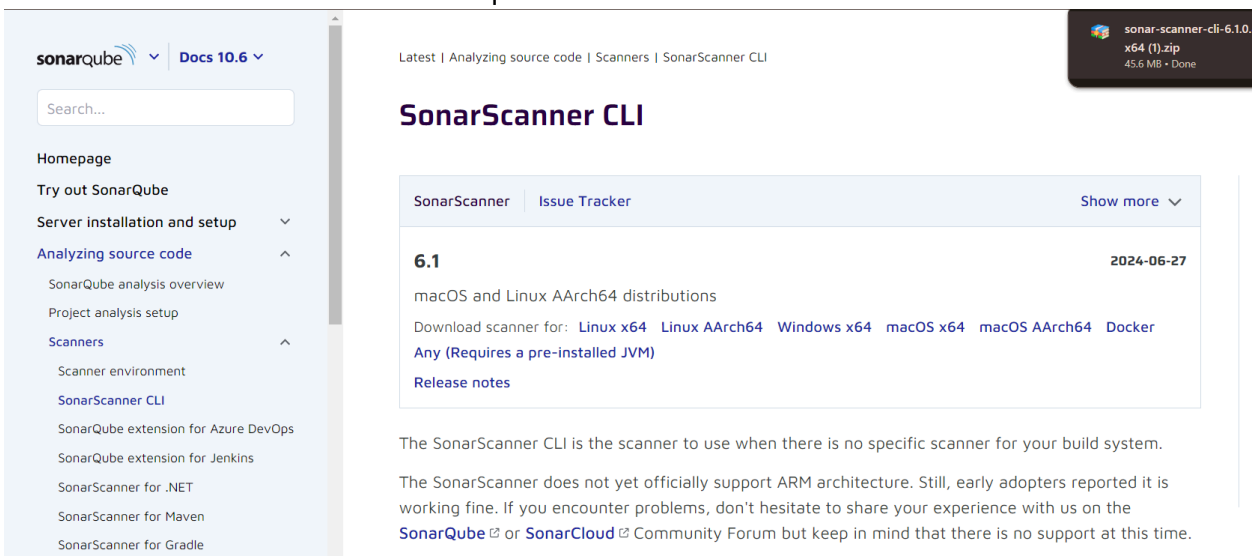
NAME: BHUSHAN MALPANI DIV: D15C ROLL NO: 33

08 Advanced DevOps Lab

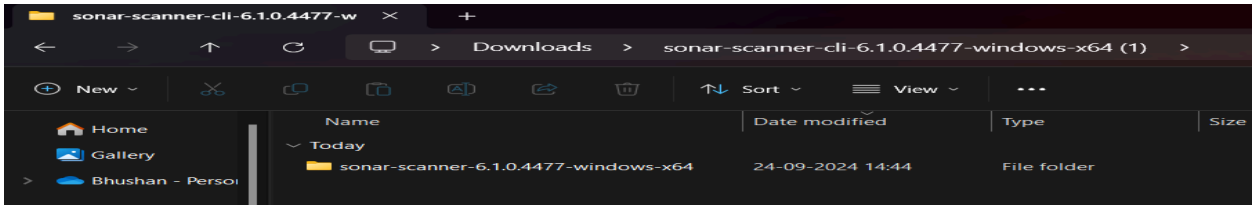
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Step 1: Download sonar scanner

https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/
Visit this link and download the sonarqube scanner CLI.

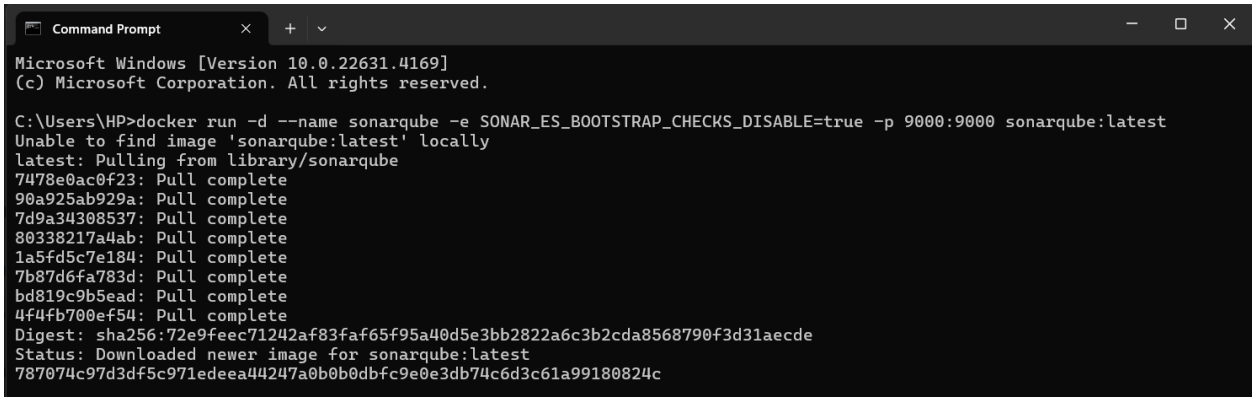


Extract the downloaded zip file in a folder.

