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.

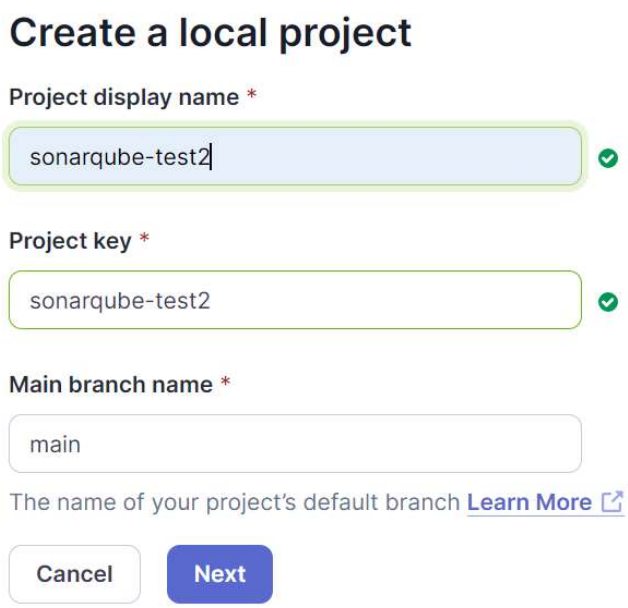
1. Open up the jenkins dashboard. Usually on localhost: 8080
2. Run SonarQube in a Docker container using this command -

```
PS C:\Windows\system32> 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
5008a1b64daaff9ca9430d6426a4ceb303d964d039798555e9fb548ff7c89073
```

3. Once the container is running, you can check it on localhost:9000. If you have already made a container named sonarqube, run the following command instead.

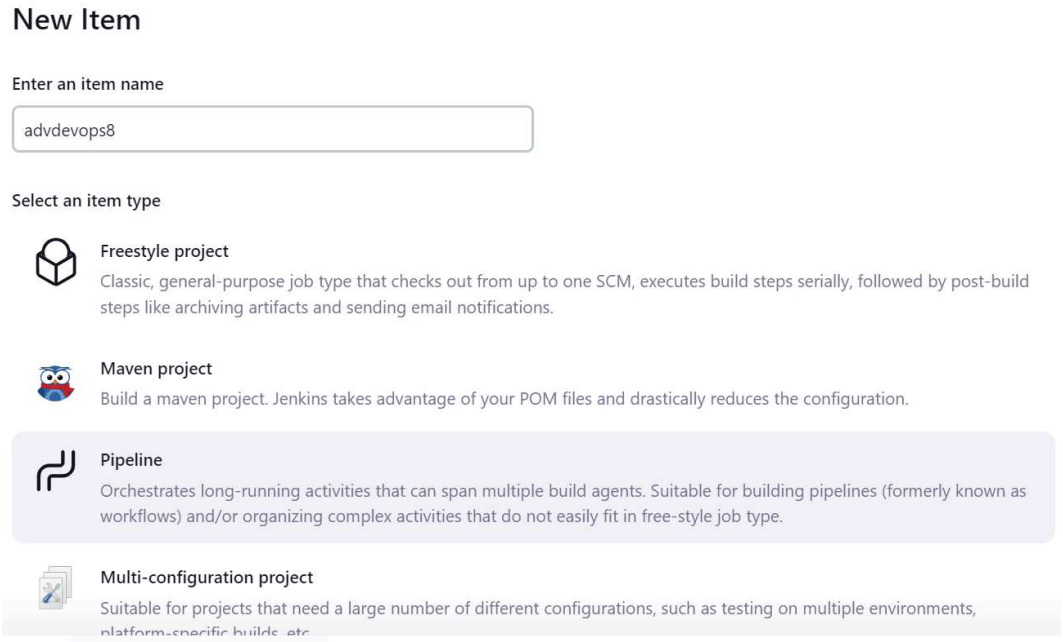
```
PS C:\Windows\system32> docker start sonarqube
>>
sonarqube
```

4. Login to sonarqube with username admin, password admin, or if you have setup any other credentials use that.
5. Create a local project. Give it a name(sonarqube-test2). Configure the project and click on create.



