

**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.

## 1. Creation of project on Sonarqube

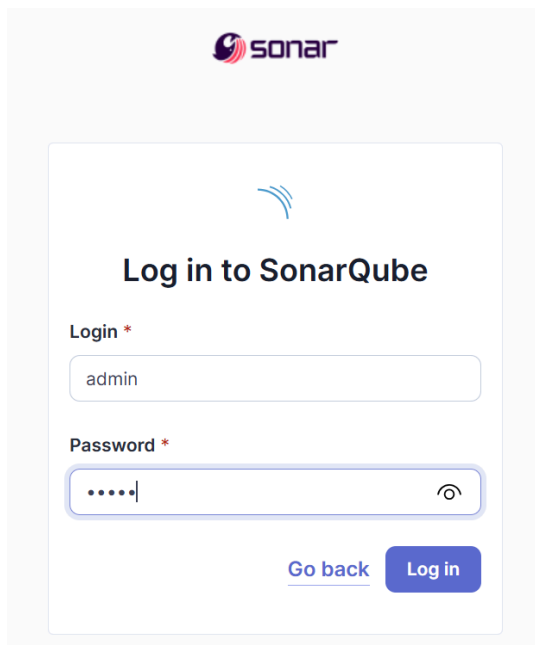
1. Open the command prompt and perform this command

(\*\* Docker must be Installed before running this command. \*\*)

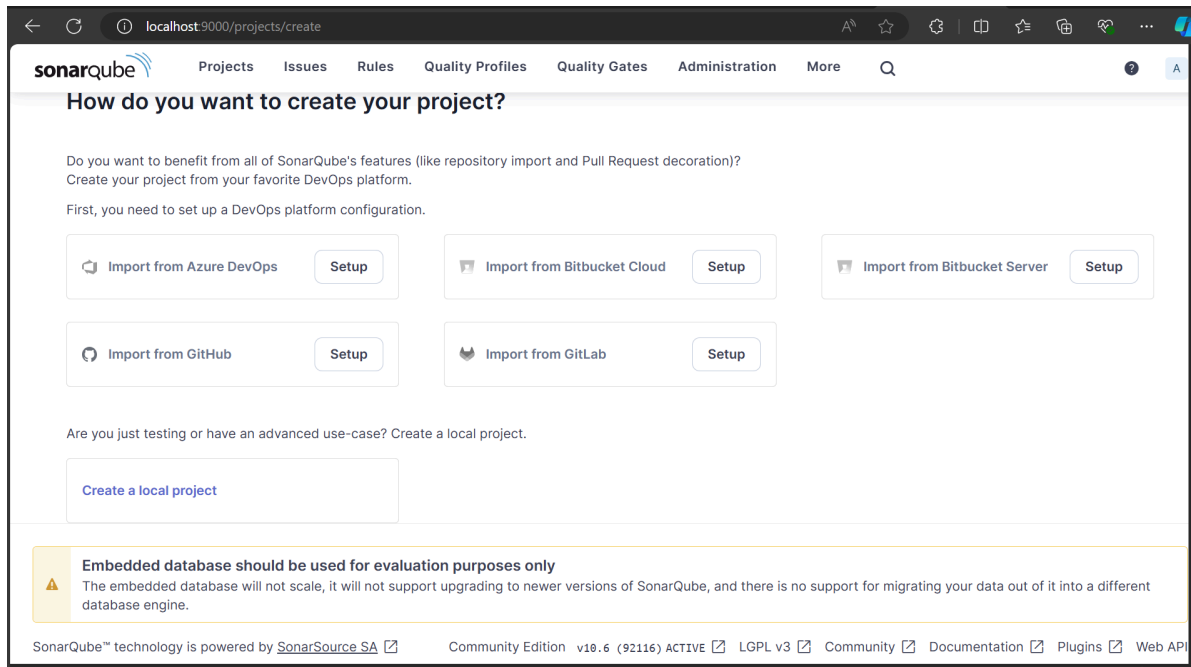
`docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest`

```
PS C:\Users\Sadneya> 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: sha256:72e9fec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
19948c8162691fd8be61d035e355c4fe4a6a2fd0a15237e94f75c0857ff7e2ff
```

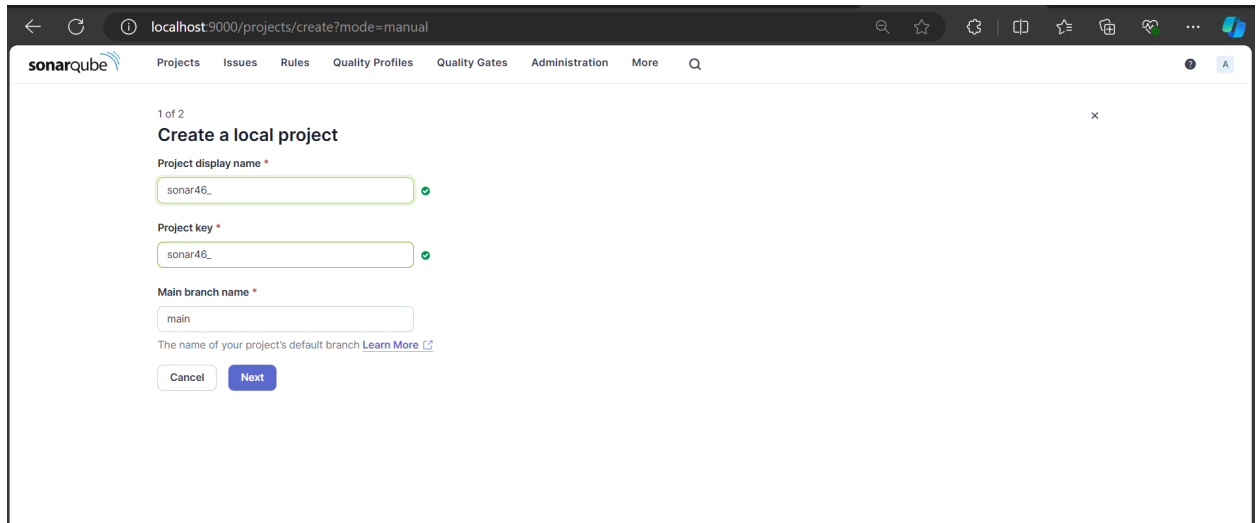
2. Then after its successful execution, run the sonarqube at <http://localhost:9000>
3. Then Login as Username admin and password admin.