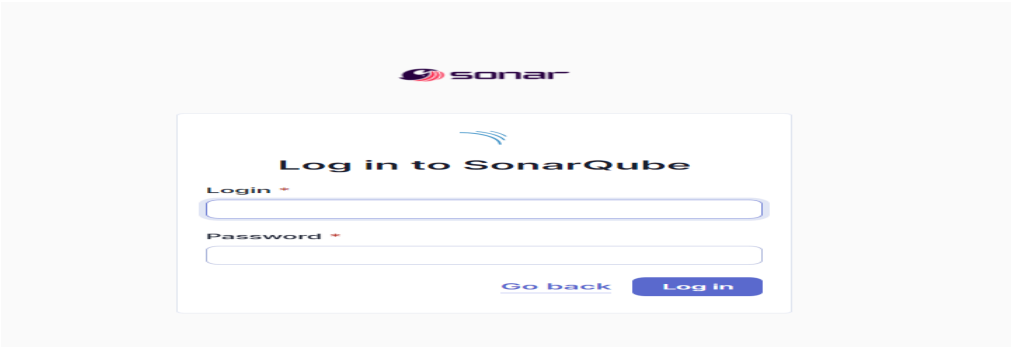


1. Install sonarqube image

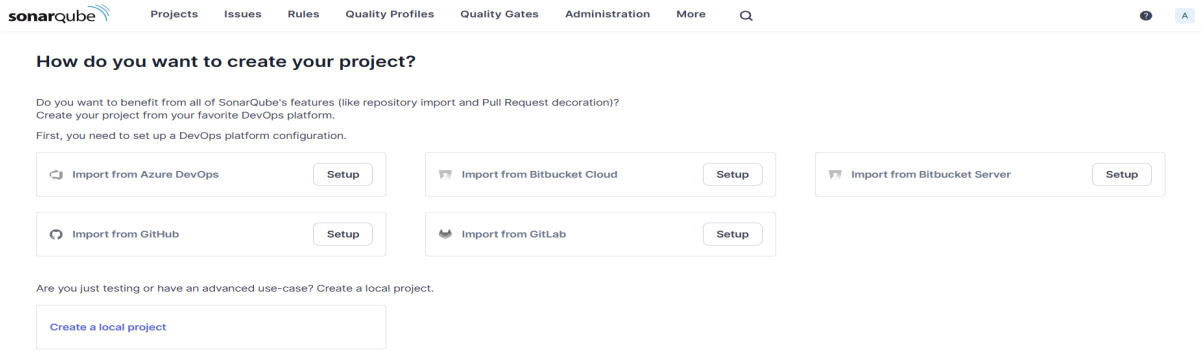
Command: **docker pull sonarqube**



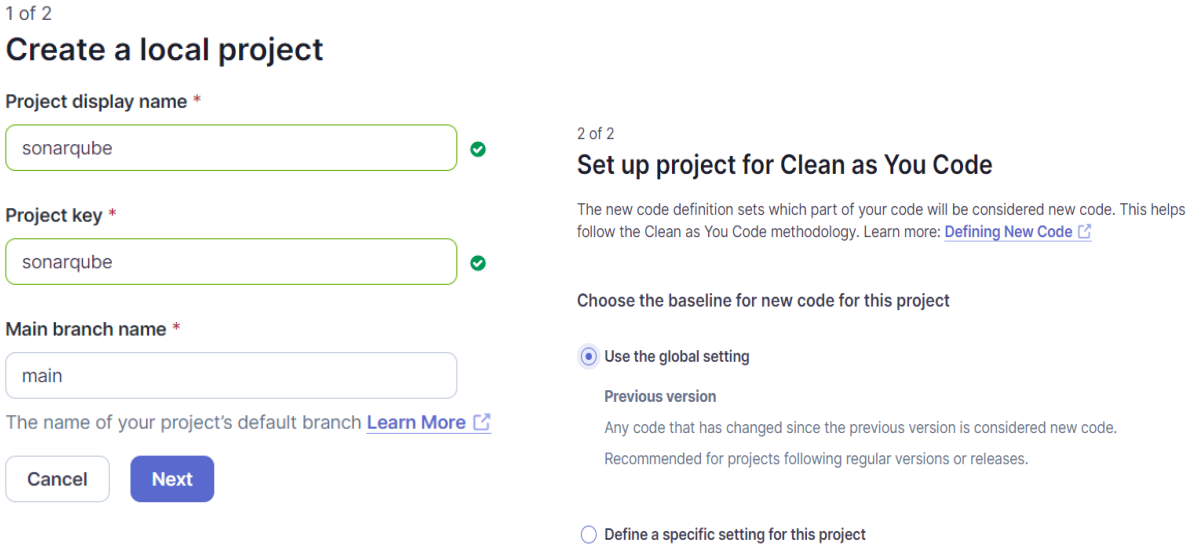
2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



