

**Aim:** Create a Jenkins CI/CD 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:

### What is SAST?

**Static Application Security Testing (SAST)** is a methodology that analyzes source code to identify security vulnerabilities before compilation. It is often referred to as white box testing and helps developers detect issues early in the software development lifecycle (SDLC).

### Problems SAST Solves

1. **Early Detection:** Identifies vulnerabilities in the initial development stages, reducing later risks.
2. **Real-Time Feedback:** Provides immediate insights, allowing developers to fix issues before moving forward.
3. **Code Navigation:** Offers visual representations of vulnerabilities for easier code understanding.
4. **Guidance on Fixes:** Suggests specific remediation steps without requiring deep security expertise.
5. **Comprehensive Coverage:** Analyzes the entire codebase quickly, outperforming manual reviews.
6. **Regular Scanning:** Ensures continuous security assessment through scheduled scans during builds or releases.

### Importance of SAST

- **Resource Efficiency:** Automates code reviews, addressing the resource gap between developers and security staff.
- **Speed:** Processes millions of lines of code in minutes, identifying critical vulnerabilities.
- **Proactive Security:** Integrates security into the development process, preventing vulnerabilities from being overlooked.

### What is a CI/CD Pipeline?

A **CI/CD Pipeline** refers to Continuous Integration and Continuous Delivery, automating software development tasks. It includes stages such as coding, building, testing, and deploying, ensuring each step is completed sequentially for efficient releases.

### What is SonarQube?

**SonarQube** is an open-source platform for continuous code quality inspection. It performs static code analysis to generate reports on bugs, vulnerabilities, and code duplications across various programming languages.

## Benefits of SonarQube

- **Sustainability:** Optimizes application lifecycle by reducing complexity and vulnerabilities.
- **Increased Productivity:** Minimizes maintenance efforts and costs.
- **Quality Control:** Integrates code quality checks into development.
- **Error Detection:** Alerts developers to fix issues before release.
- **Scalability:** Supports multiple projects without restrictions.
- **Skill Enhancement:** Provides regular feedback to improve developer skills.

## Prerequisites:

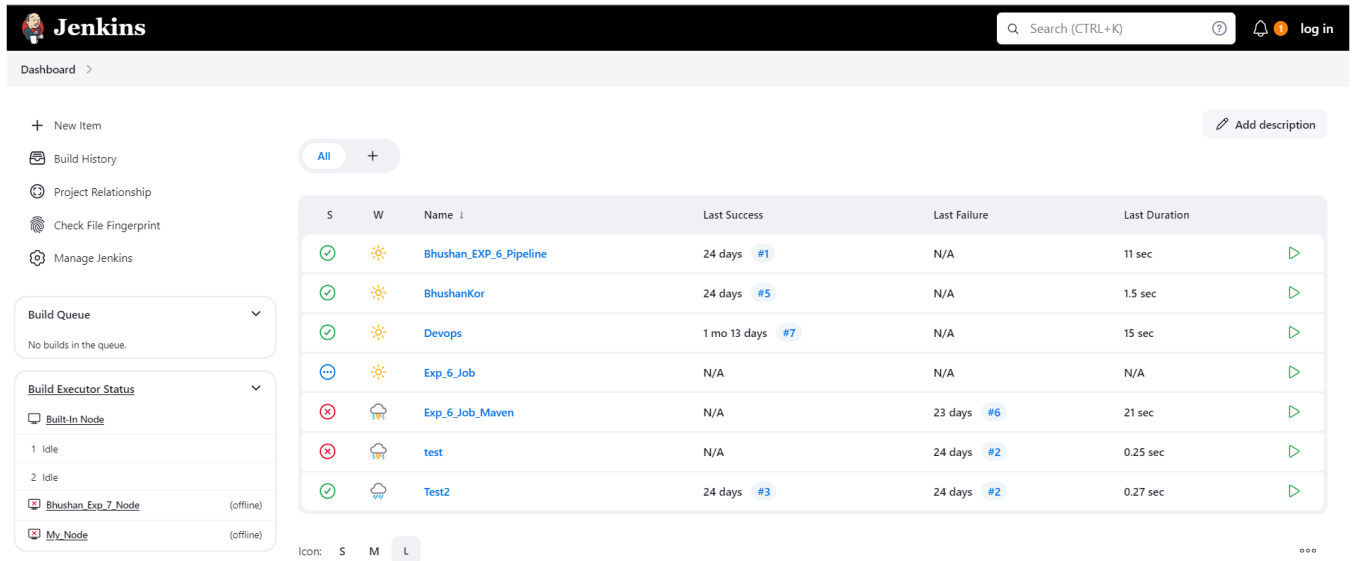
- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

## Download The SonarQube CLI according to your system :

<https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/>

The screenshot shows the SonarScanner CLI documentation page. The left sidebar contains a navigation menu with links to 'Homepage', 'Try out SonarQube', 'Server installation and setup', 'Analyzing source code', 'Scanners', 'Scanner environment', 'SonarScanner CLI', 'SonarQube extension for Azure DevOps', 'SonarQube extension for Jenkins', 'SonarScanner for .NET', 'SonarScanner for Maven', 'SonarScanner for Gradle', 'SonarScanner for NPM', 'SonarScanner for Ant (Deprecated)', and 'SonarScanner for Python (Beta)'. The main content area is titled 'SonarScanner CLI' and features a '6.2' version section with a '2024-09-17' date. It includes a 'Download scanner for:' section with links for 'Linux x64', 'Linux AArch64', 'Windows x64', 'macOS x64', 'macOS AArch64', and 'Docker'. Below this is a 'Release notes' section. The page also has a 'START FREE' button in the top right corner and an 'On this page' section on the right side listing various topics like 'Configuring your project', 'Running SonarScanner CLI from the zip file', 'Running SonarScanner CLI from the Docker image', 'Scanning C, C++, or Objective-C projects', 'Sample projects', 'Alternatives to sonar-project.properties', 'Alternate analysis directory', 'Advanced configuration', and 'Troubleshooting'. A note at the bottom states: 'The SonarScanners run on code that is checked out. See Verifying the code checkout step of your build.'

