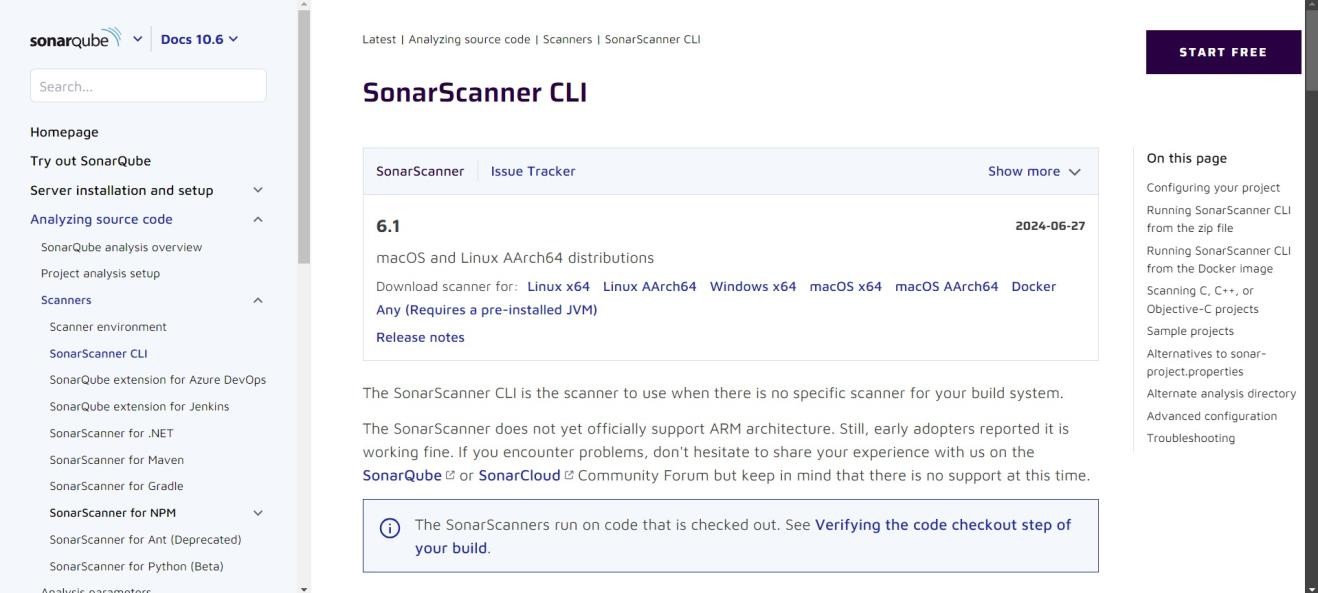
Experiment No: 8

**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.

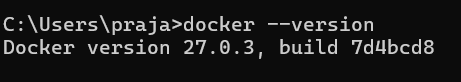
**PREREQUISITES:**

**Step 1:** Download sonar scanner [https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscan](https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/) [ner/](https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/) . Visit this link and download the sonarqube scanner CLI

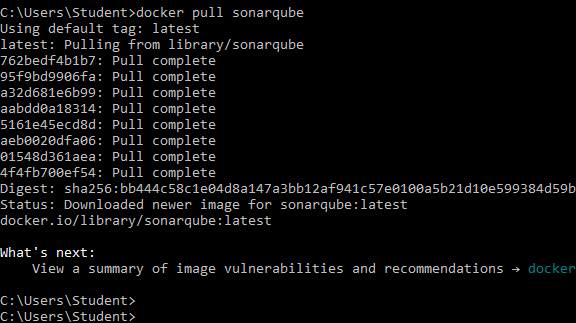


Extract the downloaded zip file in a folder.

**Step 2:** Docker Run **docker -v command .**If docker is not installed so install it



**Step 3:** Install sonarqube image Command: **docker pull sonarqube**

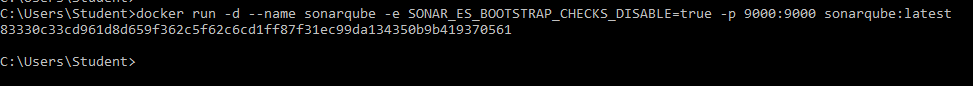


**Step 4:** Keep **Jenkins** installed on your system.

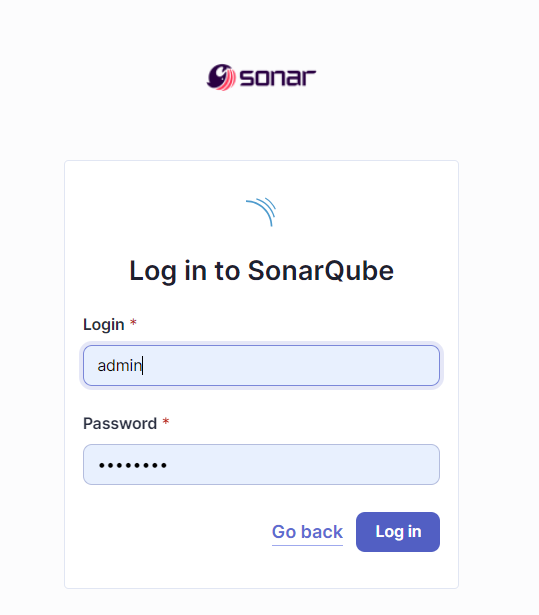
**EXPERIMENT STEPS:**

**Step1:** Run SonarQube image **docker run -d --name sonarqube -e SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000**

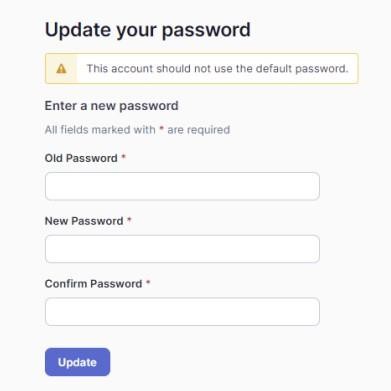
**sonarqube:latest .**This command will run the SonarQube image that was just installed using docker.



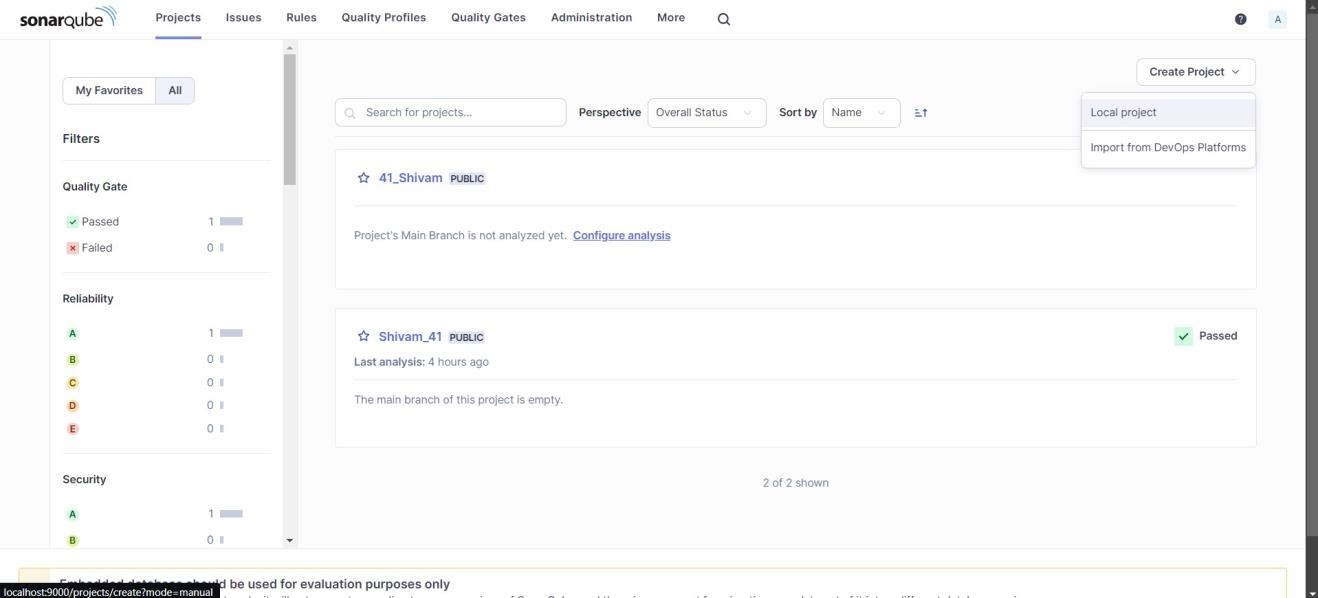
**Step 2:** Once the SonarQube image is started, you can go to **http://localhost:9000** to find the SonarQube that has started



