: ~25min 42s)	2s	5min 22s
#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

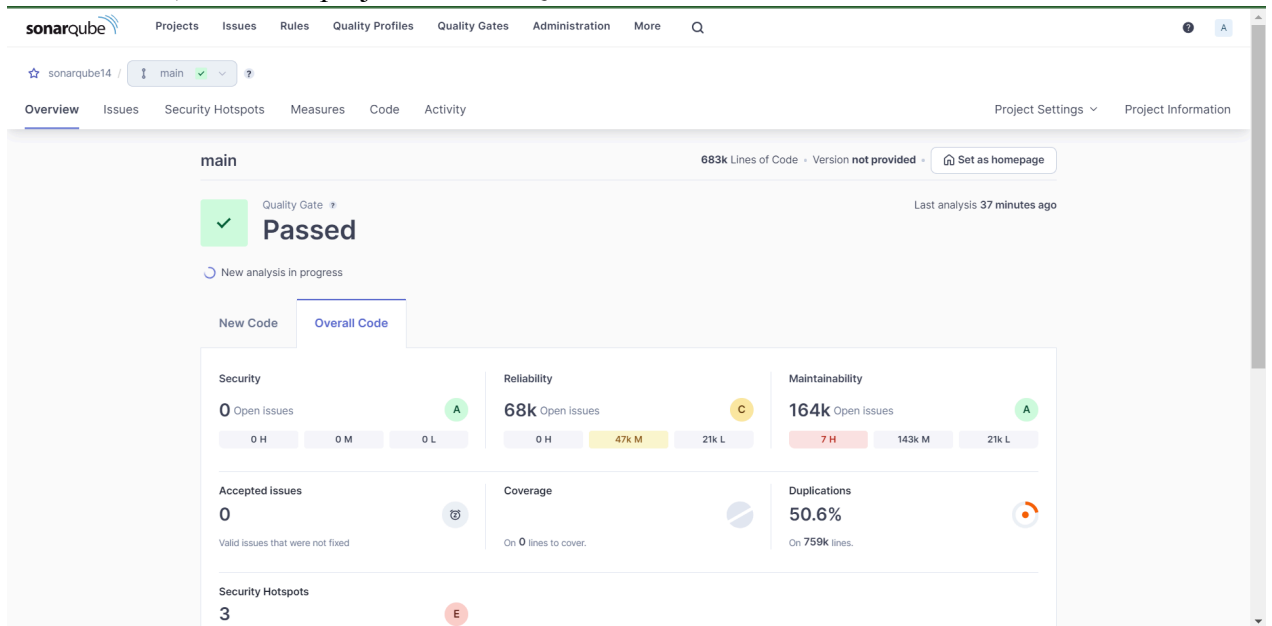
#9
Sep 27, 2024, 8:54 PM

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/BSFLListener.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/BSFLListener.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/BSFLListener.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/BSFLListener.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/BSFLListener.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/BSFLListener.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/BSFLListener.html for block at line 75. Keep only the first 100 references.
21:22:55.007 INFO CPD Executor CPD calculation finished (done) | time=443308ms
21:22:55.007 INFO SCM revision ID 'ba7999a7e1b576f04a61232b0412c56e1e5e4'
21:27:08.007 INFO Analysis report generated in 5834ms, dir size=127.2 MB
21:27:58.158 INFO Analysis report compressed in 38004ms, zip size=29.6 MB
21:27:52.802 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 Here about the report processing at http://127.0.0.1:9000/api/cx/task?id=d45a90f-fd29-48d5-bfcb-78c329f085f9
21:28:36.635 INFO Analysis total time: 25:19.835 s
21:28:36.702 INFO SonarScanner Engine completed successfully
21:28:37.627 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
```

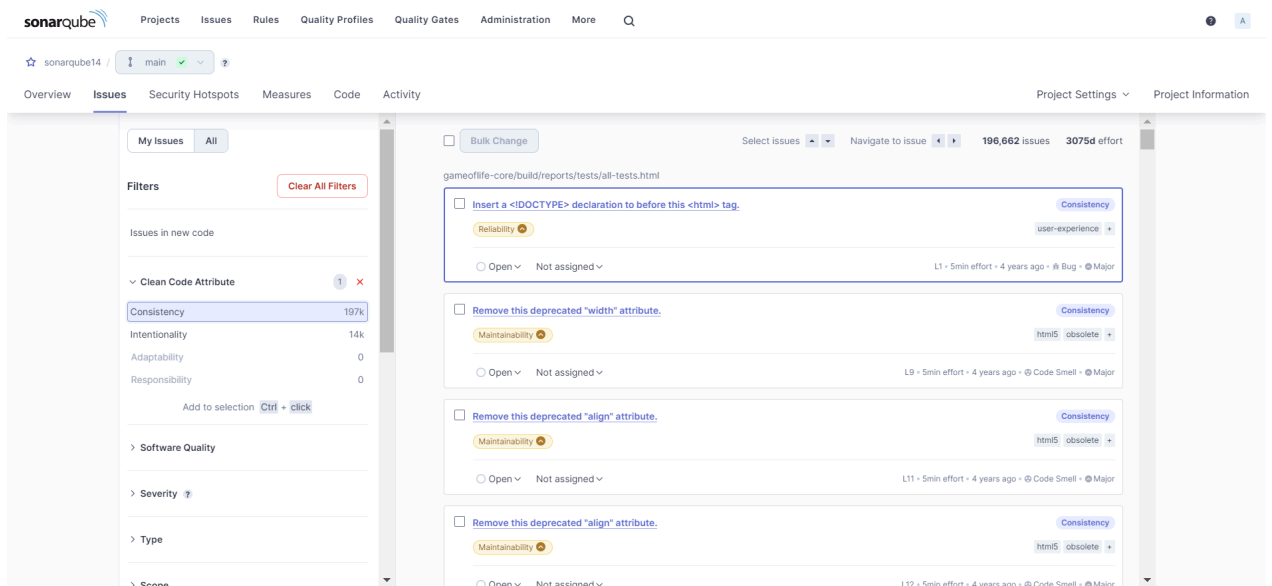
10. After that, check the project in SonarQube.



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

11. Code Problems -

Consistency

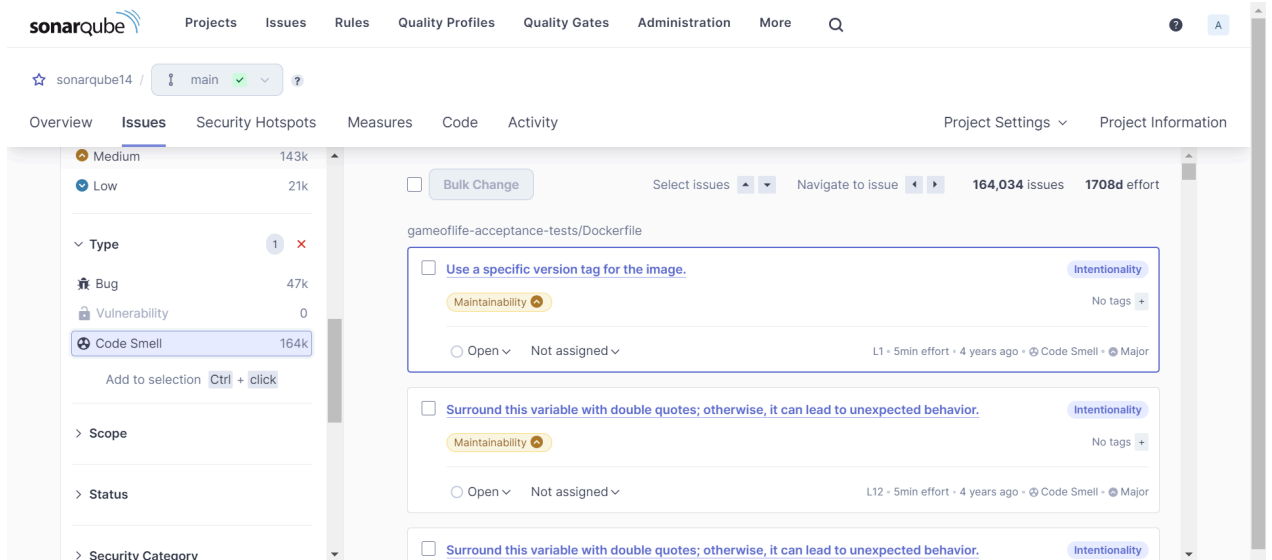


Intentionality

The screenshot displays the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The main header shows 'sonarqube14' and a dropdown menu with 'main' and a green checkmark. The left sidebar contains tabs for 'Overview', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity'. The 'Issues' tab is active, showing a list of filters on the left: 'My Issues', 'All', 'Issues in new code', 'Clean Code Attribute' (14k), 'Consistency' (197k), 'Intentionality' (14k), 'Adaptability' (0), 'Responsibility' (0), 'Software Quality', 'Severity', 'Type', and 'Scope'. The main content area shows a list of issues under the 'Intentionality' filter. The first issue is 'Use a specific version tag for the image.' with a 'Maintainability' tag. The second issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The third issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The fourth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The fifth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The sixth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The seventh issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The eighth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The ninth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The tenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The eleventh issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The twelfth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The thirteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The fourteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The fifteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The sixteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The seventeenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The eighteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The nineteenth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The twentieth issue is 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' with a 'Maintainability' tag. The total number of issues is 13,887 and the total effort is 59d.

