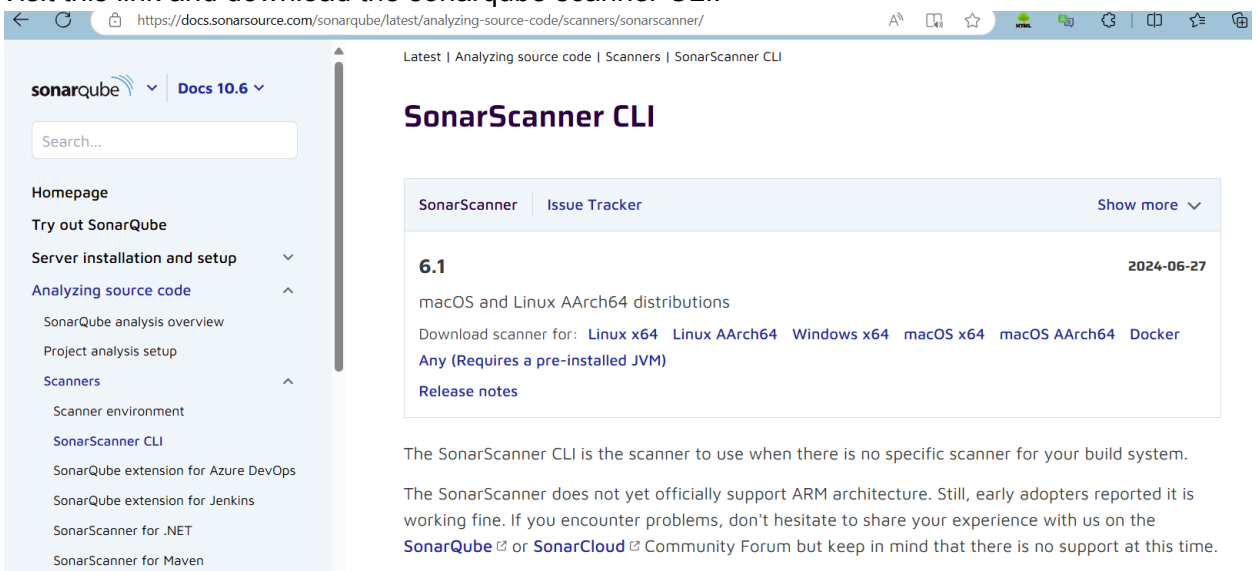
08 Advanced DevOps Lab

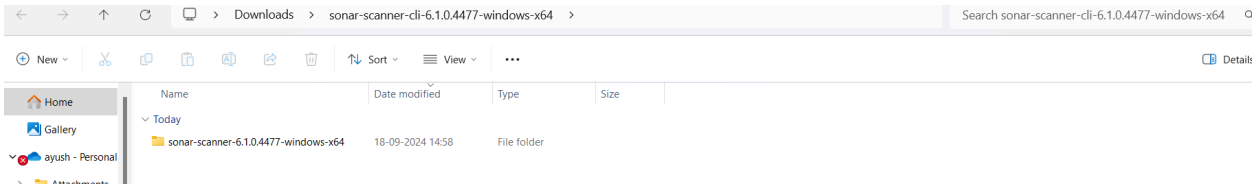
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

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.