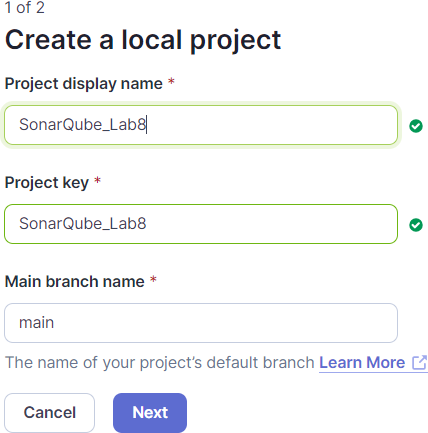
**Step 3:** On this interface, login with **username = ‘admin’** and **password = ‘admin**’. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.



**Step 4:** After changing the password, you will be directed to this screen. Click on **Create a Local Project**. Give the project a display name and project key

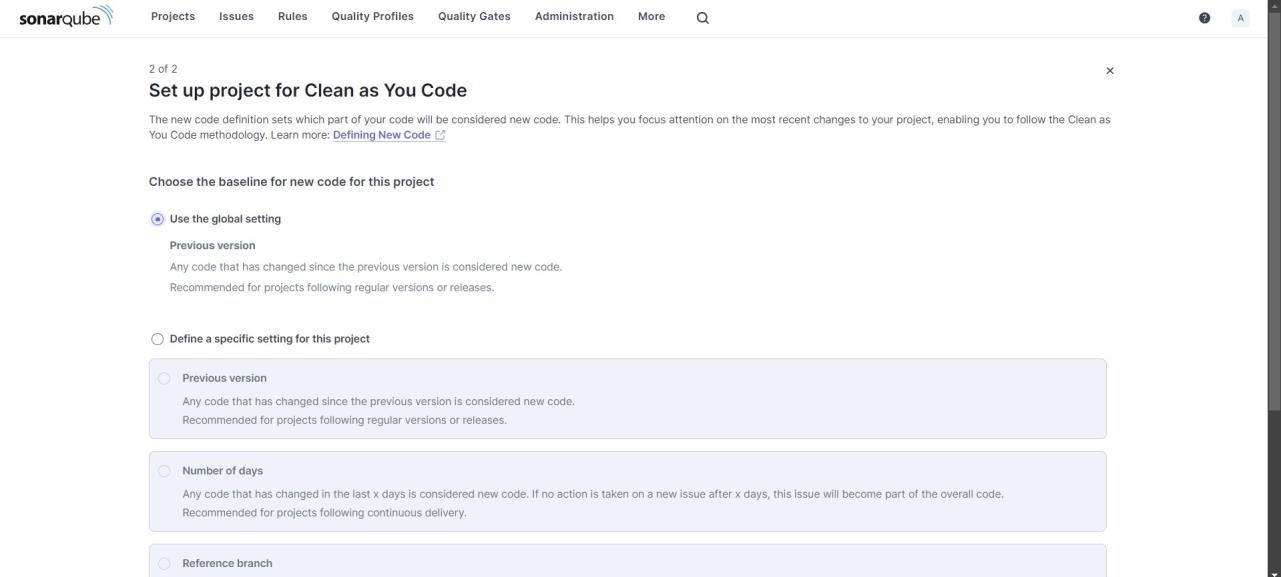


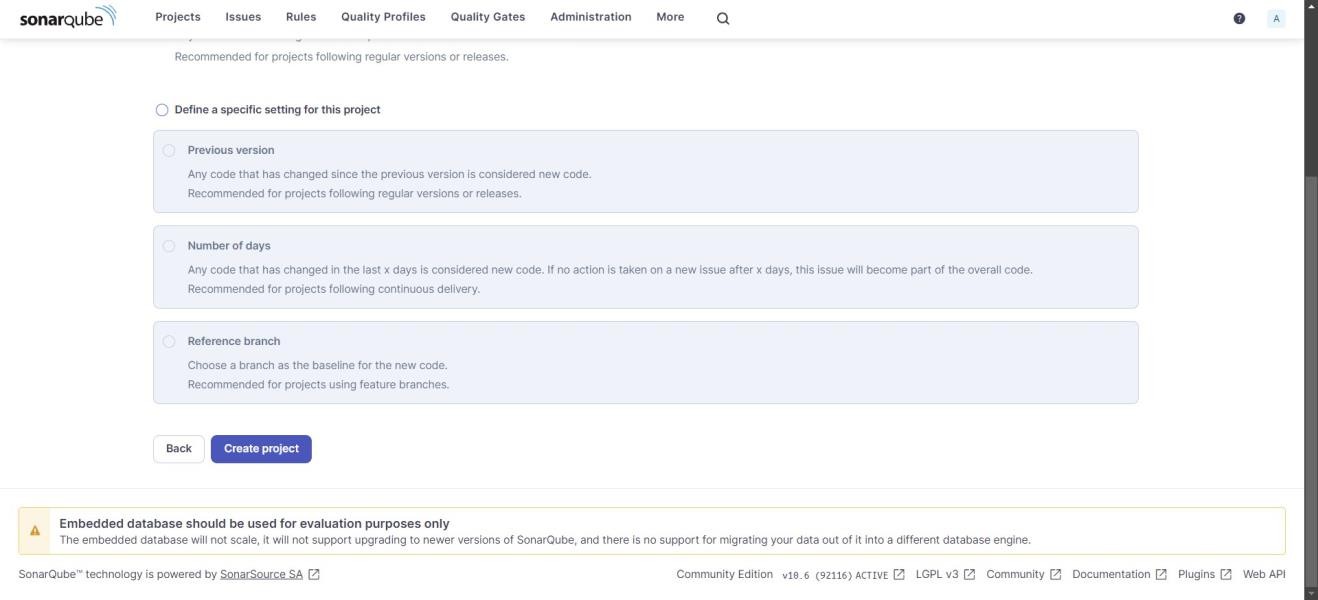
Click on Create Project



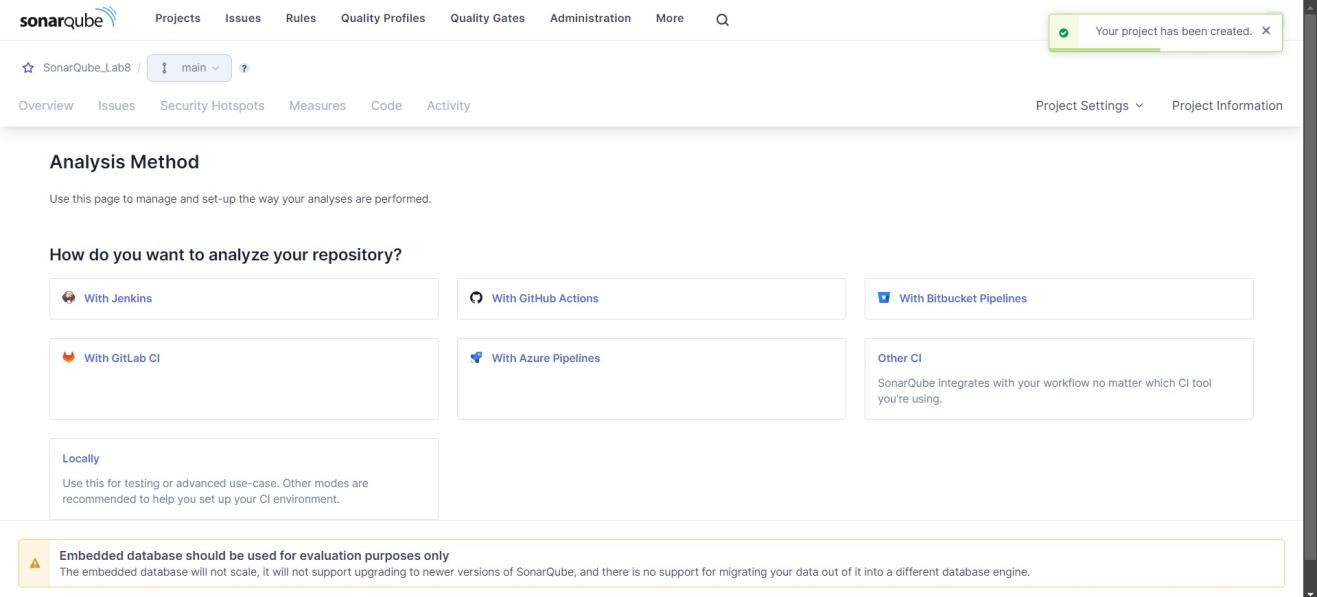
Set up the project as required and click on create.

