Shravani Anil Patil **D15A-38**

Advanced Devops
Experiment No:08

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.**

THEORY:

Static Application Security Testing (SAST) :SAST is a methodology for testing an application's source code to identify security vulnerabilities before the code is compiled. This type of testing, also referred to as white-box testing, helps improve application security by finding weaknesses early in development.

Problems SAST Solves

- **Early Detection**: SAST finds vulnerabilities early in the Software Development Life Cycle (SDLC), allowing developers to fix issues without affecting builds or passing vulnerabilities to the final release.
- **Real-Time Feedback**: Developers receive immediate feedback during coding, helping them address security issues before moving to the next stage of development.
- **Graphical Representations**: SAST tools often provide visual aids to help developers navigate the code and identify the exact location of vulnerabilities, offering suggestions for fixes.
- **Regular Scanning**: SAST tools can be configured to scan code regularly, such as during daily builds, code check-ins, or before releases.

Importance of SAST

- **Resource Efficiency**: With a larger number of developers than security experts, SAST allows full codebase analysis quickly and efficiently, without relying on manual code reviews.
- **Speed**: SAST tools can analyze millions of lines of code within minutes, detecting critical vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting (XSS) with high accuracy.

CI/CD Pipeline

A Continuous Integration/Continuous Delivery (CI/CD) pipeline is a sequence of automated tasks designed to build, test, and deploy new software versions rapidly and consistently. It plays a crucial role in DevOps practices, ensuring fast and reliable software releases.

SonarQube

SonarQube is an open-source platform from SonarSource that performs continuous code quality inspections through static code analysis. It identifies bugs, code smells, security vulnerabilities, and code duplications in a wide range of programming languages. SonarQube is extendable with plugins and integrates seamlessly into CI/CD pipelines.

Benefits of SonarQube

Sustainability: By reducing complexity and vulnerabilities, SonarQube extends the lifespan of applications and helps maintain cleaner code.

Increased Productivity: SonarQube minimizes maintenance costs and risks, resulting in fewer code changes and a more stable codebase.

Quality Code: Ensures code quality checks are integrated into the development process.

Error Detection: Automatically identifies coding errors and alerts developers to resolve them before moving to production.

Consistency: Helps maintain consistent code quality by detecting and reporting violations of coding standards.

Business Scaling: SonarQube supports scaling as the business grows without any restrictions.

Implementation:

Prerequisites

1. Jenkins installed on your machine.
2. Docker installed to run SonarQube.
3. SonarQube installed via Docker

1. Set Up Jenkins

- Open Jenkins Dashboard on localhost:8080 or your configured port

- Install the necessary plugins:
- SonarQube Scanner Plugin

2. Run SonarQube in Docker

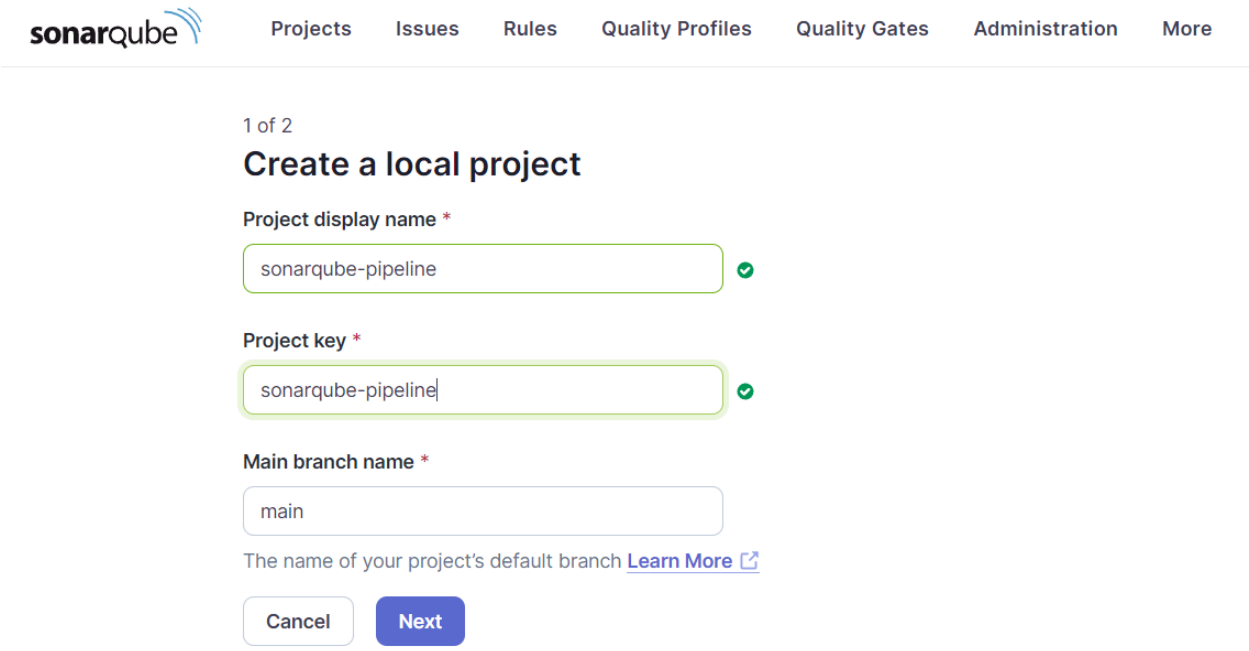
Run the following command to start SonarQube in a Docker container: command