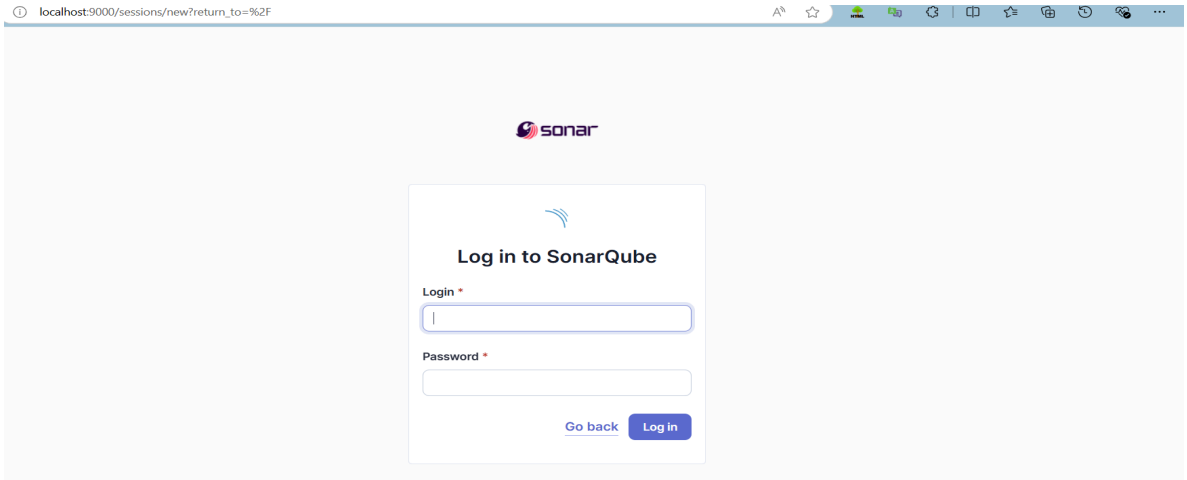


1. Install sonarqube image

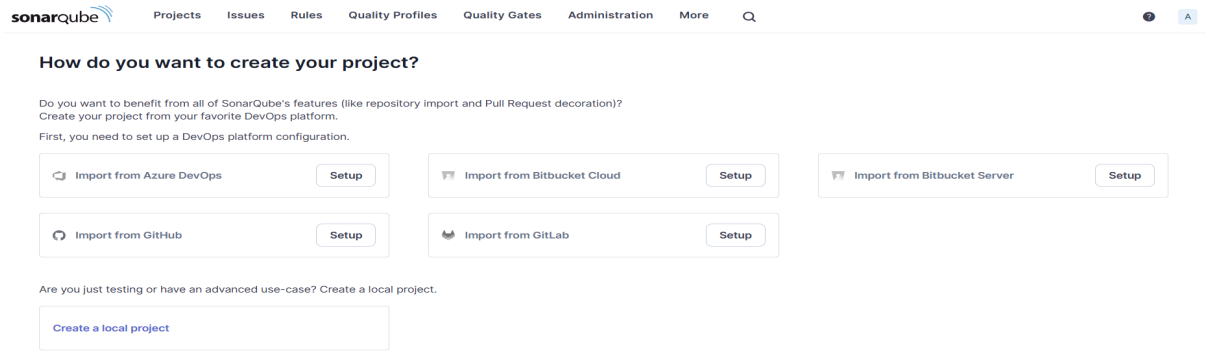
Command: **docker pull sonarqube**

```
C:\Users\Ayush Maurya>docker pull sonarqube
Using default tag: latest
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
docker.io/library/sonarqube:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview sonarqube

C:\Users\Ayush Maurya>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
dfe47cea99897c7099833dc0d7fa99279ef09d4f3038622910a74df2630afed5
```

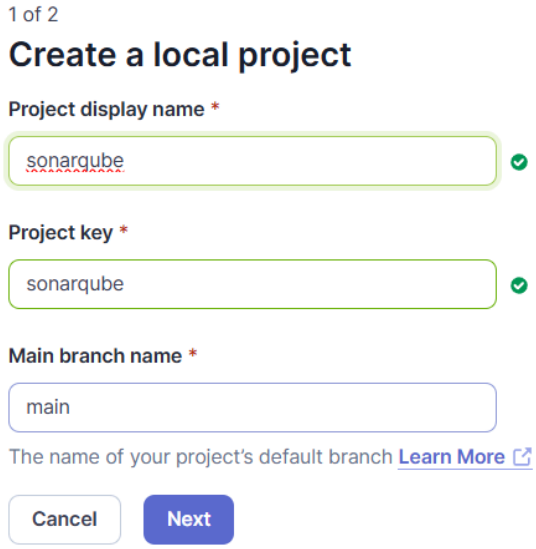
2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