In the Step 2 while creating the project,Sonarqube ask you regarding which code should be considered as the new code for examining it .



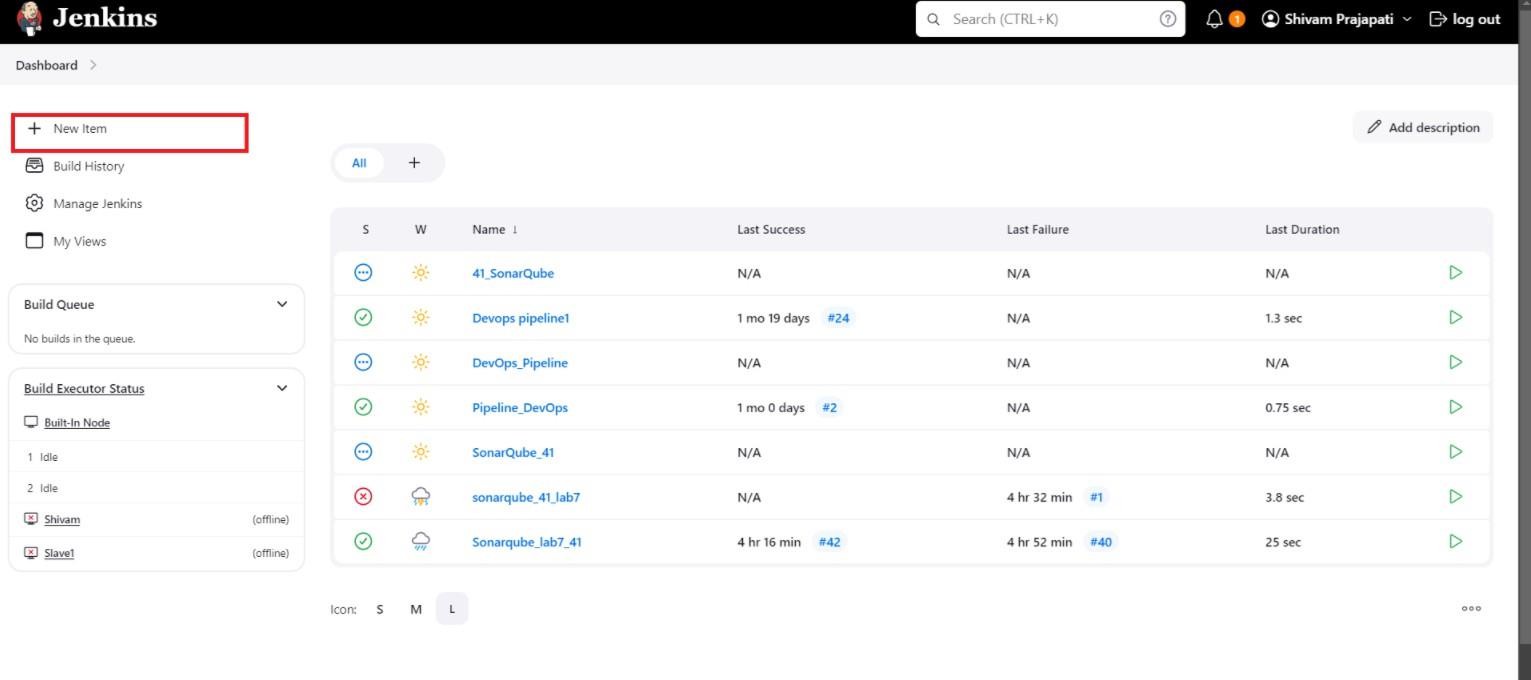


Click on Create



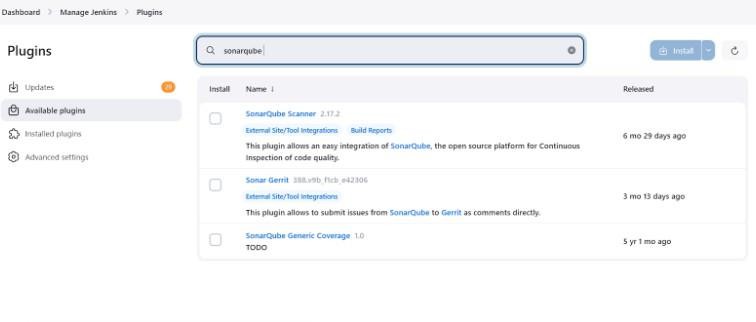
Project is created

**Step 5:** Open **Jenkins** on whichever port it is installed. (http://loaclhost:). Go to the new item

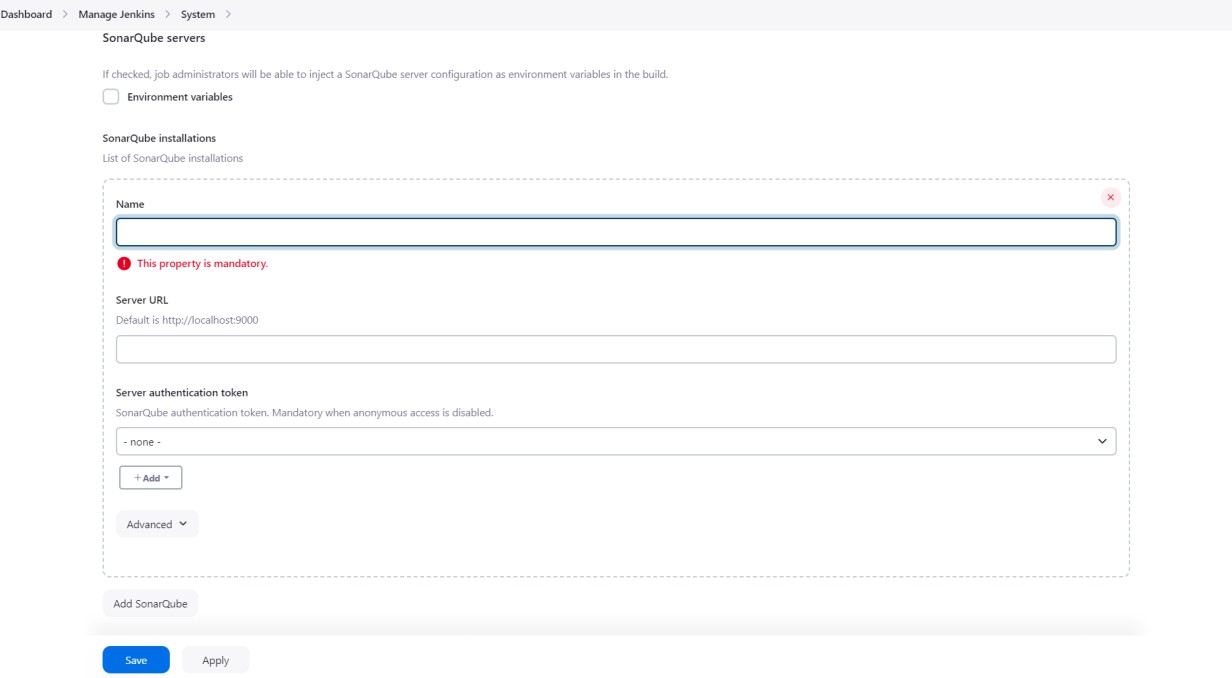


**Step 6:** Go to manage jenkins →available plugins then Search for **Sonarqube Scanner**

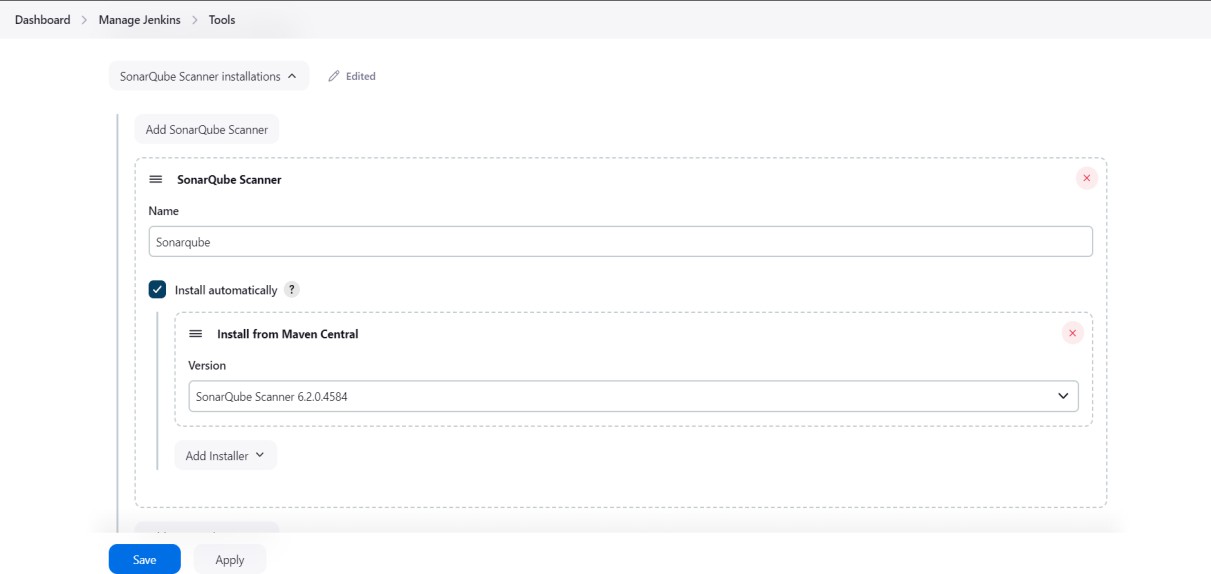
for Jenkins and install it



**Step 7:** Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.

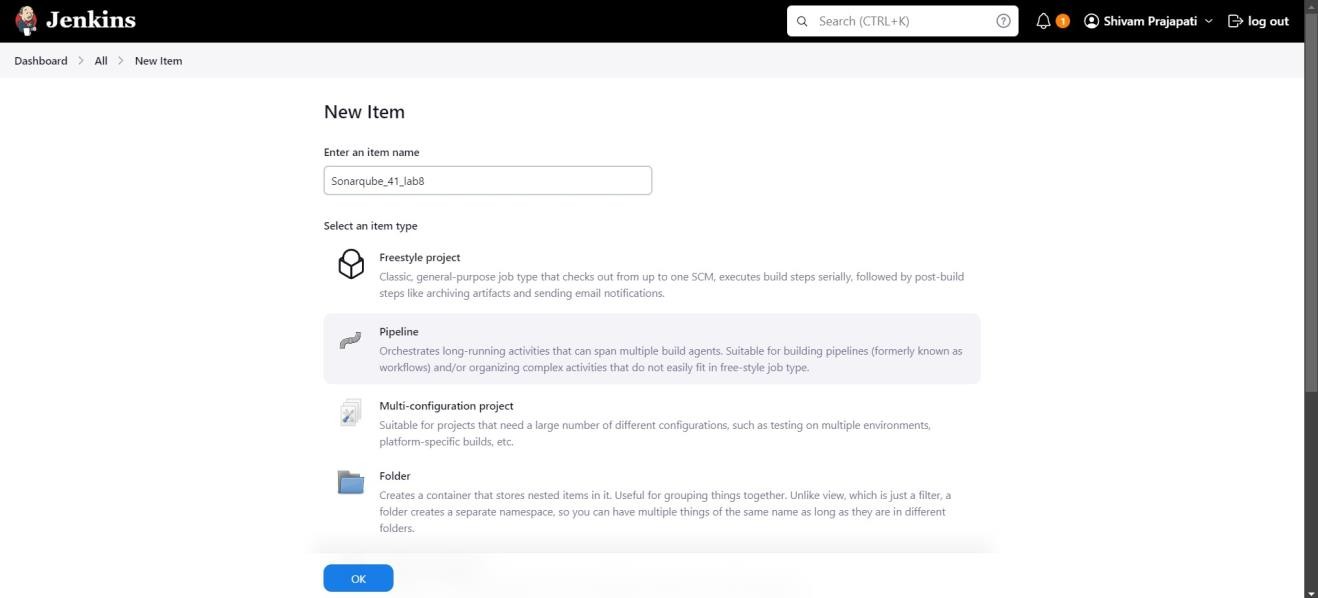