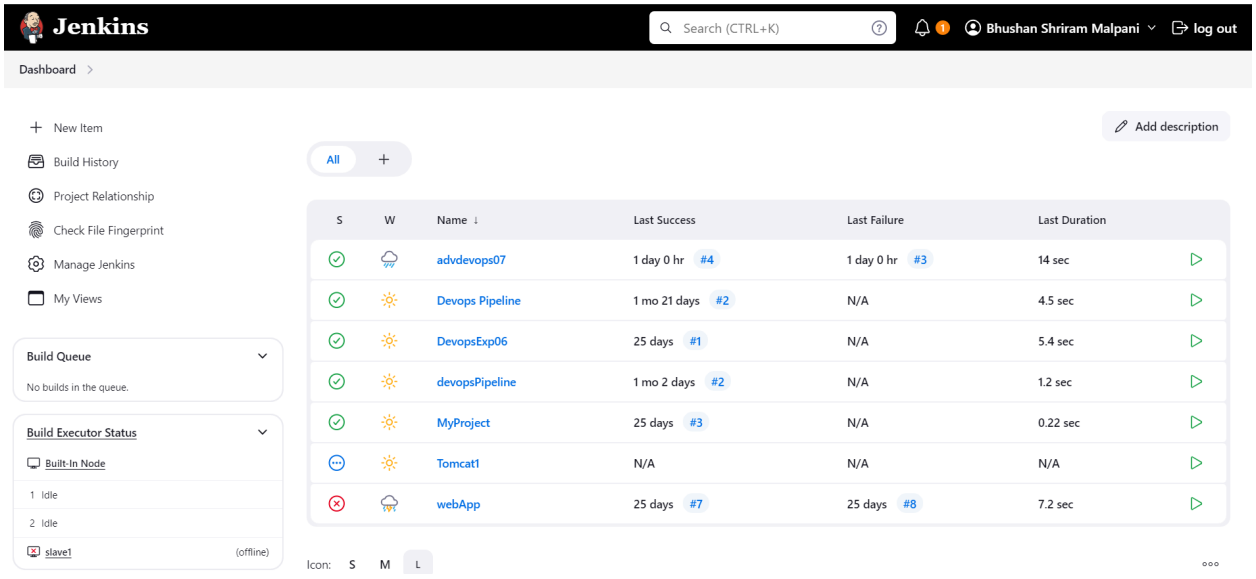
3. Login to SonarQube using username admin and password admin.



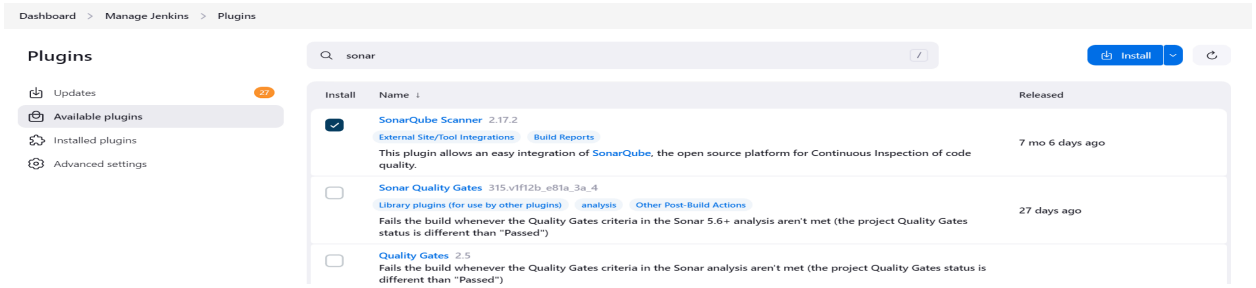
4. Create a manual project in SonarQube with the name sonarqube

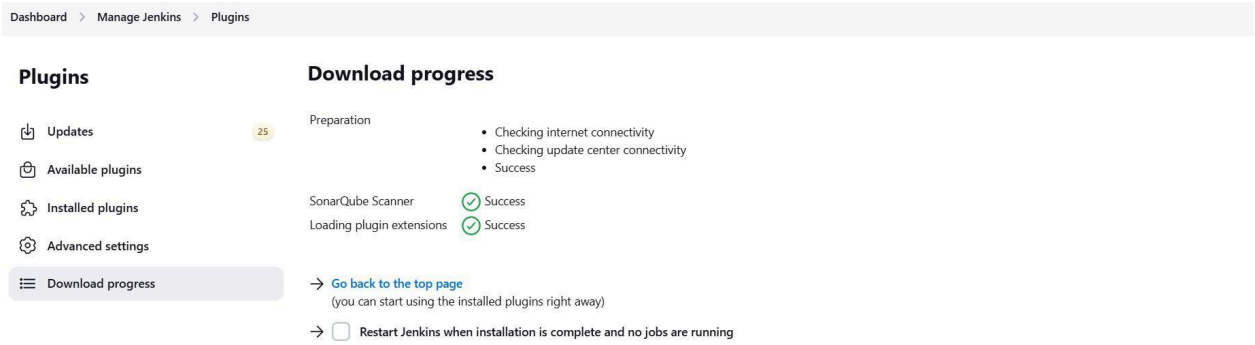


5. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



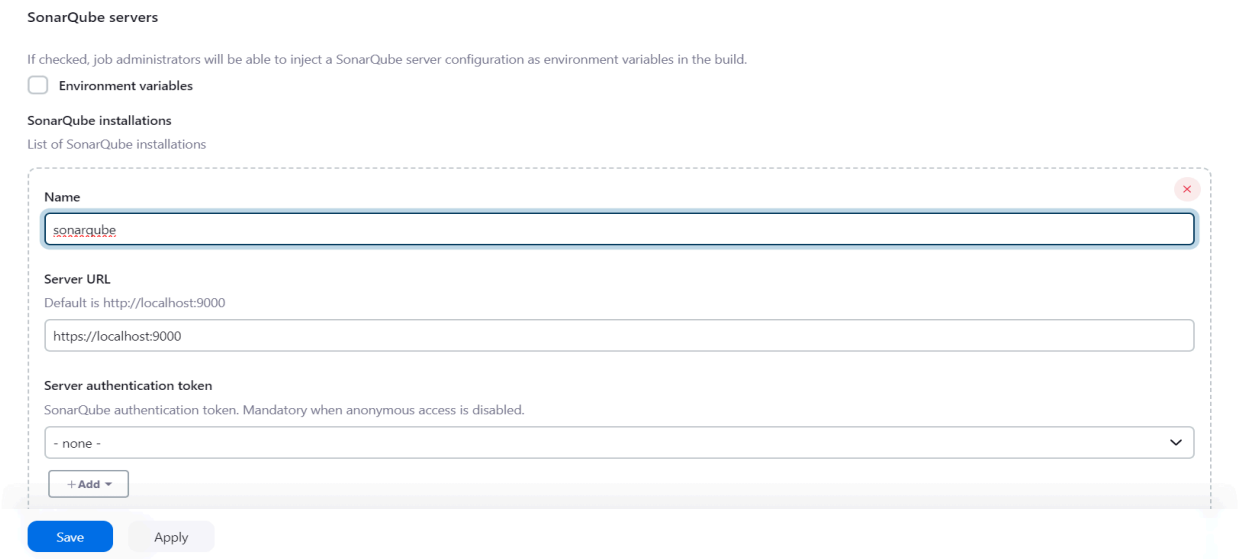
6. Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.