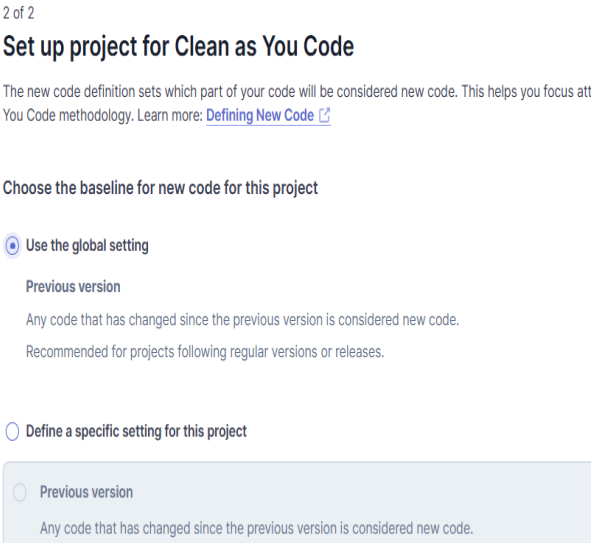


3. Login to SonarQube using username admin and password admin.

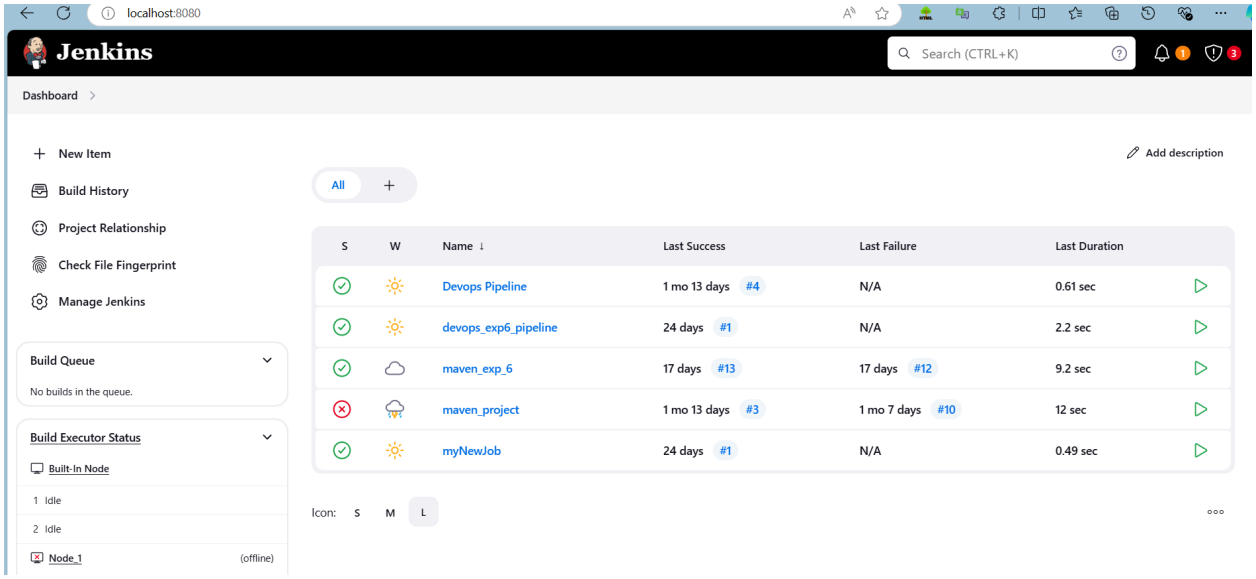


4. Create a manual project in SonarQube with the name sonarqube

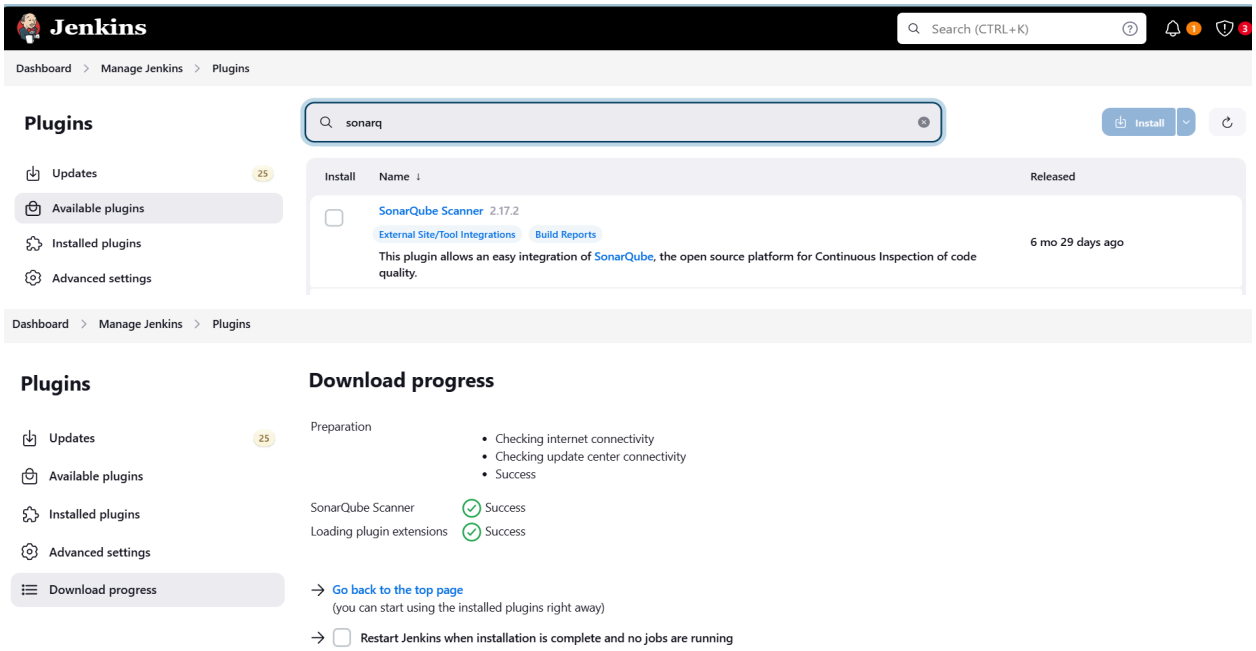




5. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



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

