

**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', and 'SonarScanner CLI'. The main content area is titled 'SonarScanner CLI' and features a table with columns for 'SonarScanner' and 'Issue Tracker'. The table lists version 6.2, dated 2024-09-17, with a description: 'Support PKCS12 truststore generated with OpenSSL'. Below the table, there is a section for 'Download scanner for:' with links for Linux x64, Linux AArch64, Windows x64, macOS x64, macOS AArch64, and Docker. A note states: 'Any (Requires a pre-installed JVM)'. There is also a link to 'Release notes'. The right sidebar contains a 'START FREE' button and a list of links under 'On this page' including '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 footer note states: 'The SonarScanners run on code that is checked out. See Verifying the code checkout step of your build.'

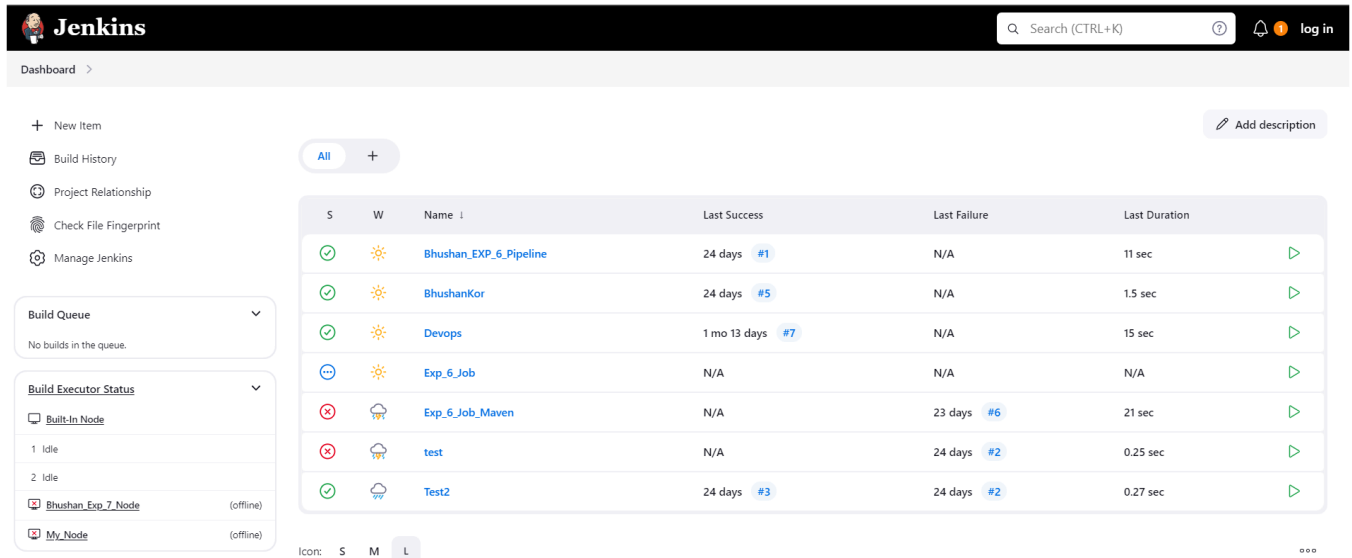
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

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



The screenshot shows the Jenkins Dashboard interface. 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 sections for Build Queue (No builds in the queue) and Build Executor Status (Built-In Node, 1 Idle, 2 Idle, Bhushan\_Exp\_7\_Node (offline), My\_Node (offline)). The main area displays a table of builds with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists several builds, including Bhushan\_EXP\_6\_Pipeline, BhushanKor, Devops, Exp\_6\_Job, Exp\_6\_Job\_Maven, test, and Test2. Each row includes a status icon (green checkmark for success, red X for failure, yellow lightning bolt for warning) and a duration. A search bar and a log in button are visible in the top right corner.

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

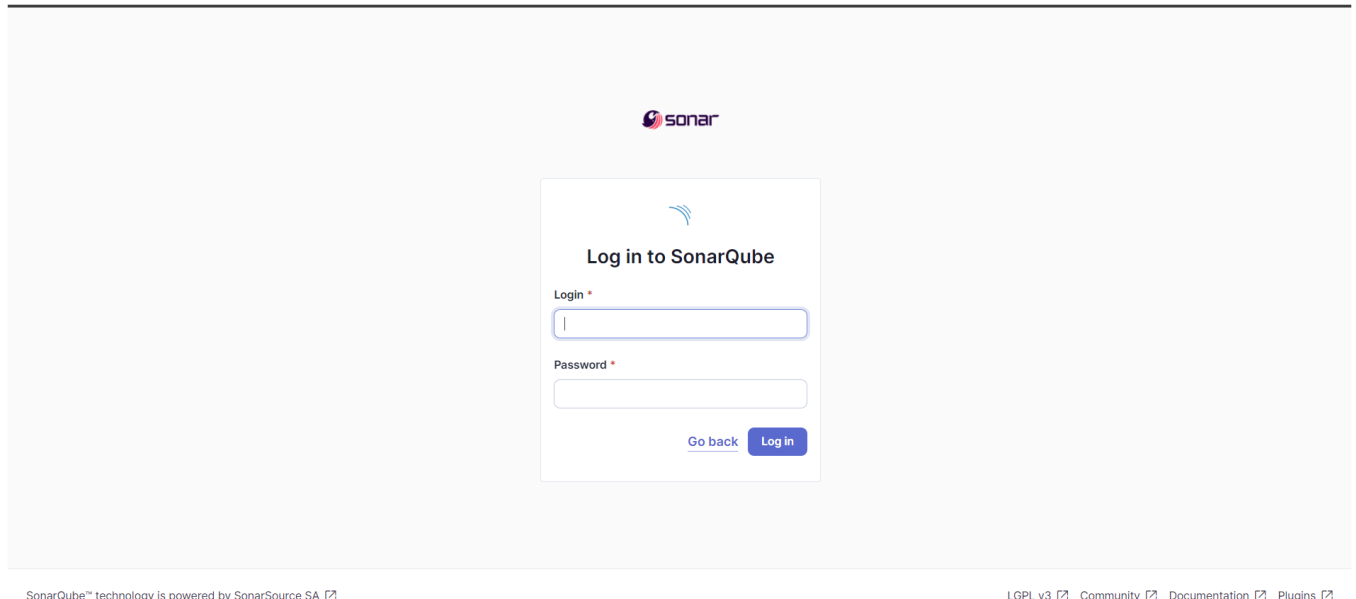
```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