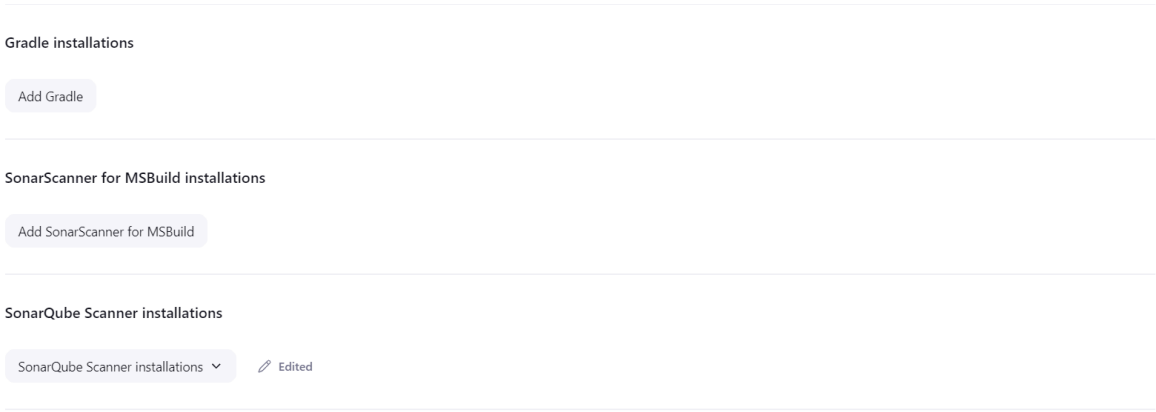
7. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**



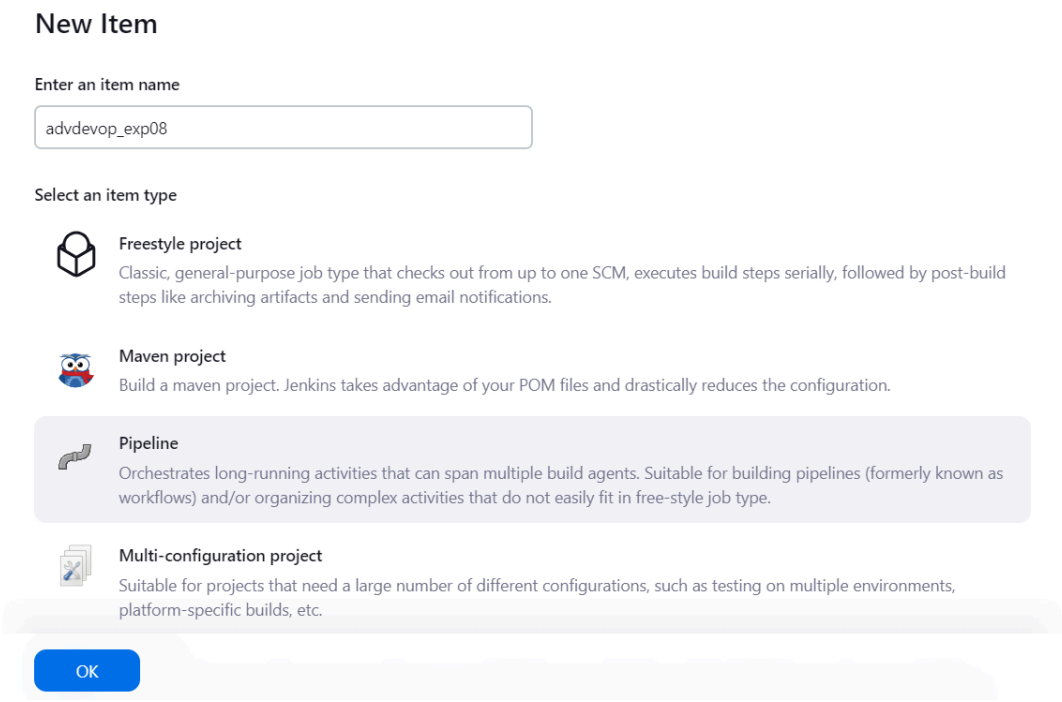
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Dashboard > Manage Jenkins > Tools



Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.