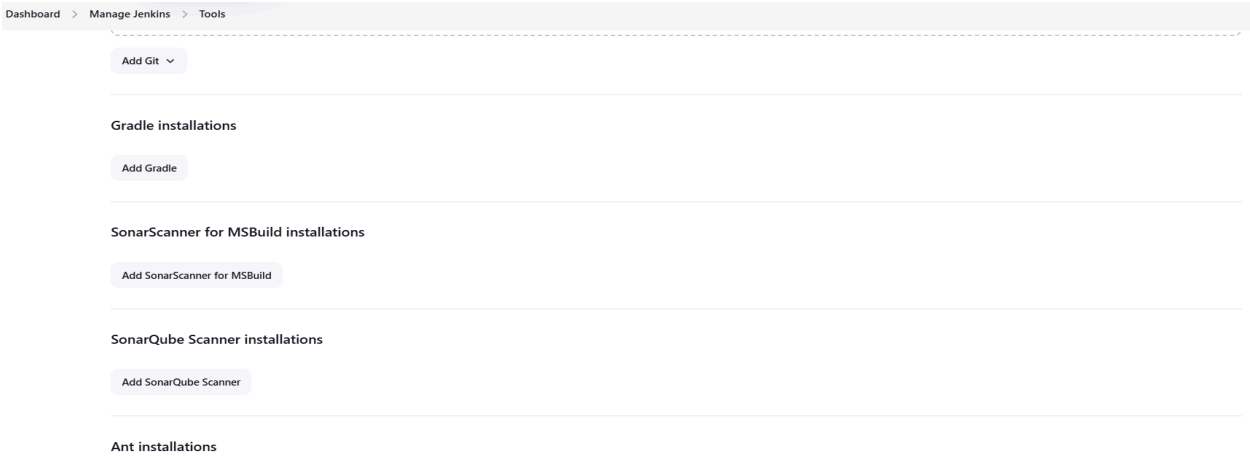


7. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.
Enter the Server Authentication token if needed.

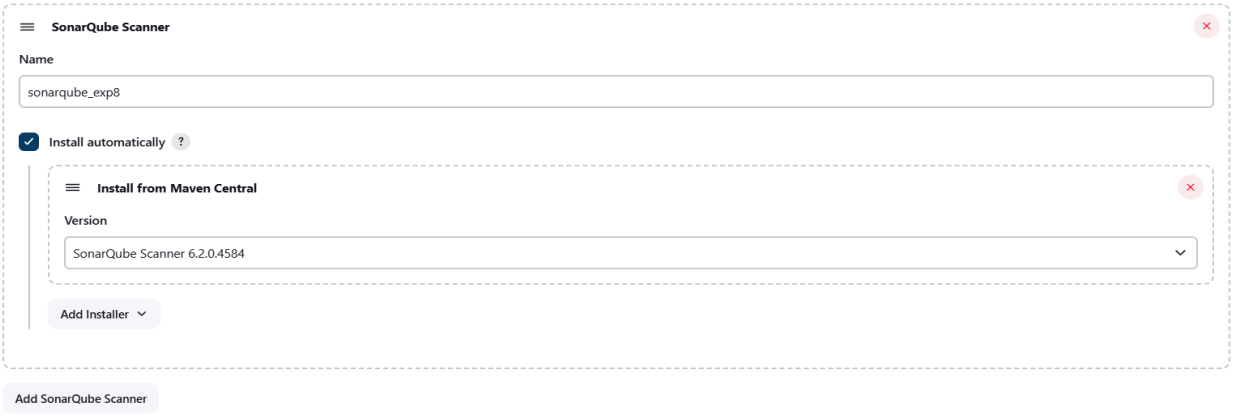
In SonarQube installations: Under **Name** add <project name of sonarqube> for me **adv_devops_7_sonarqube**
In **Server URL** Default is **http://localhost:9000**

