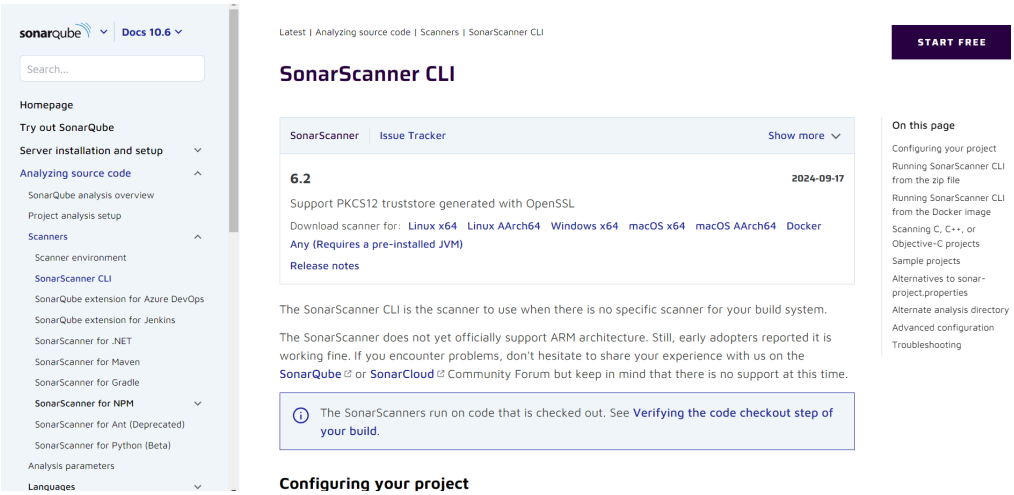
Name:Pratik Patil **Div:D15C** **Roll No:40**

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.

Prerequisites:

1. **Download Sonar Scanner:**
   Access the SonarQube documentation and download the SonarQube scanner CLI from this link:

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



2. After downloading, extract the zip file into a designated folder.

Install Docker:
Run the following command to verify Docker is installed:

```
C:\Users\91799>docker -v
Docker version 27.0.3, build 7d4bcd8

C:\Users\91799>
```

Name:Pratik Patil **Div:D15C** **Roll No:40**

3 .Pull SonarQube Docker Image:
Install the SonarQube image by executing:

Copy code
```
docker pull sonarqube
```

```
PS C:\Users\91799> docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Image is up to date for sonarqube:latest
docker.io/library/sonarqube:latest
```

4. Ensure Jenkins is installed:
Confirm that Jenkins is installed and configured on your system.

Experiment Steps:

Step 1:
Run the SonarQube Docker container by entering the command below:

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

```
PS C:\Users\91799> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
17782929ab2dbb01ea35d5bc93ed52fcffdce44ee8595a07b73a9269e0d39106
```

Step 2:

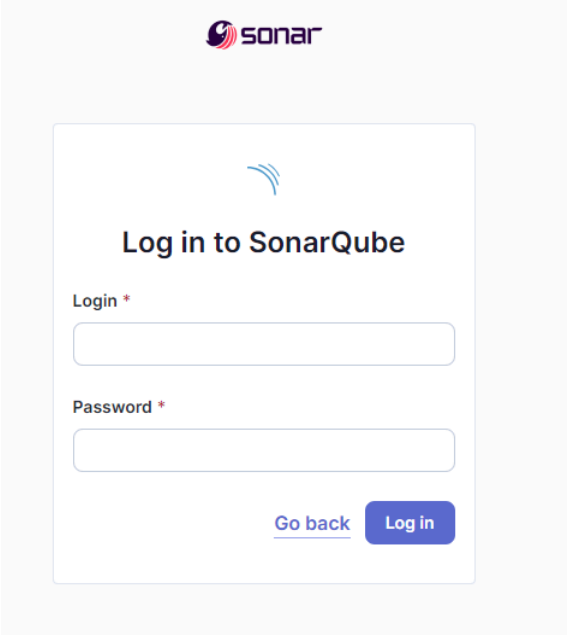
After SonarQube is running, open your browser and go to `http://localhost:9000`.

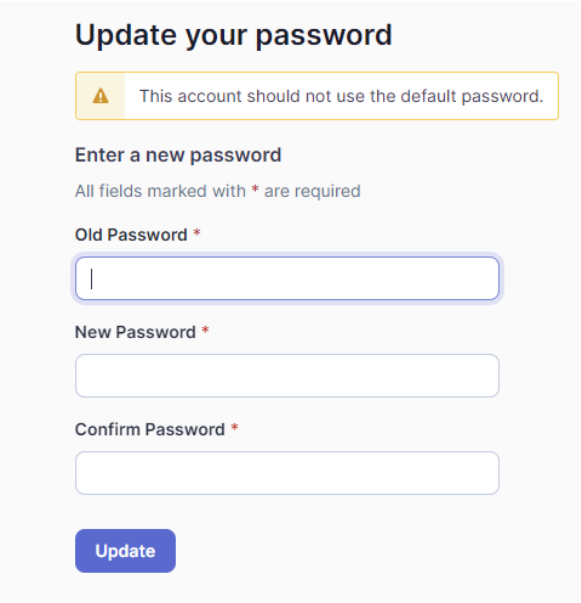
Step 3:
Log in to SonarQube using the default credentials:

`Username: admin  Password: admin`

Name:Pratik Patil **Div:D15C** **Roll No:40**

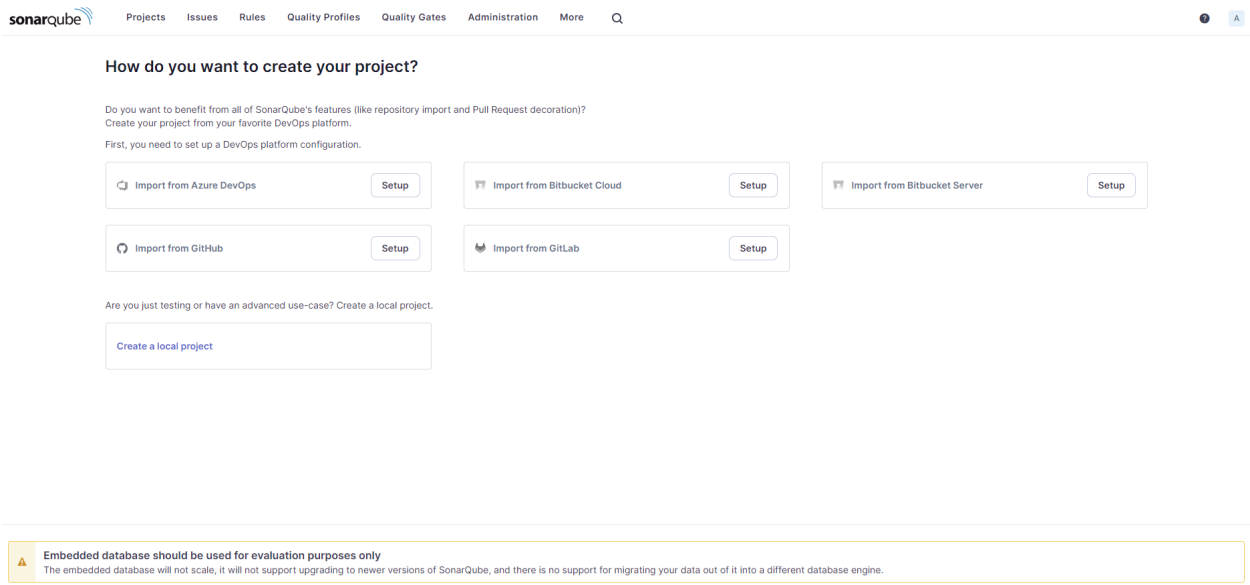
You will be asked to reset the password after logging in for the first time. Set a new password and remember it.

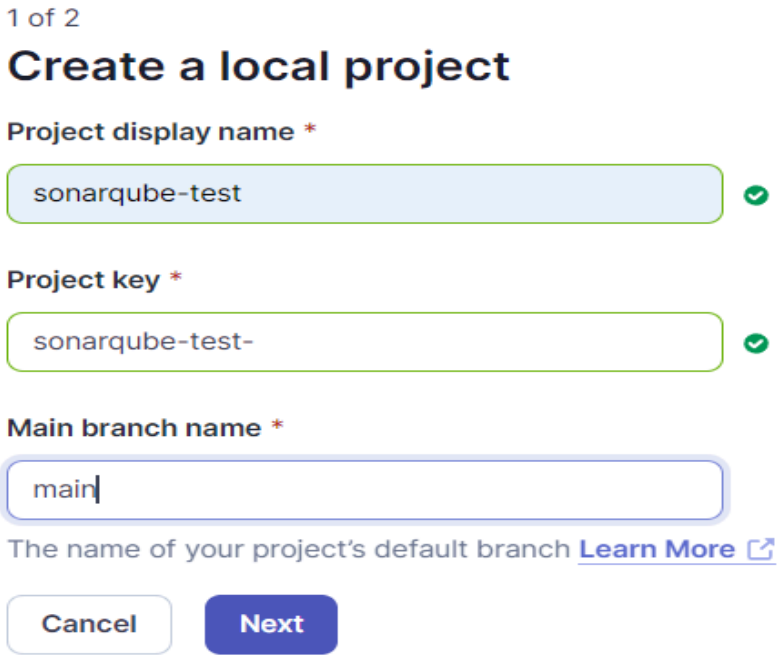




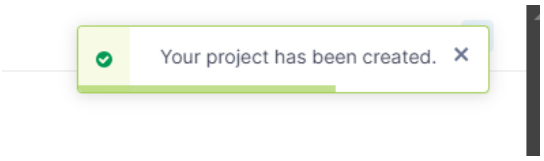
Name:Pratik Patil **Div:D15C** **Roll No:40**