```
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: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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecd
Status: Downloaded newer image for sonarqube:latest
d72b183b1866cea7ecdb976a63dfe521172c307eb45eace7b769f726f0bbf989

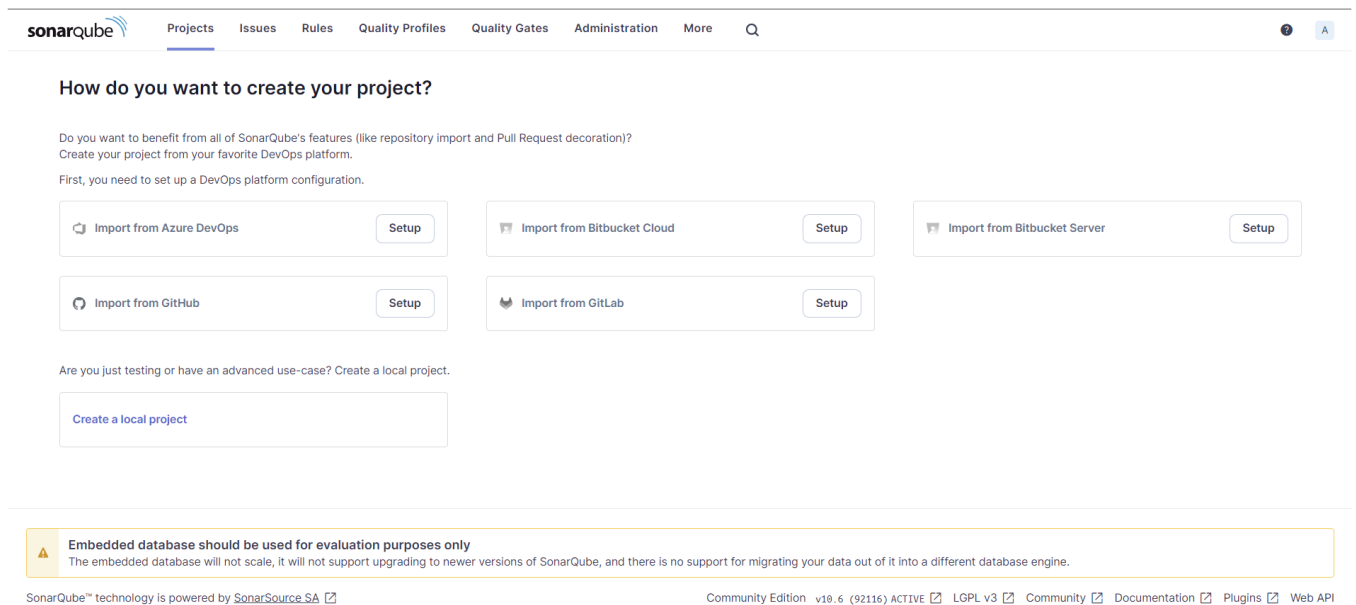
C:\Users\INFT505-11>
```

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

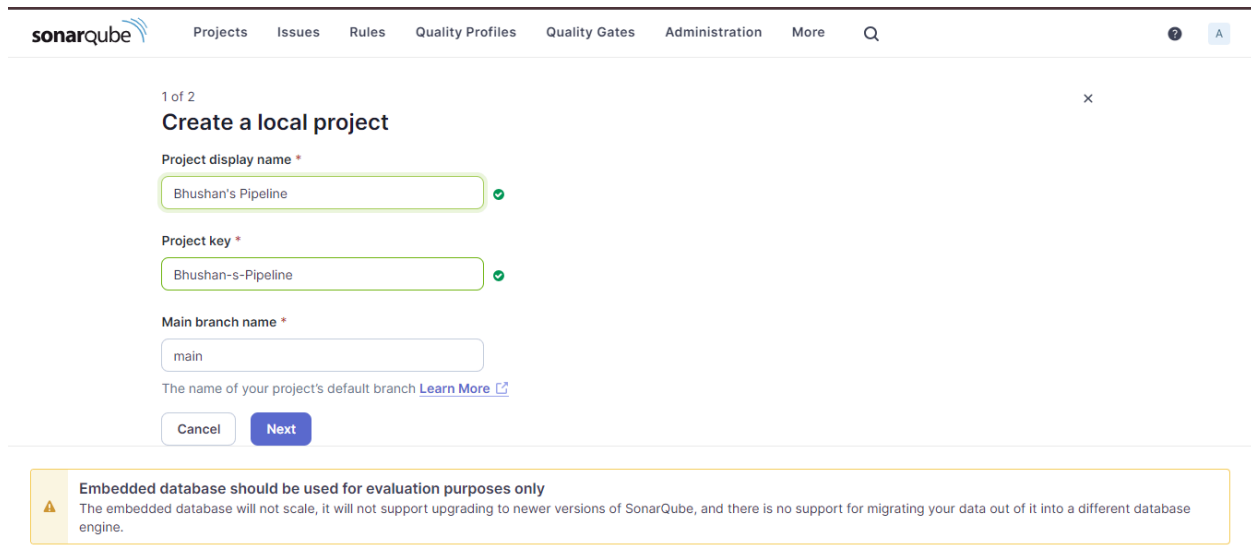


The image shows the SonarQube login page. At the top center is the Sonar logo. Below it is a white box with the title "Log in to SonarQube". Inside this box are two input fields: "Login \*" and "Password \*". Below the password field are two buttons: "Go back" and "Log in". At the bottom of the page, there is a footer with the text "SonarQube™ technology is powered by SonarSource SA" and a link to the SonarSource website. To the right of this, there are links for "LGPL v3", "Community", "Documentation", and "Plugins".

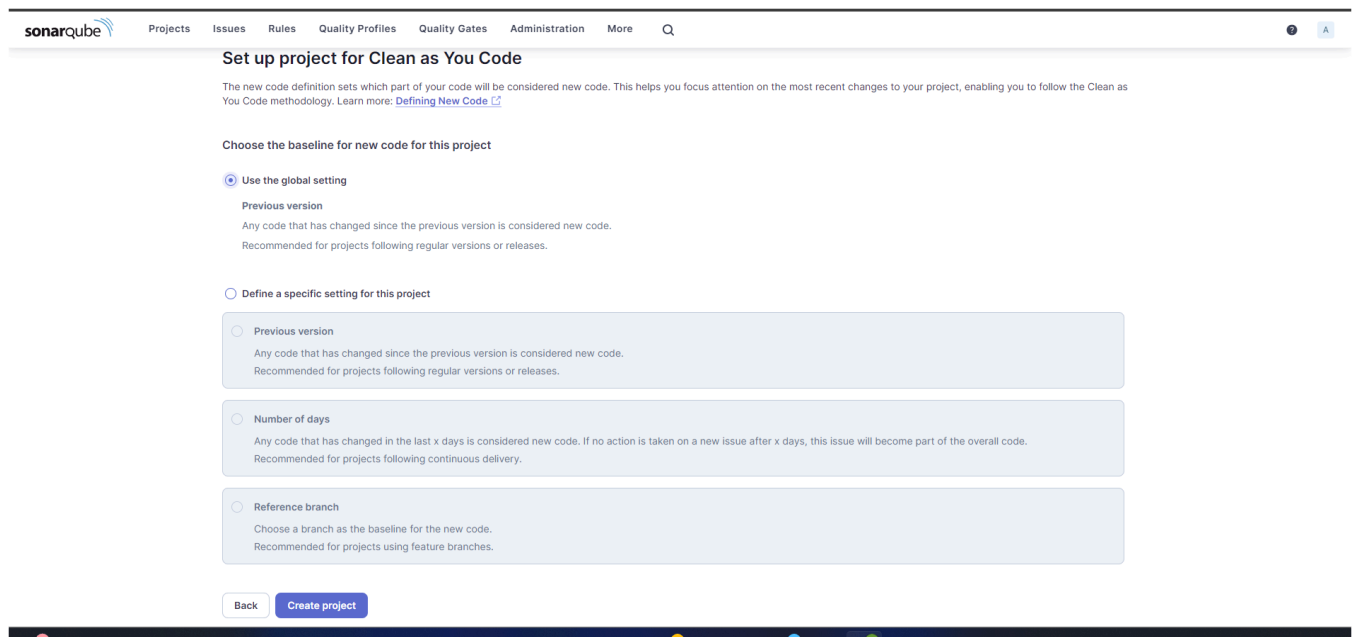
**Start 4:** Login to SonarQube using username admin and password admin.



The image shows the SonarQube project creation page. At the top is the SonarQube logo and a navigation bar with links for "Projects", "Issues", "Rules", "Quality Profiles", "Quality Gates", "Administration", and "More". Below the navigation bar is a section titled "How do you want to create your project?". It contains two paragraphs of text: "Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform." and "First, you need to set up a DevOps platform configuration." Below this are five buttons: "Import from Azure DevOps", "Import from Bitbucket Cloud", "Import from Bitbucket Server", "Import from GitHub", and "Import from GitLab". Each button has a "Setup" button next to it. Below these buttons is a paragraph of text: "Are you just testing or have an advanced use-case? Create a local project." and a button labeled "Create a local project". At the bottom of the page, there is a yellow warning box with the text "Embedded database should be used for evaluation purposes only" and a link to the SonarSource website. To the right of this, there are links for "Community Edition", "v10.6 (92116)", "ACTIVE", "LGPL v3", "Community", "Documentation", "Plugins", and "Web API".