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

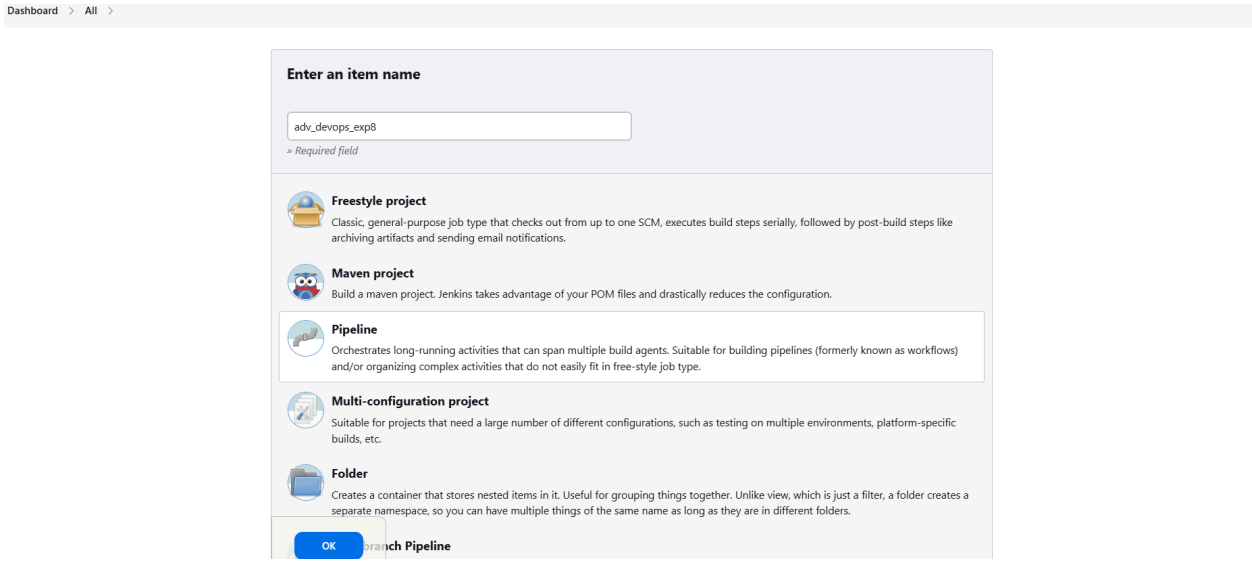
Dashboard > Manage Jenkins > Tools



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.

Pipeline

Definition

Pipeline script

Script

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

    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {  // Ensure this matches the SonarQube environment name in Jenkins
            bat """
                "C:\\Users\\Ayush Maurya\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.
                -D sonar.login=admin ^
                -D sonar.password=Ayush3114 ^
                -D sonar.projectKey=sonarqube ^
                -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
                -D sonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

11. Build project

Dashboard > adv_devops_exp8 >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- SonarQube
- Stages
- Rename
- Pipeline Syntax

Build History trend

Filter...

#9

Sep 18, 2024, 4:14 PM

adv_devops_exp8

Stage View

	Cloning the GitHub Repo	SonarQube analysis
Average stage times: (Average full run time: ~6min 4s)	3s	40s
#9 Sep 18 16:14 No Changes	2s	6min 2s
#8 Sep 18 16:12 No Changes	2s	1s failed
#7 Sep 18 16:10 No Changes	2s	120ms failed