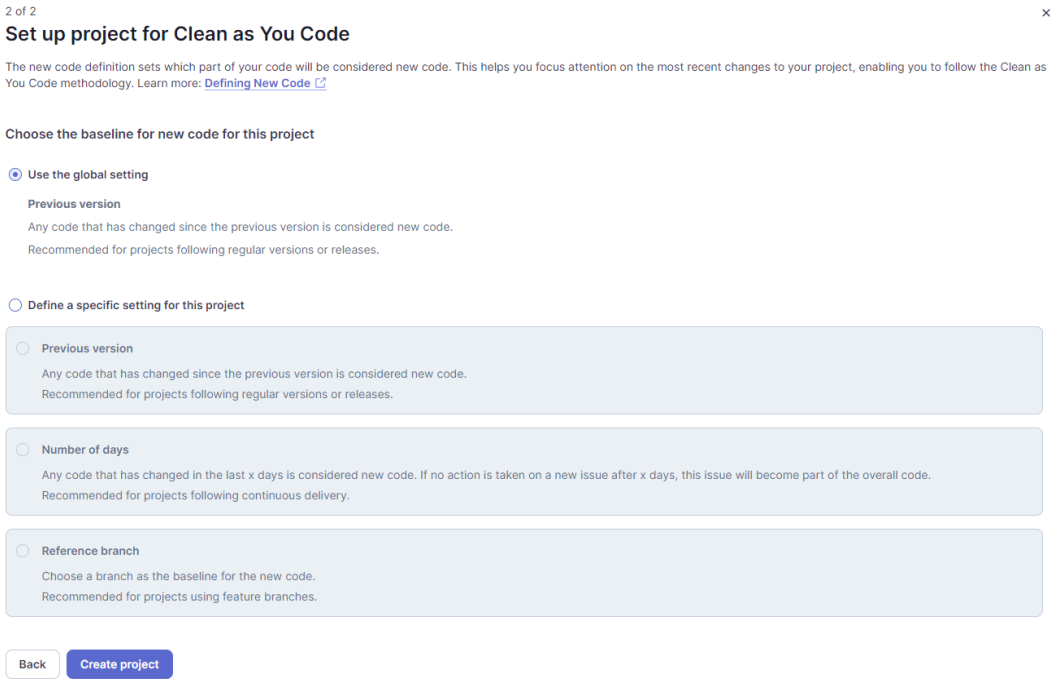
Step 4:
On the SonarQube dashboard, click **Create a Local Project**. Provide a project name and a unique project key.





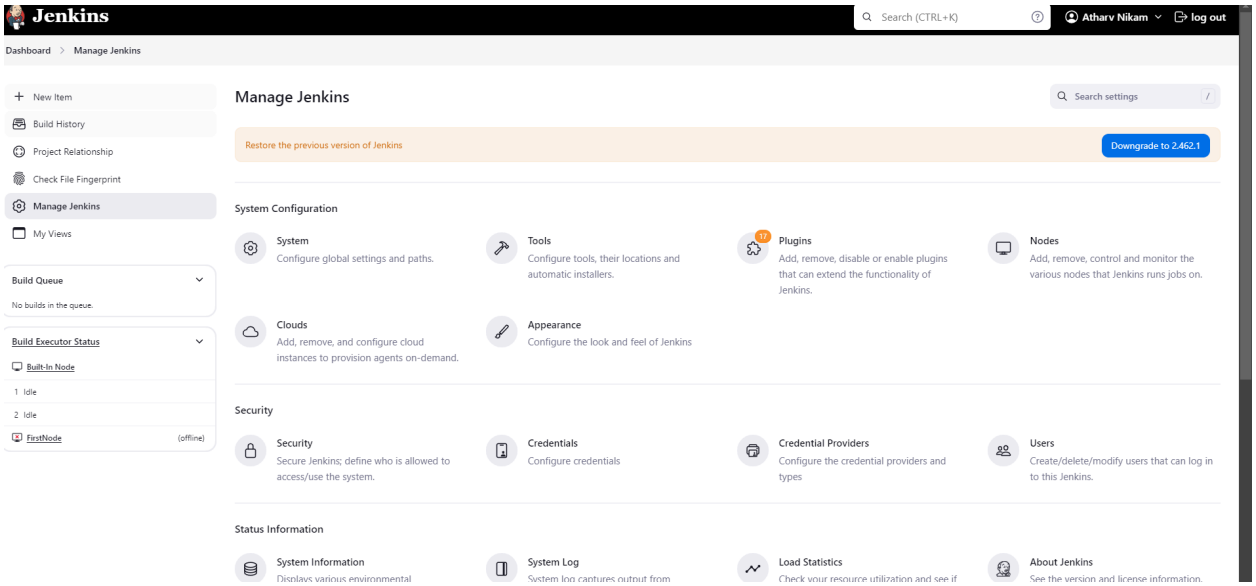
Name:Pratik Patil **Div:D15C** **Roll No:40**