**Step 1:** Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



The screenshot shows the Jenkins Dashboard. On the left, there is a sidebar with navigation links: New Item, Build History, Project Relationship, Check File Fingerprint, and Manage Jenkins. Below these are two expandable sections: 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing 'Built-In Node' with 1 idle and 2 offline executors). The main area displays a table of build jobs. A filter button 'All' is present. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. The jobs listed are Bhushan\_EXP\_6\_Pipeline, BhushanKor, Devops, Exp\_6\_Job, Exp\_6\_Job\_Maven, test, and Test2. At the bottom, there are icons for 'Icon: S M L' and a menu icon '...'.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	Bhushan_EXP_6_Pipeline	24 days #1	N/A	11 sec
✓	☀	BhushanKor	24 days #5	N/A	1.5 sec
✓	☀	Devops	1 mo 13 days #7	N/A	15 sec
...	☀	Exp_6_Job	N/A	N/A	N/A
✗	☁	Exp_6_Job_Maven	N/A	23 days #6	21 sec
✗	☁	test	N/A	24 days #2	0.25 sec
✓	☁	Test2	24 days #3	24 days #2	0.27 sec

**Step 2:** Run SonarQube in a Docker container using this command

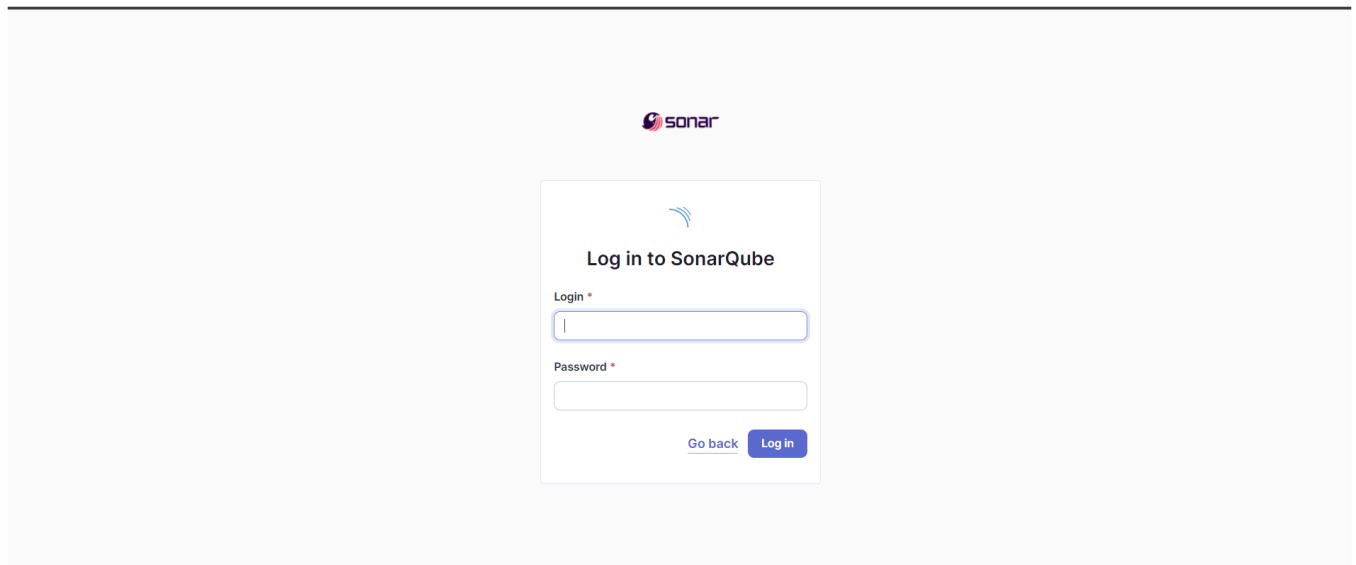
```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

```
Microsoft Windows [Version 10.0.22621.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\INFT505-11>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
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
d72b183b1866cea7ecdb976a63dfe521172c307eb45eace7b769f726f0bbf989