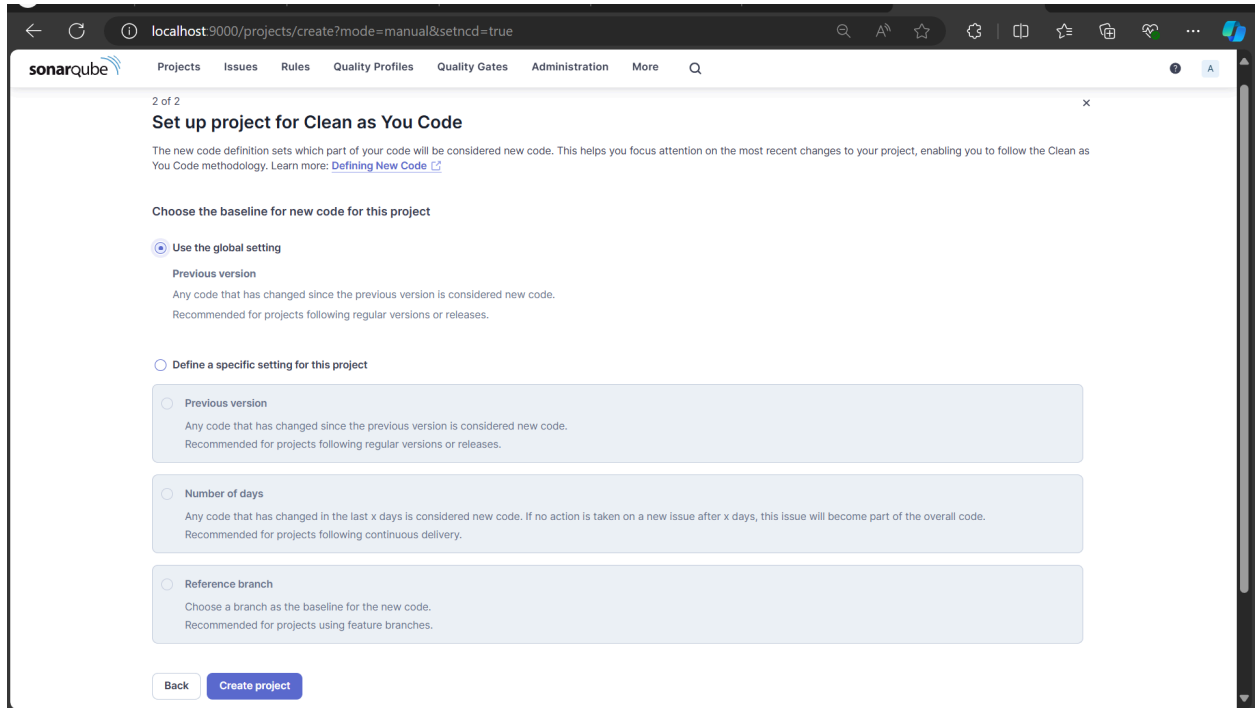
The image shows the SonarQube login page. At the top is the Sonar logo. Below it is a white box with a blue Sonar icon and the text "Log in to SonarQube". There are two input fields: "Login \*" with the text "admin" and "Password \*" with masked characters ".....". To the right of the password field is an eye icon. At the bottom of the box are two buttons: "Go back" (a link) and "Log in" (a blue button).



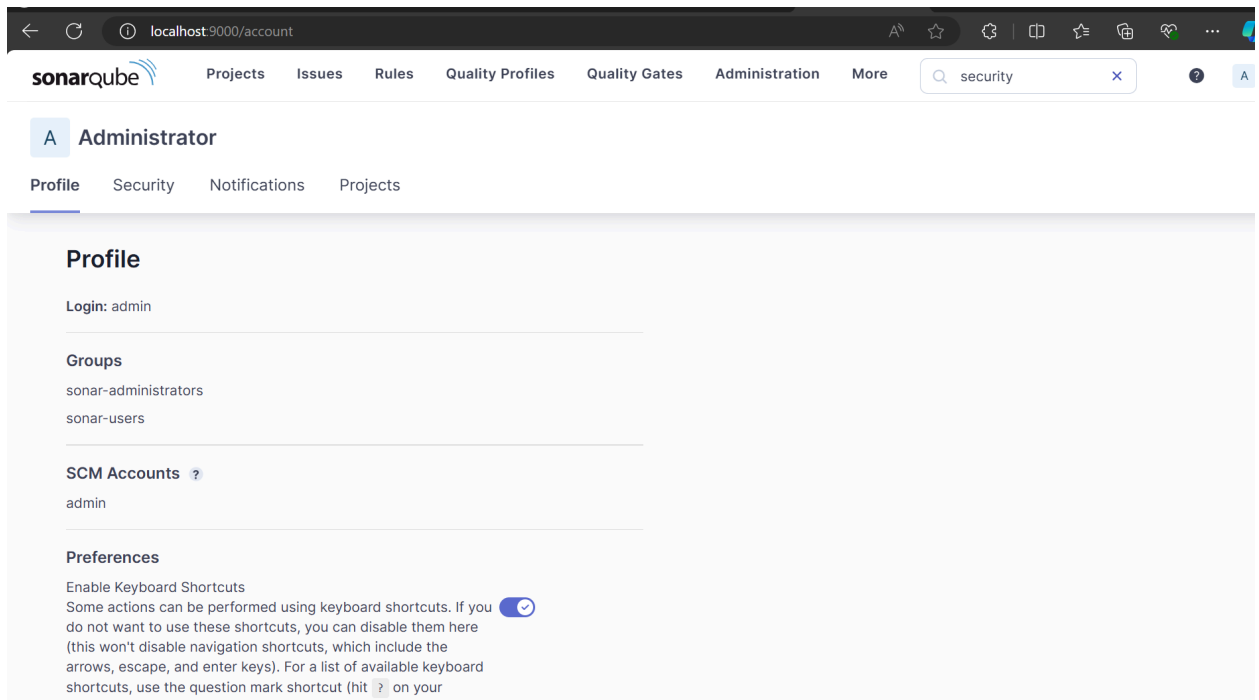
4. Then create a project. Here I have given the name as “sonar46”. Keep branch name main only and then click on next.