Name:Pratik Patil **Div:D15C** **Roll No:40**

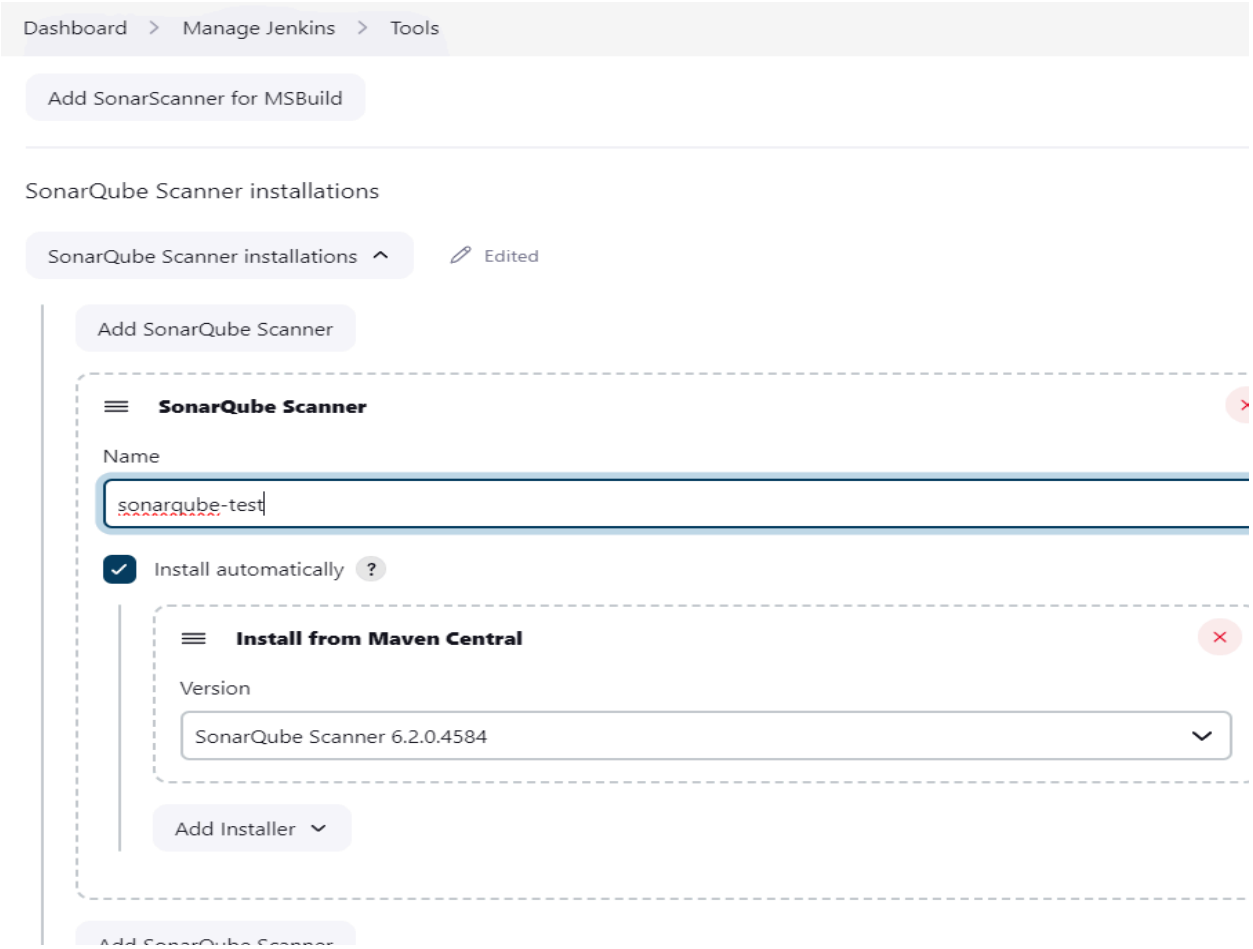
Step 6:
In Jenkins, go to **Manage Jenkins** → **Plugins** and search for **SonarQube Scanner for Jenkins**. Install the plugin.



Step 7:
Once installed, head to **Manage Jenkins** → **System**. Under **SonarQube Servers**, add your SonarQube server, and provide any necessary authentication tokens.

Name:Pratik Patil **Div:D15C** **Roll No:40**

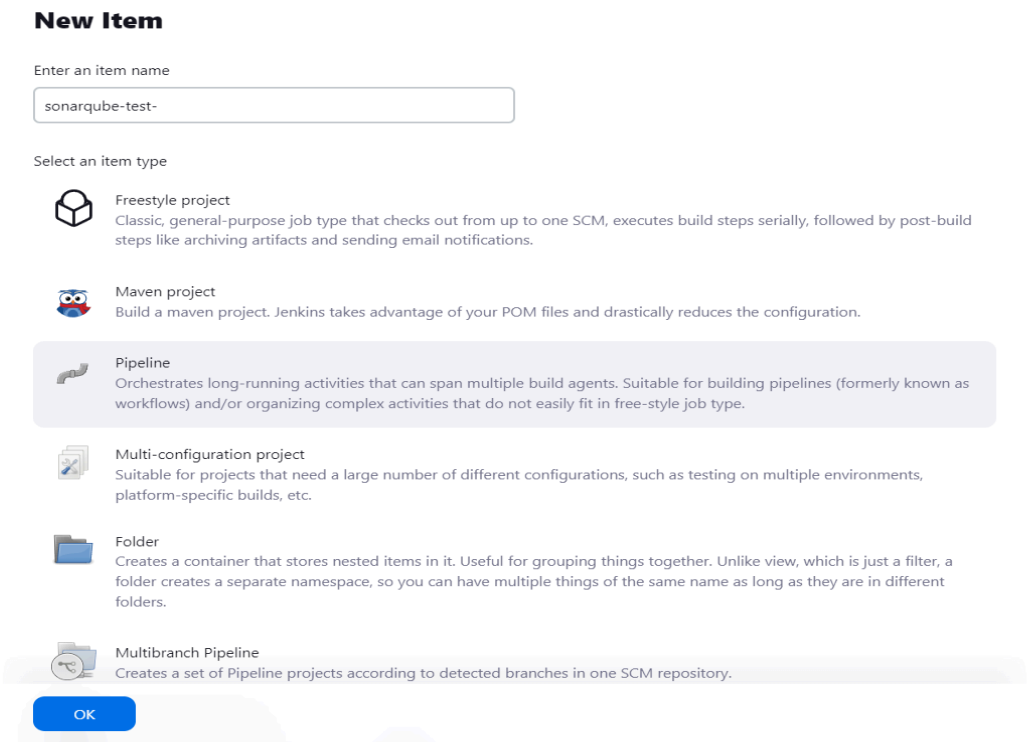
Step 8:
Next, under **Manage Jenkins** → **Tools**, navigate to **SonarQube Scanner** and configure it to automatically install the latest version.