6. Now we need to install the SonarQube CLI.
   Go to following link:https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/
   Download the suitable version compatible with your device.
   Extract the zip file downloaded. Inside the folder go to bin/sonar-scanner, copy the path of this file.

7. Create a new project in Jenkins. Give it a name and select the project type as pipeline in Jenkins.



8. Inside the pipeline script, give the following script. Use the path copied in earlier step instead of Path to sonarqube folder.

```
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/"
```

```
        }
    }
}
```

Click on save to create the pipeline.

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

9. Run the build.

Stage View

	Cloning the GitHub Repo	SonarQube analysis
Average stage times: (Average full run time: ~15min 44s)	1s	7min 51s
#2 Sep 28 09:18 No Changes	1s	15min 42s

10. Check the console output

Console Output

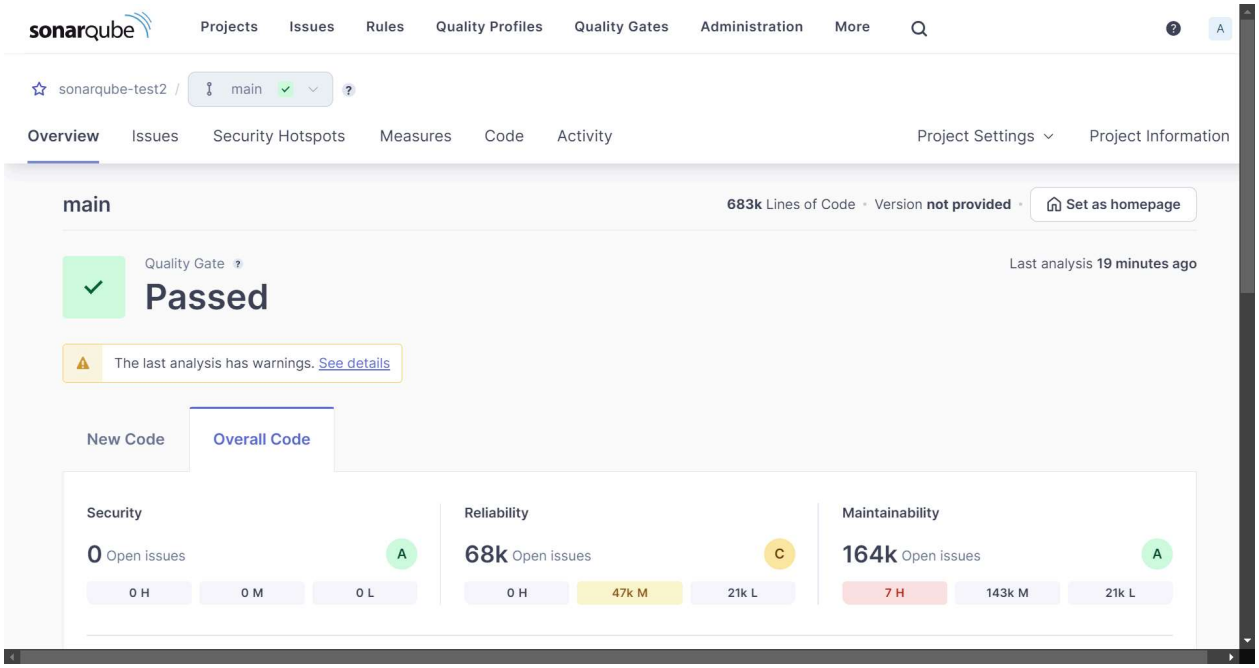
Download Copy View as plain text

Skipping 4,249 KB.. Full Log

```
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 296. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 17. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 212. Keep
only the first 100 references.
09:27:10.137 WARN  Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/gui/ReportMainFrame.WindowHappenings.html for block at line 215. Keep
```

```
09:27:15.430 INFO  CPD Executor CPD calculation finished (done) | time=28705ms
09:27:15.451 INFO  SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
09:30:22.050 INFO  Analysis report generated in 138747ms, dir size=127.2 MB
09:33:36.168 INFO  Analysis report compressed in 194118ms, zip size=29.6 MB
09:33:36.981 INFO  Analysis report uploaded in 813ms