5. Then in the Setup project for clean as you code they will ask to choose the baseline for new code for this project. Choose Use the global setting. Then click on create project.




6. Then click on your account profile and select my account. Inside this click on security option.



7. Then set the name of your token and here I have chosen the global scope of the token and then clicked on generate token, thus token created successfully. (Here I have given my token name as 'jenkins token'.)

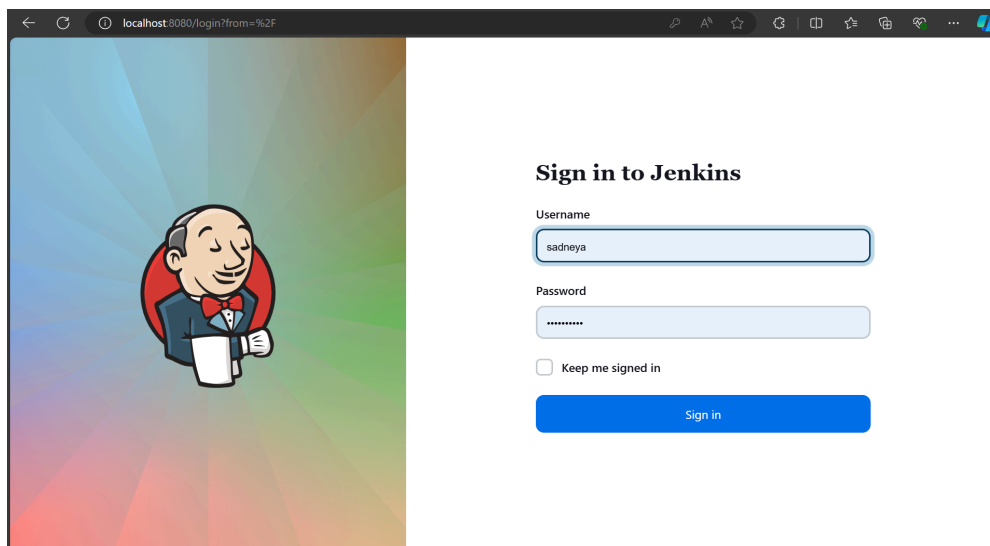
✓ New token "Jenkins Token" has been created. Make sure you copy it now, you won't be able to see it again!

sqa\_ab7e392ec51669ad950952a9bdbd2dbac3b7498e 

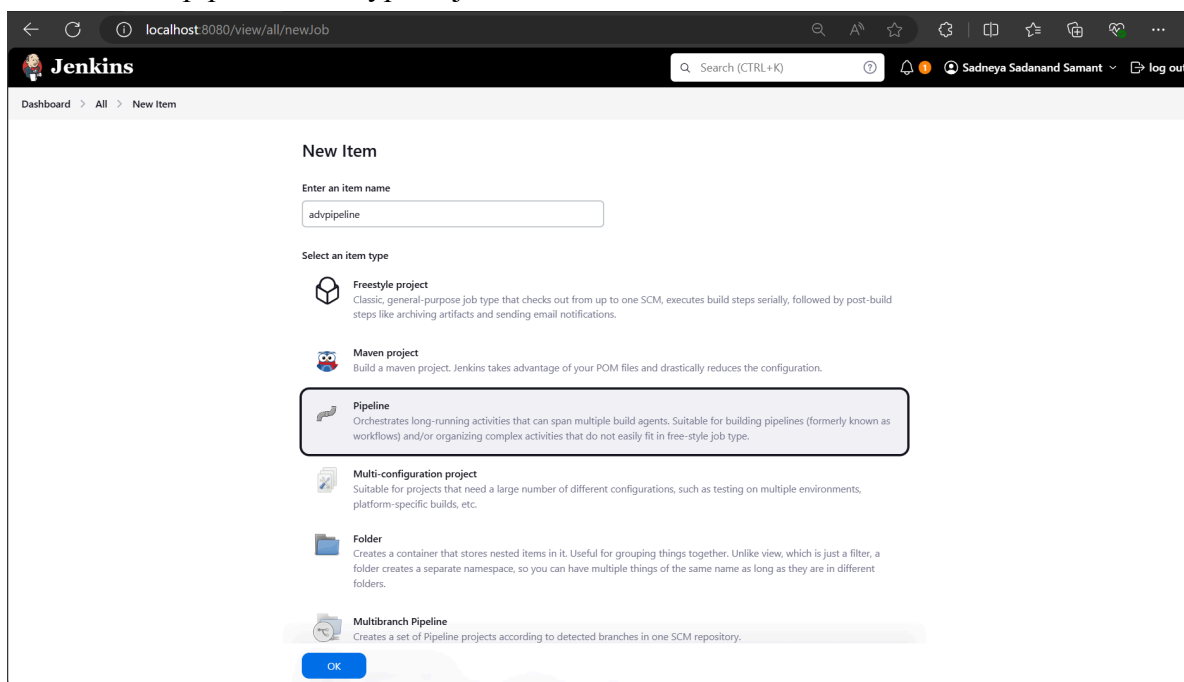
Name	Type	Project	Last use	Created	Expiration	
Jenkins Token	Global		Never	September 21, 2024	October 21, 2024	<a href="#">Revoke</a>