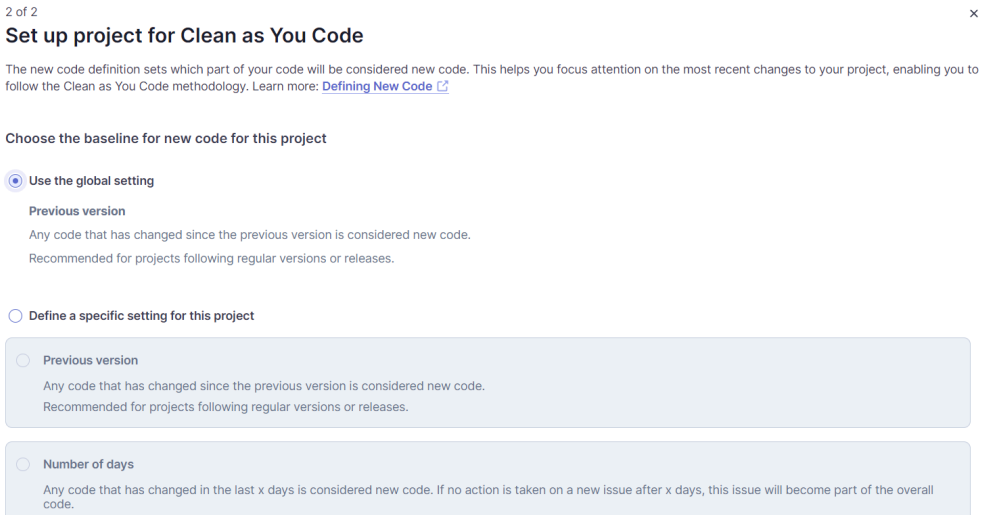
:

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

- Check SonarQube status at http://localhost:9000.
- Login with your credentials:

Step 1: Log in to sonarqube portal and create a local project.