9. After configuration, create a New Item → choose a pipeline project.



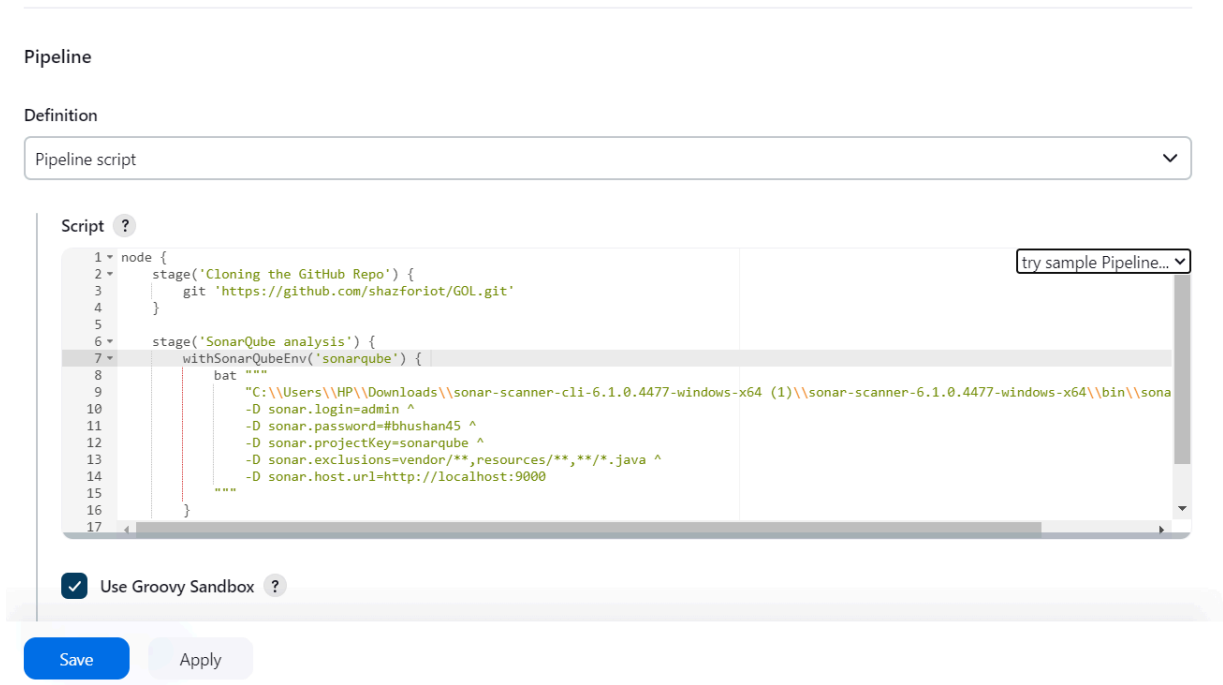
10. Under Pipeline script, enter the following:

```
node {
stage('Cloning the GitHub Repo') {
  git 'https://github.com/shazforiot/GOL.git'
```

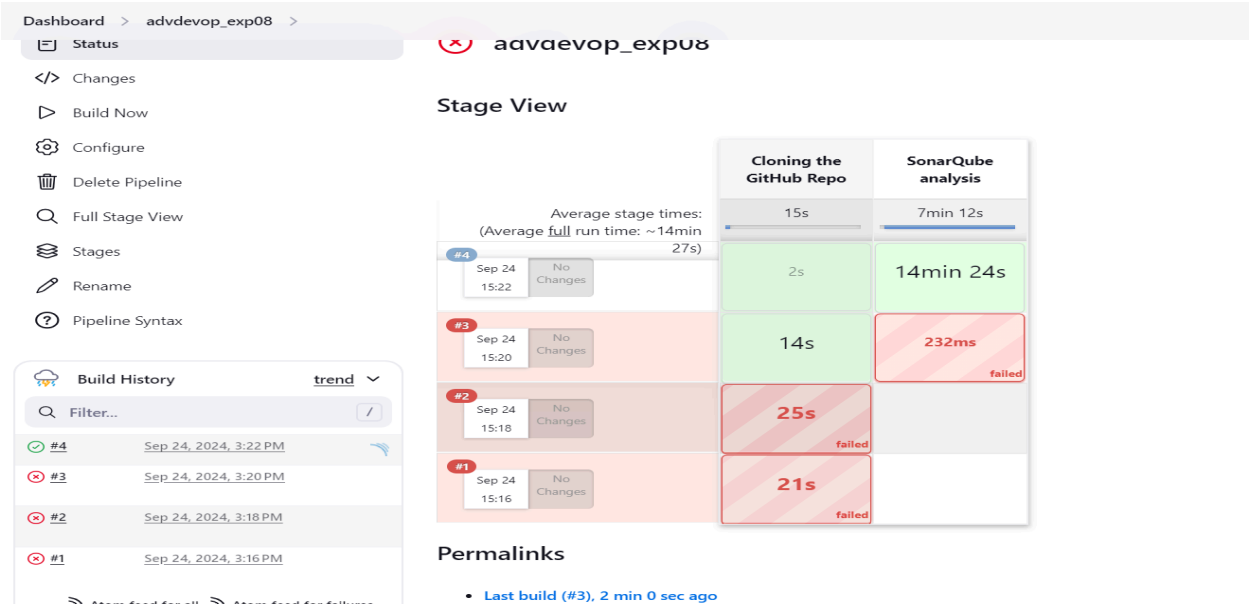
```
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('<Name_of_SonarQube_environment_on_Jenkins>') {
            sh """
                <PATH_TO_SONARQUBE_SCANNER_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=<SonarQube_URL>(default: http://localhost:9000/)
            """
        }
    }
}
```

