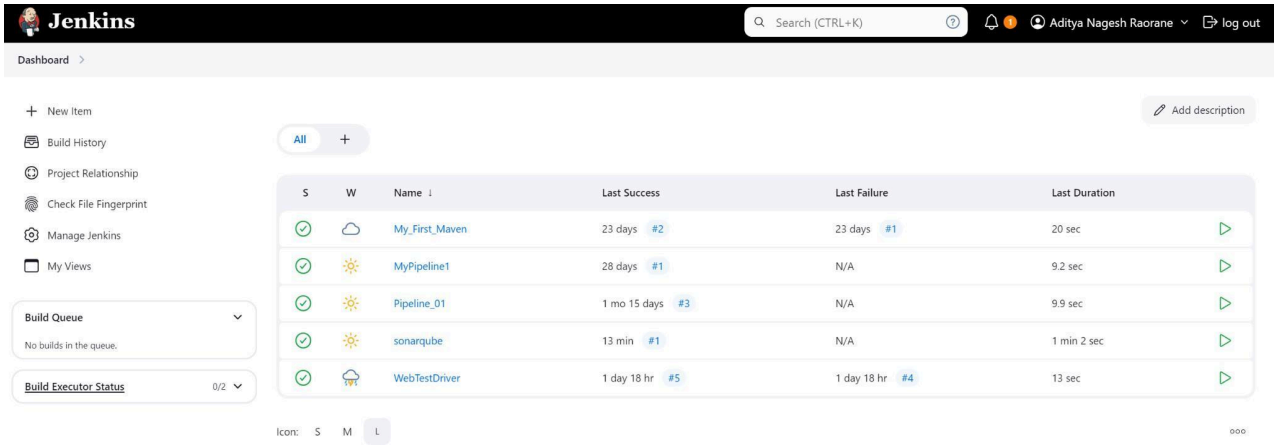
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

1. Open up Jenkins Dashboard on localhost:8080.



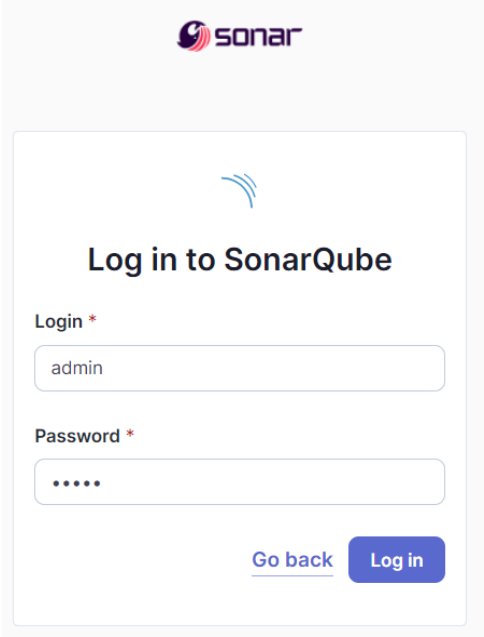
2. Run SonarQube in a Docker container using this command: a] docker -v

b] docker pull sonarqube

c] docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

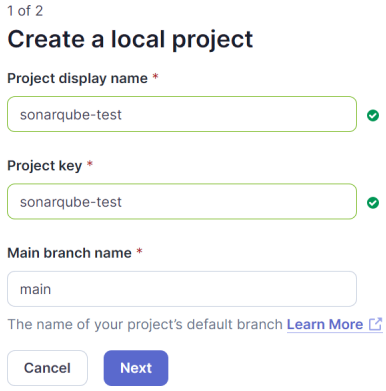
```
C:\Users\adity>docker -v
Docker version 27.0.3, build 7d4bcd8

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
```

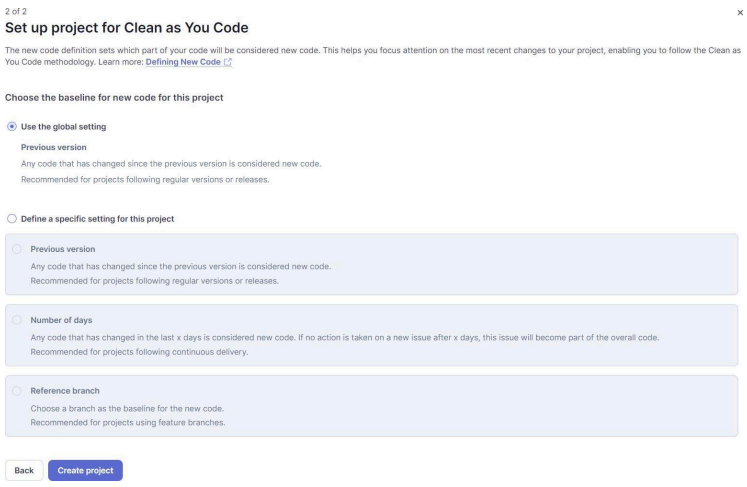
3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is



“**aditya**”.