12. Check console

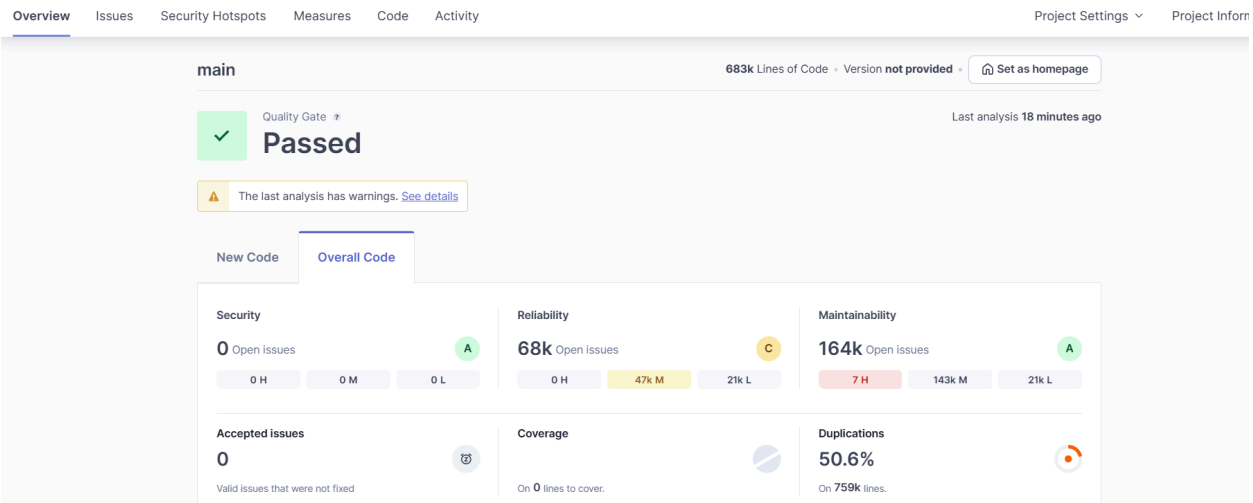
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#9'
- Timings
- Git Build Data
- Pipeline Overview
- Pipeline Console
- Replay
- Pipeline Steps
- Workspaces
- Previous Build

Console Output

Skipping 4,246 KB.. Full Log

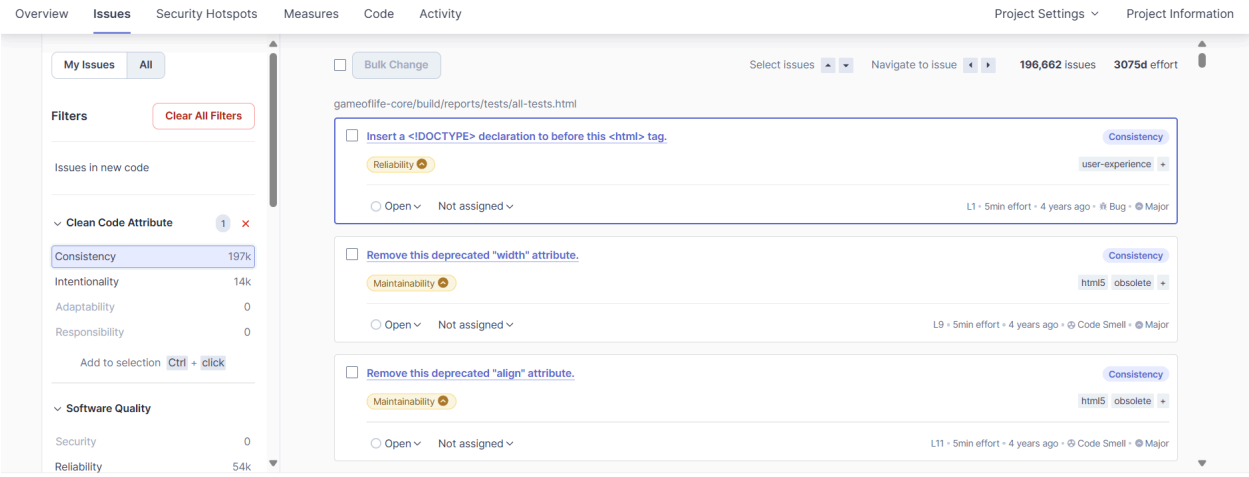
```
16:19:49.751 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
512. Keep only the first 100 references.
16:19:49.751 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
248. Keep only the first 100 references.
16:19:49.751 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.
16:19:49.751 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.
16:19:49.751 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.
16:19:49.751 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.
16:19:49.751 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.
16:19:49.751 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.
16:19:49.752 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.
16:19:49.752 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.
16:19:49.752 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
548. Keep only the first 100 references.
16:19:49.752 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
543. Keep only the first 100 references.
16:19:49.752 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
152. Keep only the first 100 references.
16:19:49.752 WARN  Too many duplication references on file gameoflife-web/tools/jmeter/docs/api/org/apache/jmeter/visualizers/PropertyControlGui.html for block at line
```