## 2. Creation and building of jenkins pipeline

1. Run the jenkins at <http://localhost:8080> and enter username and password and click on sign in



2. Then create a pipeline item type in jenkins.



- Go to [SonarScanner CLI \(sonarsource.com\)](https://sonarsource.com) this website and download the latest version of sonar scanner on the computer.
- Then select the machine here I am using windows x64

Latest | Analyzing source code | Scanners | SonarScanner CLI

## SonarScanner CLI

SonarScanner
Issue Tracker
Show more

**6.2**
2024-09-17

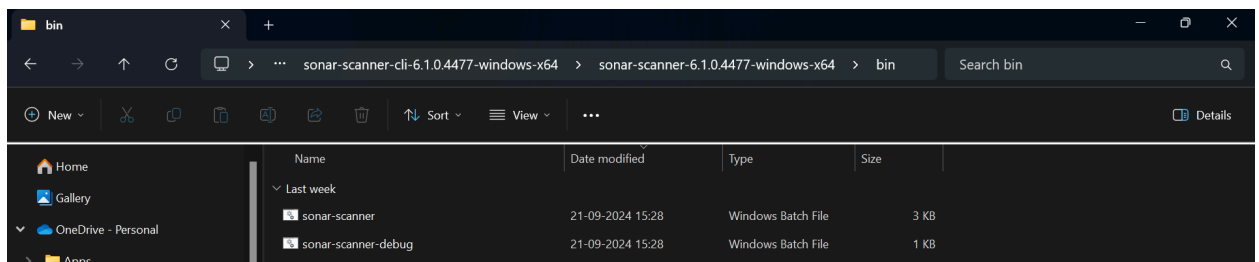
Support PKCS12 truststore generated with OpenSSL

Download scanner for: [Linux x64](#) [Linux AArch64](#) [Windows x64](#) [macOS x64](#) [macOS AArch64](#) [Docker](#) [Any \(Requires a pre-installed JVM\)](#)

[Release notes](#)

Once it is downloaded, copy its path as it is required in the pipeline script.

C:\\Users\\Sadneya\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat"



- Inside the pipeline script section write the following script.

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

```
"C:\\Users\\Sadneya\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat" ^
-Dsonar.login=sqp_fab91d70fc5db32bb83641a23decbbf04470ba2d ^ (login token)
```

```
-Dsonar.projectKey=sonar46 ^ (your project name)
-Dsonar.exclusions=vendor/**,resources/**,**/*.java ^ (libraries u want to
exclude)
-Dsonar.host.url=http://localhost:9000 (server URL)
""""
}
}
}
```

The screenshot shows the Jenkins configuration page for a pipeline named 'soanrpipeline'. The page is titled 'Configure' and has a sidebar with 'General', 'Advanced Project Options', and 'Pipeline'. The 'Advanced Project Options' tab is selected. Under 'Pipeline', the 'Definition' is set to 'Pipeline script'. The 'Script' section contains a Groovy script for a pipeline with two stages: 'Cloning the GitHub Repo' and 'SonarQube analysis'. The 'SonarQube analysis' stage uses the 'withSonarQubeEnv' plugin and sets various Sonar properties. The 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons.

```
1 node {
2   stage('Cloning the GitHub Repo') {
3     git "https://github.com/shazforiot/GOL.git"
4   }
5   stage('SonarQube analysis') {
6     withSonarQubeEnv('sonarqube') {
7       bat """
8         "C:\Users\Sadneya\Downloads\sonar-scanner-cli-6.1.0.4477-windows-x64\sonar-scanner-6.1.0.4477-windows-x64\bin\sonar-scanner.bat" ^
9         -Dsonar.login=sqp_fab91d78fc5db32bb83641a23decbbf04470ba2d ^
10        -Dsonar.projectKey=sonar46 ^
11        -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
12        -Dsonar.host.url=http://localhost:9000
13        """
14      }
15    }
16  }
17 }
```

6. Then save and apply the changes. Then click on build option.

The screenshot shows the Jenkins dashboard for the 'advdevops project'. The dashboard has a sidebar with 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'SonarQube', and 'Rename'. The 'Build Now' button is highlighted.

Then it throws an error.