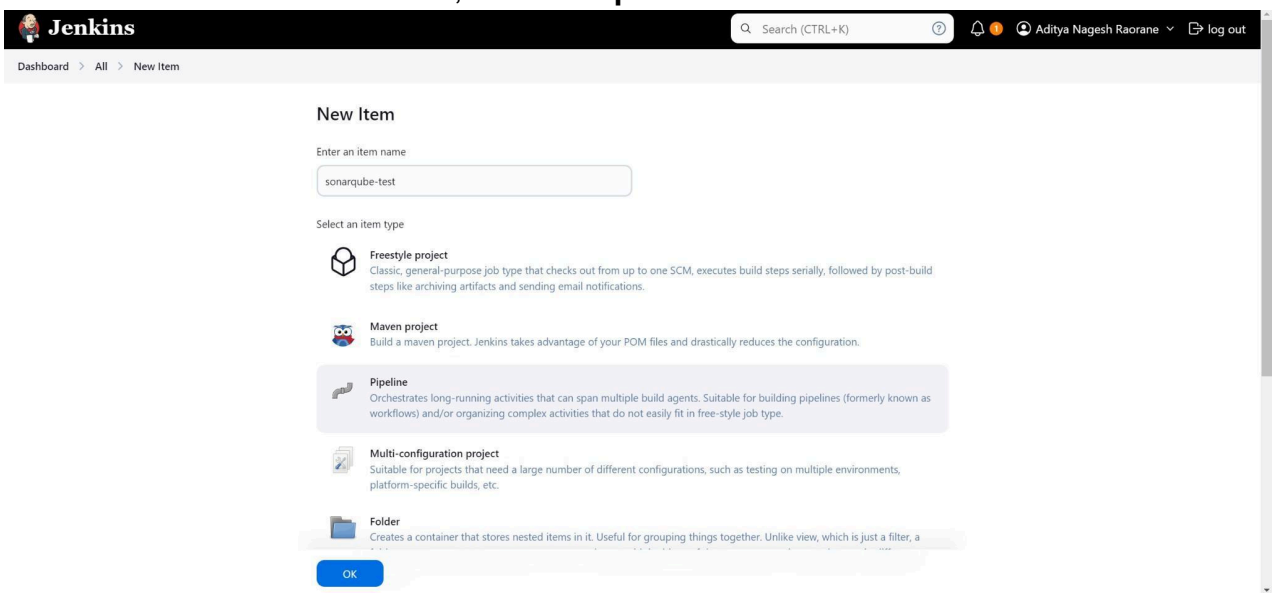
4. **Create a local project in SonarQube** with the name **sonarqube-test**.





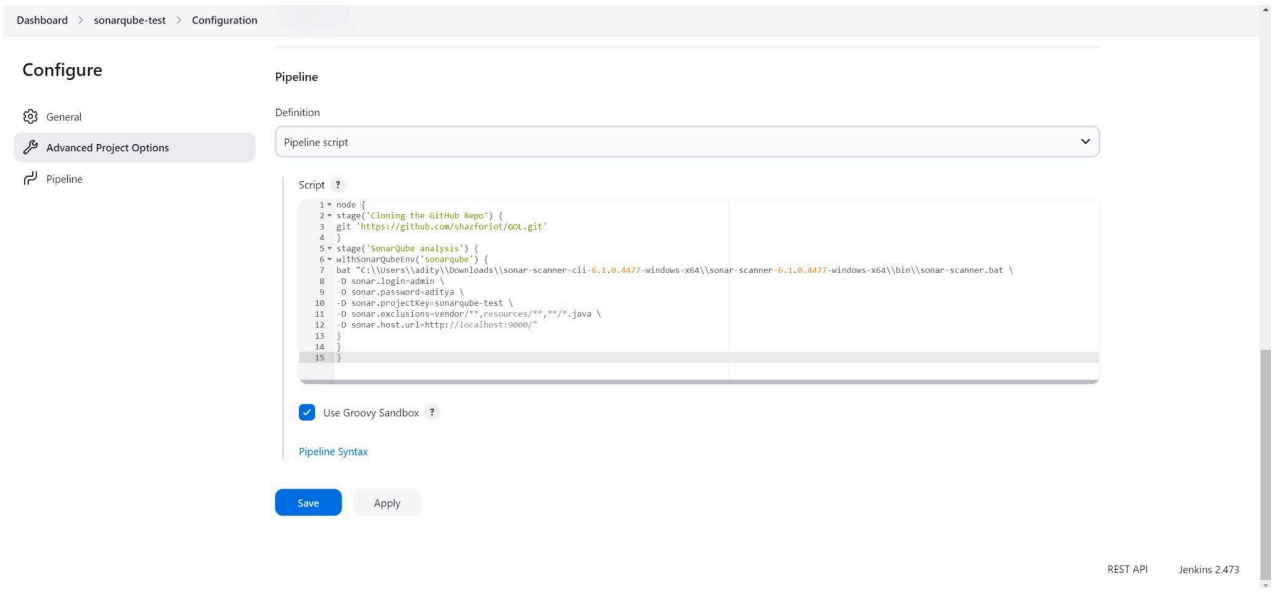
Setup the project and come back to Jenkins Dashboard.