**Step 8:** Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose to install automatically.



**Step 9**: After configuring, click on **New Item** and select **Pipeline Project**

.



**Step 10**: Under Pipeline script, enter the following: node {

stage('Cloning the GitHub Repo') {

git 'https://github.com/shazforiot/GOL.git'

}

stage('SonarQube analysis') { withSonarQubeEnv('sonarqube') {

bat """

C:\\Users\\praja\\Downloads\\SonarqubeCLI\\sonar-scanner-6.1.0.4477-windows-x64\\bi n\\sonar-scanner.bat ^

-D sonar.login=admin ^

-D sonar.password=Shivam2@ ^

-D sonar.projectKey=SonarQube\_Lab8 ^

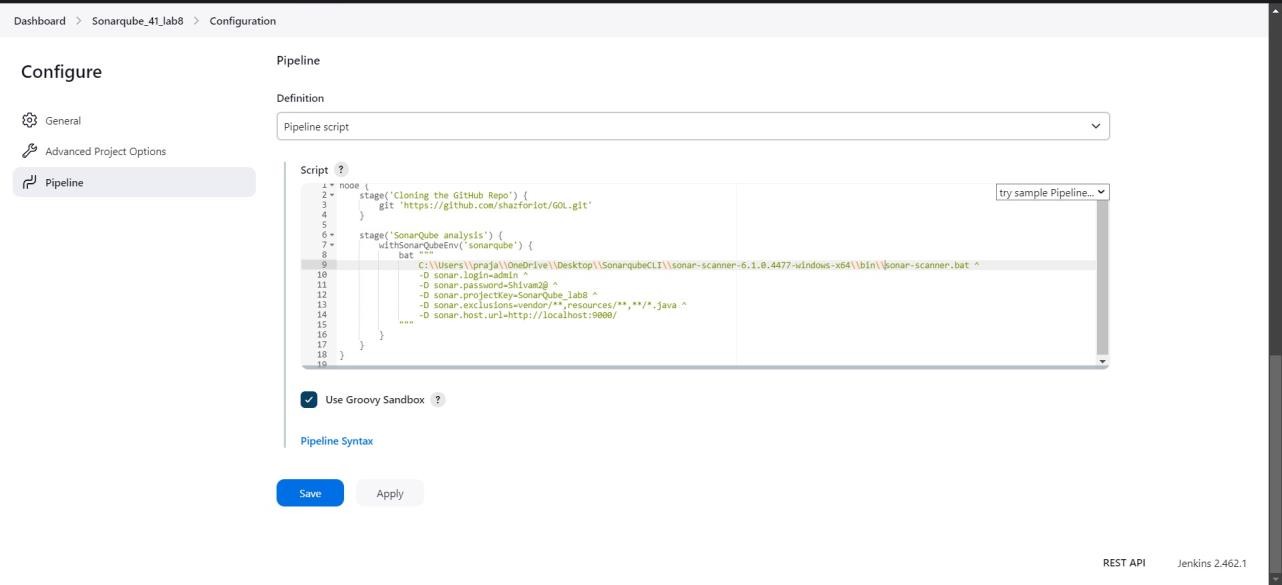
-D sonar.exclusions=vendor/\*\*,resources/\*\*,\*\*/\*.java ^