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

The screenshot shows the 'Create a local project' form in SonarQube. The form is titled '1 of 2' and 'Create a local project'. It contains three input fields: 'Project display name' with the value 'Bhushan's Pipeline', 'Project key' with the value 'Bhushan-s-Pipeline', and 'Main branch name' with the value 'main'. Each field has a green checkmark icon to its right. Below the 'Main branch name' field, there is a link 'Learn More' with an external link icon. At the bottom of the form, there are two buttons: 'Cancel' and 'Next'. A yellow warning box at the bottom of the form contains the text: '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.'



The screenshot shows the 'Set up project for Clean as You Code' form in SonarQube. The form is titled 'Set up project for Clean as You Code'. It contains a paragraph of text: '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](#)'. Below this, there is a section 'Choose the baseline for new code for this project'. There are three radio button options: 'Use the global setting', 'Previous version', and 'Define a specific setting for this project'. The 'Use the global setting' option is selected. Below the 'Previous version' option, there is a paragraph of text: 'Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases.' Below the 'Define a specific setting for this project' option, there are three sub-options: 'Previous version', 'Number of days', and 'Reference branch'. Each sub-option has a paragraph of text: 'Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases.', '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.', and 'Choose a branch as the baseline for the new code. Recommended for projects using feature branches.' At the bottom of the form, there are two buttons: 'Back' and 'Create project'.

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

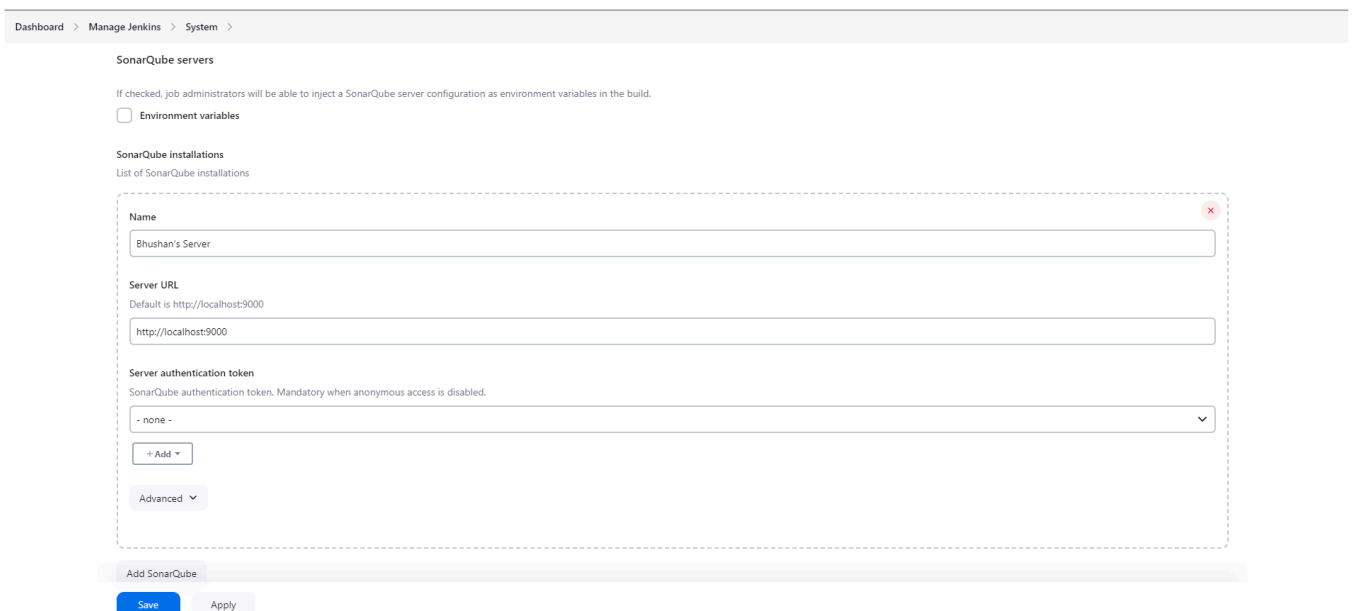
Roll No: 28

**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.



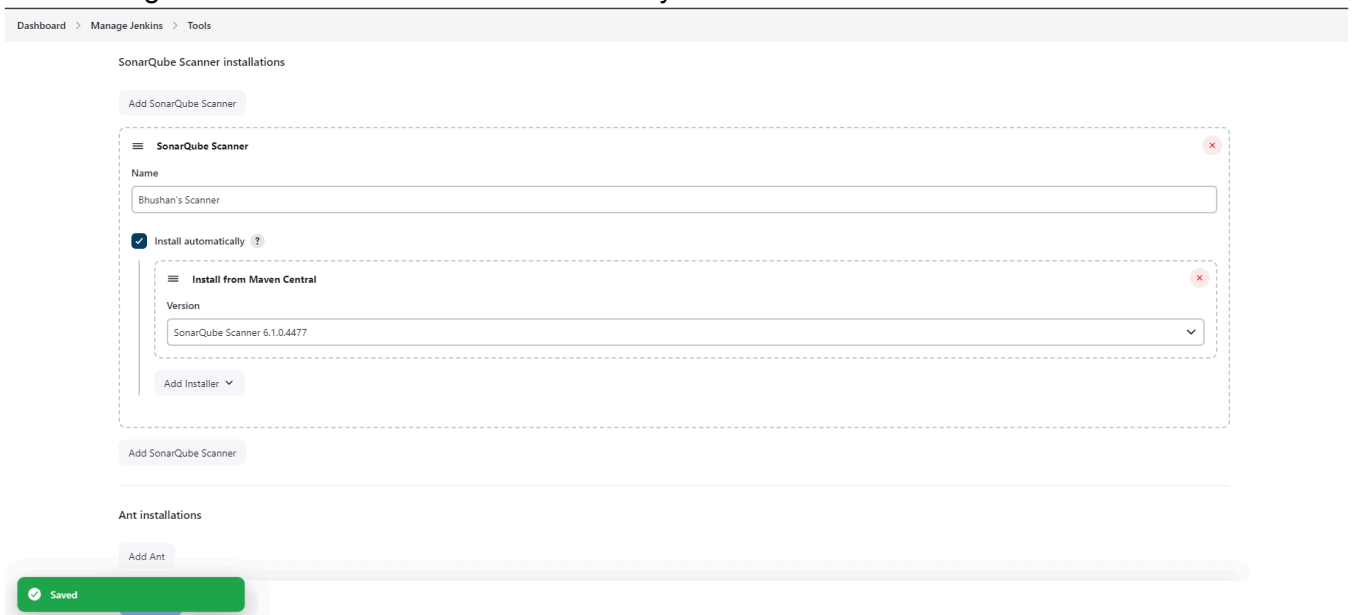