### ❌ Console Output

```

Started by user Sadneya Sadanand Samant
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\soanrpipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/shazforiot/GOL.git
> git.exe init C:\ProgramData\Jenkins\.jenkins\workspace\soanrpipeline # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git.exe --version # timeout=10
> git --version # 'git version 2.43.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
17:42:26.066 INFO   JRE PROVIDING: OS[WINDOWS], ARCH[amd64]
17:42:26.066 INFO   EXECUTION FAILURE
17:42:26.066 INFO   Total time: 0.766s
17:42:26.067 ERROR Error during SonarScanner CLI execution
java.lang.IllegalStateException: Error status returned by url [http://localhost:9000/api/v2/analysis/jres?os=windows&arch=amd64]: 401
    at org.sonarsource.scanner.lib.internal.http.ServerConnection.callUrl(ServerConnection.java:182)
    at org.sonarsource.scanner.lib.internal.http.ServerConnection.callApi(ServerConnection.java:145)
    at org.sonarsource.scanner.lib.internal.http.ServerConnection.callRestApi(ServerConnection.java:123)
    at org.sonarsource.scanner.lib.internal.JavaRunnerFactory.getJreMetadata(JavaRunnerFactory.java:159)
    at org.sonarsource.scanner.lib.internal.JavaRunnerFactory.getJreFromServer(JavaRunnerFactory.java:138)
    at org.sonarsource.scanner.lib.internal.JavaRunnerFactory.createRunner(JavaRunnerFactory.java:85)
    at org.sonarsource.scanner.lib.internal.ScannerEngineLauncherFactory.createLauncher(ScannerEngineLauncherFactory.java:53)
    at org.sonarsource.scanner.lib.ScannerEngineBootstrapper.bootstrap(ScannerEngineBootstrapper.java:118)
    at org.sonarsource.scanner.cli.Main.analyze(Main.java:75)
    at org.sonarsource.scanner.cli.Main.main(Main.java:63)
17:42:26.068 ERROR
17:42:26.068 ERROR Re-run SonarScanner CLI using the -X switch to enable full debug logging.
[Pipeline] }
WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE

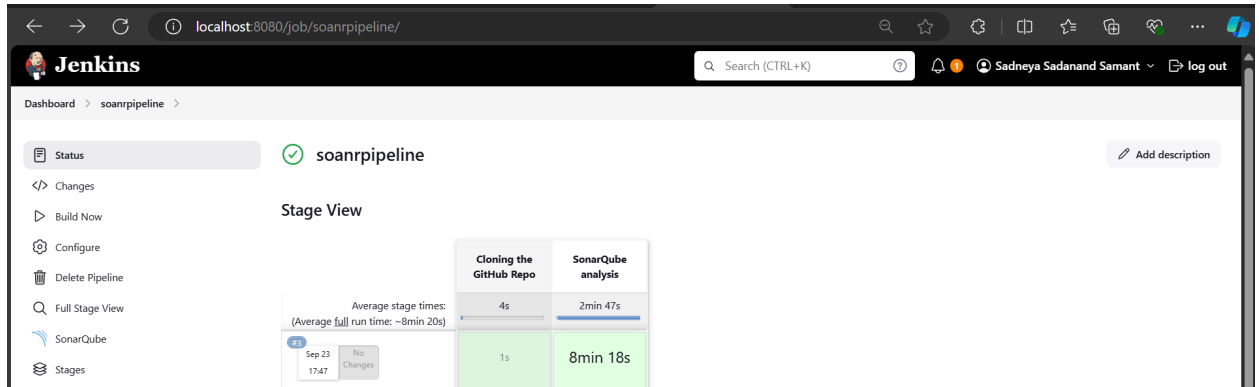
```

This error was due to failure in login and connection. So I generated the token again and replaced it in the pipeline script inside the configure of the pipeline.