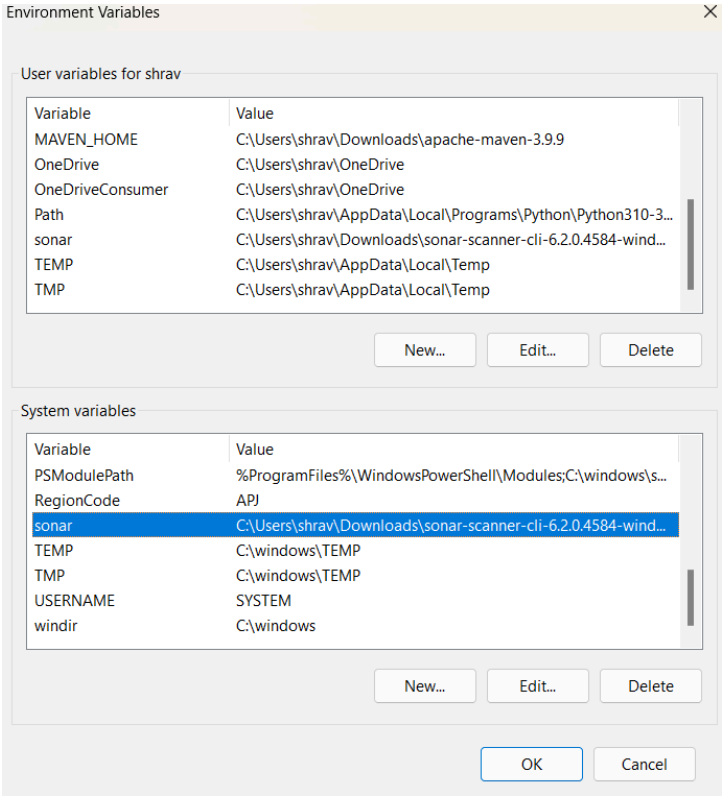


Step 2: Go to [download_sonarscanner]() to download sonar scanner

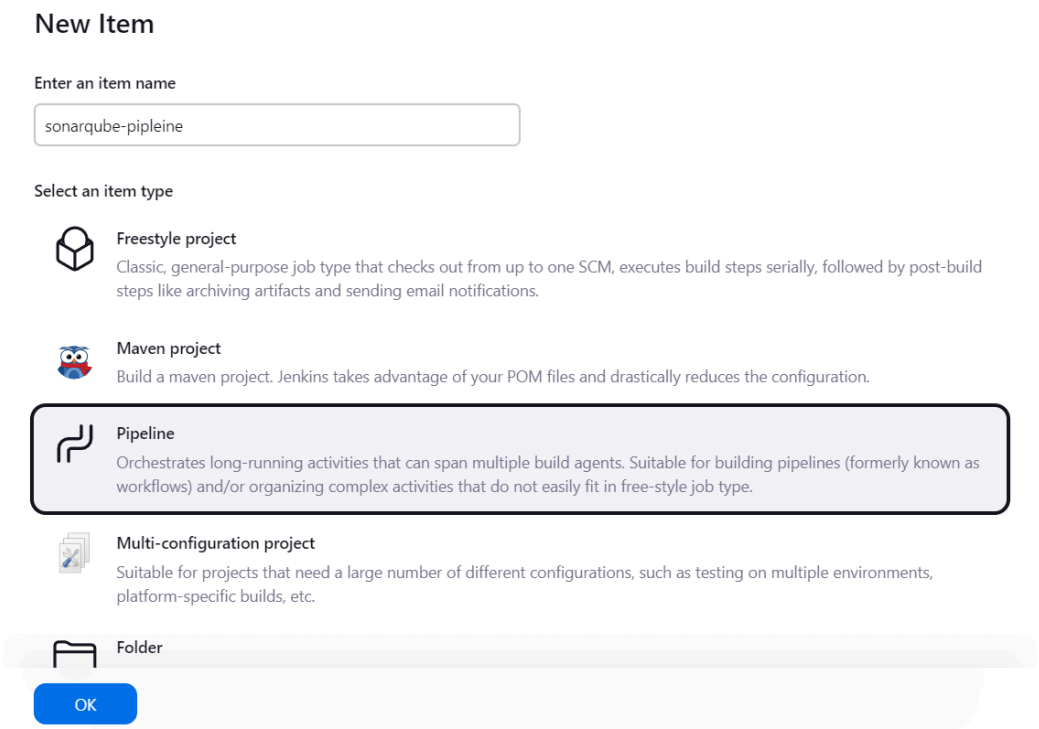
After the download is complete, extract the file and copy the path to bin folder

Go to environment variables, system variables and click on path

Add a new path, paste the path copied earlier.

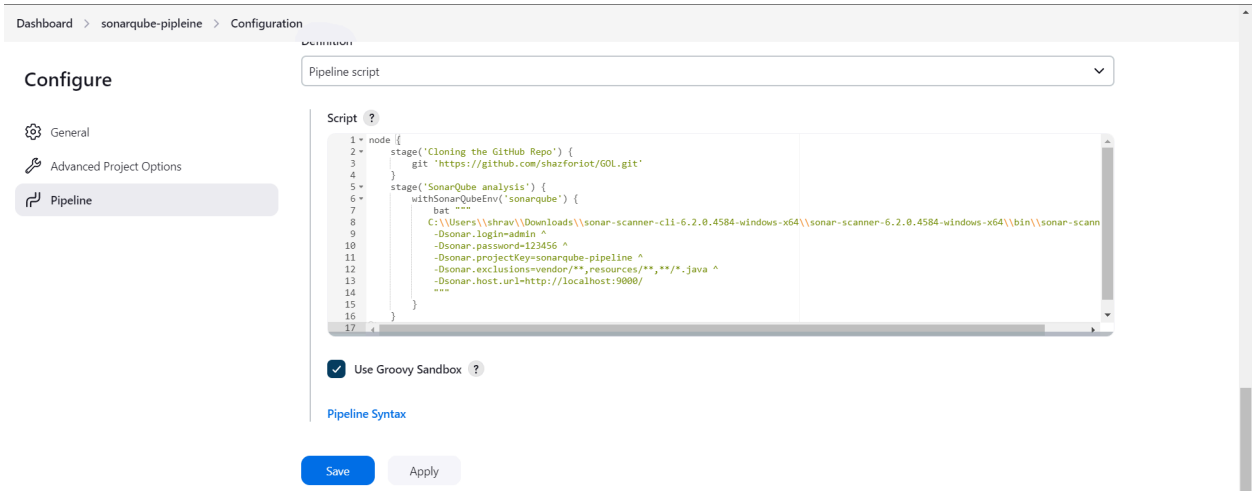


Step 3: Create a New Item in Jenkins, choose Pipeline.

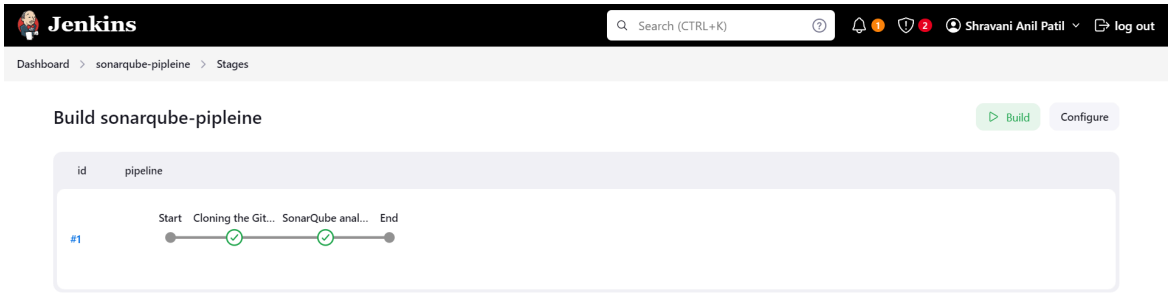


Add pipeline script :