The screenshot shows the Jenkins 'Manage Jenkins' > 'Plugins' page. A search bar at the top contains 'sonar'. On the left sidebar, 'Updates' is highlighted with a badge showing 17 updates. The main area displays a table of installed plugins. The first entry is 'SonarQube Scanner for Jenkins' version 2.17.2, which is marked as 'Enabled' with a toggle switch and a red 'X' icon for removal. The description states: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality. Report an issue with this plugin'.

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



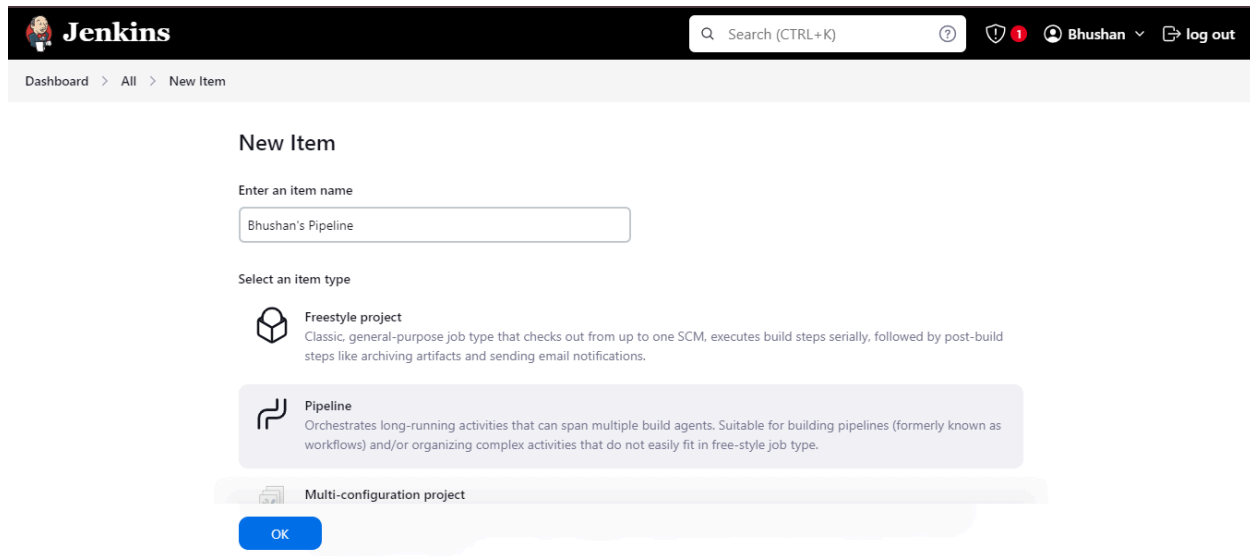
The screenshot shows the 'Dashboard' > 'Manage Jenkins' > 'System' configuration page. The 'SonarQube servers' section is active. It includes a checkbox for 'Environment variables' (unchecked). Below, the 'SonarQube installations' section shows a list of installations. A new installation is being configured with the following details: Name: 'Bhushan's Server', Server URL: 'http://localhost:9000', and Server authentication token: '- none -'. There is an 'Add' button and an 'Advanced' dropdown. At the bottom, there is an 'Add SonarQube' button and 'Save' and 'Apply' buttons.

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



The screenshot shows the 'Dashboard' > 'Manage Jenkins' > 'Tools' configuration page. The 'SonarQube Scanner installations' section is active. It shows a list of installations. A new installation is being configured with the following details: Name: 'Bhushan's Scanner', 'Install automatically' is checked, and the version is 'SonarQube Scanner 6.10.0.4477'. There is an 'Add Installer' dropdown. At the bottom, there is an 'Add SonarQube Scanner' button. A green 'Saved' notification banner is visible at the bottom of the page.

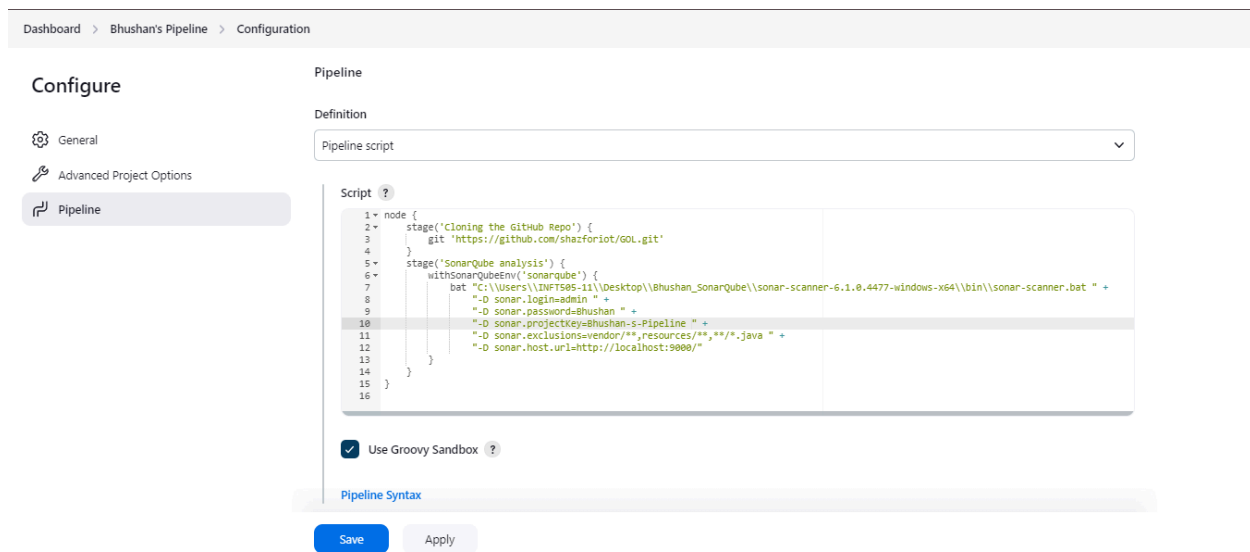
**Step 9:** Now click on the new item and select the pipeline project and give name.



The image shows the Jenkins 'New Item' form. At the top, there's a search bar and a user profile 'Bhushan'. The breadcrumb trail is 'Dashboard > All > New Item'. The form has two main sections: 'Enter an item name' and 'Select an item type'. In the name field, 'Bhushan's Pipeline' is entered. Under 'Select an item type', three options are listed: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is highlighted with a blue border and a blue 'OK' button is visible at the bottom.

**Step 10:** Under the scripts add the following script. (Do not forget to replace details with your details.)