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



11. Build project



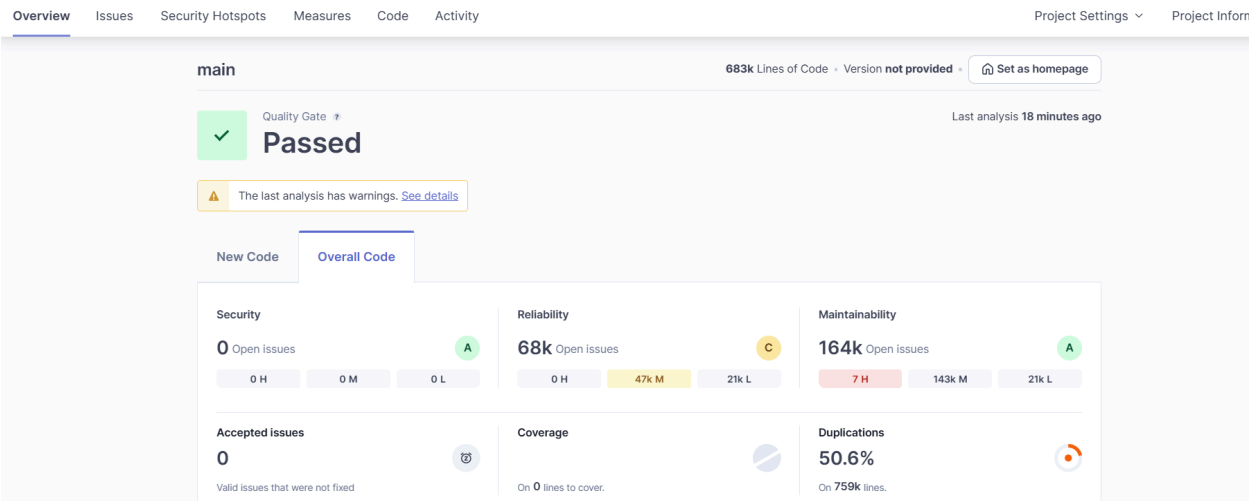
12. Check console

Console Output

Skipping 4,250 KB.. Full Log

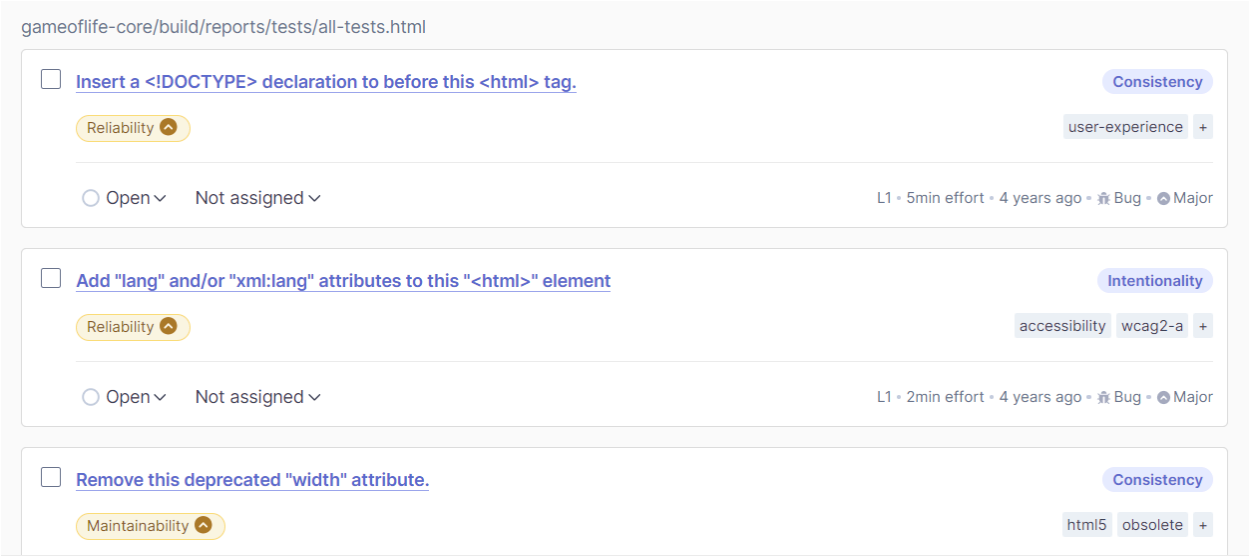
```
15:36:08.390 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 789. Keep only the first 100 references.
15:36:08.390 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 512. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 248. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 886. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 249. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 662. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 615. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 664. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 913. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 810. Keep only the first 100 references.
15:36:08.391 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html
for block at line 668. Keep only the first 100 references.
```