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



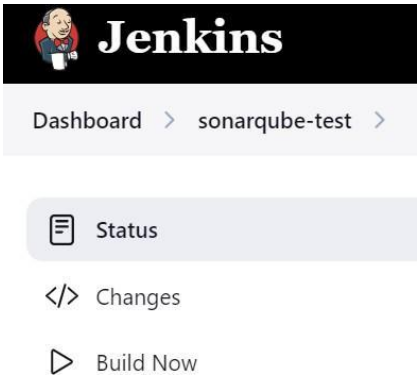
7. Under **Pipeline Script**, enter the following -

```
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/"
                }
        }
}
```

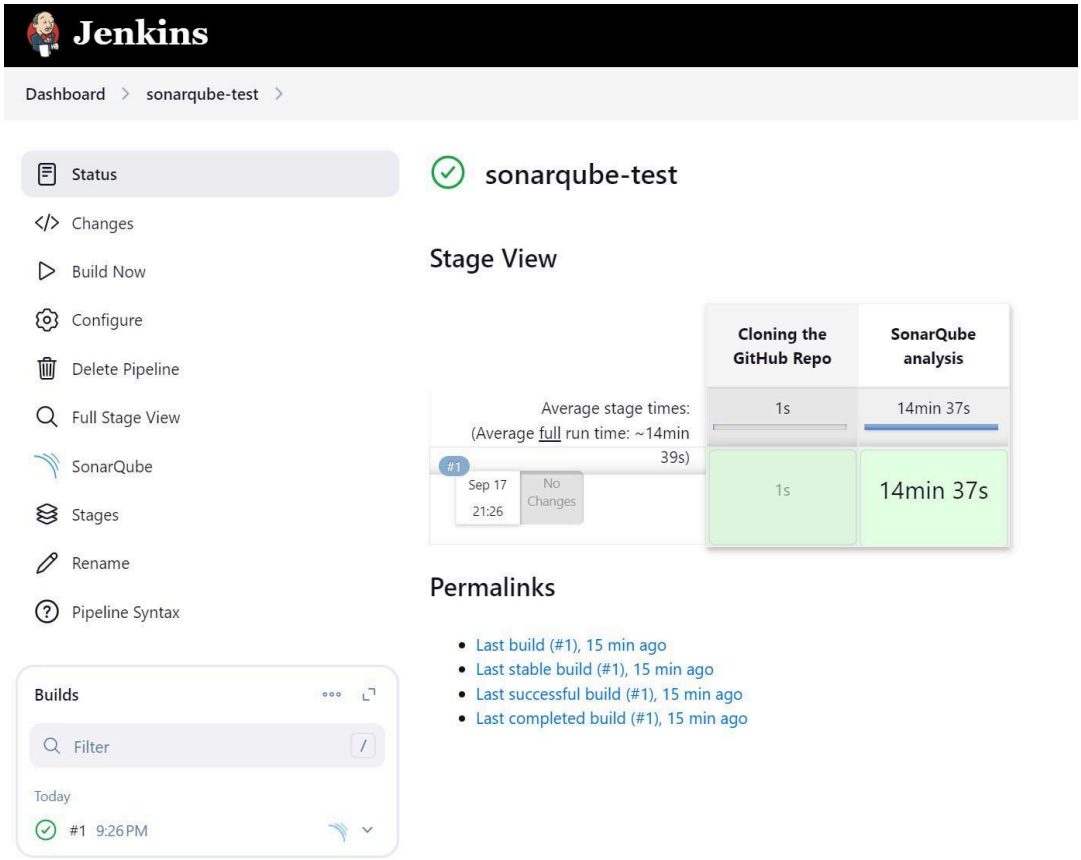


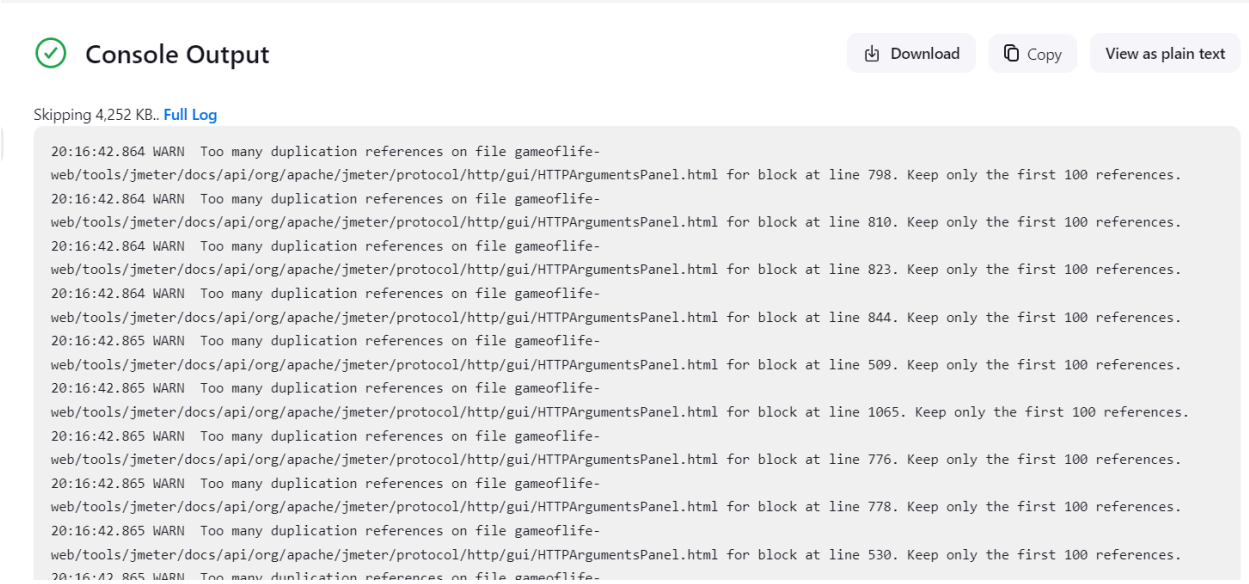
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8. Run The Build.