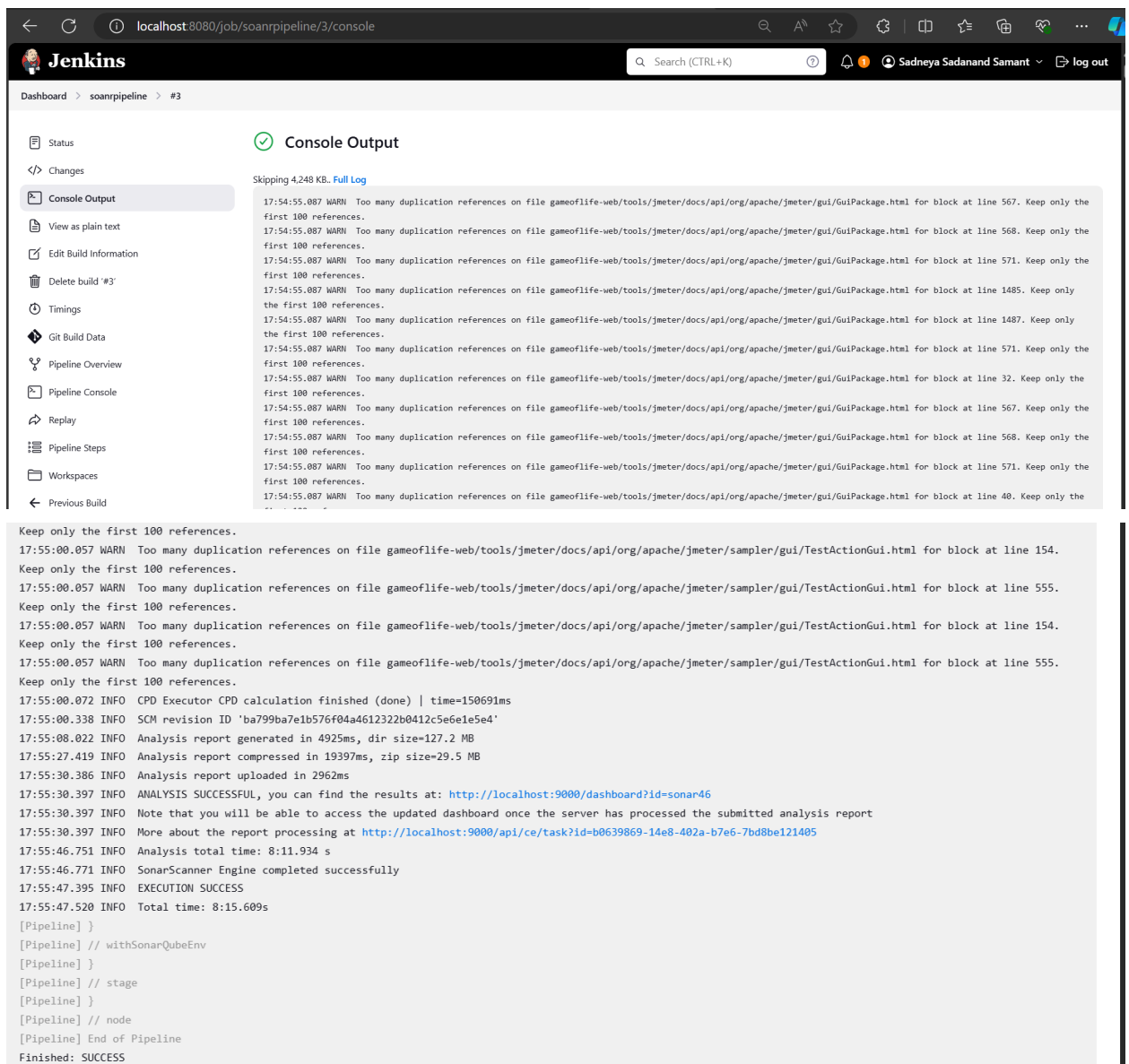
ie. `-Dsonar.login=(new token)`

7. Then I again clicked on build ,but this time the build got executed successfully.

### 3. Analysis by sonarqube

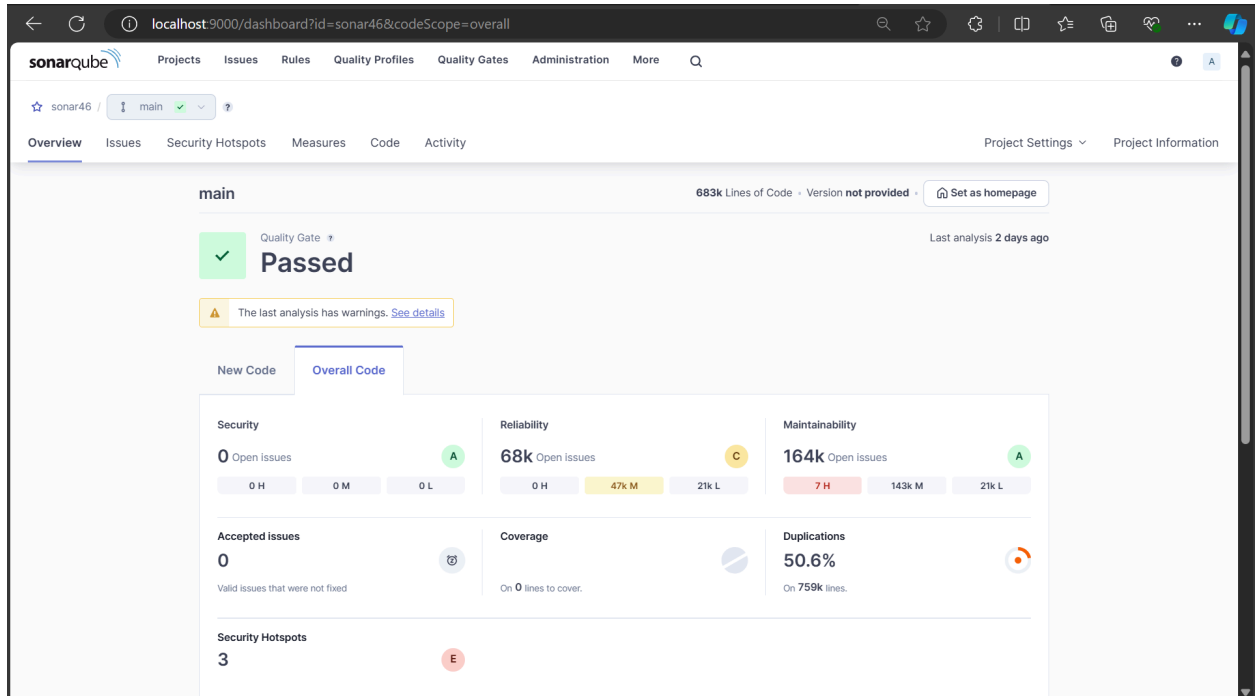


### 8. Check the Console Output.





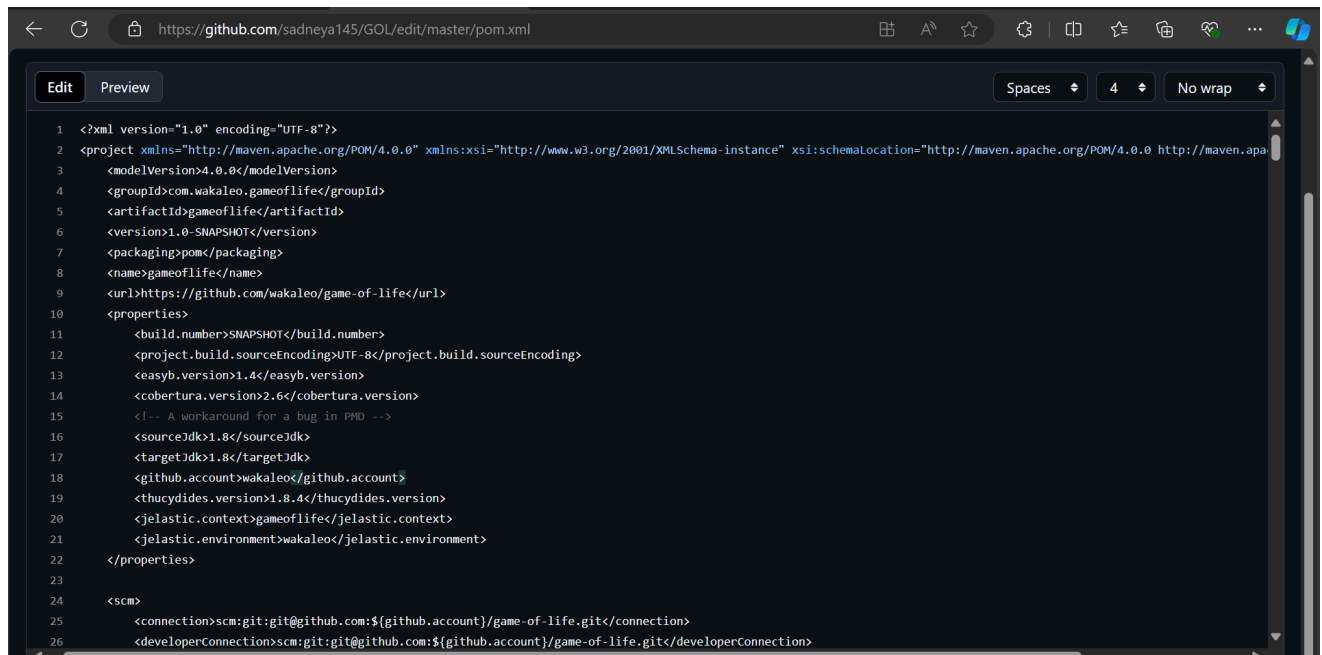
- Then go to SonarQube. Then go inside the project that you created, where it will show output passed.