```
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 image shows the Jenkins 'Configure' page for 'Bhushan's Pipeline'. The left sidebar has 'General', 'Advanced Project Options', and 'Pipeline' tabs, with 'Pipeline' selected. The main area is titled 'Pipeline' and 'Definition'. A dropdown menu shows 'Pipeline script'. Below this, a 'Script' section contains a Groovy script with line numbers 1 to 16. The script is the same as the one in Step 10. At the bottom, there's a checkbox 'Use Groovy Sandbox' which is checked. Below the script area is a 'Pipeline Syntax' link. At the very bottom 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.

**Jenkins** Search (CTRL+K) ? 1 log in

Dashboard > Bhushan's Pipeline >

**Status** **Bhushan's Pipeline** Add description

Changes

Workspace

Build Now

Configure

Delete Project

SonarQube

Rename

**Permalinks**

- Last build (#2), 7 days 23 hr ago
- Last stable build (#2), 7 days 23 hr ago
- Last successful build (#2), 7 days 23 hr ago
- Last failed build (#1), 7 days 23 hr ago
- Last unsuccessful build (#1), 7 days 23 hr ago
- Last completed build (#2), 7 days 23 hr ago

**Build History** trend

Filter...

#2	
Sep 19, 2024, 9:01 PM	

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

**Jenkins** Search (CTRL+K) ? 1 Bhushan log out

Dashboard > Bhushan's Pipeline > #11

**Console Output** Download Copy View as plain 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
```



Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

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

The screenshot shows the SonarQube 'Projects' page. On the left, there are filters for Quality Gate (Passed, Failed), Reliability (A, B, C, D, E), and Security (A, B). The main area displays a list of projects, with 'Bhushan's Pipeline' selected. It shows a 'Passed' status, last analysis 12 minutes ago, and 683k Lines of Code. Metrics include Security (0), Reliability (68k), Maintainability (164k), Hotspots Reviewed (0.0%), Coverage (50.6%), and Duplications (50.6%).

**Quality Gate:** Passed (1)

**Reliability:** A (0), B (0), C (1), D (0), E (0)

**Security:** A (1), B (0)

**Project Details:** Bhushan's Pipeline (PUBLIC) - Passed  
Last analysis: 12 minutes ago - 683k Lines of Code - HTML, XML, ...

**Metrics:** Security: 0, Reliability: 68k, Maintainability: 164k, Hotspots Reviewed: 0.0%, Coverage: 50.6%, Duplications: 50.6%

1 of 1 shown

**Warning:** Embedded database should be used for evaluation purposes only. The embedded database will not scale. It will not respect credentials to access resources of SonarQube, and there is no support for migrating your data out of it into a different database engine.

The screenshot shows the SonarQube 'Project Overview' page for 'Bhushan's Pipeline'. The project is in the 'main' branch and is 'Passed'. The last analysis was 10 minutes ago. The page shows metrics for Security, Reliability, Maintainability, Accepted issues, Coverage, and Duplications.

**Project Overview:** Bhushan's Pipeline / main - Passed  
Last analysis 10 minutes ago