13. Now, check the project in SonarQube

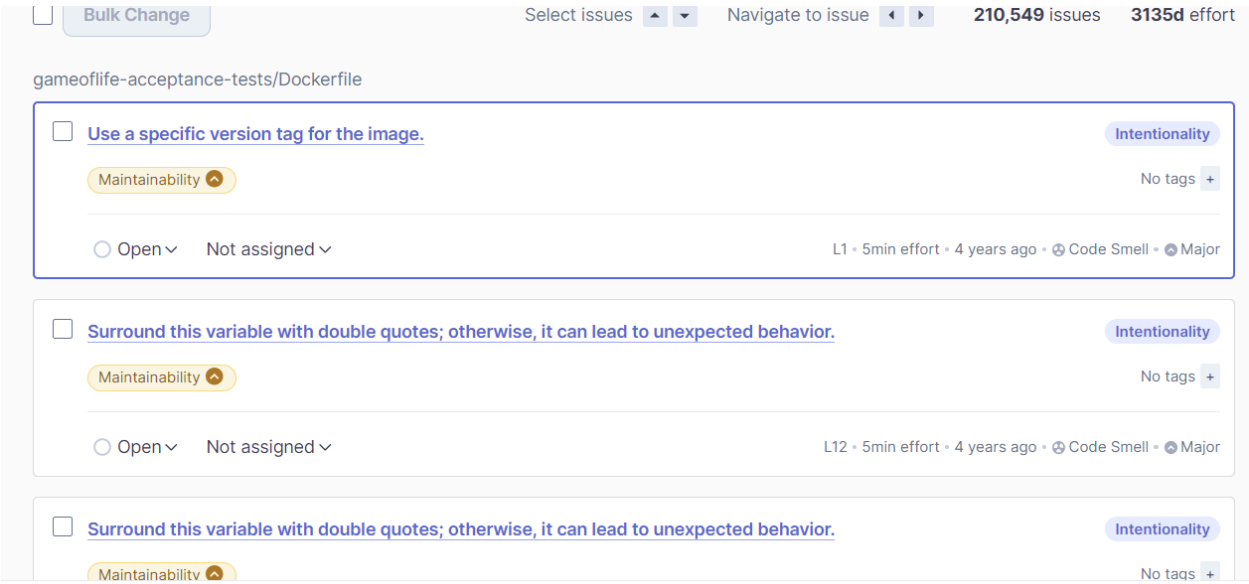


14. Code Problems

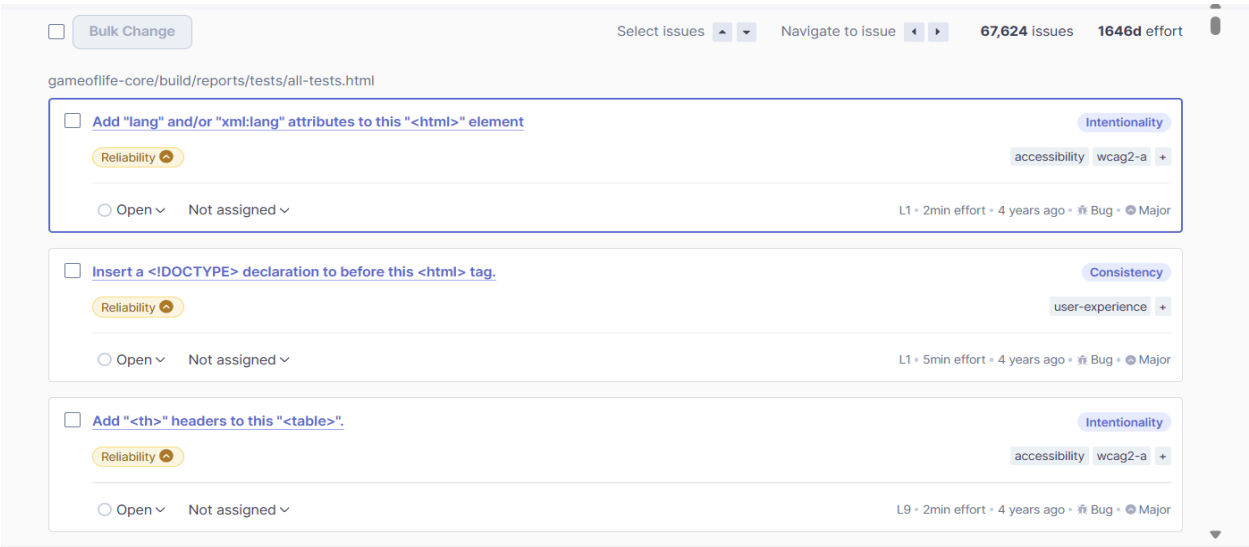
- Consistency



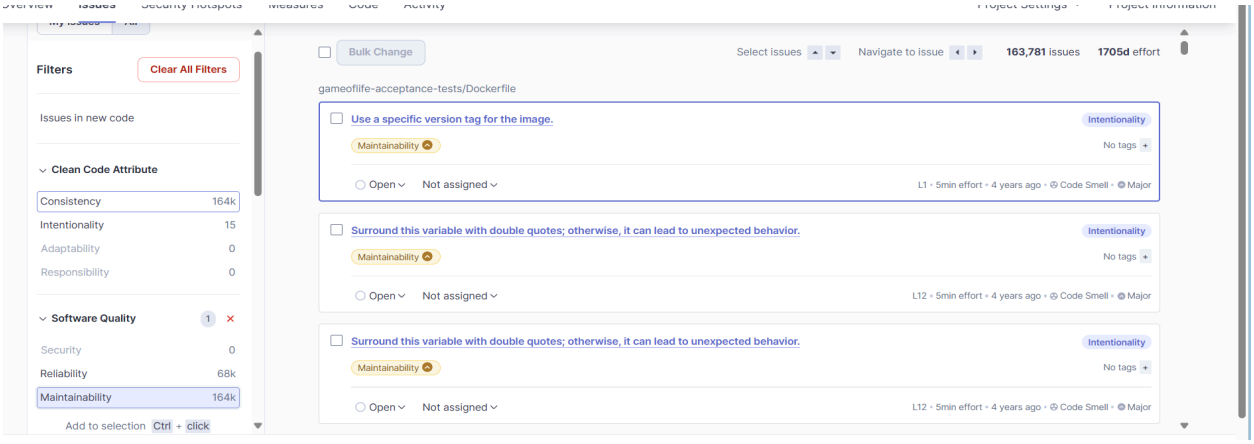
- Intentionality



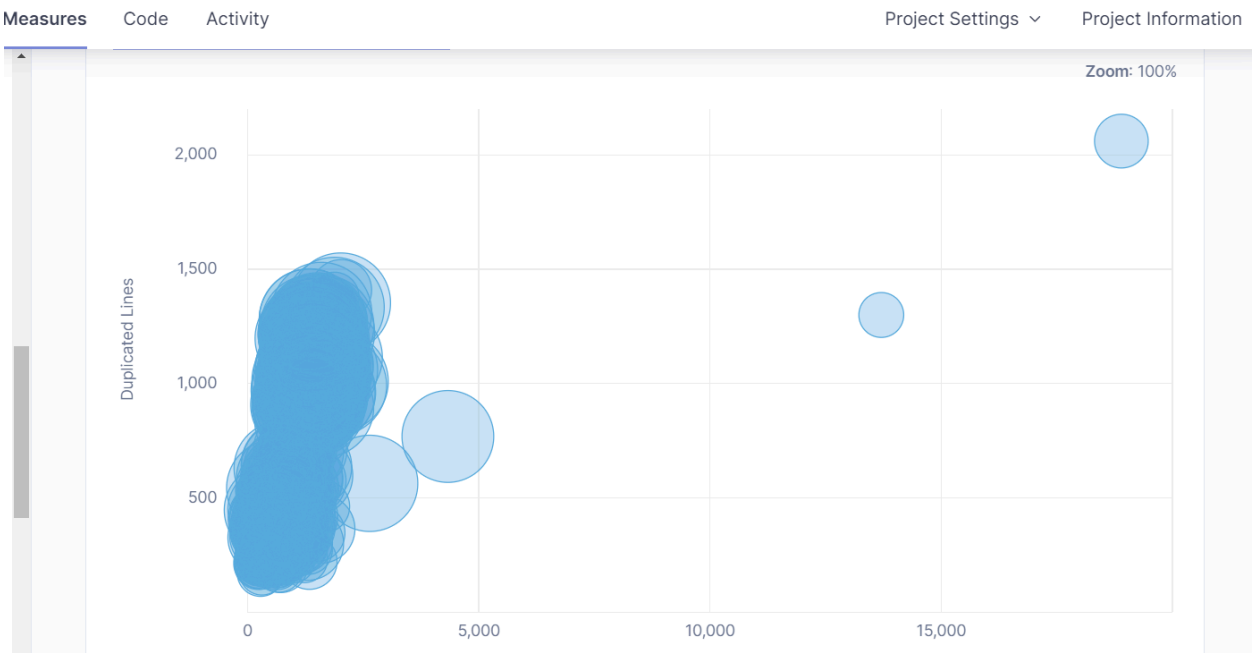
- Bugs



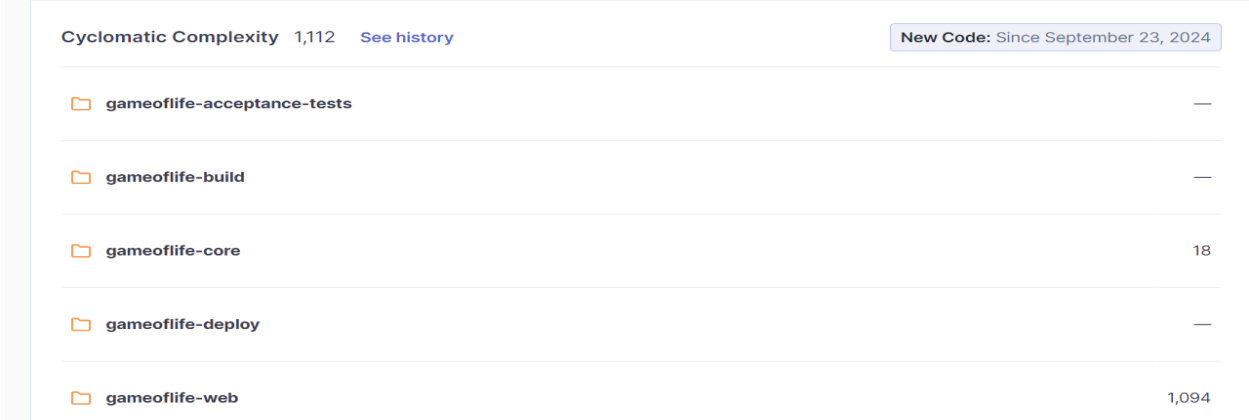
- Code Smells



- Duplications



- Cyclomatic Complexities



In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

In this experiment, we integrated Jenkins with SonarQube to implement automated code quality checks within our CI/CD pipeline. The process began with deploying SonarQube using Docker, followed by setting up a dedicated project within SonarQube to perform detailed code quality analysis. We then configured Jenkins by installing the SonarQube Scanner plugin, which allowed Jenkins to communicate with SonarQube. This included adding the necessary SonarQube server details and configuring the scanner tool to analyze code efficiently.

A Jenkins pipeline was developed to automate key tasks, such as cloning the code from a GitHub repository and triggering the SonarQube analysis on the codebase. Through this

integration, the pipeline continuously monitors code quality, providing detailed reports on issues such as bugs, code smells, and security vulnerabilities. This setup not only ensures early detection of potential problems but also promotes consistent improvements in code quality throughout the development lifecycle.