Thus Project Build successfully.

#### 4. Analysis by sonarqube after changing code

- Then I made the changes in the [pom.xml](#) file to see the analysis given by sonarqube. In these I have introduced the hardcoded credentials and an unused variable:
- I have changed the properties in the [pom.xml](#) from



**To this**

```

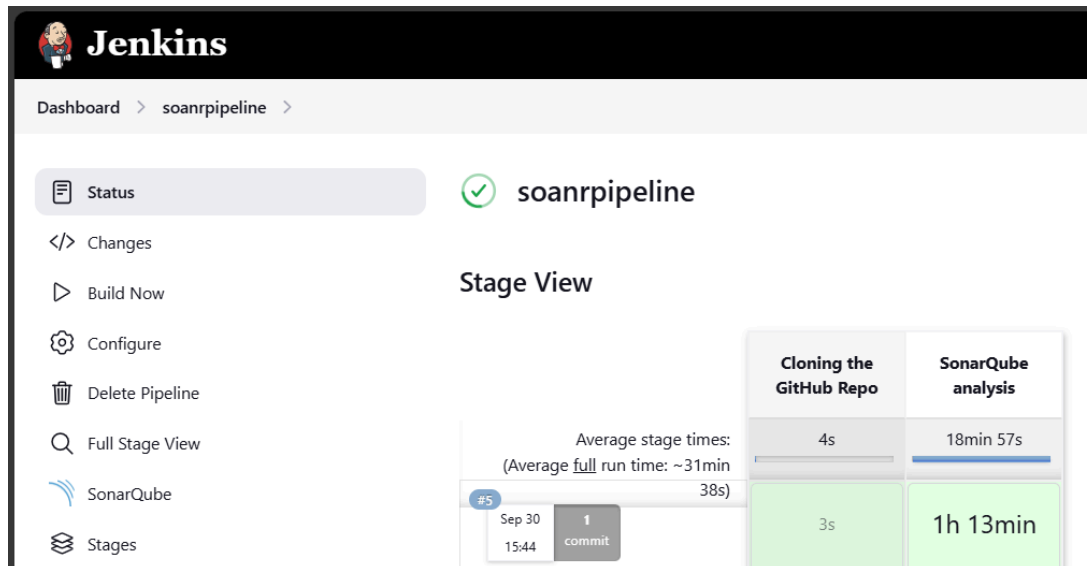
<properties>
  <build.number>SNAPSHOT</build.number>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <easyb.version>1.4</easyb.version>
  <cobertura.version>2.6</cobertura.version>
  <!-- A workaround for a bug in PMD -->
  <sourceJdk>1.8</sourceJdk>
  <targetJdk>1.8</targetJdk>
  <github.account>wakaleo</github.account>
  <thucydides.version>1.8.4</thucydides.version>
  <jelastic.context>gameoflife</jelastic.context>
  <jelastic.environment>wakaleo</jelastic.environment>

  <!-- Hardcoded username and password (Security issue) -->
  <jelastic.username>admin</jelastic.username>
  <jelastic.password>password123</jelastic.password>

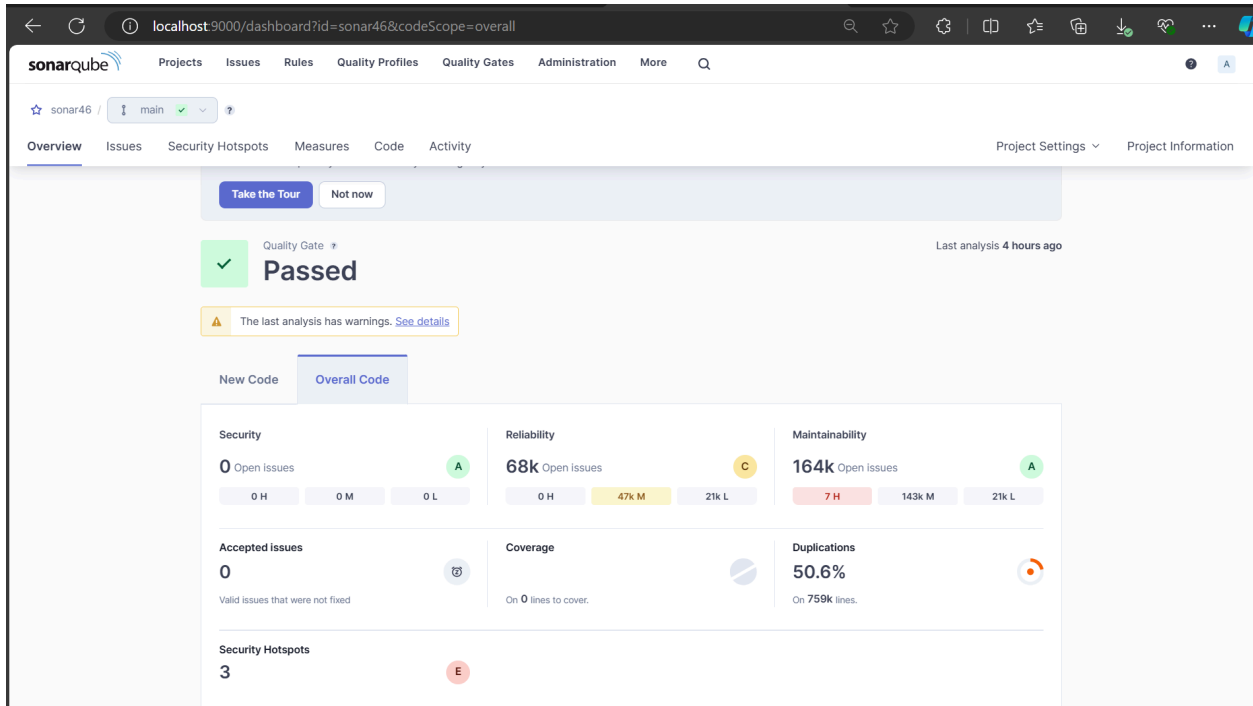
  <!-- Unused property (Maintainability issue) -->
  <unused.property>ThisIsUnused</unused.property>
</properties>