Name:Pratik Patil **Div:D15C** **Roll No:40**

Step 9:

Create a new pipeline item in Jenkins



Step 10:

In the pipeline script section, input the following:

```
node {
  stage('Cloning the GitHub Repo') {
     git 'https://github.com/shazforiot/GOL.git'
  }

  stage('SonarQube Analysis') {
     withSonarQubeEnv(‘sonarqube-test’) {
        bat """
        <PATH_TO_SONARSCANNER_FOLDER>\\bin\\sonar-scanner.bat ^
        -D sonar.login=<SONARQUBE_LOGIN> ^
        -D sonar.password=<SONARQUBE_PASSWORD> ^
        -D sonar.projectKey=<PROJECT_KEY> ^
        -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
        -D sonar.host.url=http://localhost:9000/
        """
     }
  }
}
```

Name:Pratik Patil **Div:D15C** **Roll No:40**

Pipeline

Definition

Pipeline script

Script

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube Analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """
            C:\\Users\\91799\\Downloads\\sonar-scanner-cli-6.2.0.4584-windows-x64\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat
            -D sonar.login=admin
            -D sonar.password=pratik
            -D sonar.projectKey=sonarqube-test
            -D sonar.exclusions=vendor/**,resources/**,**/*.java
            -D sonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

Use Groovy Sandbox

Pipeline Syntax

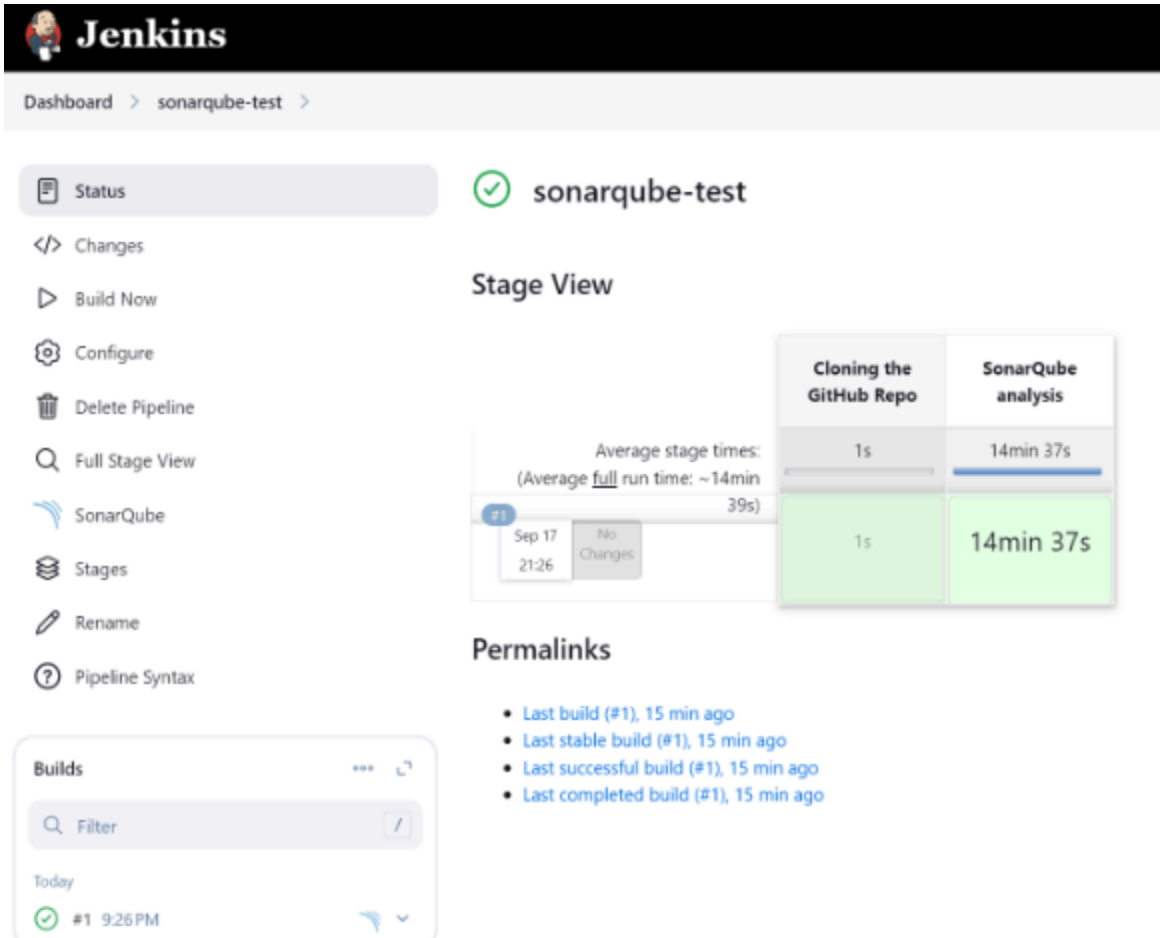
Save Apply

This script clones a sample Java project from GitHub, which has several issues that SonarQube will detect.

Step 11:

Go back to Jenkins, select the job you just created, and click **Build Now** to run the pipeline.