-D sonar.host.url=http://localhost:9000/ """

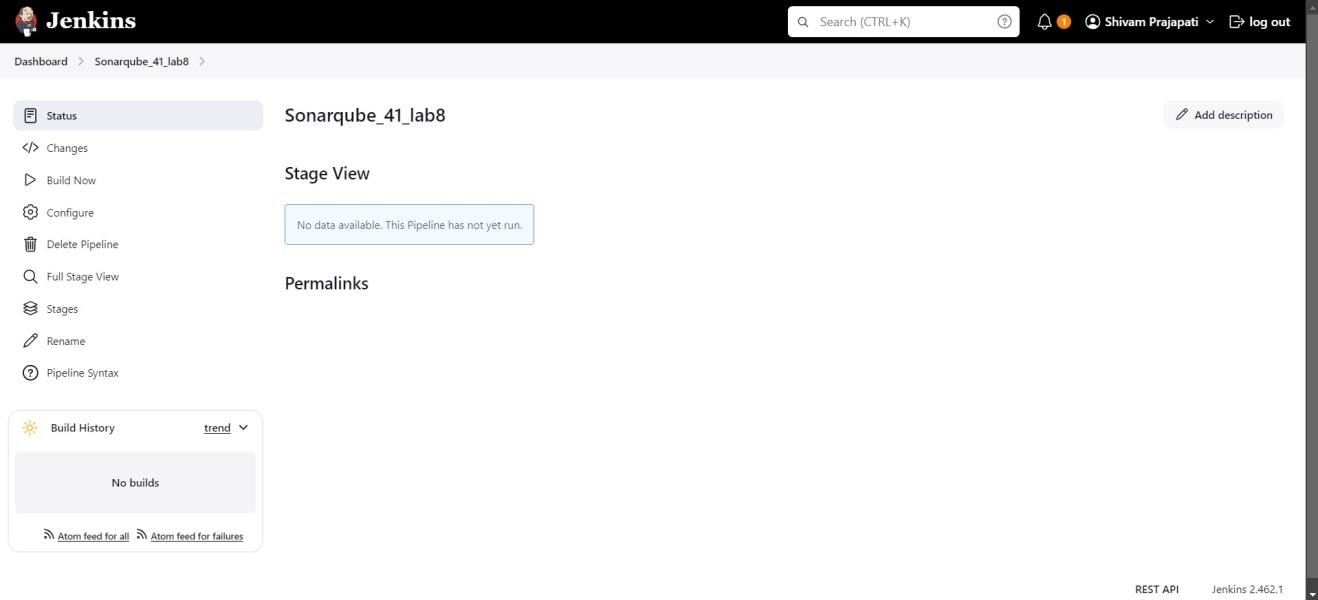
}

}

}

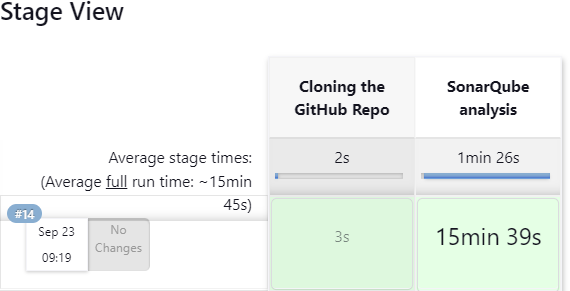


Click on save.



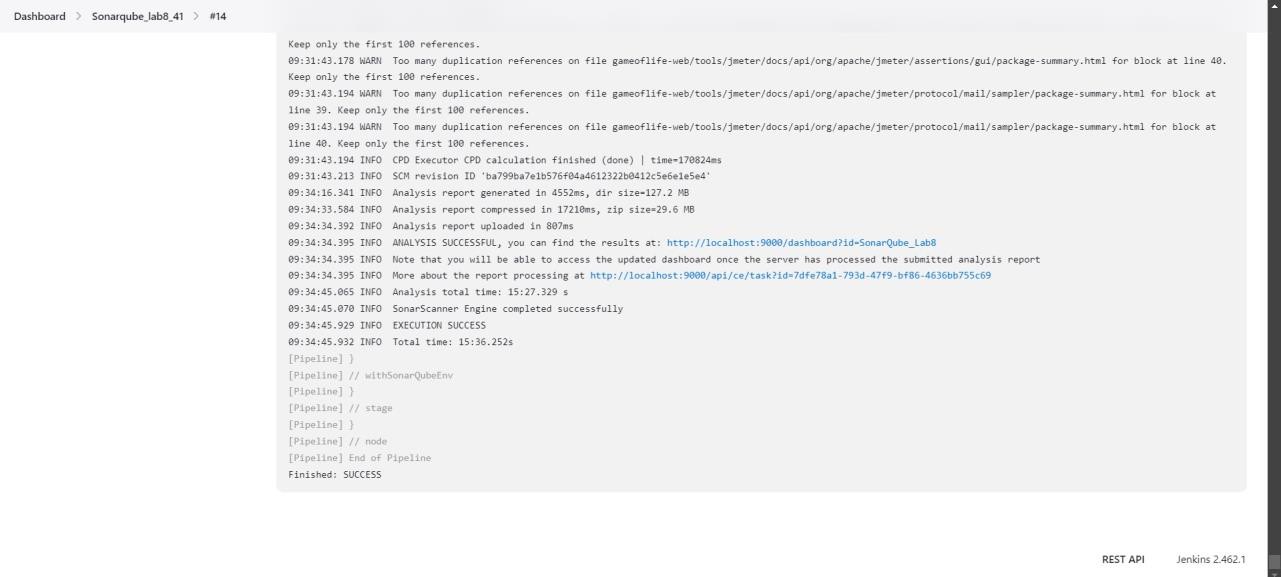
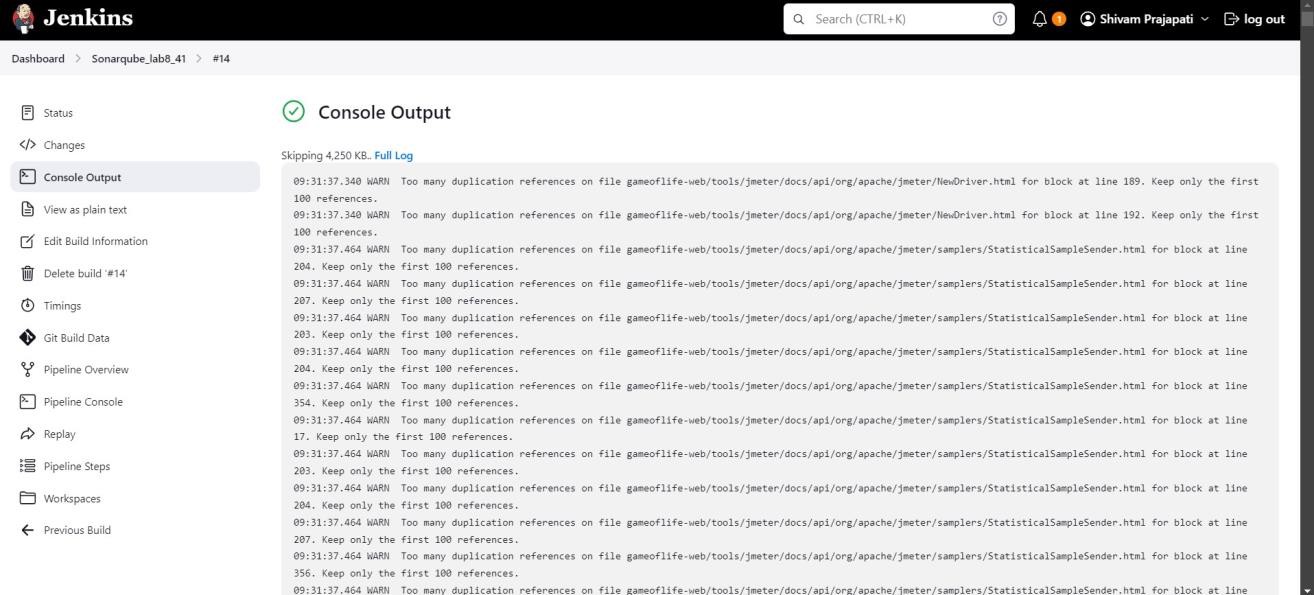
This is a Java sample project with many repetitive sections and coding issues that SonarQube will be able to detect during analysis.