Bugs and Code Smells

The screenshot displays the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The main header shows 'sonarqube14' and a dropdown menu with 'main' and a green checkmark. The left sidebar contains tabs for 'Overview', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity'. The 'Issues' tab is active, showing a list of filters on the left: 'My Issues', 'All', 'Issues in new code', 'Clean Code Attribute' (14k), 'Consistency' (197k), 'Intentionality' (14k), 'Adaptability' (0), 'Responsibility' (0), 'Software Quality', 'Severity', 'Type', and 'Scope'. The main content area shows a list of issues under the 'Bugs and Code Smells' filter. The first issue is 'Add "lang" and/or "xml:lang" attributes to this "<html>" element' with a 'Reliability' tag. The second issue is 'Insert a <IDOCYTYPE> declaration to before this <html> tag.' with a 'Consistency' tag. The third issue is 'Add "<th>" headers to this "<table>." with a 'Reliability' tag. The total number of issues is 46,515 and the total effort is 1426d.

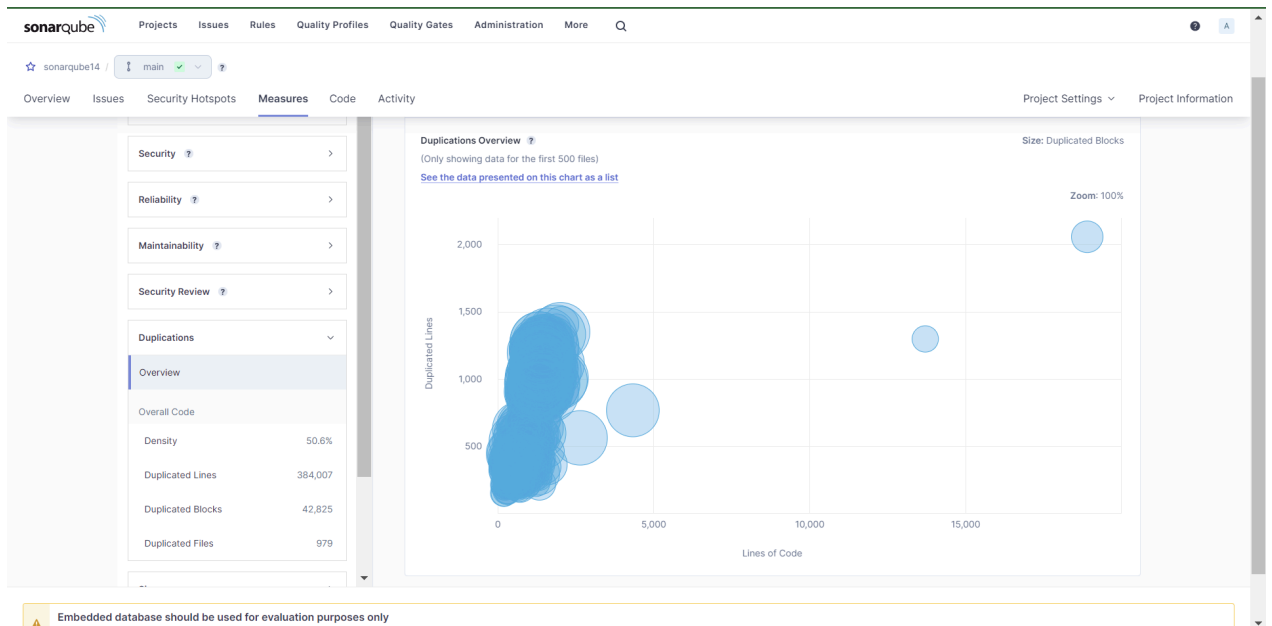


The screenshot shows the SonarQube interface for the 'sonarqube14' project. The 'Issues' tab is active, displaying a list of issues. On the left, a sidebar shows filters for 'Type' (Bug, Vulnerability, Code Smell) and 'Scope'. The main area shows a list of issues, including 'Use a specific version tag for the image' and 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior'. The interface includes a 'Bulk Change' button and a 'Navigate to issue' dropdown. The total number of issues is 164,034, and the total effort is 1708d.