C:\Users\INFT505-11>
```

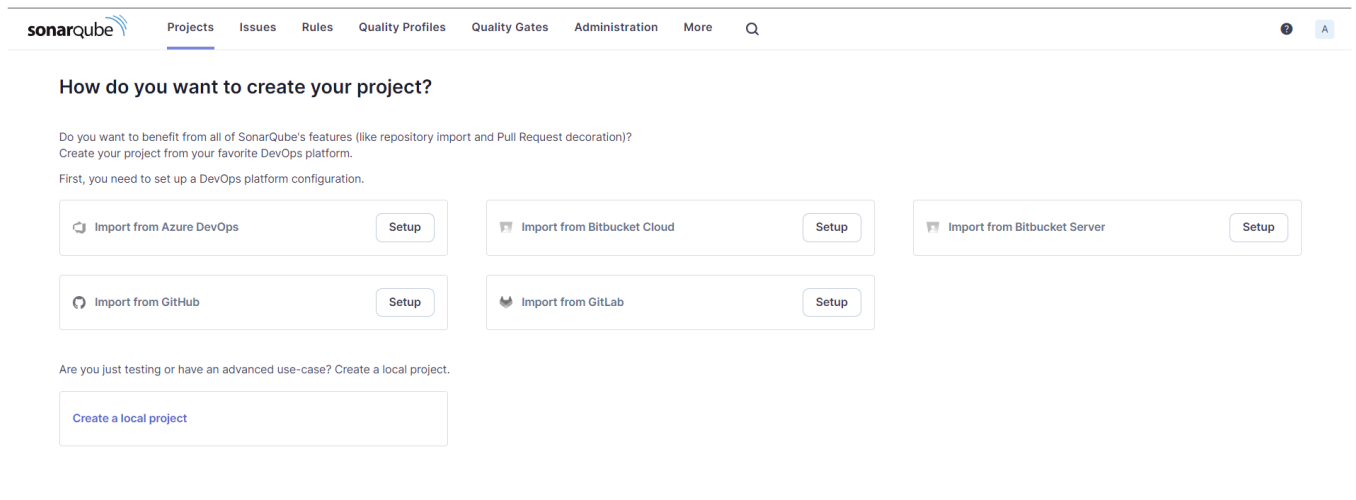
**Step 3:** Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



SonarQube™ technology is powered by [SonarSource SA](#)

[LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#)

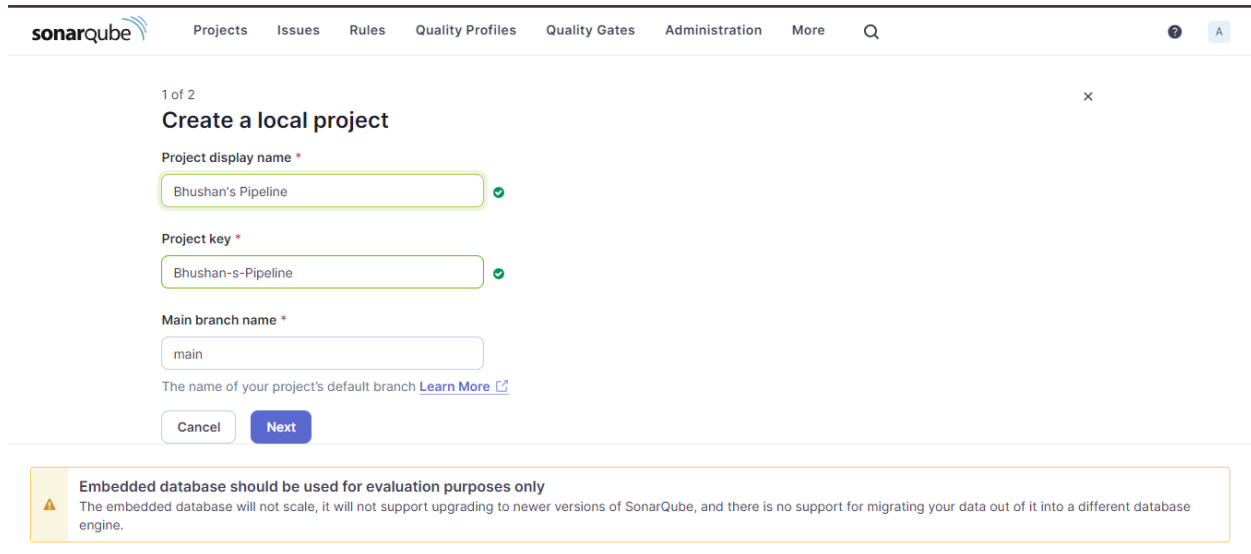
**Start 4:** Login to SonarQube using username admin and password admin.



**Embedded database should be used for evaluation purposes only**  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) ACTIVE [LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#) [Web API](#)

**Step 5: Create a manual project in SonarQube with any Name**

1 of 2

### Create a local project

Project display name \*

Bhushan's Pipeline ✓

Project key \*

Bhushan-s-Pipeline ✓

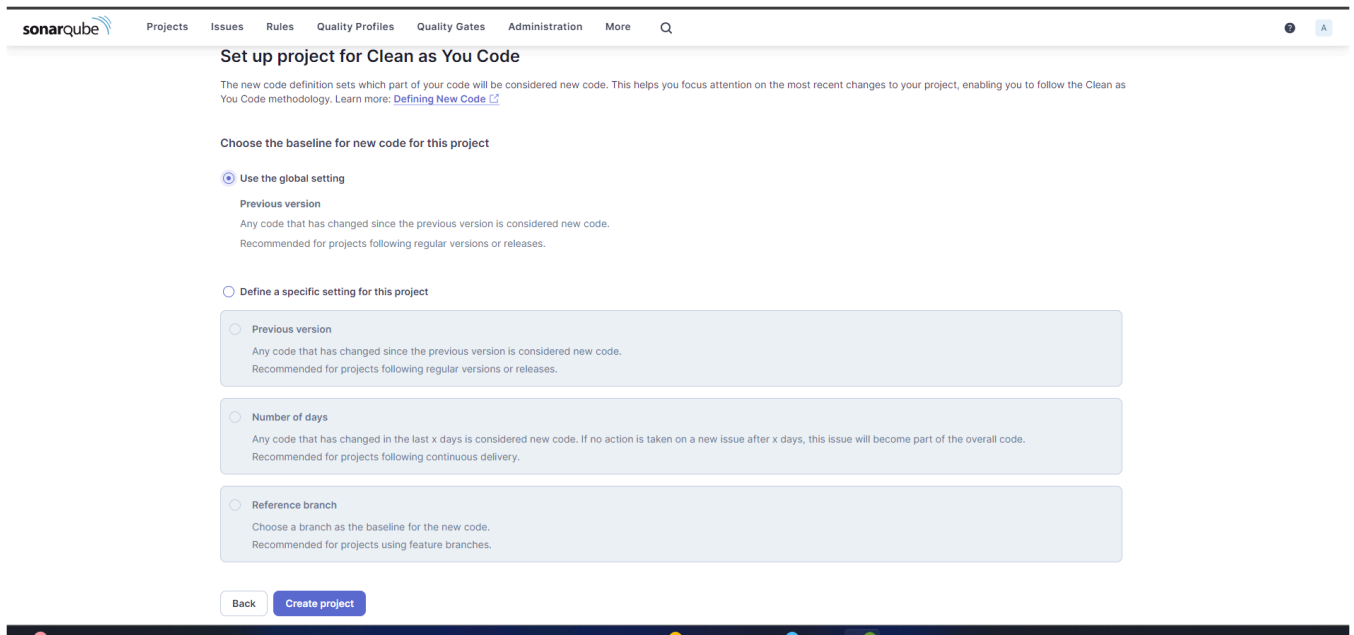
Main branch name \*

main

The name of your project's default branch [Learn More](#)

Cancel Next

**Embedded database should be used for evaluation purposes only**  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.



sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

### Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

**Previous version**  
Any code that has changed since the previous version is considered new code.  
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

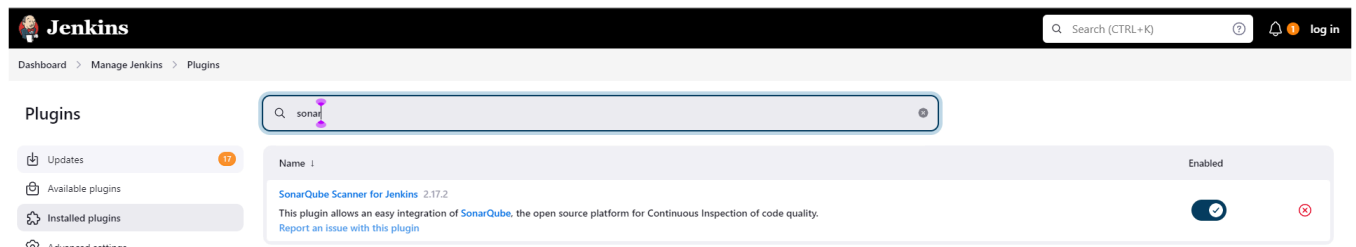
☐ Previous version  