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



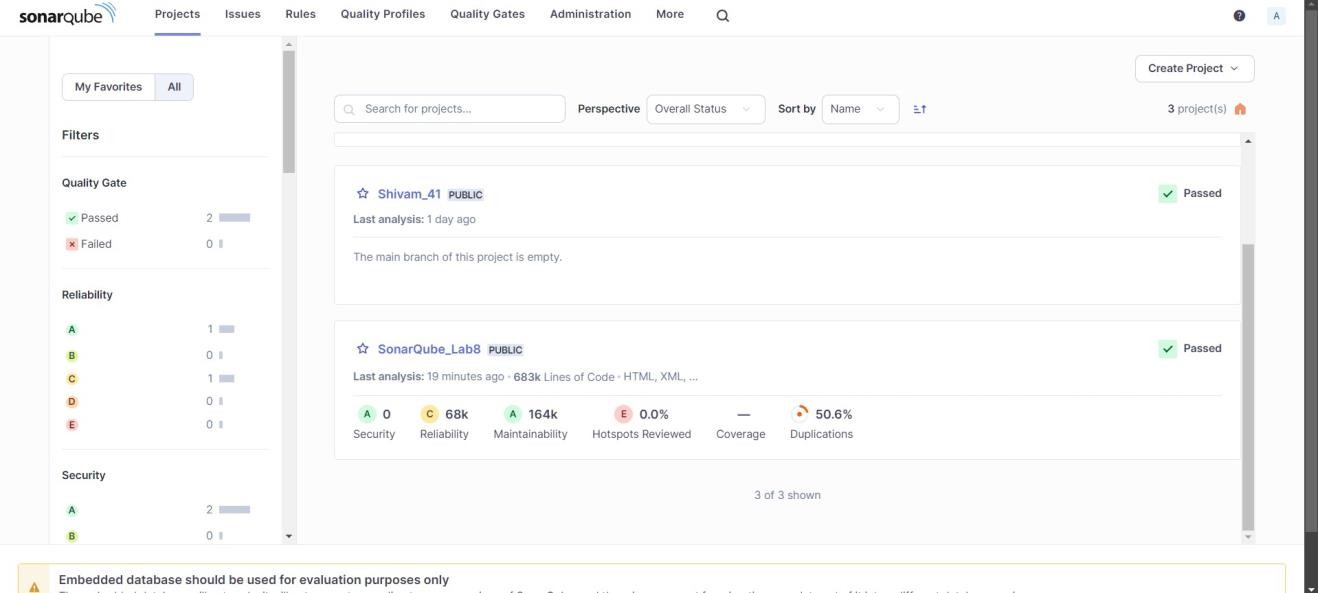
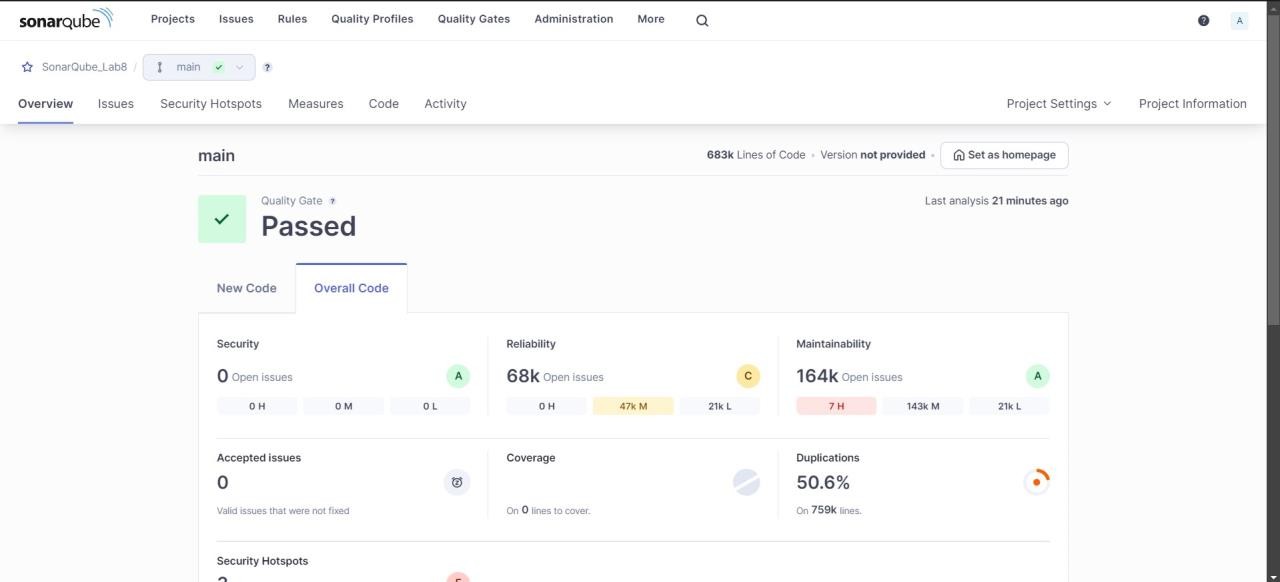
The problem was C:\windows\system32 was not there so we need to add in our environment variable .

Now Check the console output once



Successfully BUILD

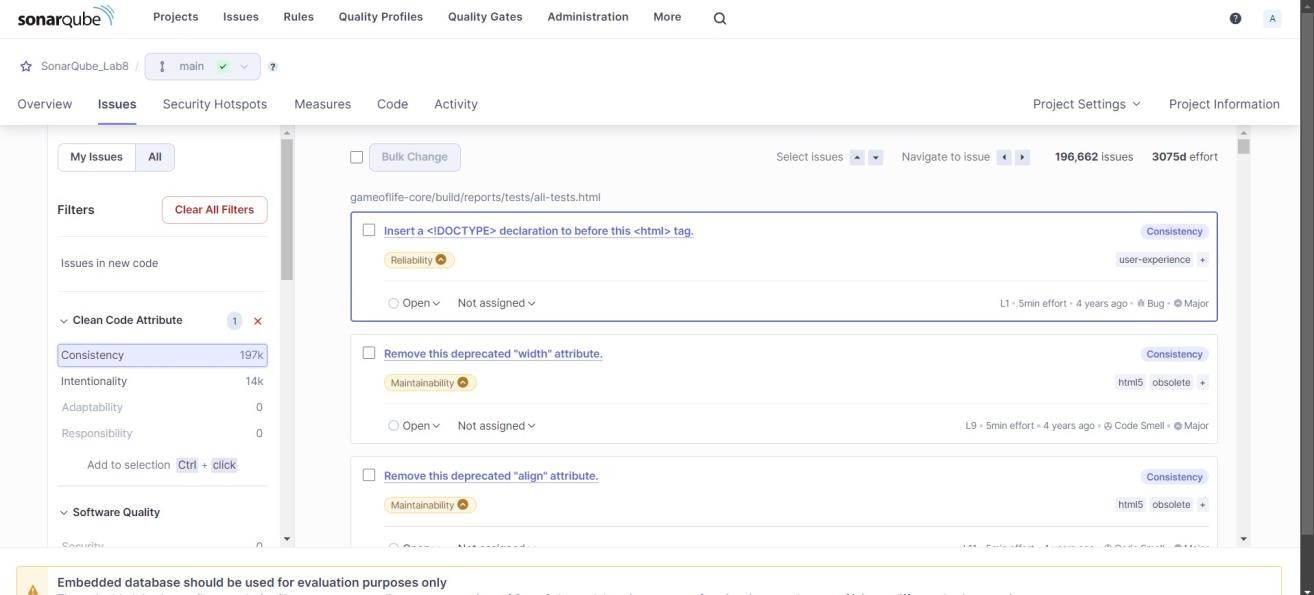
**Step 12**: After the build is finished, return to SonarQube and review the linked project in detail.