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



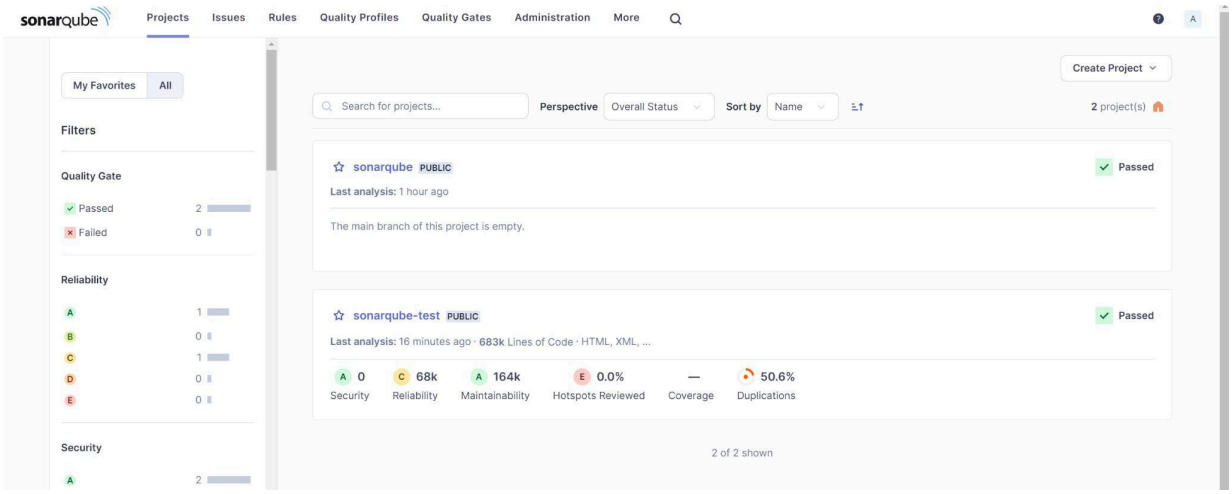


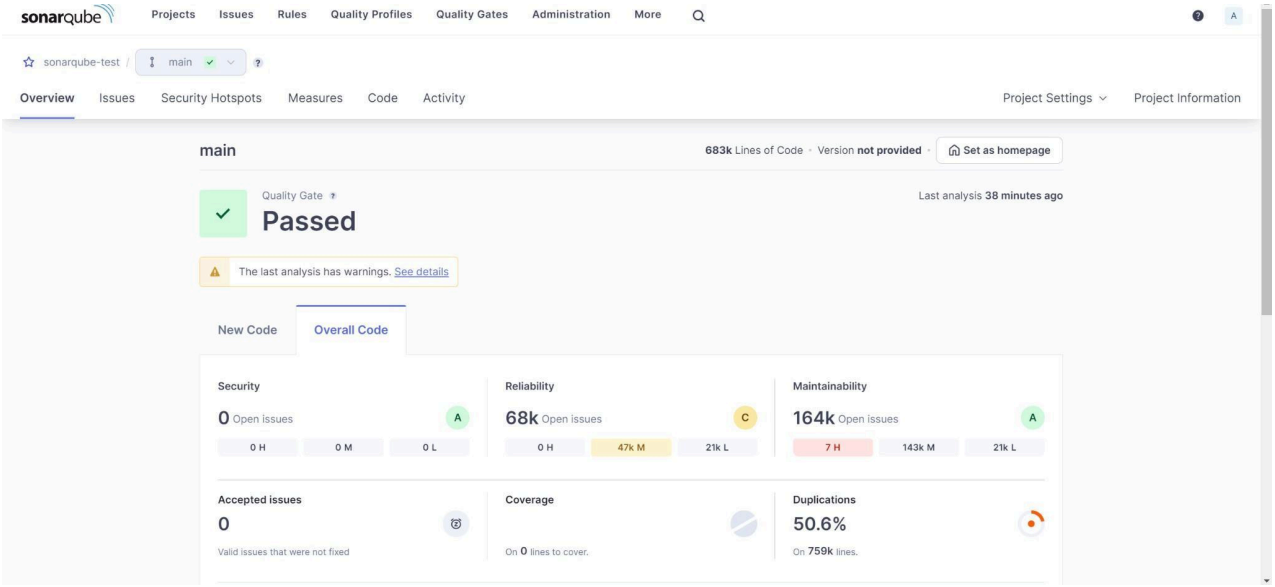
Dashboard > sonarqube-test > #1

```
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 