Any code that has changed since the previous version is considered new code.  
Recommended for projects following regular versions or releases.

☐ Number of days  
Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.  
Recommended for projects following continuous delivery.

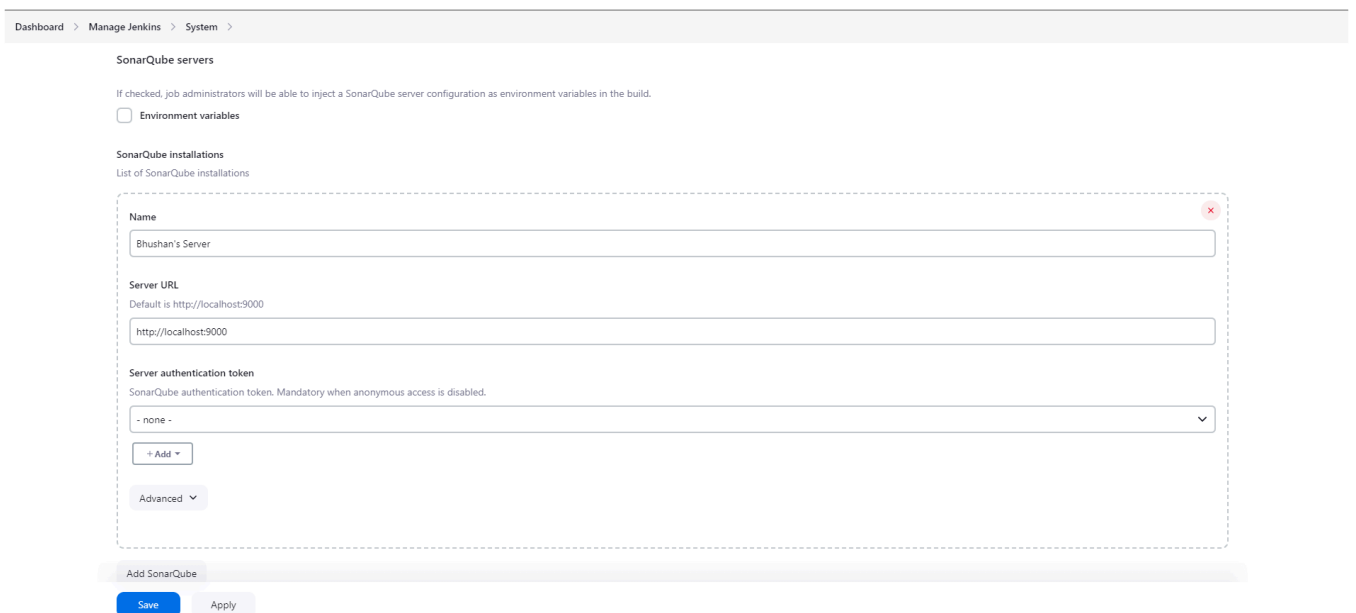
☐ Reference branch  
Choose a branch as the baseline for the new code.  
Recommended for projects using feature branches.

Back Create project

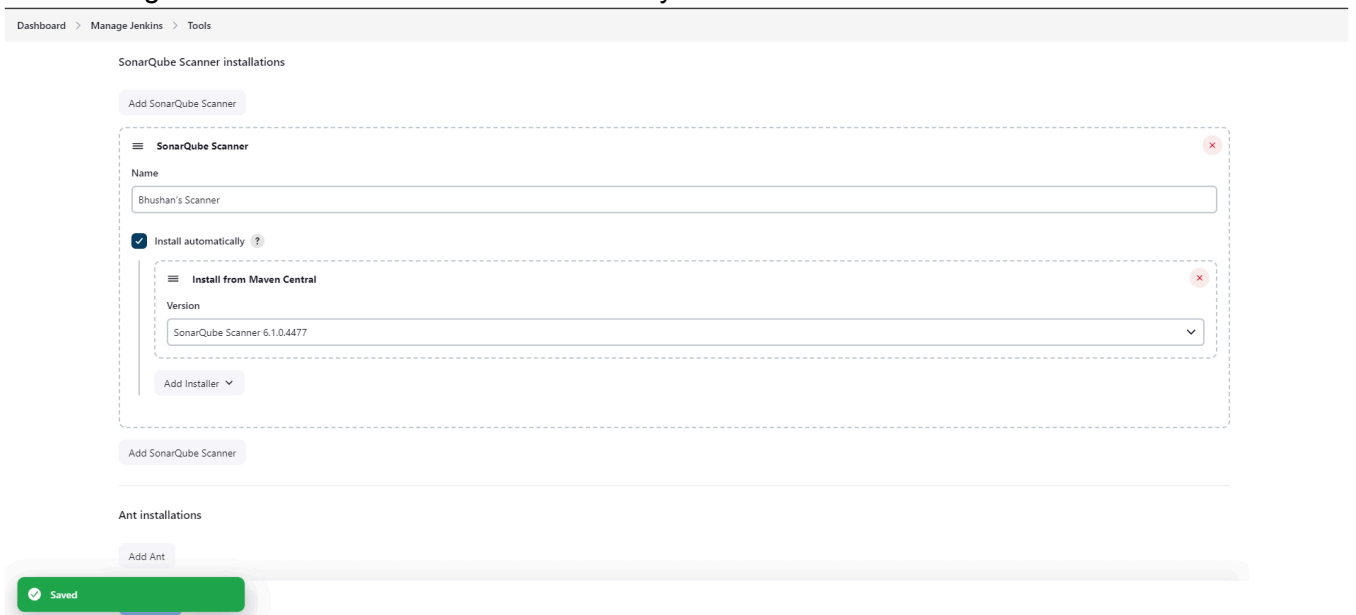
**Step 6:** Setup the project and come back to Jenkins Dashboard.  
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



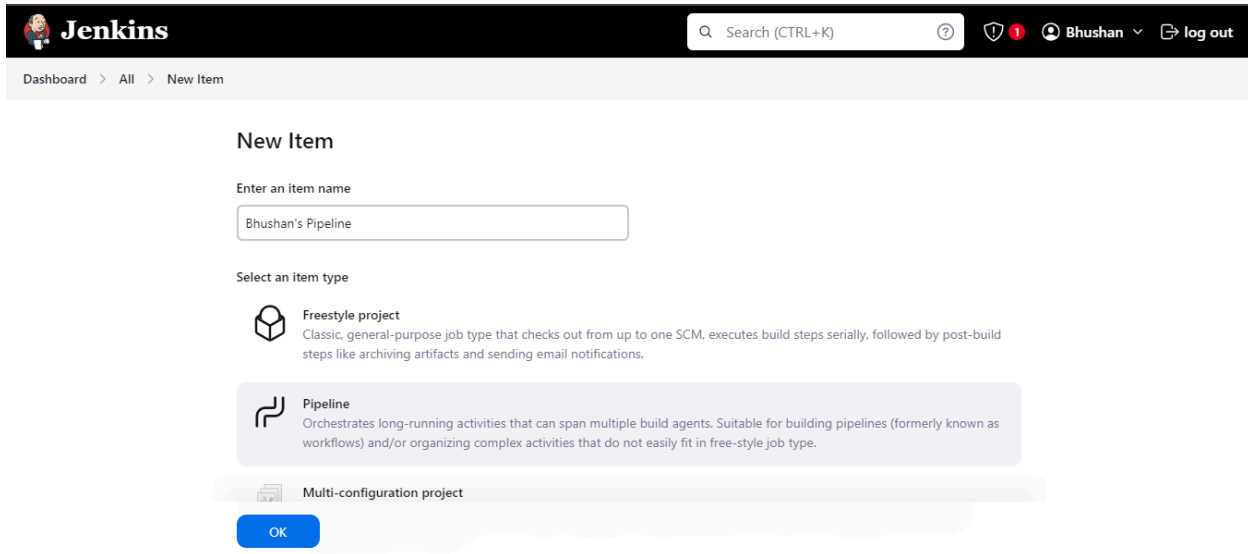
**Step 7:** Under Jenkins 'Configure System', look for SonarQube Servers and enter the details.  
Enter the Server Authentication token if needed.



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



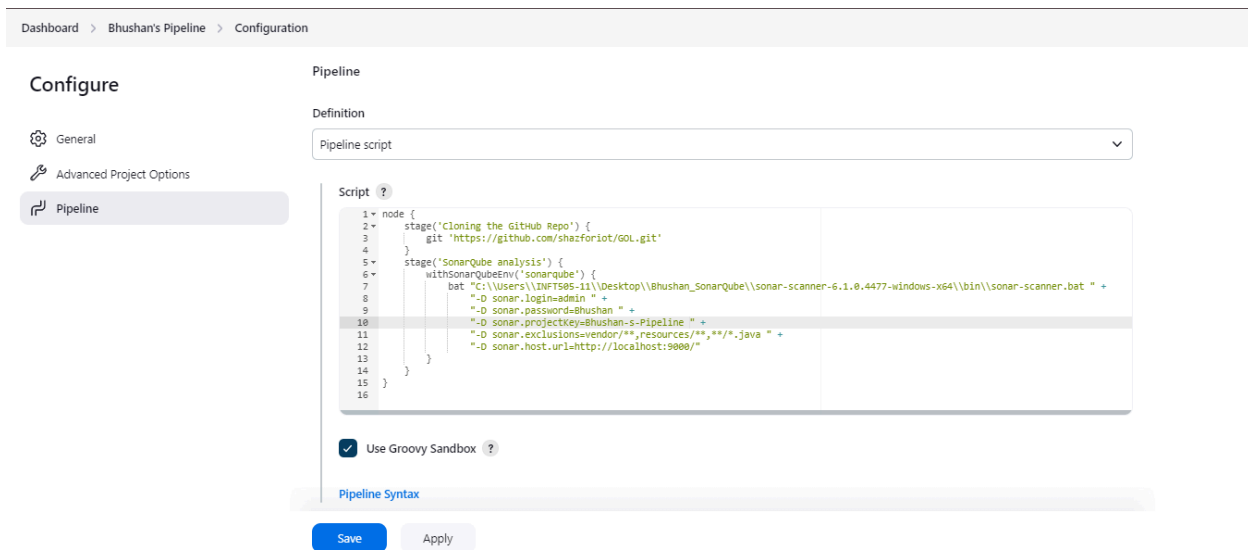
**Step 9:** Now click on the new item and select the pipeline project and give name.



The screenshot shows the Jenkins 'New Item' page. At the top, there's a search bar and user information. Below, the 'Enter an item name' field contains 'Bhushan's Pipeline'. Under 'Select an item type', three options are listed: 'Freestyle project', 'Pipeline' (highlighted in blue), and 'Multi-configuration project'. An 'OK' button is at the bottom.

**Step 10:** Under the scripts add the following script