Name:Pratik Patil **Div:D15C** **Roll No:40**

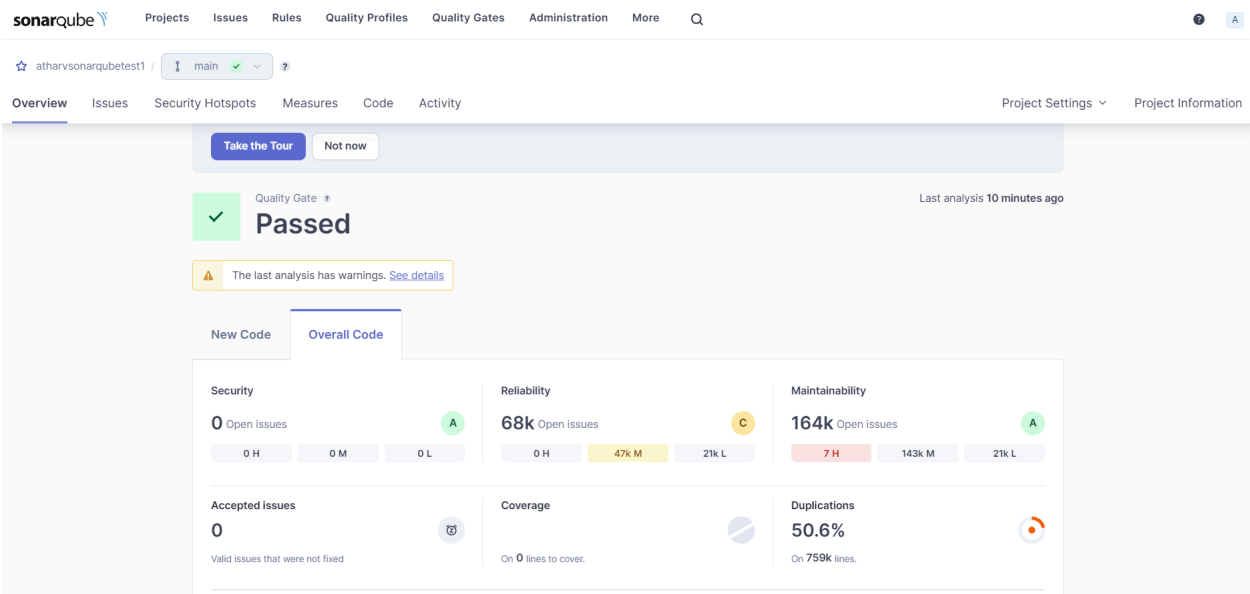



```
Started by user Pratik Manish Patil
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.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/* # timeou
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
```

Name:Pratik Patil **Div:D15C** **Roll No:40**

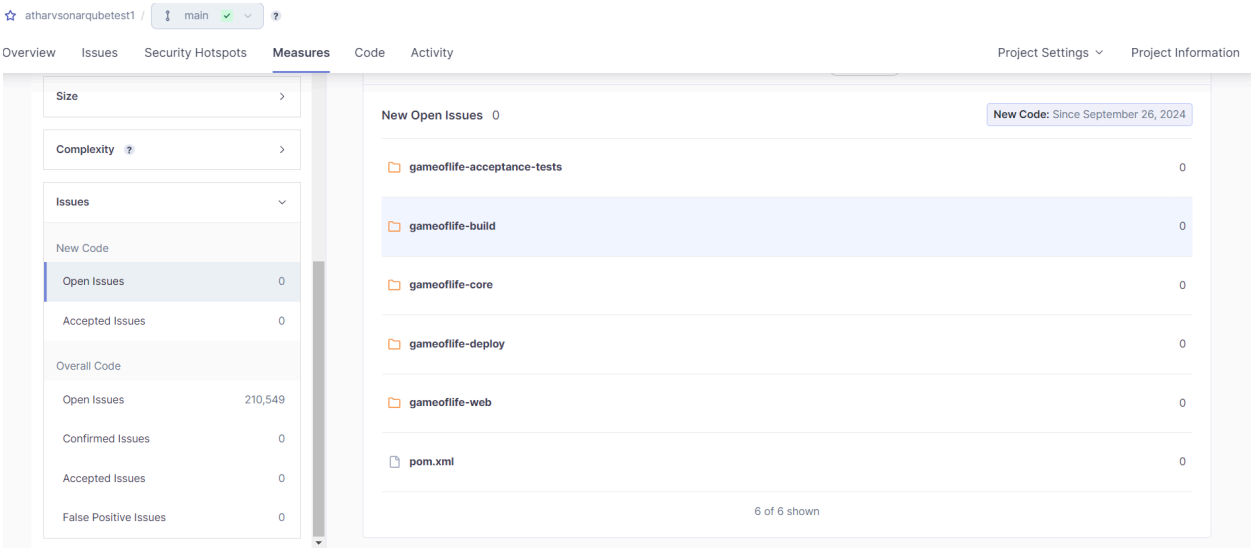
Step 12:

Once the build is complete, return to SonarQube to view the analysis of your project. Check for bugs, code smells, duplications, and other metrics related to the quality of your code.