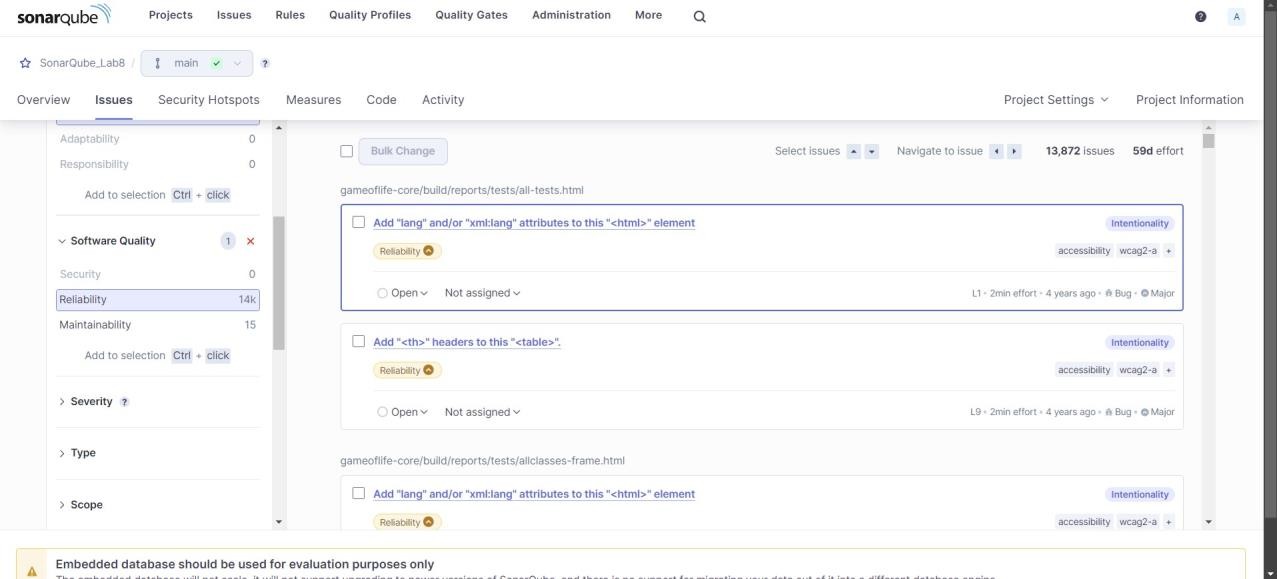
Under different options on the navbar , we can check all the issues with the code.