```
node {
    stage('Cloning the GitHub Repo'){
        git 'https://github.com/shazforiot/GOL.git'
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat "<your bin file location of SonarQube CLI>"+
                "-D sonar.login=<your user name>"+
                "-D sonar.password=<your password>"+
                "-D sonar.projectkey=<your projectkey>"+
                "-D sonar.exclusions=vendor/**,resources/**,**/*.java "+
                "-D sonar.host.url=http://localhost:9000/"
        }
    }
}
```



The screenshot shows the Jenkins 'Configure' page for 'Bhushan's Pipeline'. The 'Definition' dropdown is set to 'Pipeline script'. The 'Script' field contains the Groovy script from Step 10. The 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons.

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

**Step 11:** Go back to jenkins. Go to the job you had just built and click on Build Now.

The screenshot shows the Jenkins dashboard for a pipeline named 'Bhushan's Pipeline'. The pipeline is in a successful state, indicated by a green checkmark. The left sidebar contains a list of actions: Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main area displays the pipeline's status and a list of build history items. The build history shows a single build (#2) that was successful and completed on Sep 19, 2024, at 9:01 PM. The SonarQube icon is visible next to the pipeline name.

**Step 12:** Check the console output.

The screenshot shows the Jenkins console output for the pipeline 'Bhushan's Pipeline'. The console output displays the following text:

```
Started by user Bhushan
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\Bhushan's Pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\Bhushan's Pipeline\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git.exe --version # timeout=10
> git --version # 'git version 2.46.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master" # timeout=10
Checking out Revision ba799ba7e1b576f04a612322b0412c5e61e5e4 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f ba799ba7e1b576f04a612322b0412c5e61e5e4 # timeout=10
```



**Step 13:** Once the build is complete, go back to SonarQube and check the project linked.

The image shows two screenshots of the SonarQube web interface. The top screenshot displays the 'Projects' page with a list of projects. The bottom screenshot shows the detailed view of a project named 'Bhushan's Pipeline'.

**Top Screenshot: SonarQube Projects Page**

- Navigation: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More.
- Search: Search for projects...
- Filters: Quality Gate (Passed: 1, Failed: 0), Reliability (A: 0, B: 0, C: 1, D: 0, E: 0), Security (A: 1, B: 0).
- Project List: 1 project(s) shown. Project: Bhushan's Pipeline (PUBLIC). Last analysis: 12 minutes ago. 683k Lines of Code + HTML, XML, ...
- Metrics: Security (A 0), Reliability (C 68k), Maintainability (A 164k), Hotspots Reviewed (E 0.0%), Coverage (50.6%), Duplications (50.6%).

**Bottom Screenshot: Project Overview - Bhushan's Pipeline**

- Navigation: Overview, Issues, Security Hotspots, Measures, Code, Activity.
- Project Info: 683k Lines of Code, Version not provided, Set as homepage.
- Quality Gate: Passed (Last analysis 10 minutes ago).