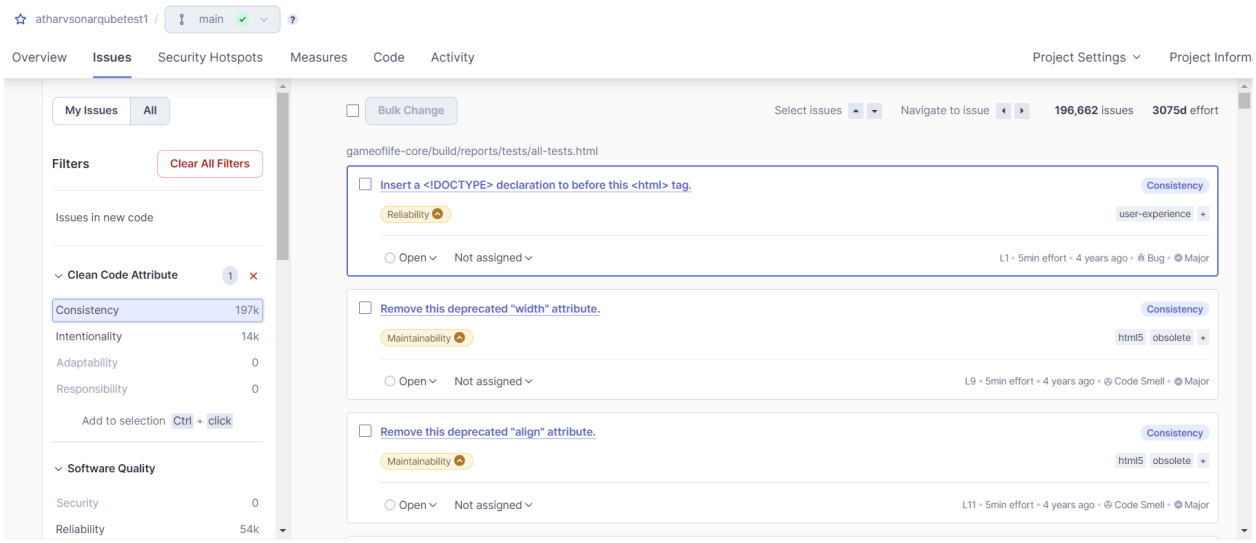
Under different tabs, check all the issues with the code.