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.

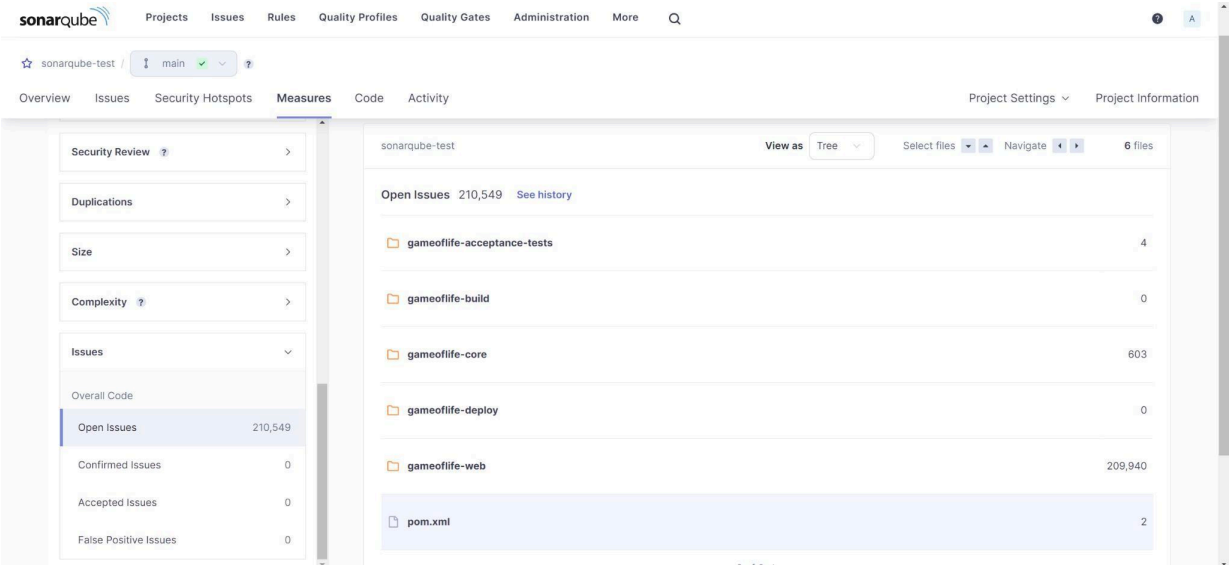




Under different tabs, check all different issues with the code.

11. **Code**

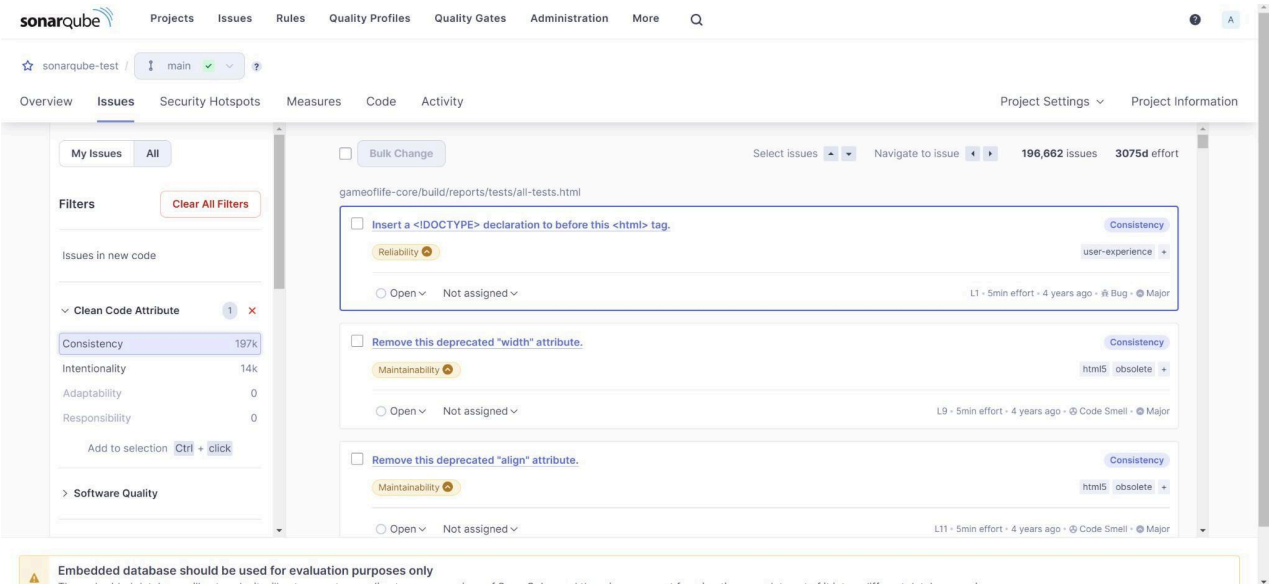