- Overall Code Metrics:
  - Security: 0 Open issues (A).
  - Reliability: 68k Open issues (C).
  - Maintainability: 164k Open issues (A).
  - Accepted issues: 0 (Valid issues that were not fixed).
  - Coverage: 0 lines to cover.
  - Duplications: 50.6% (On 759k lines).

Click on Issues and see the different Issues.

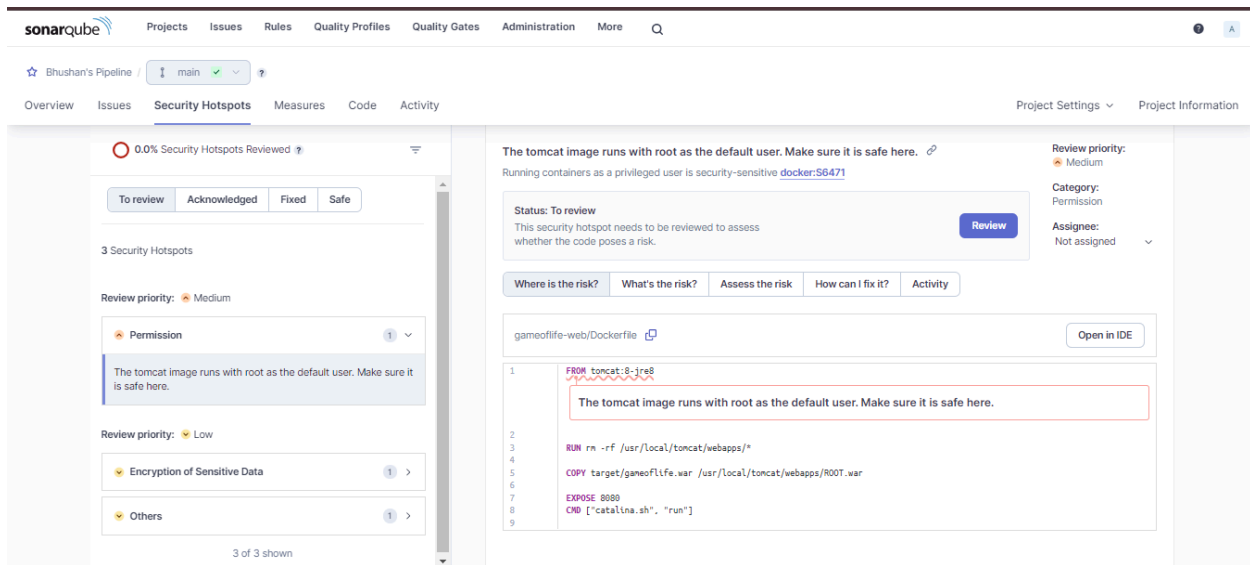
## Codesmell:

The screenshot shows the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. Below the navigation bar, the 'Issues' tab is selected, showing a list of issues for the 'gameoflife-acceptance-tests/Dockerfile' project. The left sidebar shows the 'Software Quality' section with 'Code Smell' selected, indicating 164k issues. The main content area displays three issues, all related to 'Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.' The issues are categorized as 'Intentionality' and 'Maintainability'. The bottom of the page has a warning message: 'localhost:9000/security\_hotspots?id=Bhushan-s-Pi for evaluation purposes only'.

## Bugs:

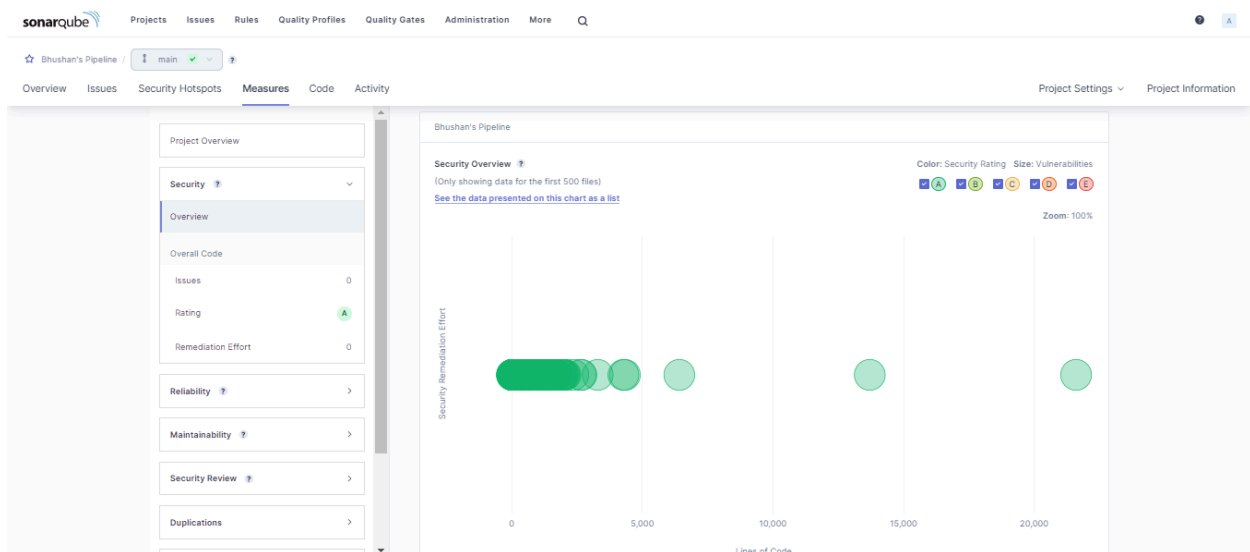
The screenshot shows the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. Below the navigation bar, the 'Issues' tab is selected, showing a list of issues for the 'gameoflife-core/build/reports/tests/all-tests.html' project. The left sidebar shows the 'Software Quality' section with 'Bug' selected, indicating 47k issues. The main content area displays three issues, all related to 'Insert a <IDOCYTE> declaration to before this <html> tag.' The issues are categorized as 'Consistency' and 'Reliability'. The bottom of the page has a warning message: 'Embedded database should be used for evaluation purposes only'.