**Quality Gate:** Passed

**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)

**Security Hotspots:**

**Academic Year:2024-2025**

**Roll No: 28**

**Click on Issues and see the different Issues.**

## Codesmell:

**Sonarqube** | Projects | Issues | Rules | Quality Profiles | Quality Gates | Administration | More | 🔍

Bhushan's Pipeline | [main] [dropdown] [help]

Overview | **Issues** | Security Hotspots | Measures | Code | Activity

Project Settings ▾ | Project Information

**Software Quality**

- Security 0
- Reliability 21k
- Maintainability 164k

**> Severity ?**

- Type** 1 ✕
  - Bug 47k
  - Vulnerability 0
  - Code Smell 164k**

Add to selection Ctrl + click

☐ Bulk Change
 Select issues [x] [y] Navigate to issue [left] [right] **164,034 issues** **1708d effort**
  
gameoflife-acceptance-tests/Dockerfile
 

☐ Use a specific version tag for the image. Intentionality
 

Maintainability ⓘ No tags ▾

Open ▾ Not assigned ▾ L1 - 5min effort - 4 years ago - @ Code Smell - ⚡ Major

☐ Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality
 

Maintainability ⓘ No tags ▾

Open ▾ Not assigned ▾ L12 - 5min effort - 4 years ago - @ Code Smell - ⚡ Major

☐ Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. Intentionality
 

Maintainability ⓘ No tags ▾

Open ▾ Not assigned ▾ L12 - 5min effort - 4 years ago - @ Code Smell - ⚡ Major

localhost:9000/security\_hotspots?id=Bhushan-s-Pit for evaluation purposes only

### Bugs:

**sonarqube**

Projects Issues Rules Quality Profiles Quality Gates Administration More

Bhushan's Pipeline / main

Overview **Issues** Security Hotspots Measures Code Activity Project Settings Project Information

▼ Software Quality

Security	0
Reliability	47k
Maintainability	0

> Severity

▼ Type

- Bug** 47k
- Vulnerability 0
- Code Smell 164k

Add to selection Ctrl + click

> Scope

☐ Bulk Change

Select issues Navigate to issue 46,515 issues 1426d effort

gameoflife-core/build/reports/tests/all-tests.html

☐ Insert a <DOCTYPE> declaration to before this <html> tag.

Consistency Reliability user-experience

Open Not assigned L1 • 5min effort • 4 years ago • Bug • Major

☐ Add "lang" and/or "xml:lang" attributes to this "<html>" element

Intentionality accessibility wcag2-a

Open Not assigned L1 • 2min effort • 4 years ago • Bug • Major

☐ Add "<th>" headers to this "<table>"

Intentionality accessibility wcag2-a

Open Not assigned L1 • 2min effort • 4 years ago • Bug • Major

Embedded database should be used for evaluation purposes only

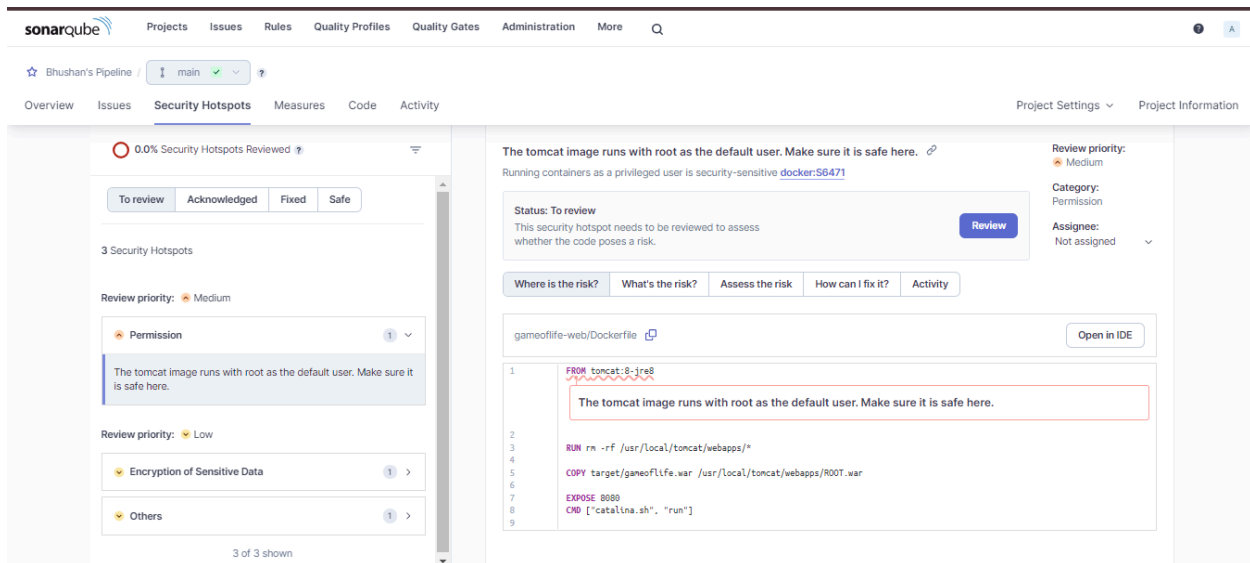
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

Click on Security hotspots and see the different Security hotspots.

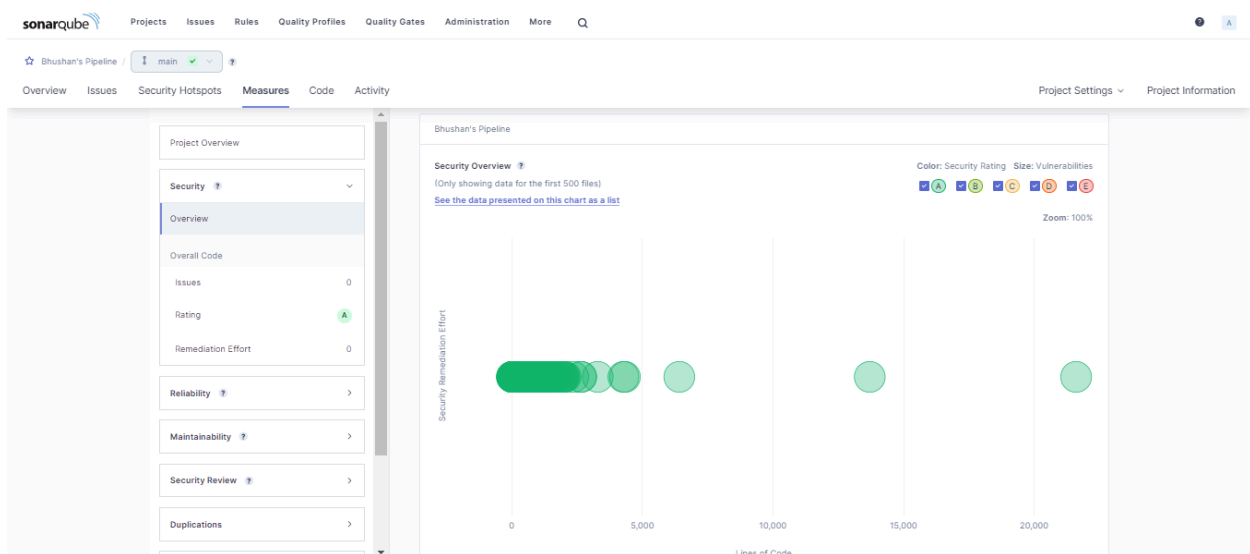


The screenshot shows the SonarQube interface for the 'Bhushan's Pipeline' project. The 'Security Hotspots' tab is active, displaying a list of 3 hotspots. The first hotspot, 'Permission', is highlighted. It has a 'Review priority' of Medium and a 'Category' of Permission. The description states: 'The tomcat image runs with root as the default user. Make sure it is safe here.' The code snippet shows a Dockerfile with the following content:

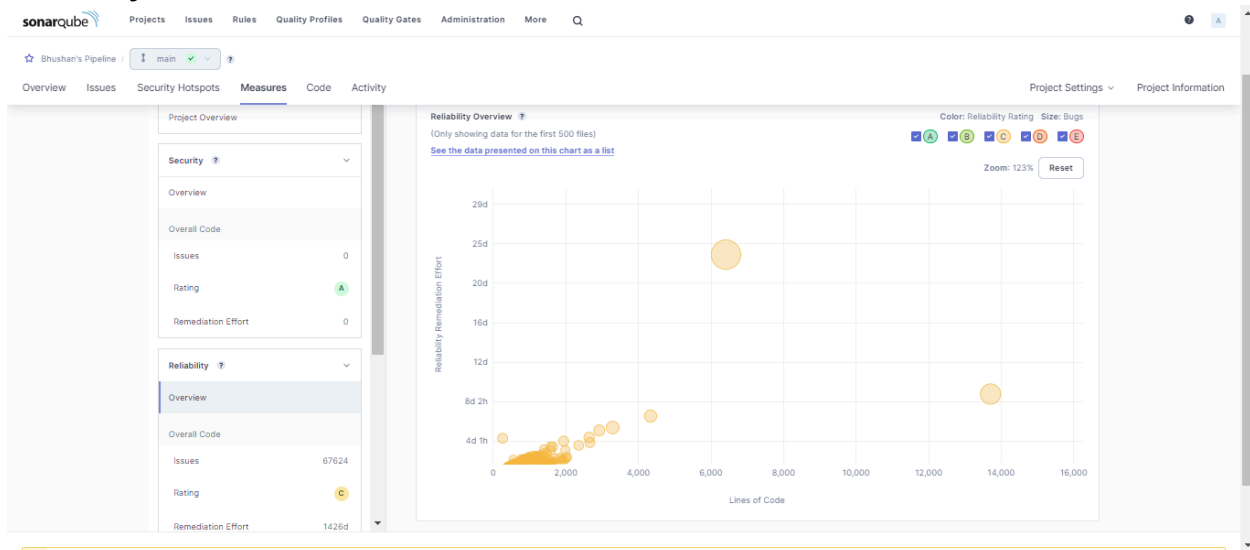
```
FROM tomcat:8-jre8
RUN rm -rf /usr/local/tomcat/webapps/*
COPY target/gameoflife.war /usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