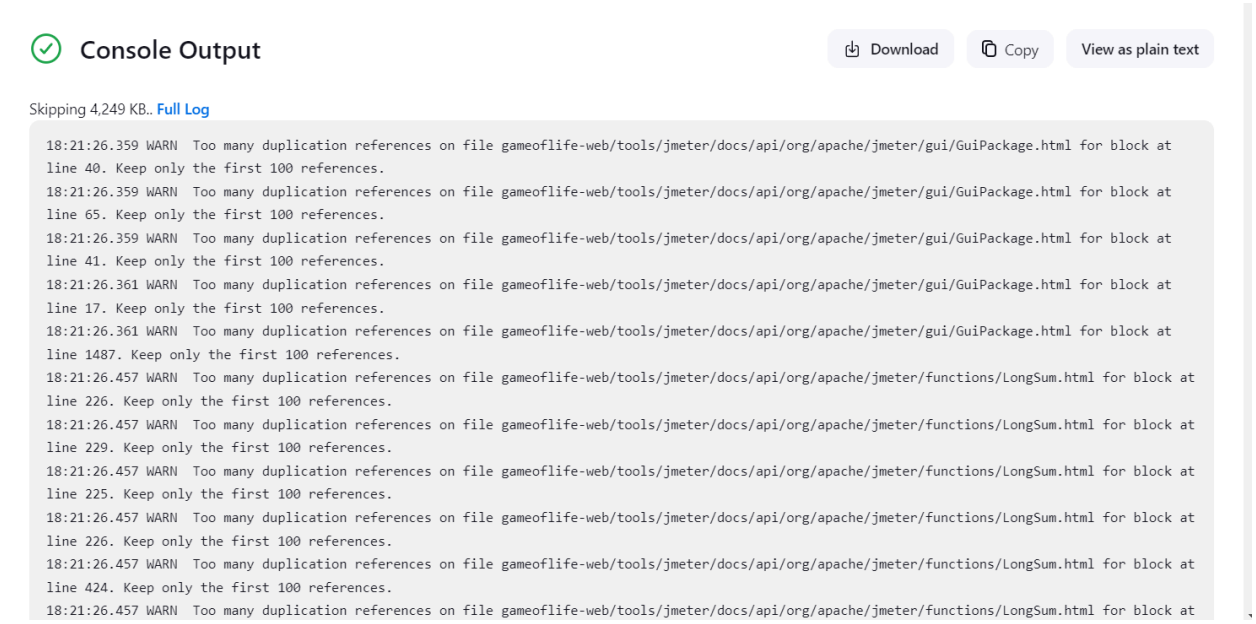
```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """

C:\\Users\\shrav\\Downloads\\sonar-scanner-cli-6.2.0.4584-windows-x64\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat ^
            -Dsonar.login=admin ^
            -Dsonar.password=123456 ^
            -Dsonar.projectKey=sonarqube-pipeline ^
            -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
            -Dsonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