Click on Security hotspots and see the different Security hotspots.

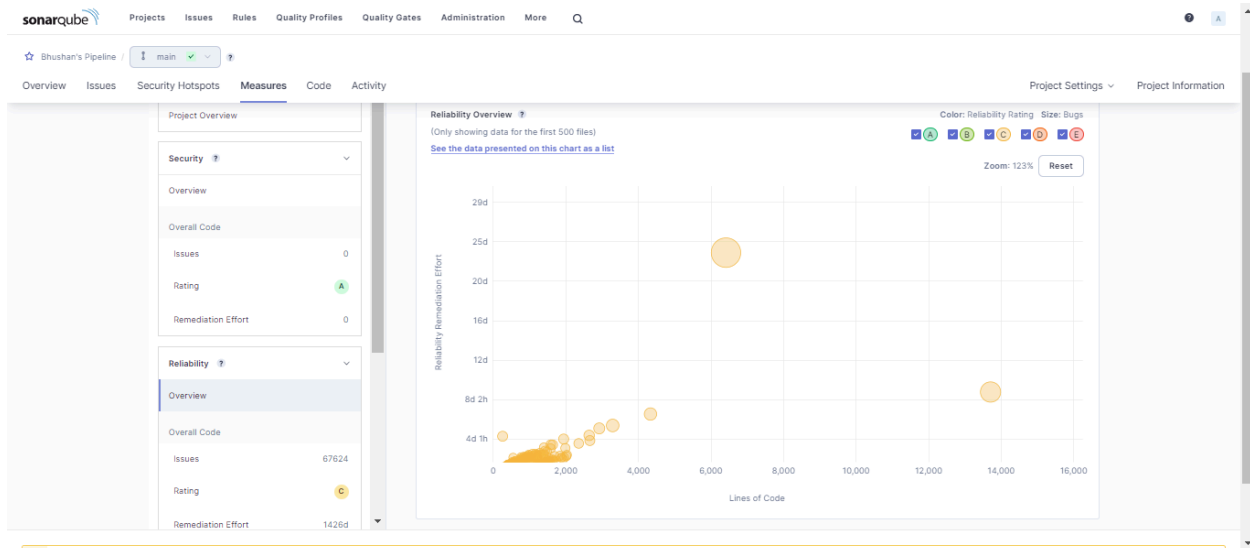


The screenshot shows the SonarQube interface for the 'Bhushan's Pipeline' project. The 'Security Hotspots' tab is active, displaying a list of 3 hotspots. The first hotspot, 'Permission', has a 'Review priority' of Medium and a description: 'The tomcat image runs with root as the default user. Make sure it is safe here.' The right panel shows the details of this hotspot, including its status 'To review', a description, and a code snippet from 'gameoflife-web/Dockerfile' showing the 'FROM tomcat:8-jre8' line. The 'Review' button is visible.

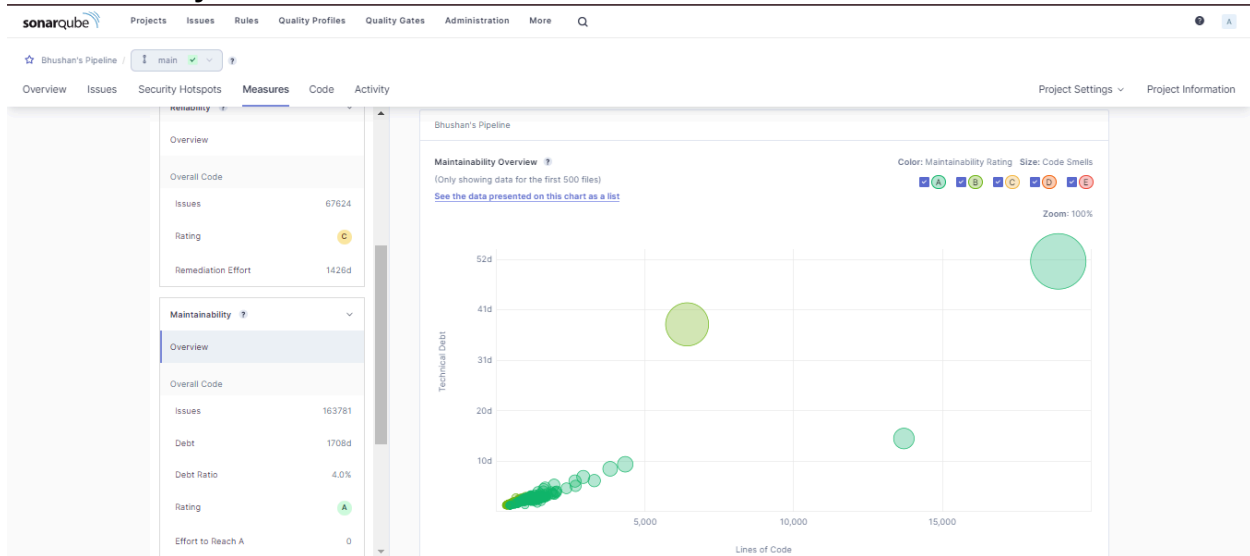
Click on Measures and see the Measures in the form of Graphs.  
Security:



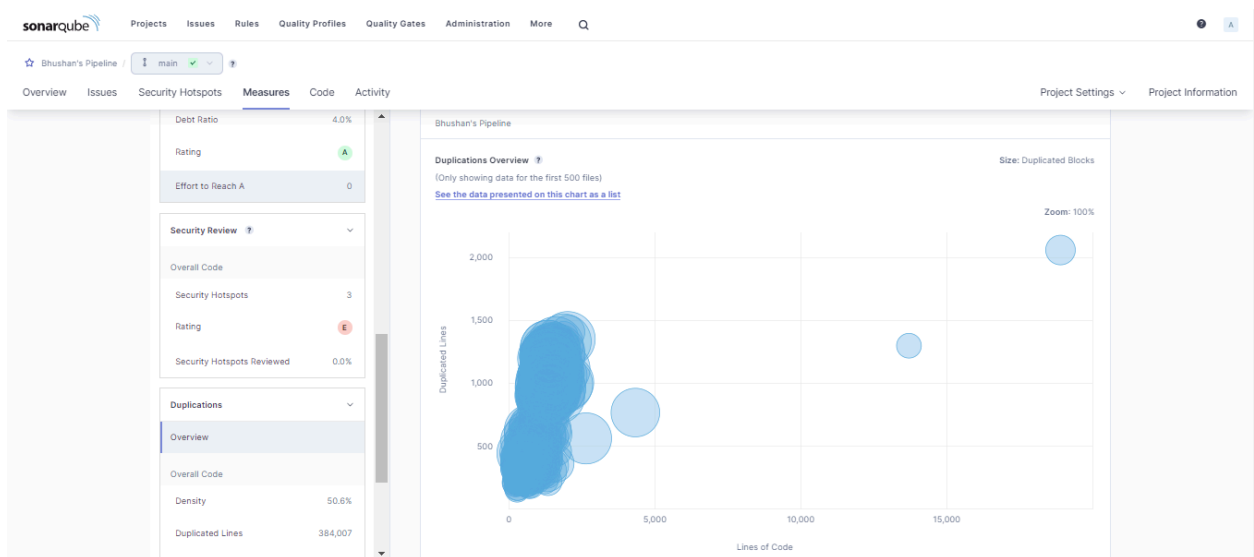
Reliability:



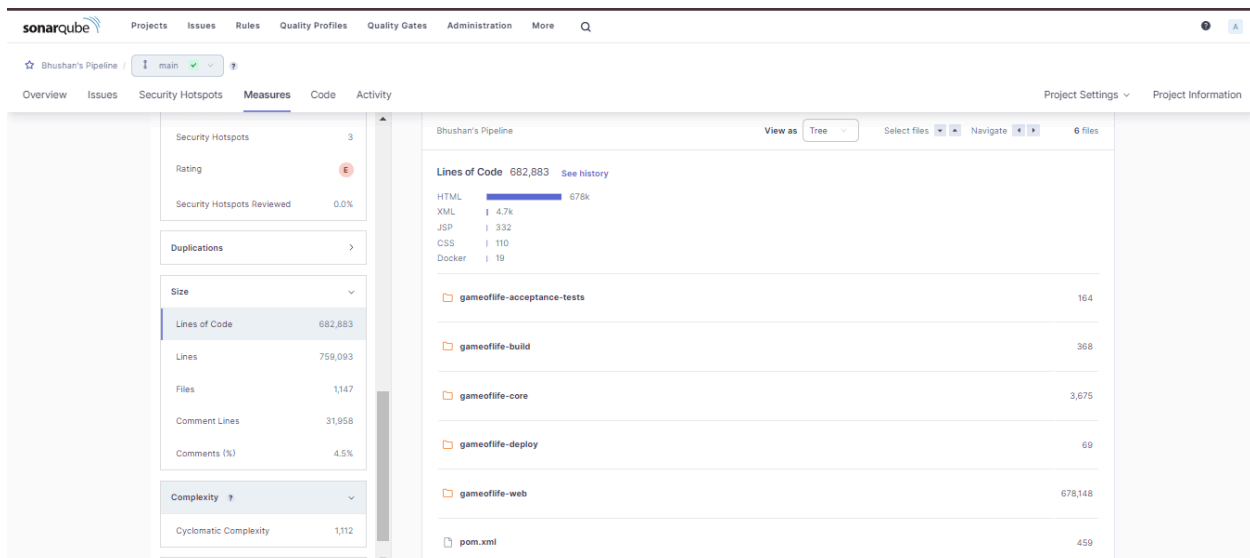
## Maintainability:

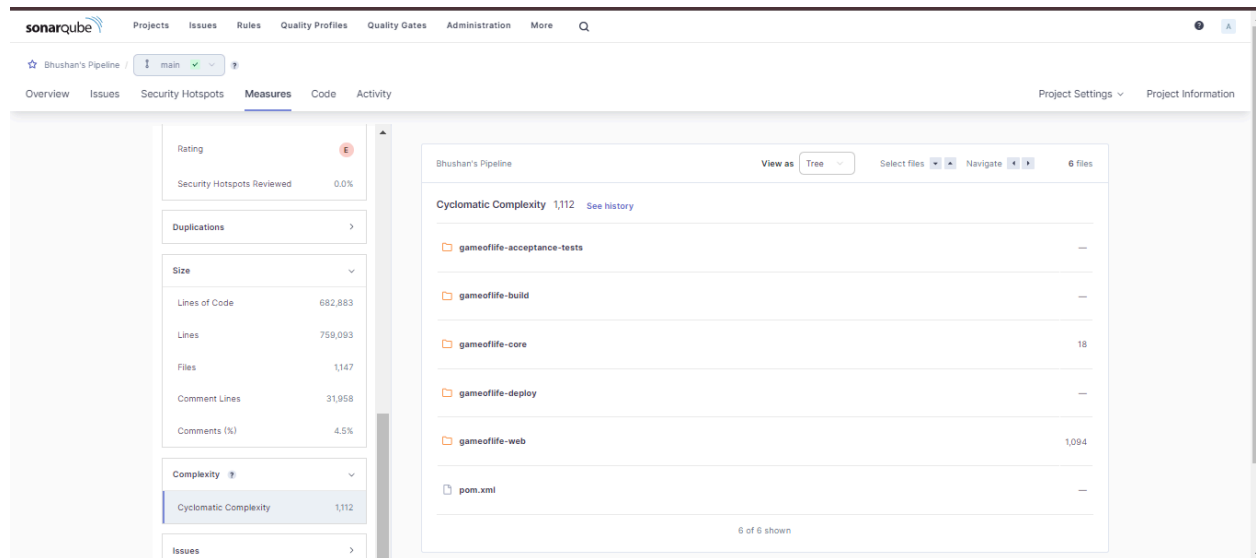
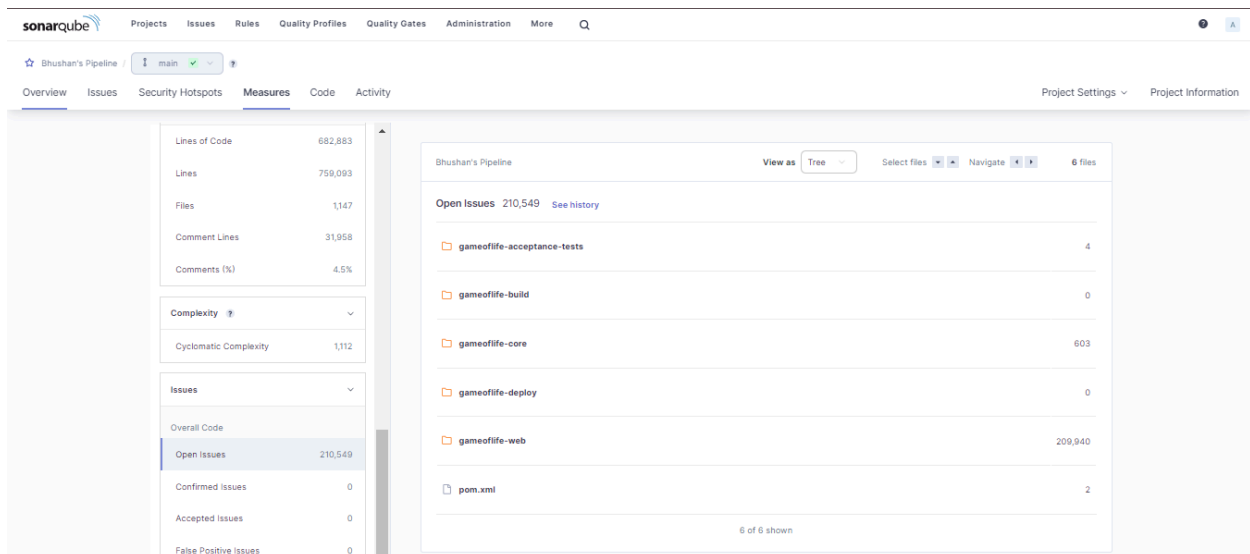


## Duplication:



## Sizes:



**Complexity:****Issues:**

**Conclusion:** In this experiment, we performed static analysis of a code using Jenkins CI/CD Pipeline with SonarQune analysis. A pipeline project is to be created which is given a pipeline script. This script contains all the information needed for the project to run the SonarQube analysis. After the necessary configurations are made on Jenkins, the Jenkins project is built. The code provided in this experiment contains lots of errors, bugs, duplications which can be checked on the SonarQube project linked with this build.It streamlines the process of detecting errors in the code.