```

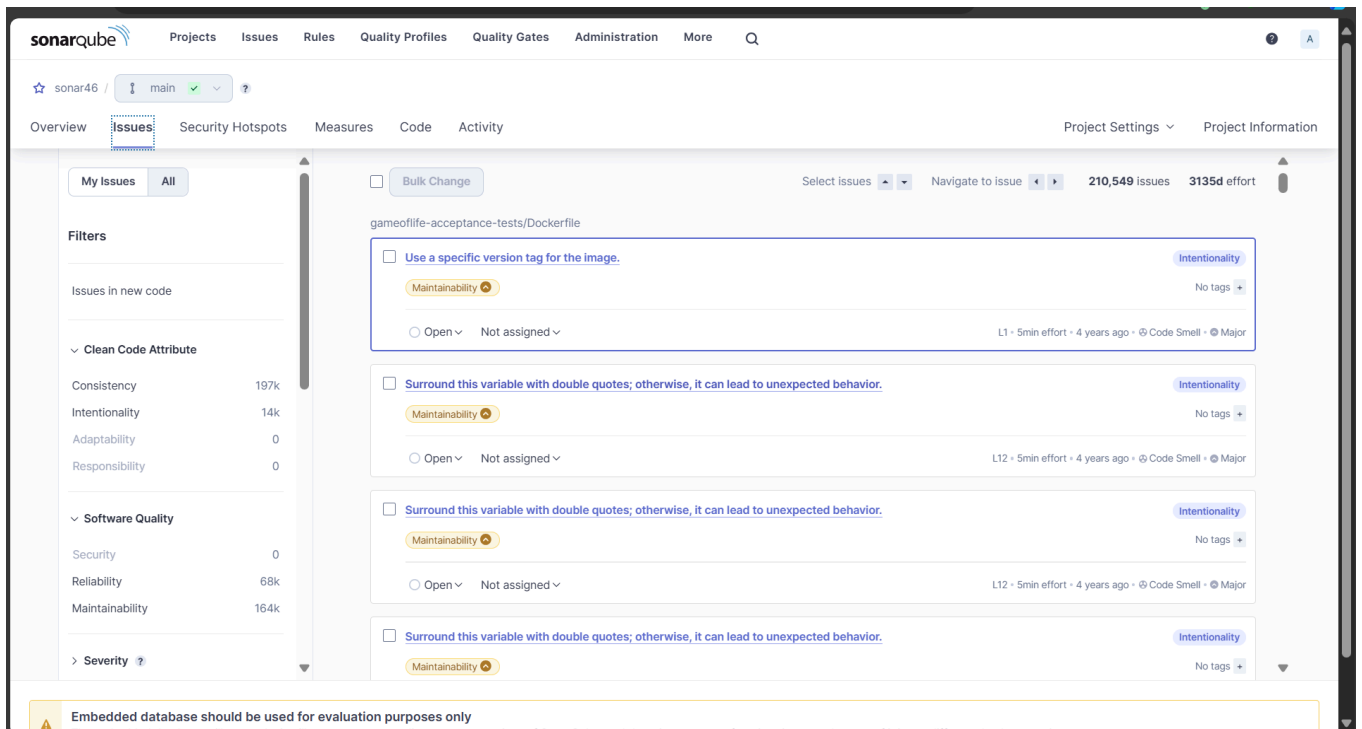
3. Then again I build the code on Jenkins.



4. Then I again go to the sonarqube to see the analysis it given me output passed



But in issues section there are many issues after making changes.



Thus sonarqube analysis is done.

**Conclusion:** Here we created the “sonarpipeline” project locally successfully. Then created a pipeline on Jenkins and installed a plugin named SonarQube Scanner. Then we build that project by adding the script inside pipeline which consists of project name, key, token and then GitHub repository to clone. Then after saving and applying the changes, the build executed successfully. Initially there was an issue in connection to solve this I created the token again and it executed successfully. Then we also checked that SonarQube gives response as passed. Thus, Build is done successfully. Then again I made some changes in pom.xml and again built the code which was also done successfully and then I opened SonarQube which gave me detailed analysis of issues arising due to new changes in code.