Problems - Open



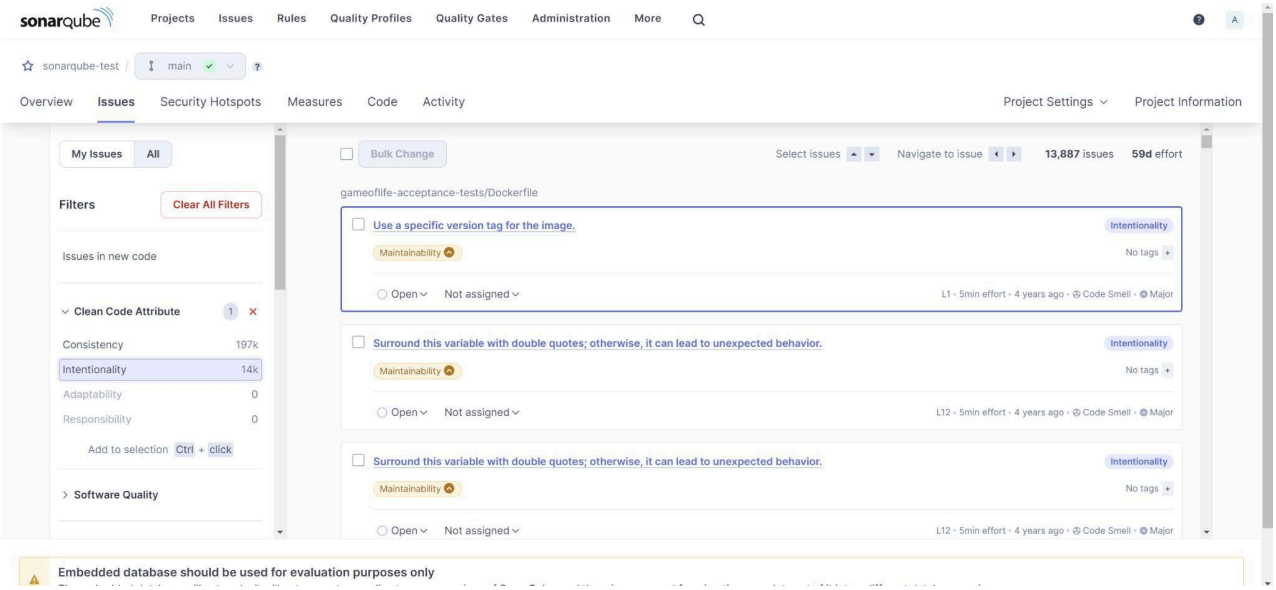
Issues

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

Consistency

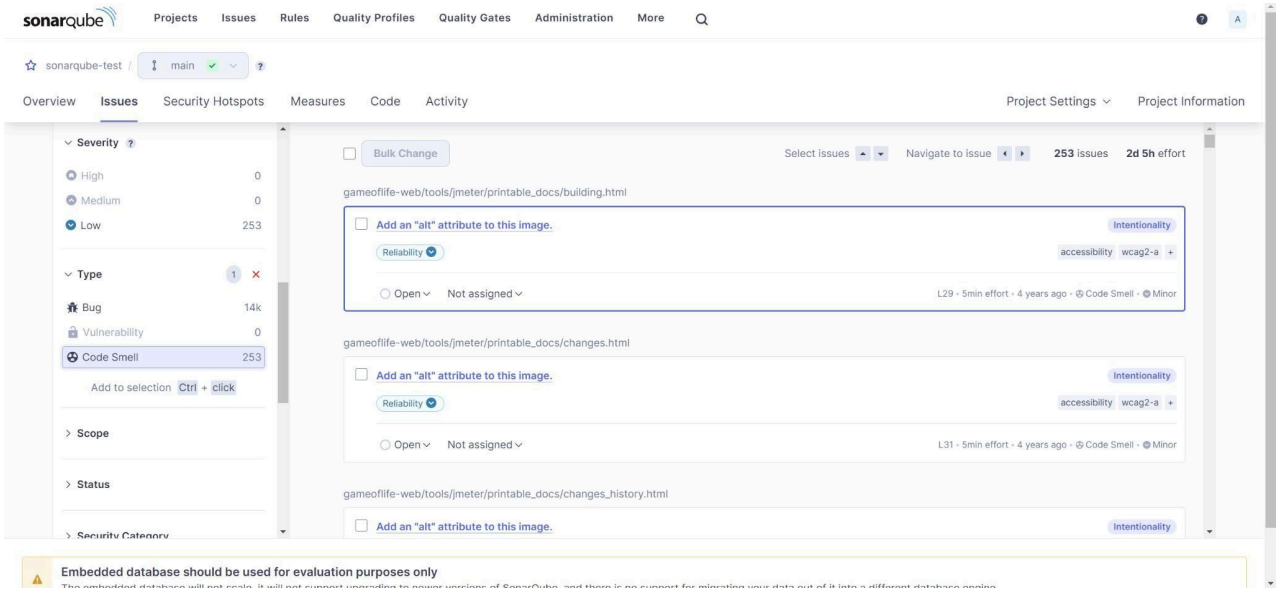


Intentionality