13. Now, check the project in SonarQube

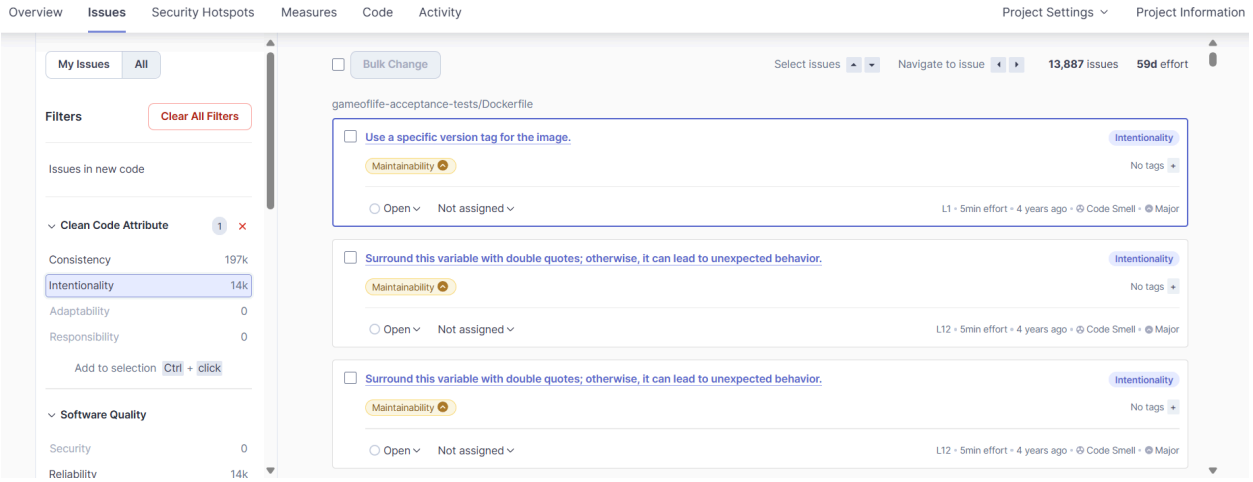


14. Code Problems

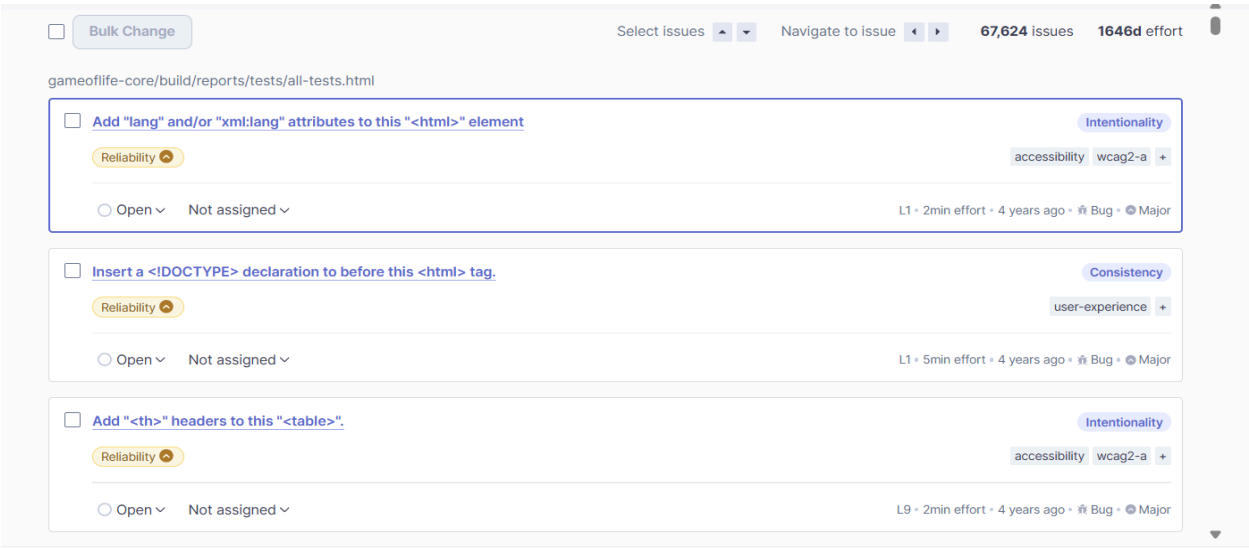
- Consistency



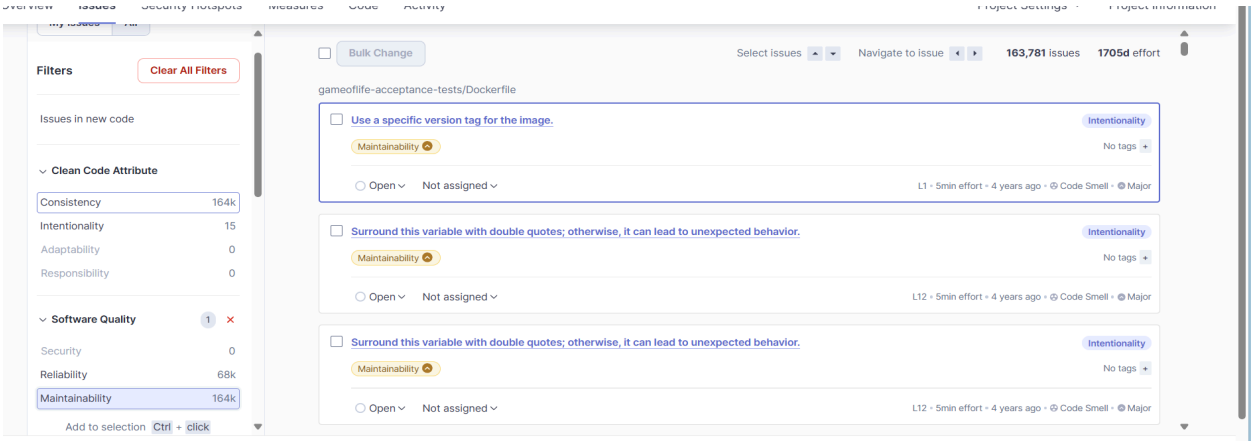
- Intentionality



- Bugs

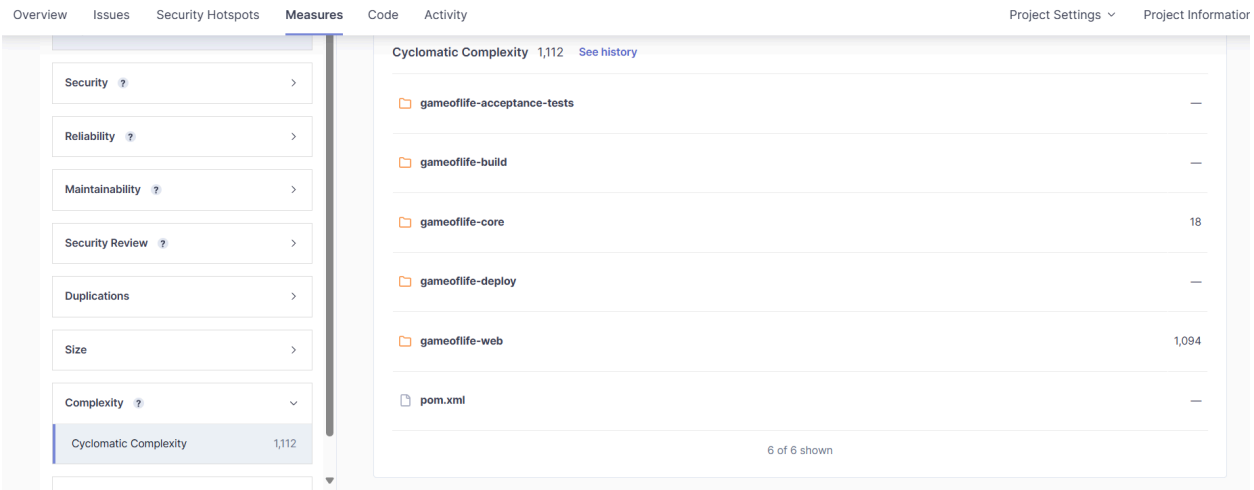


- Code Smells



- Duplications

- Cyclomatic Complexities



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

Conclusion:

This experiment established a seamless integration of Jenkins with SonarQube to automate code quality assessments within the CI/CD pipeline. SonarQube was deployed using Docker, and after setting up a project, it was configured to analyze the codebase for potential quality issues. Jenkins was configured with the necessary SonarQube plugins, allowing automated code checks through a pipeline that cloned a GitHub repository and performed a SonarQube scan. This integration ensures continuous monitoring throughout the development cycle, effectively identifying and addressing bugs, code smells, and security vulnerabilities, thereby enhancing code quality and security.