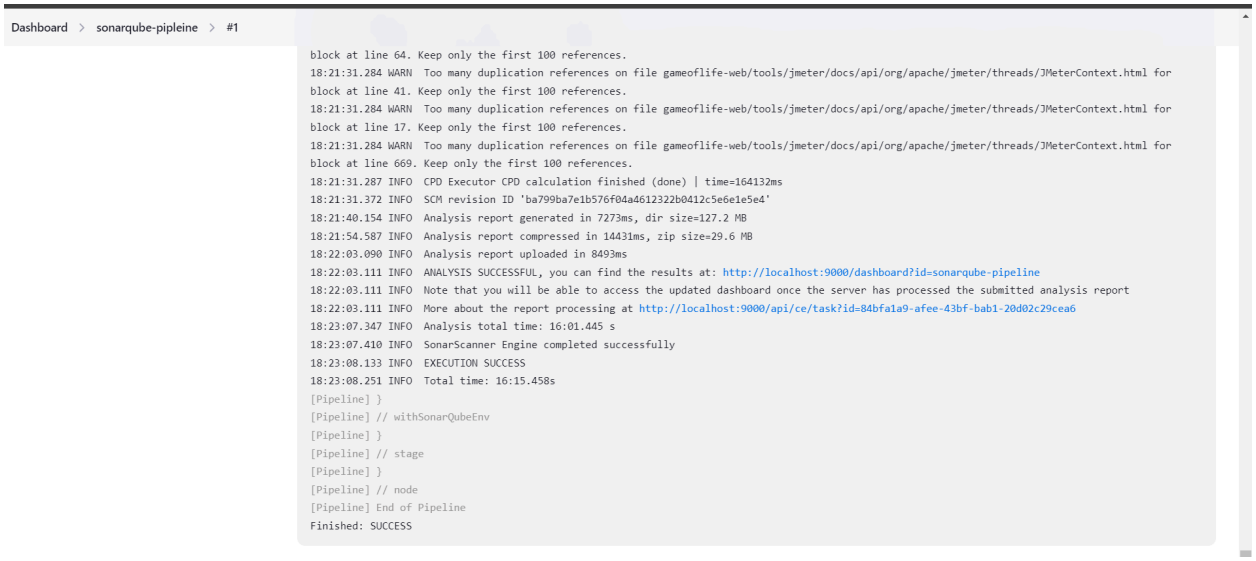


Step 4: Save the pipeline and build it.

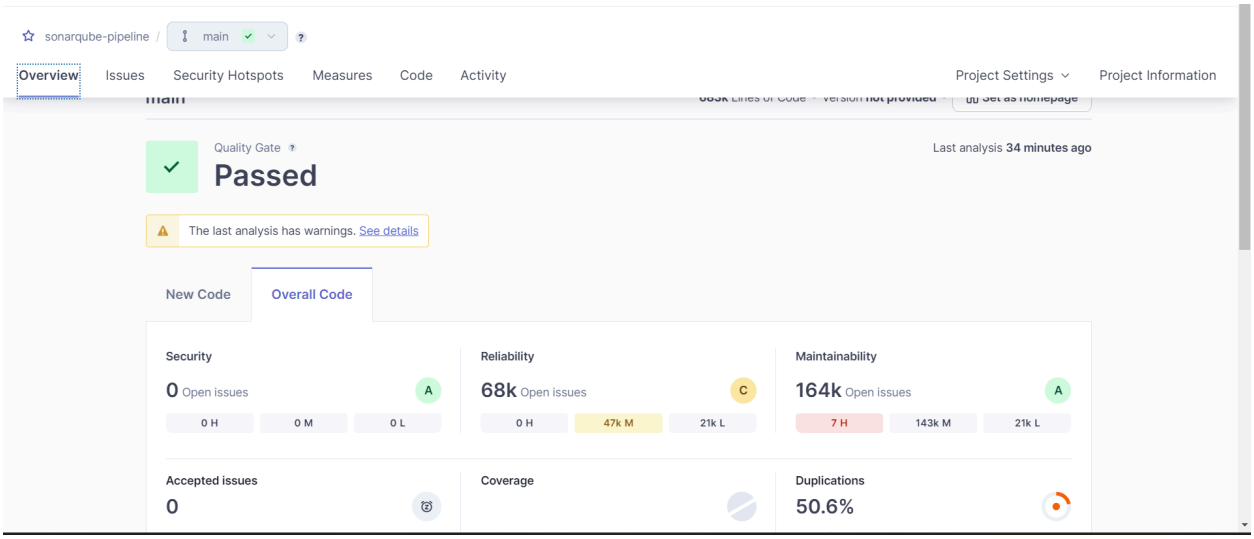


Console output:

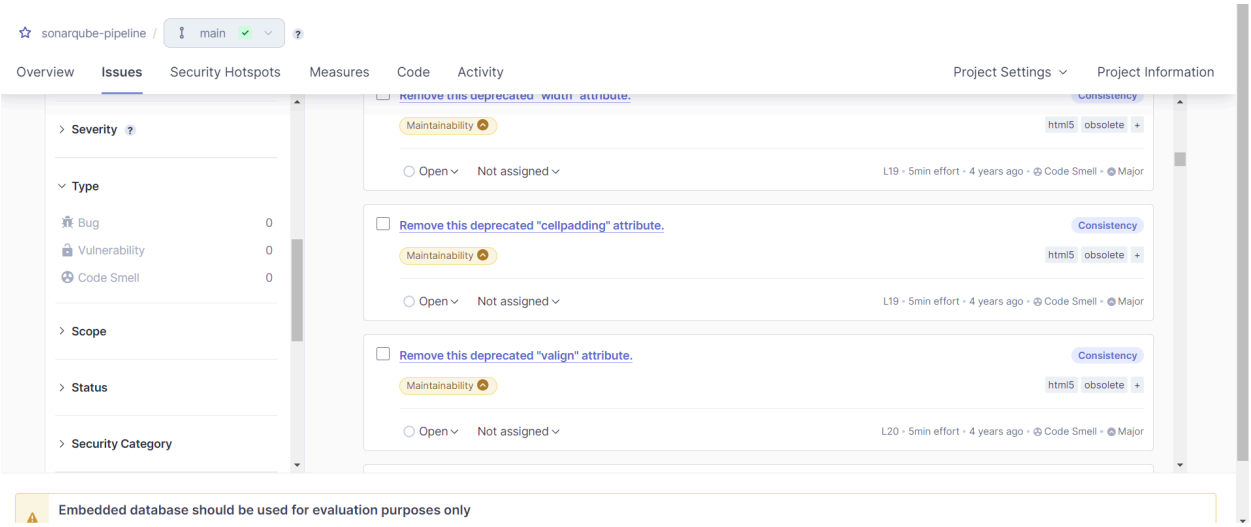


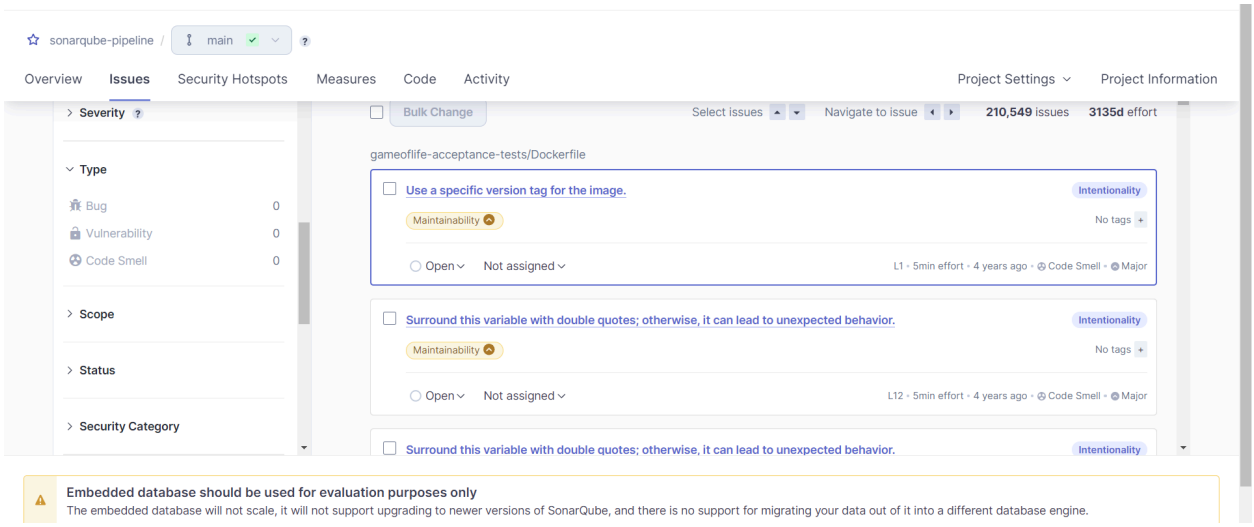


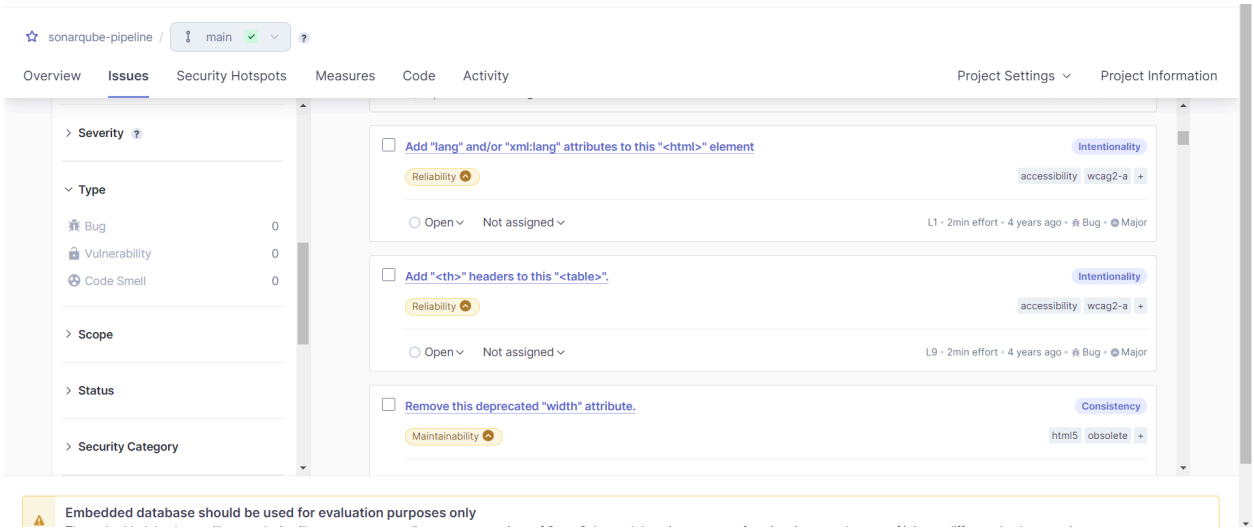
Step 5: After that, check the project in SonarQube

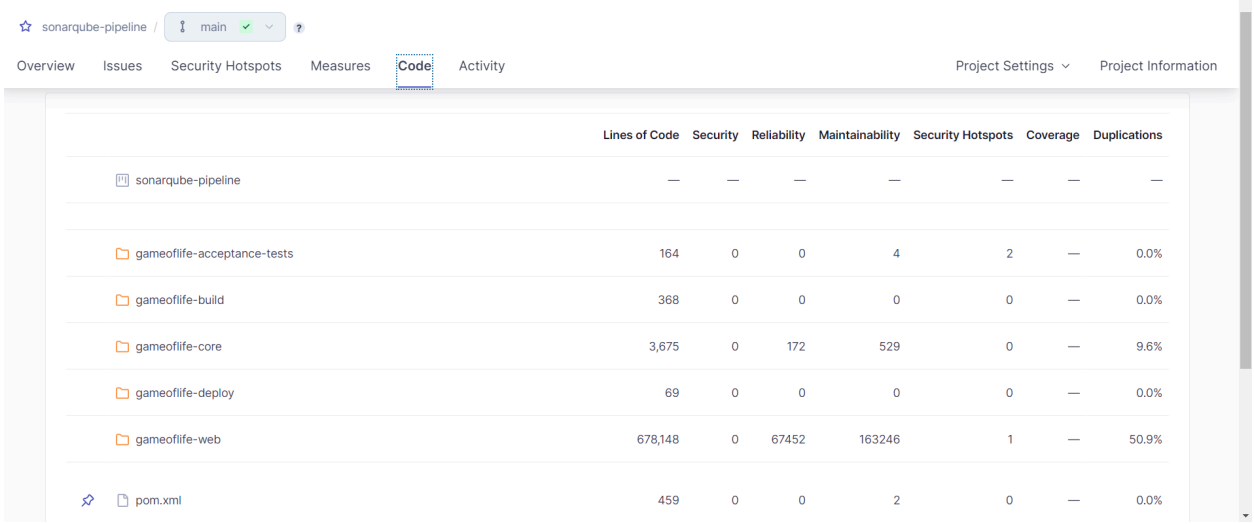


Under different tabs, check all different issues with the code

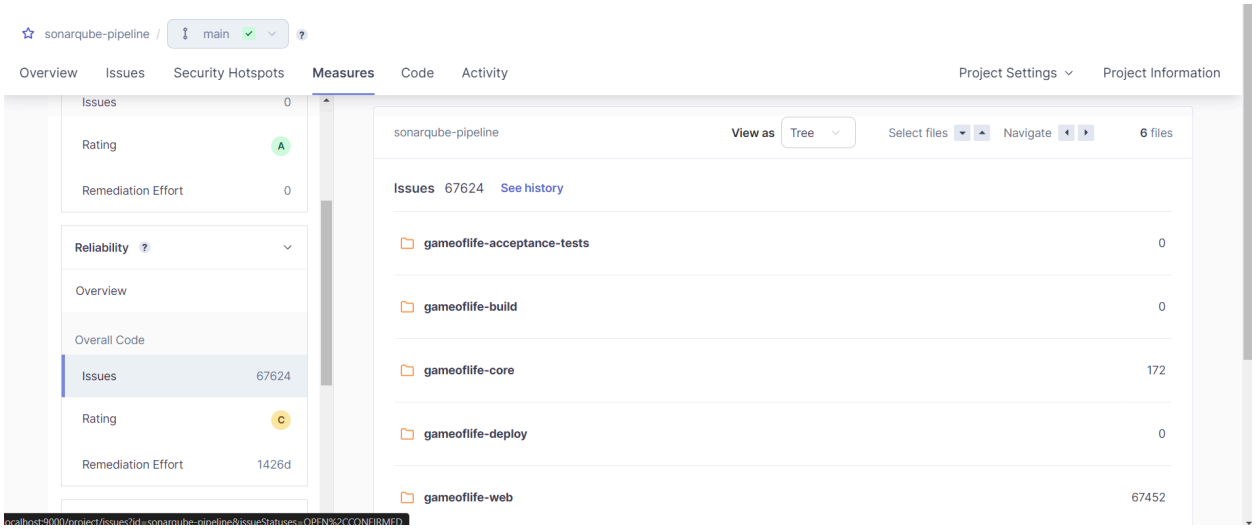


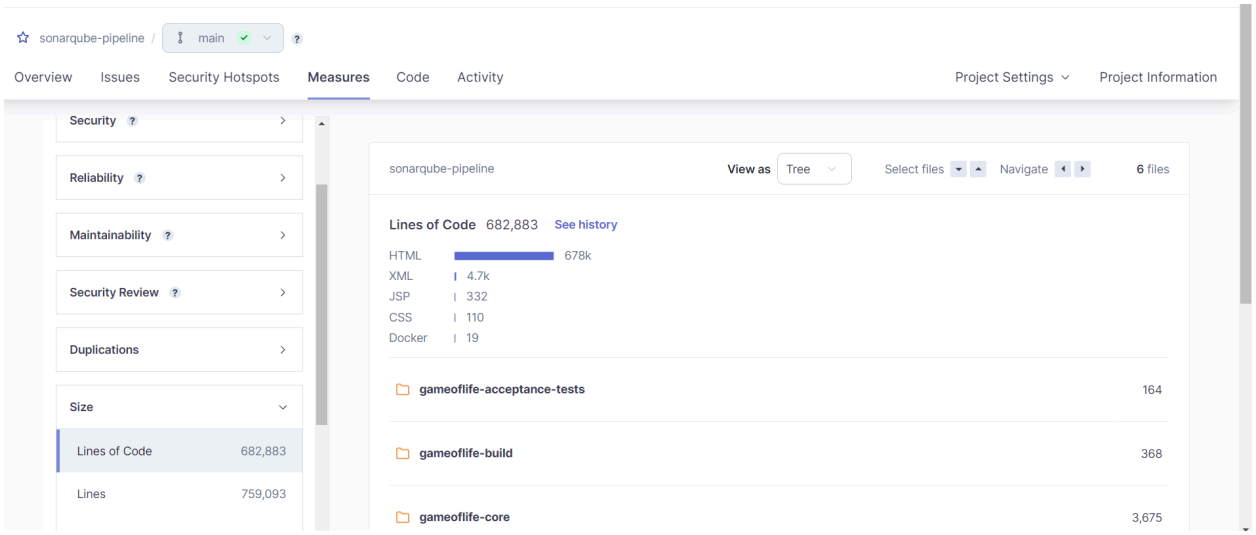