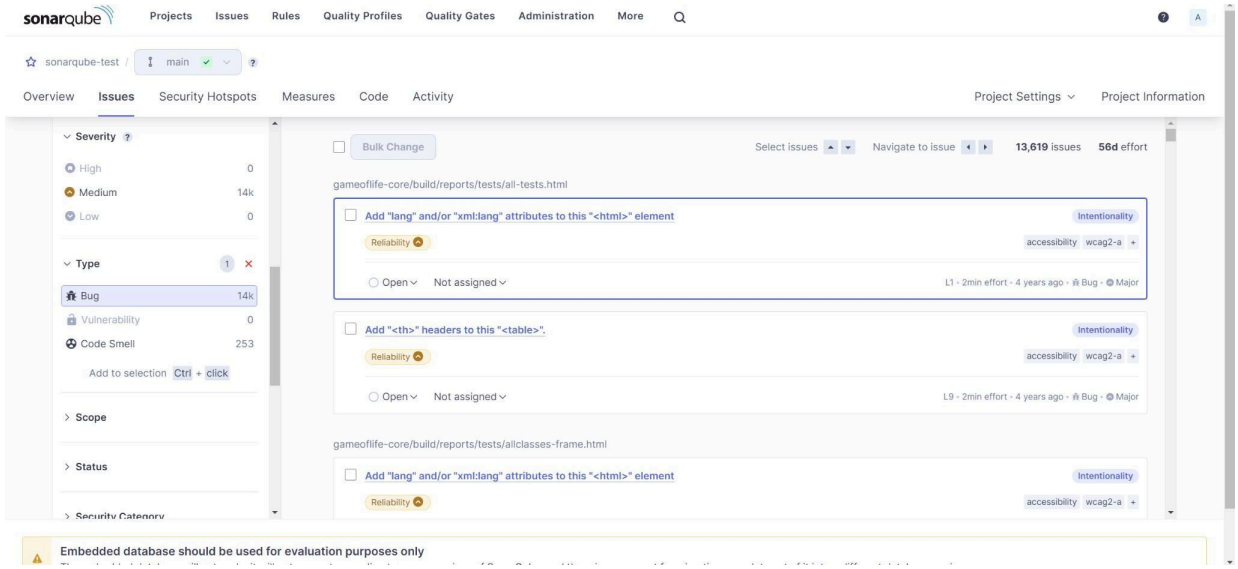


Code Smells

Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

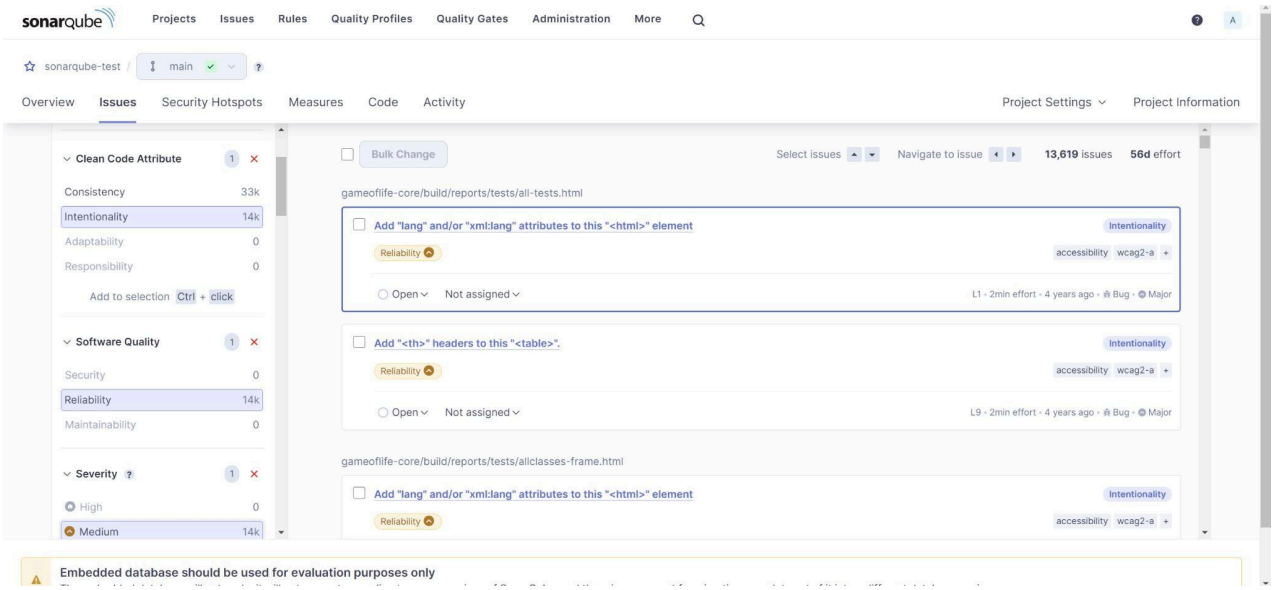


Bugs

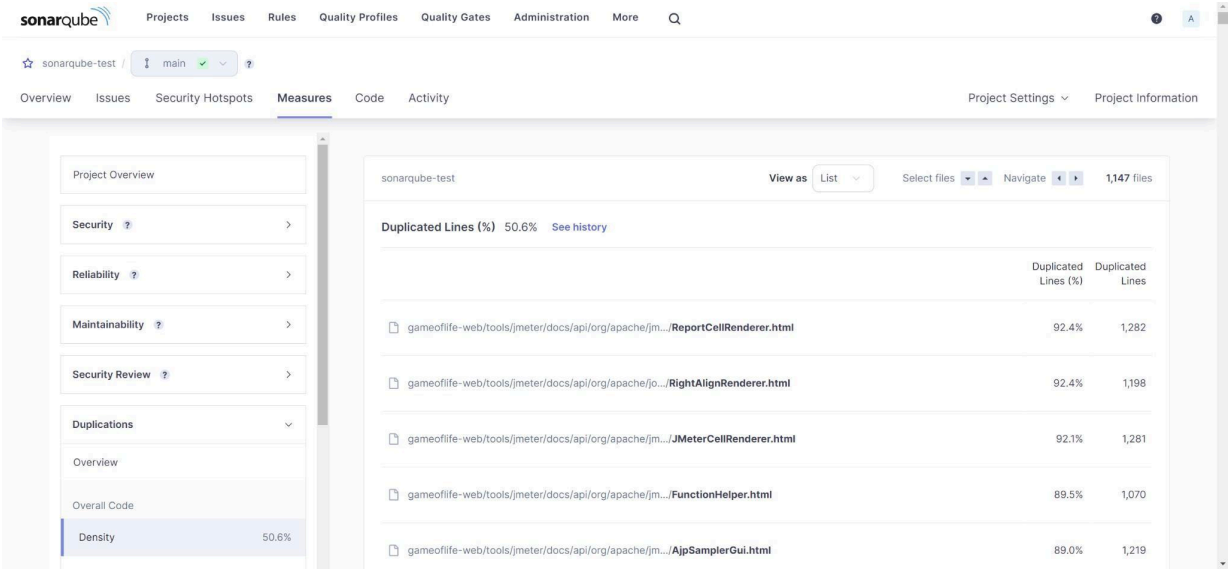