The right sidebar shows the 'Review priority' as Medium, 'Category' as Permission, and 'Assignee' as Not assigned. The bottom of the list shows '3 of 3 shown'.

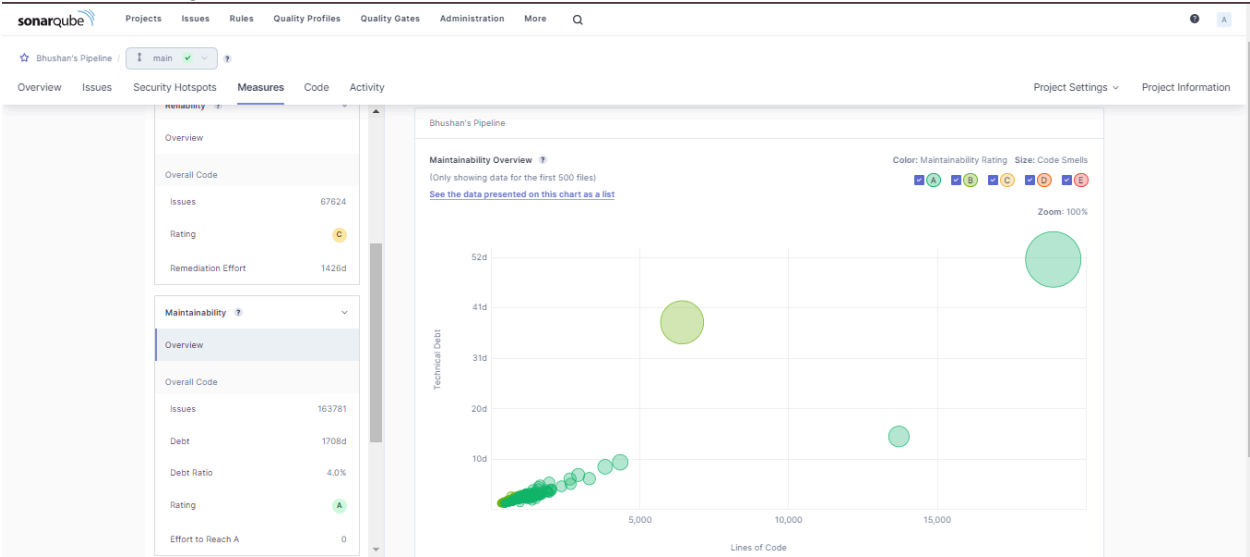
Click on Measures and see the Measures in the form of Graphs.  
Security:



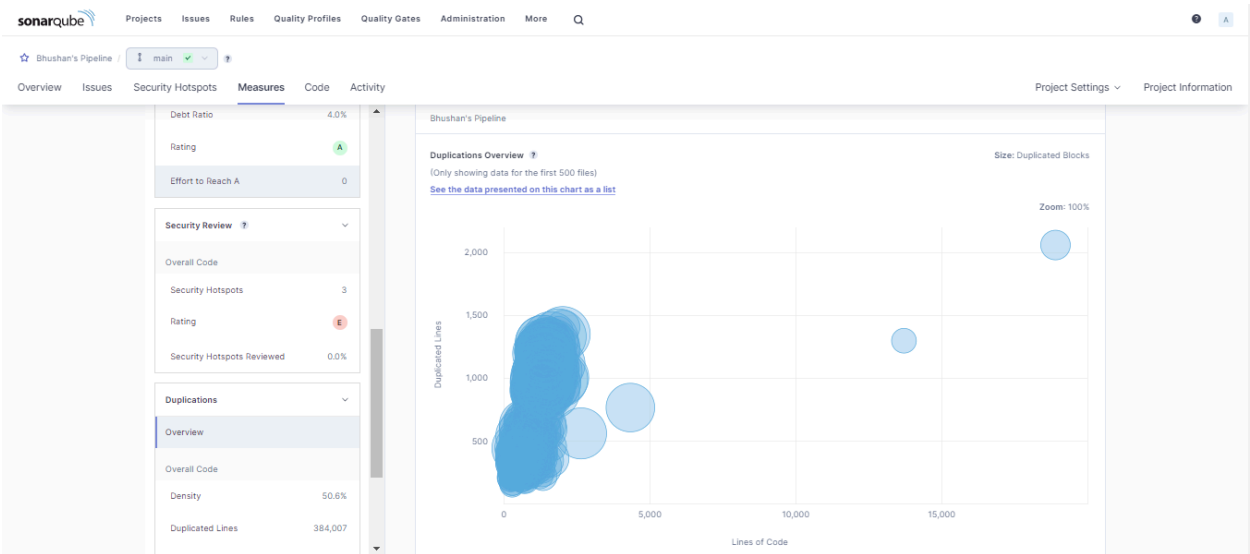
Reliability:



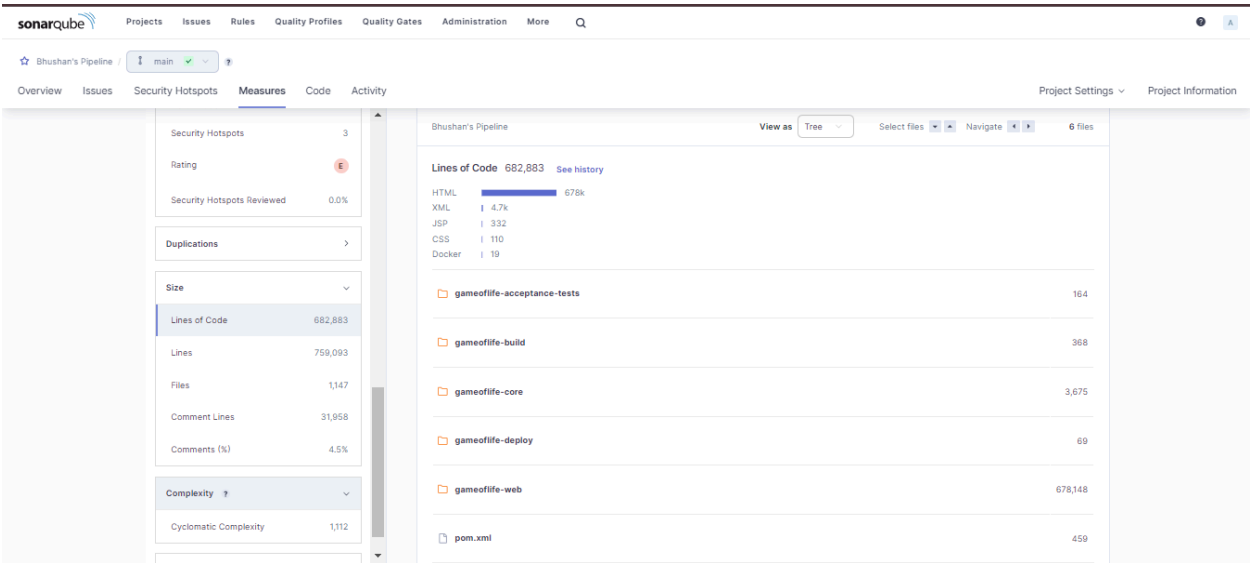
Maintainability:



Duplication:



Sizes:



Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

Complexity:

The screenshot shows the SonarQube interface for a project named 'Bhushan's Pipeline'. The 'Measures' tab is selected, displaying a table of metrics. The 'Cyclomatic Complexity' measure is highlighted, showing a value of 1,112. The table lists various files and their complexity values.

File	Cyclomatic Complexity
gameoflife-acceptance-tests	—
gameoflife-build	—
gameoflife-core	18
gameoflife-deploy	—
gameoflife-web	1,084
pom.xml	—

Issues:

The screenshot shows the SonarQube interface for the same project, 'Bhushan's Pipeline', with the 'Issues' tab selected. The 'Open Issues' measure is highlighted, showing a value of 210,549. The table lists various files and their issue counts.

File	Open Issues
gameoflife-acceptance-tests	4
gameoflife-build	0
gameoflife-core	603
gameoflife-deploy	0
gameoflife-web	209,940
pom.xml	2

**Conclusion:** In this experiment, we have learned how to perform 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.