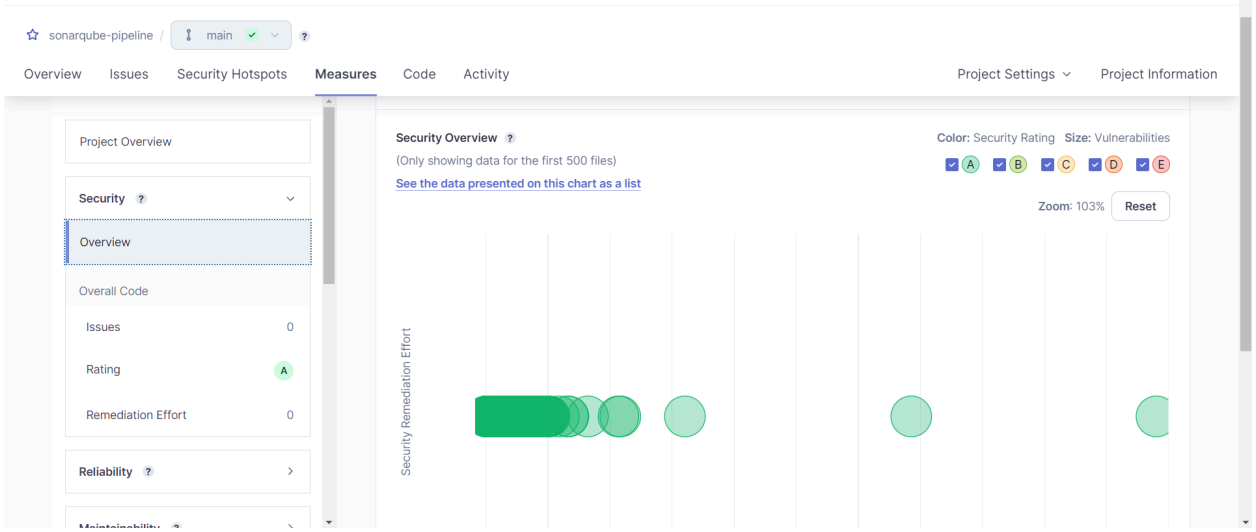




	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
sonarqube-pipeline	—	—	—	—	—	—	—
gameoflife-acceptance-tests	164	0	0	4	2	—	0.0%
gameoflife-build	368	0	0	0	0	—	0.0%
gameoflife-core	3,675	0	172	529	0	—	9.6%
gameoflife-deploy	69	0	0	0	0	—	0.0%
gameoflife-web	678,148	0	67452	163246	1	—	50.9%
pom.xml	459	0	0	2	0	—	0.0%


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Issues 0
Rating A
Remediation Effort 0
Reliability
Overview
Overall Code
Issues 67624
Rating C
Remediation Effort 1426d
sonarqube-pipeline
View as Tree
Select files Navigate
6 files
Issues 67624 See history
gameoflife-acceptance-tests 0
gameoflife-build 0
gameoflife-core 172
gameoflife-deploy 0
gameoflife-web 67452


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Security
Reliability
Maintainability
Security Review
Duplications
Size
Lines of Code 682,883
Lines 759,093
sonarqube-pipeline
View as Tree
Select files Navigate
6 files
Lines of Code 682,883 See history
HTML 678k
XML 4.7k
JSP 332
CSS 110
Docker 19
gameoflife-acceptance-tests 164
gameoflife-build 368
gameoflife-core 3,675


sonarqube-pipeline / main
Overview Issues Security Hotspots Measures Code Activity
Project Settings Project Information
Project Overview
Security
Overview
Overall Code
Issues 0
Rating A
Remediation Effort 0
Reliability
Security Overview
(Only showing data for the first 500 files)
See the data presented on this chart as a list
Color: Security Rating Size: Vulnerabilities
A B C D E
Zoom: 103% Reset
Security Remediation Effort