Name: Laukik Padgaonkar Class: D15C/ Batch B Roll No: 37

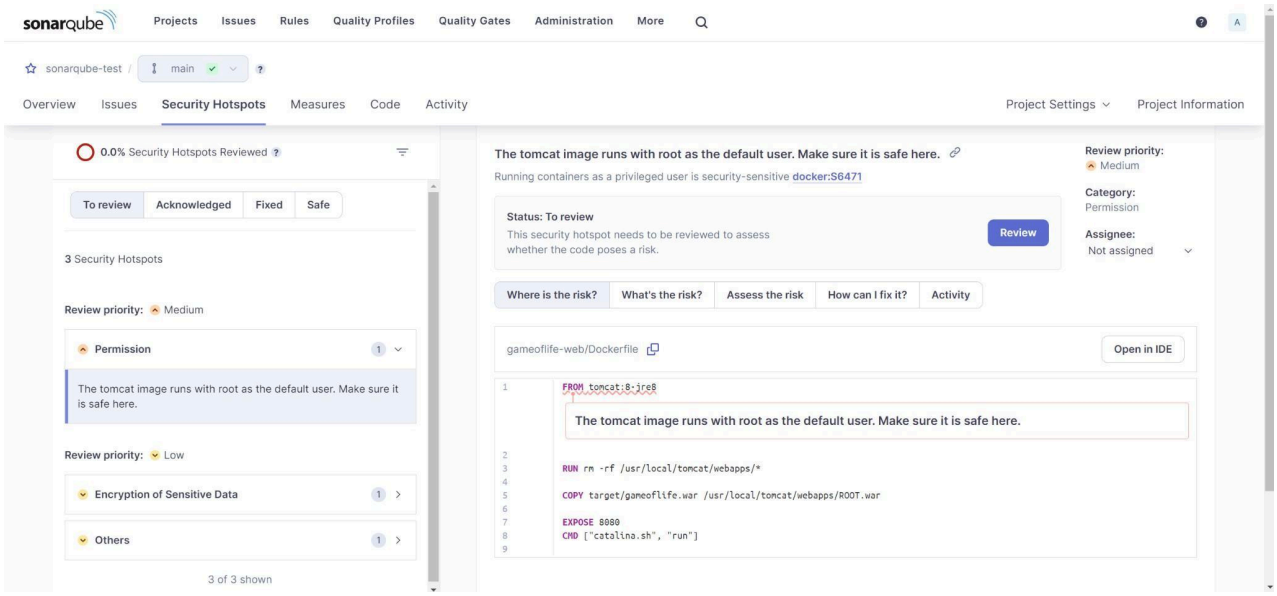
Reliability



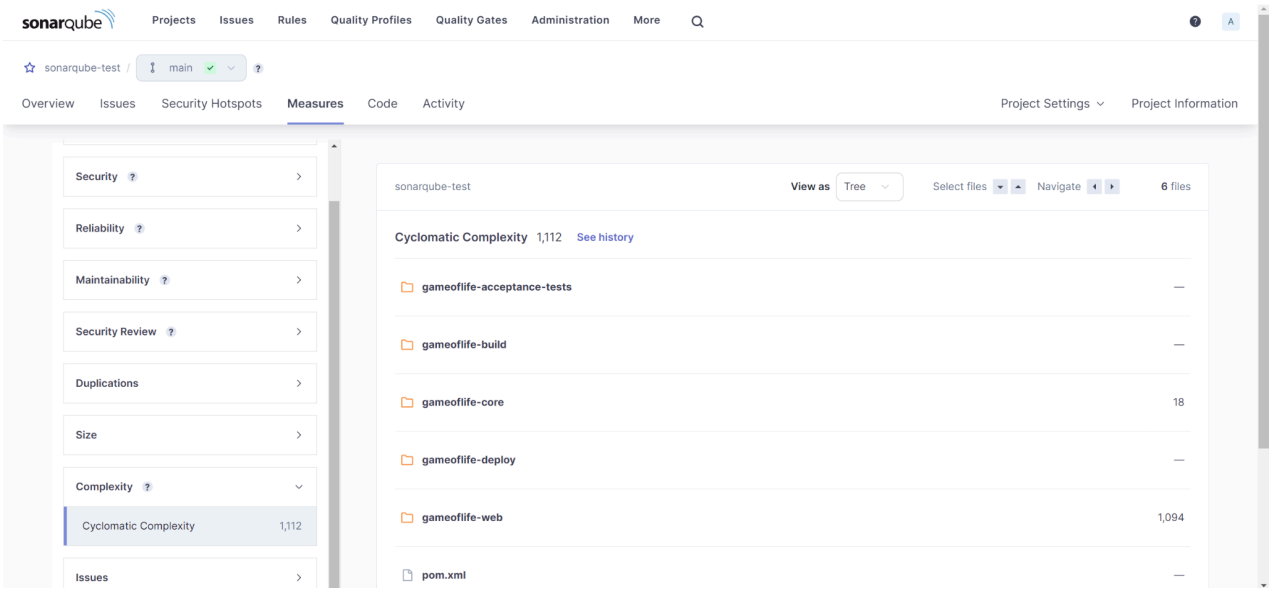
Duplicates



Security Hotspot



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.