**UNDER ISSUES:**

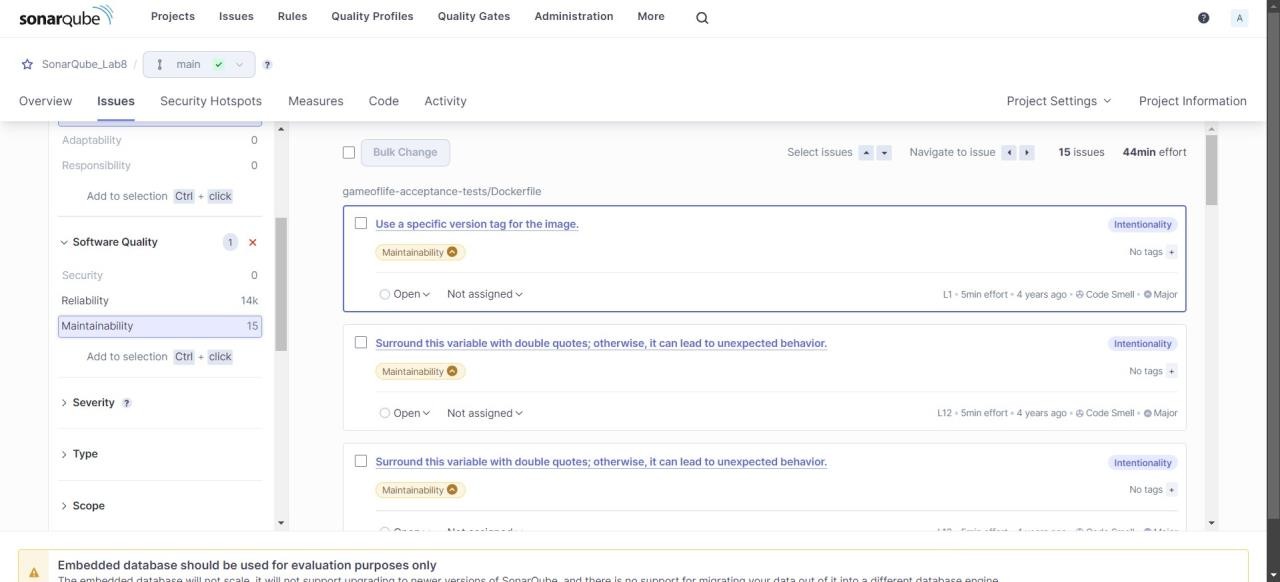
1. Consistency



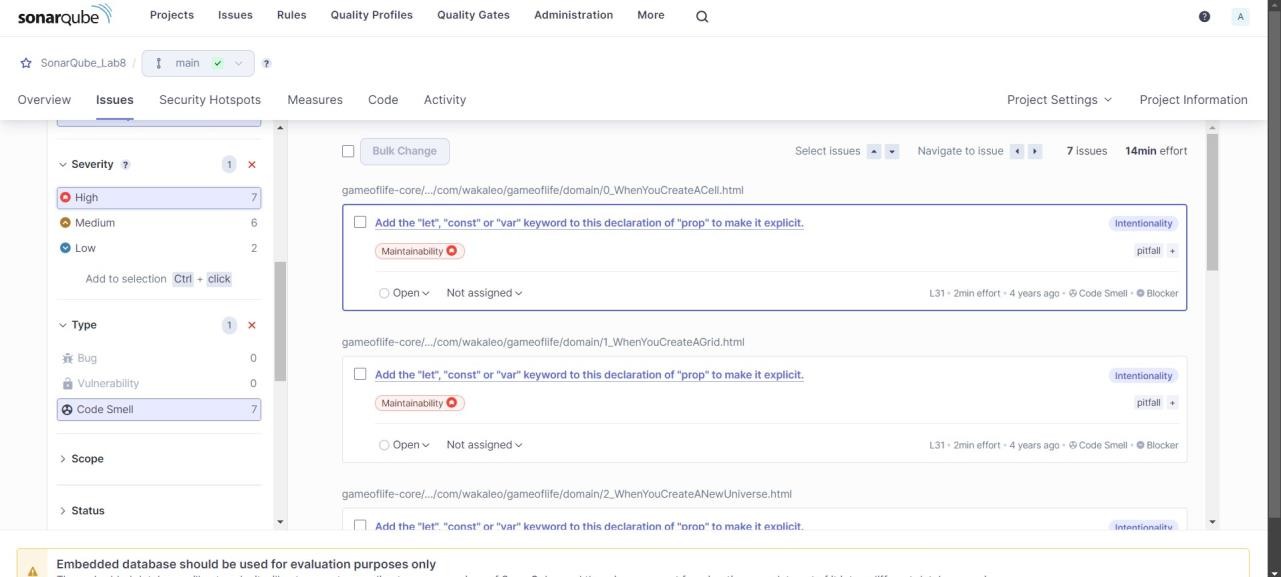
1. Reliability



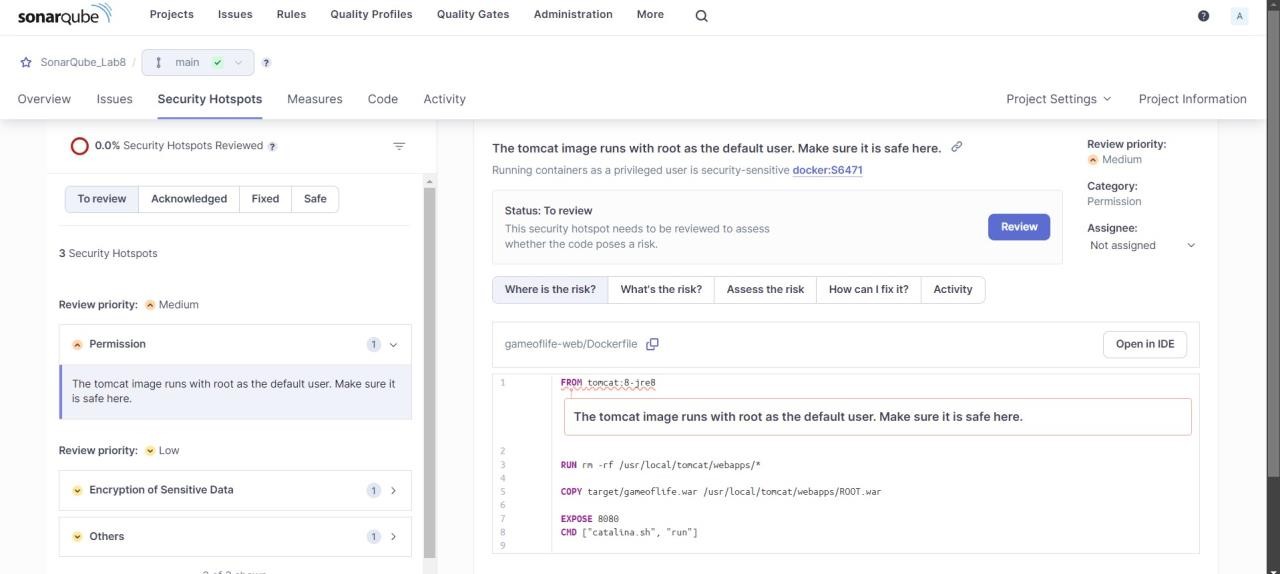
1. Maintainability



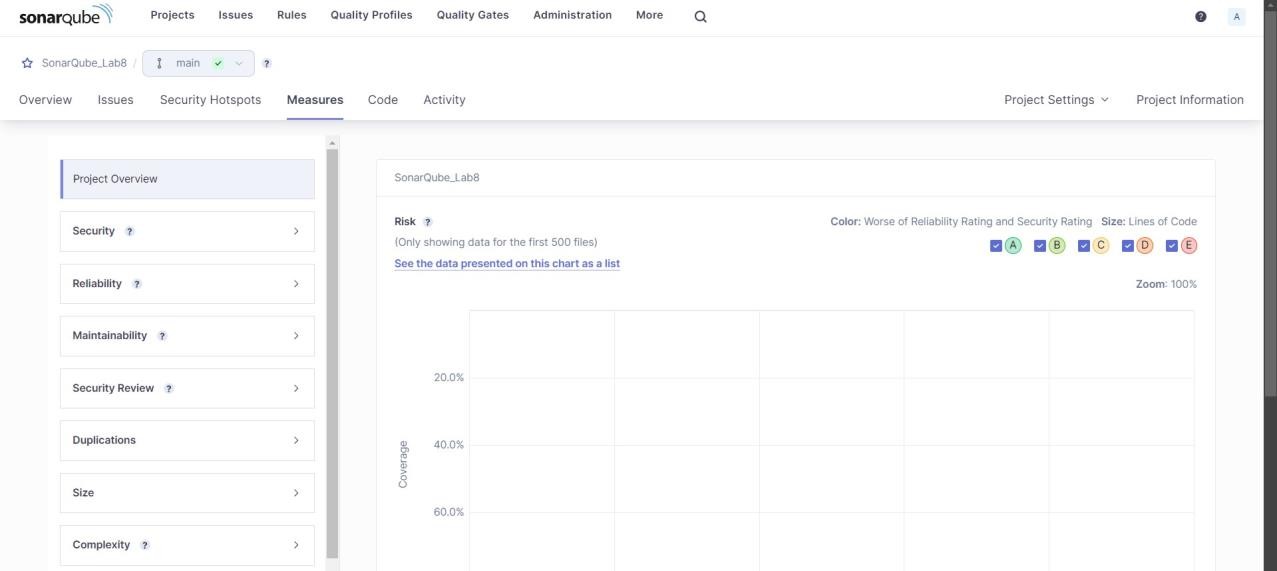
1. Severity

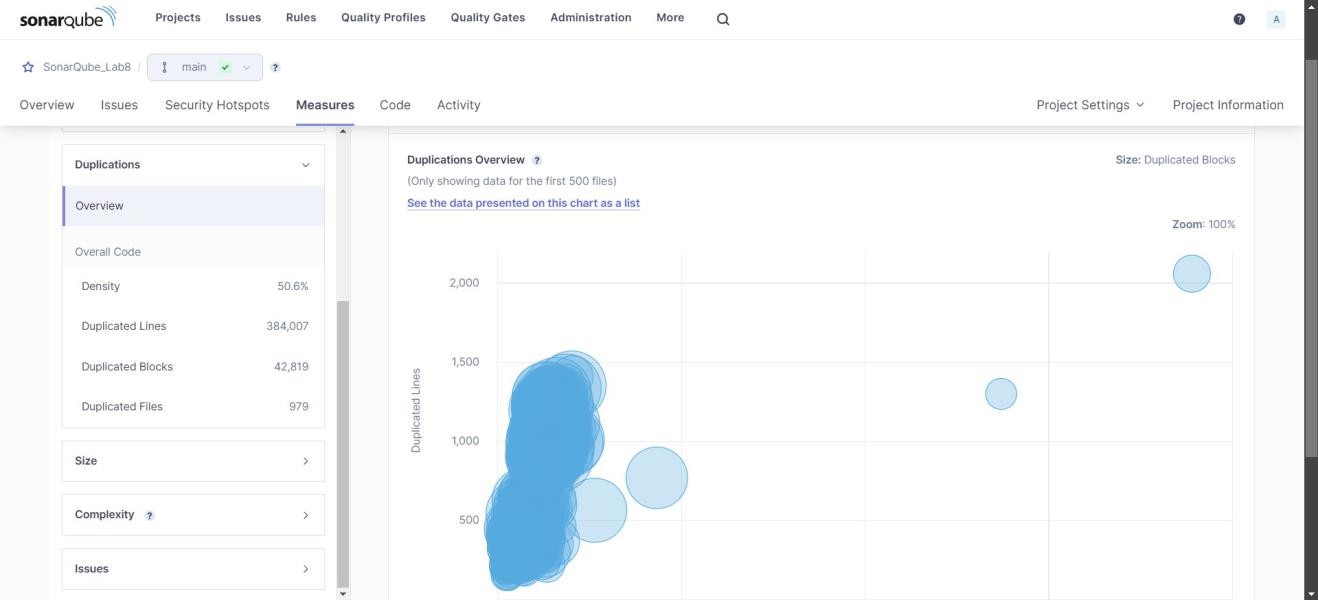


**UNDER SECURITY HOTSPOT:**



**UNDER MEASURES:**





**CONCLUSION:**

In this experiment, we demonstrated how to perform static code analysis using Jenkins CI/CD Pipeline with SonarQube integration. We created a pipeline project with a specific script that contains all the instructions necessary to run SonarQube analysis. After configuring Jenkins appropriately, we built the project. The analyzed code in this experiment had several issues, such as errors, bugs, and duplications, all of which were detected and displayed in the linked SonarQube project.