09:33:36.982 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube-
test2
09:33:36.982 INFO  Note that you will be able to access the updated dashboard once the server has processed the
submitted analysis report
09:33:36.982 INFO  More about the report processing at http://127.0.0.1:9000/api/ce/task?id=0fd50499-d2c7-460f-
a5c0-06a3b9908b8b
09:33:43.711 INFO  Analysis total time: 15:38.015 s
09:33:43.729 INFO  SonarScanner Engine completed successfully
09:33:44.597 INFO  EXECUTION SUCCESS
09:33:44.597 INFO  Total time: 15:41.397s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

11. Since the build is successful, now check the project in sonarqube.

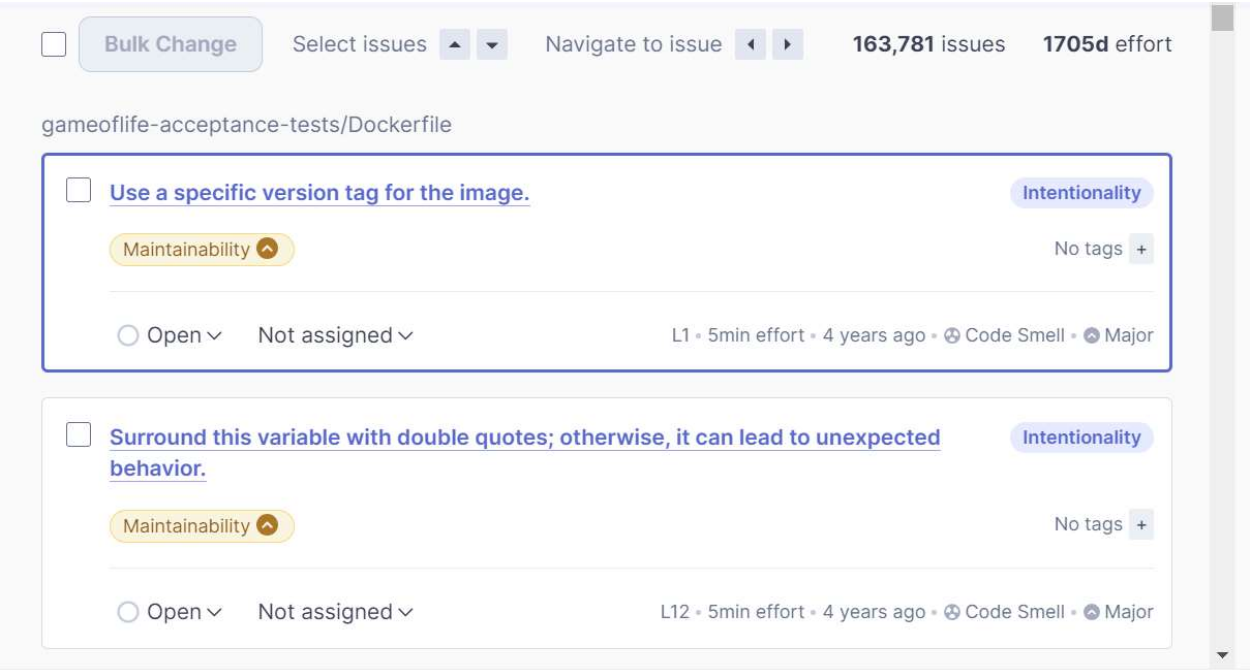


You will be able to see different issues with the code under different tabs listed.

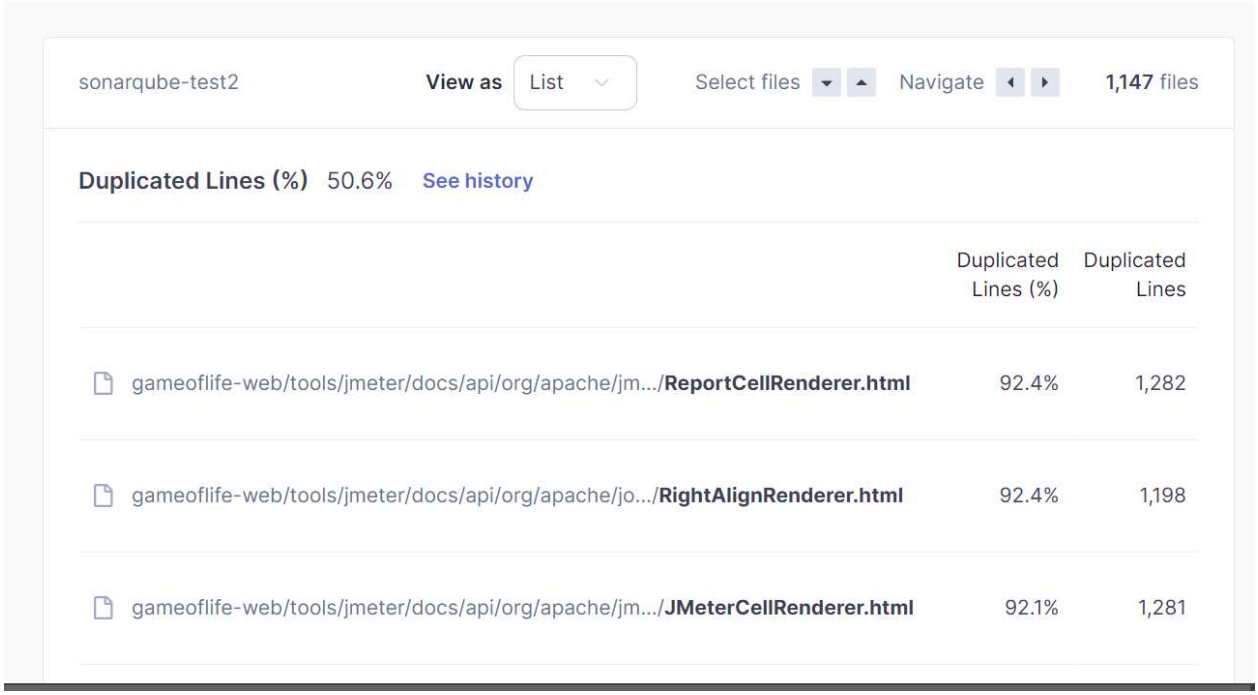
12. Code problems:

Reliability:

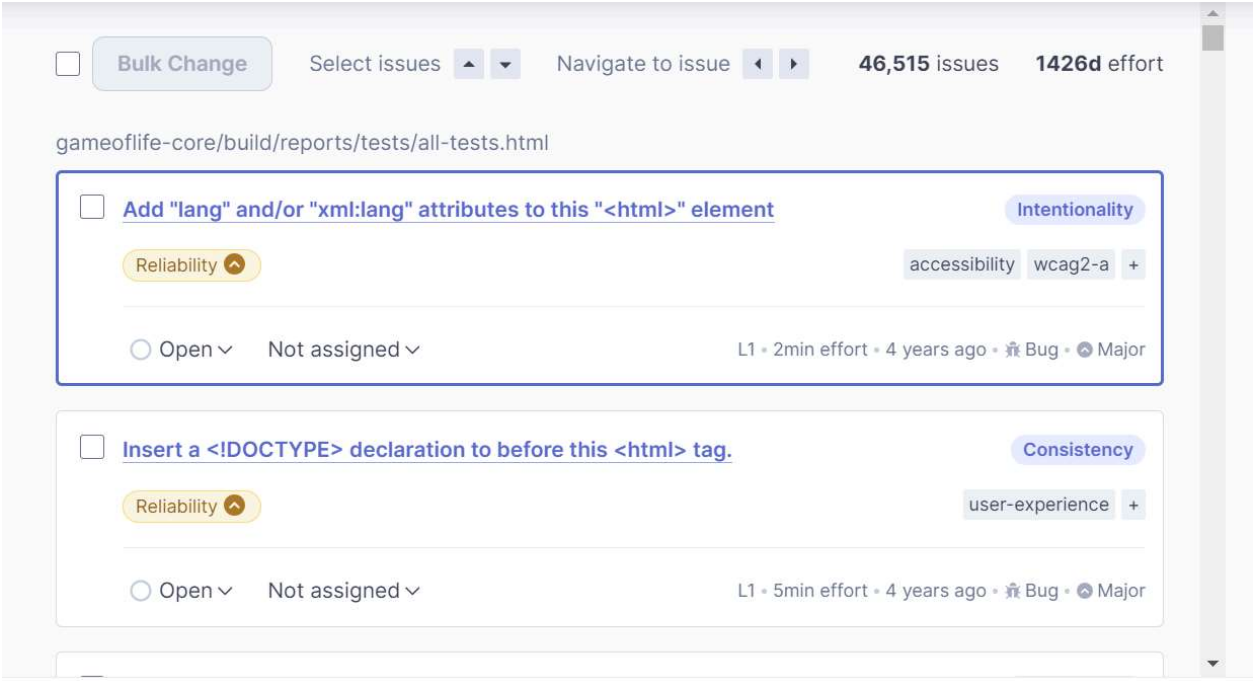
Maintainability:



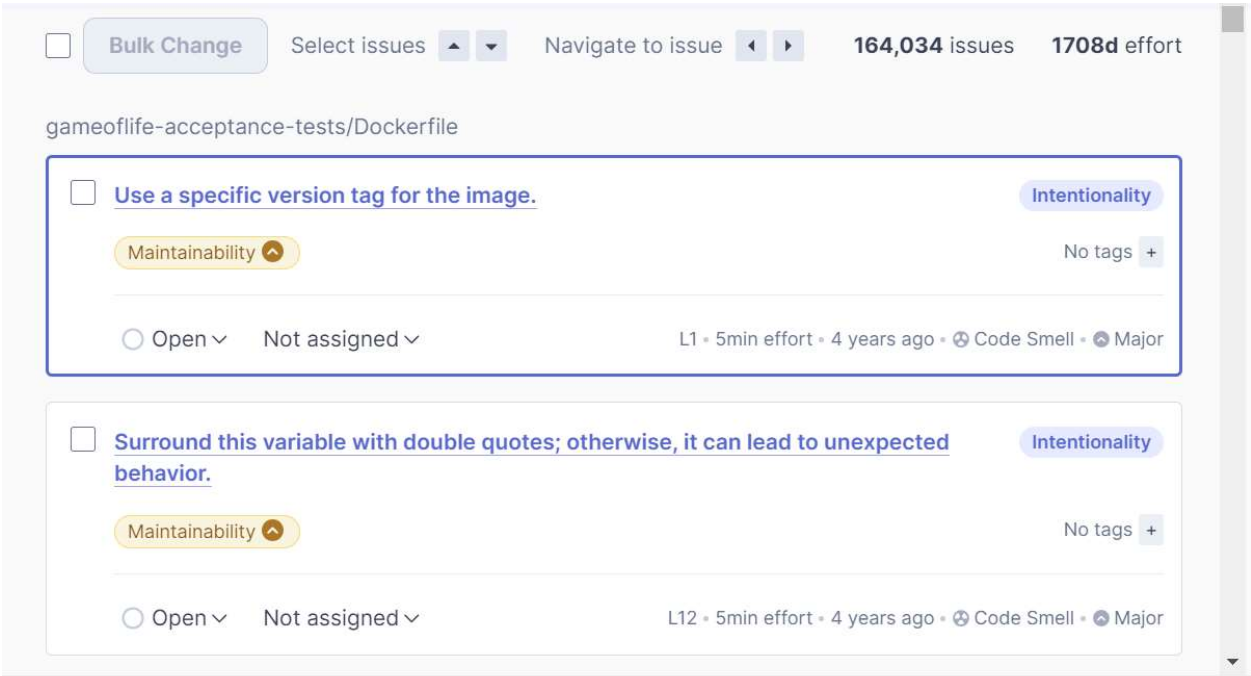
Duplications:



Bug:



Code smells:



And various other issues or problems can be seen in the sonarqube project dashboard.

Conclusion:

In this experiment, a Jenkins CI/CD pipeline was successfully integrated with SonarQube to perform static code analysis on a sample application. SonarQube, running in a Docker

container, analyzed the code for issues like bugs, code smells, and security vulnerabilities. The pipeline was automated by cloning the source code from a GitHub repository and using the SonarQube Scanner to detect potential code problems. After a successful Jenkins build, the SonarQube project dashboard provided information about code's reliability, maintainability, duplications, and detected bugs, offering feedback that help improve the code quality.