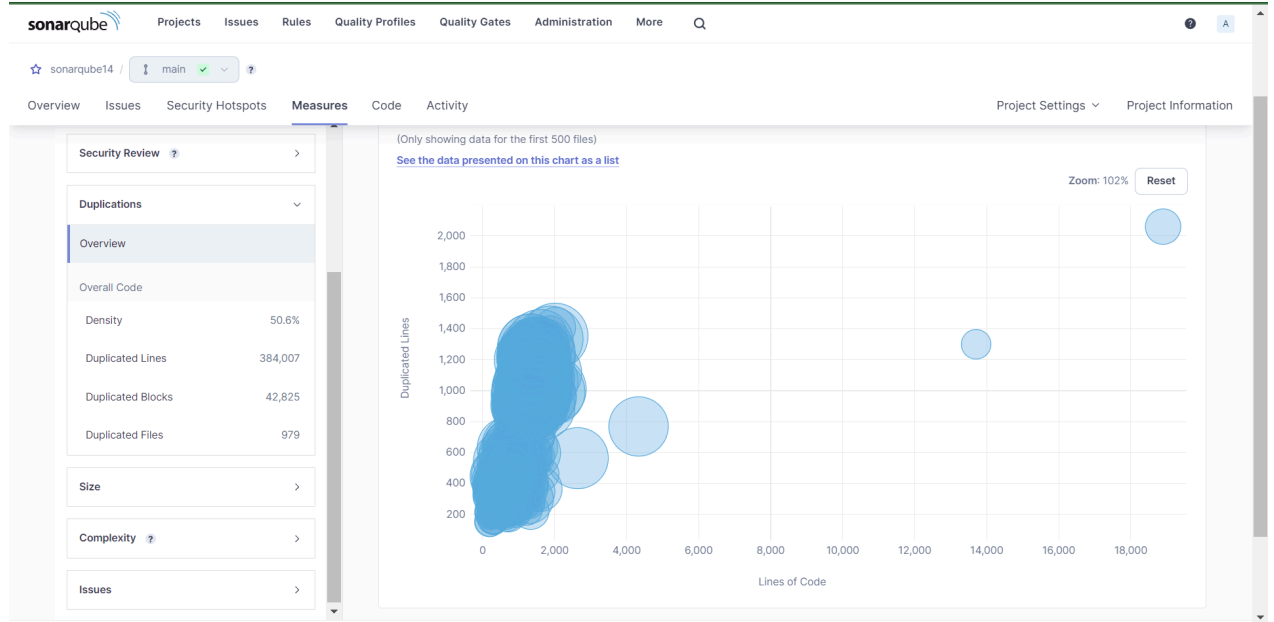
Type	Count
Bug	47k
Vulnerability	0
Code Smell	164k

Issue	Severity	Effort	Age	Category
Use a specific version tag for the image	Intentionality	5min	4 years ago	Code Smell - Major
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior	Intentionality	5min	4 years ago	Code Smell - Major

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 comprehensive code analysis. SonarQube provided valuable insights into various types of program issues, such as:

Consistency: Detected non-adherence to coding standards and formatting rules.

Intentionality: Flagged potential logical or structural errors within the code.

Severity: Classified issues by their level of criticality (e.g., critical, major, minor).

Duplicates: Identified redundant code segments, suggesting potential optimization.

Cyclomatic Complexity: Measured the complexity of the code based on the number of control flow paths, highlighting sections that might be difficult to maintain or prone to errors.

Issues Faced:

SonarQube Scanner Path Error: Initially, Jenkins failed to detect the correct path for the SonarQube Scanner. This required manual intervention to adjust the pipeline script and specify the correct path for the scanner bash file, allowing the pipeline to execute successfully.

SonarQube Login Issues: Default credentials for logging into SonarQube (username: admin, password: admin) did not work due to a configuration reset or password change in a previous session. This necessitated troubleshooting and reconfiguration of access credentials.

Slow Analysis Process: During the scan of larger repositories, the SonarQube analysis process was notably slow.