- Code Problems

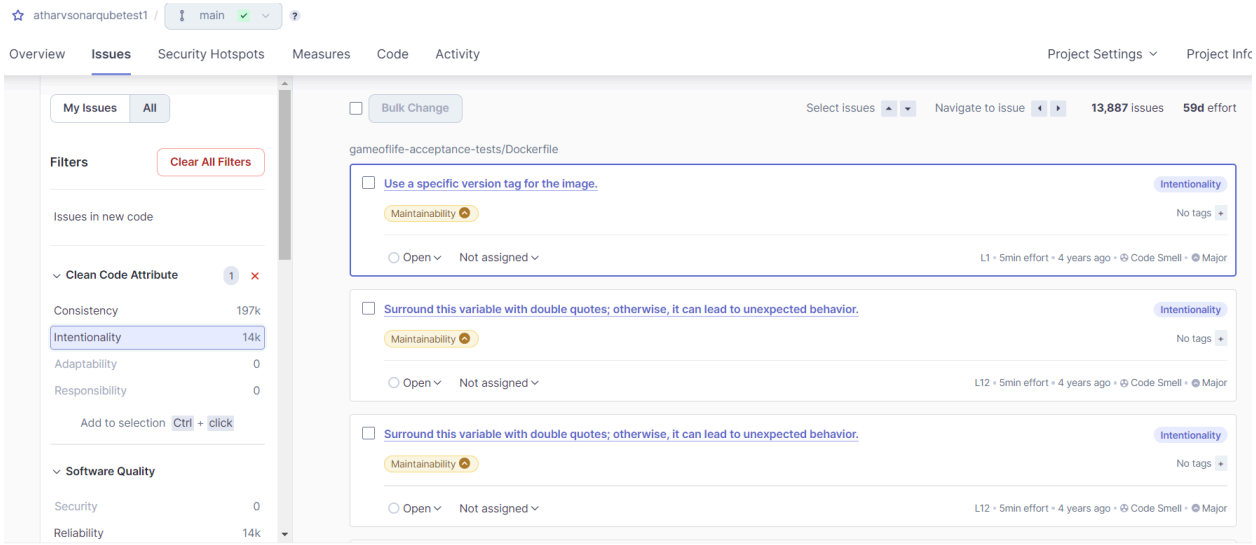


Name:Pratik Patil **Div:D15C** **Roll No:40**

- Consistency

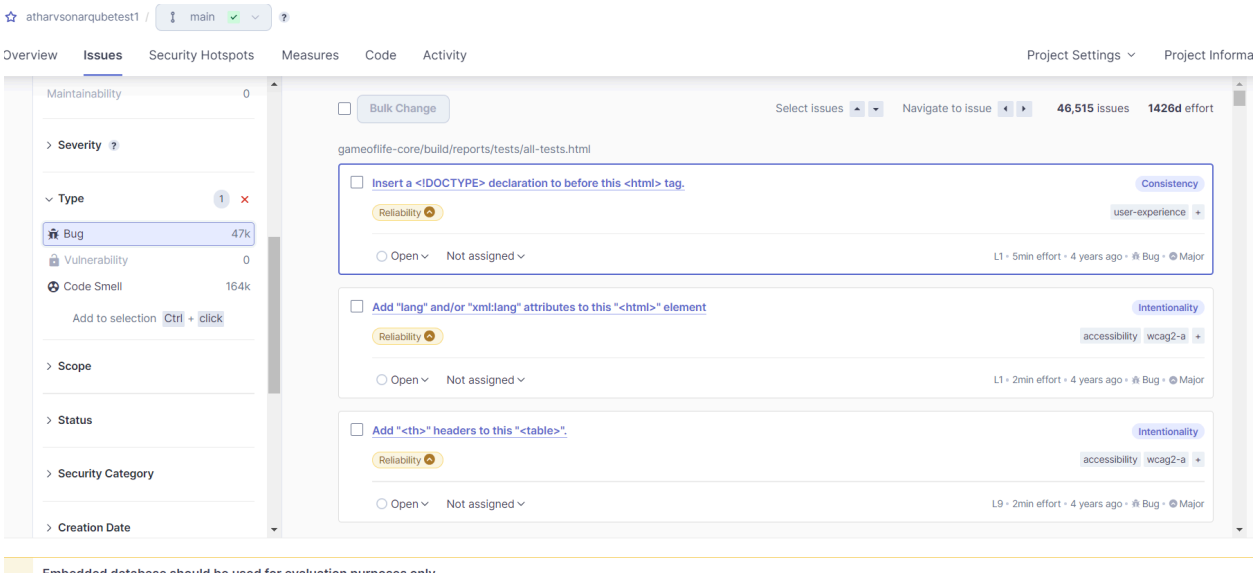


- Intentionality

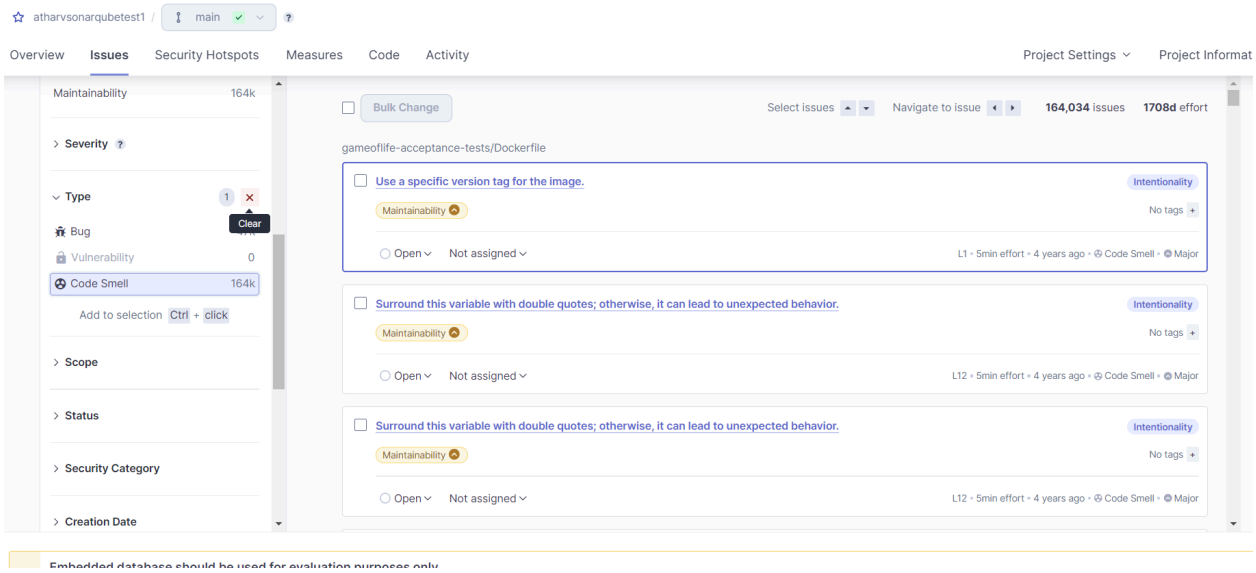


Name:Pratik Patil **Div:D15C** **Roll No:40**

- Bugs

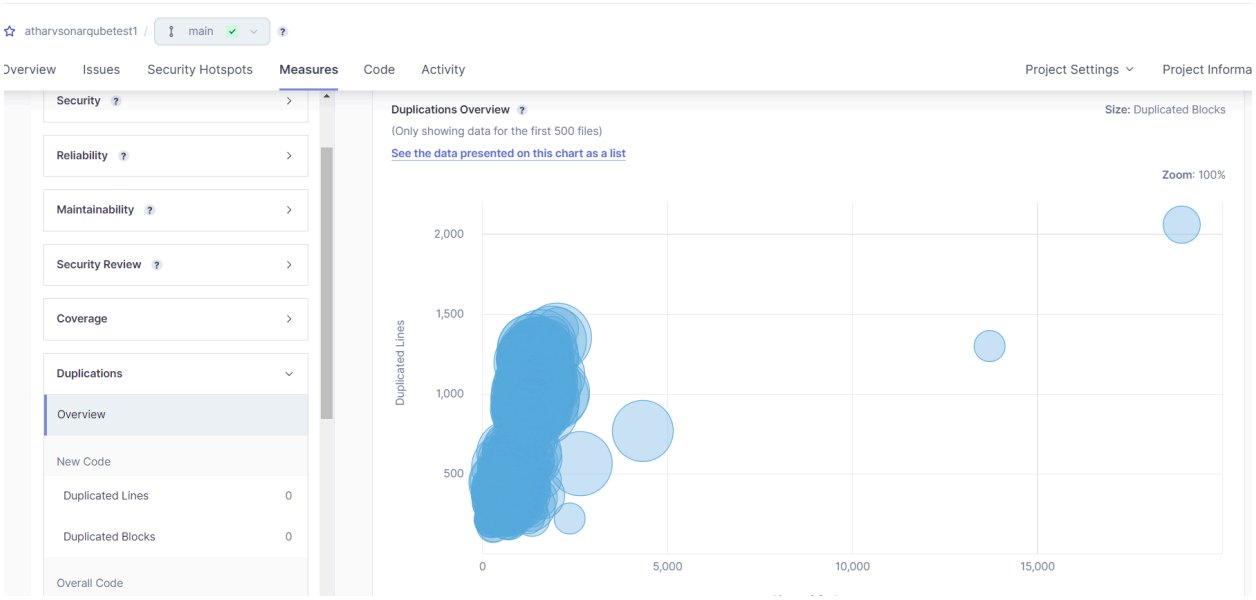


- Code Smells

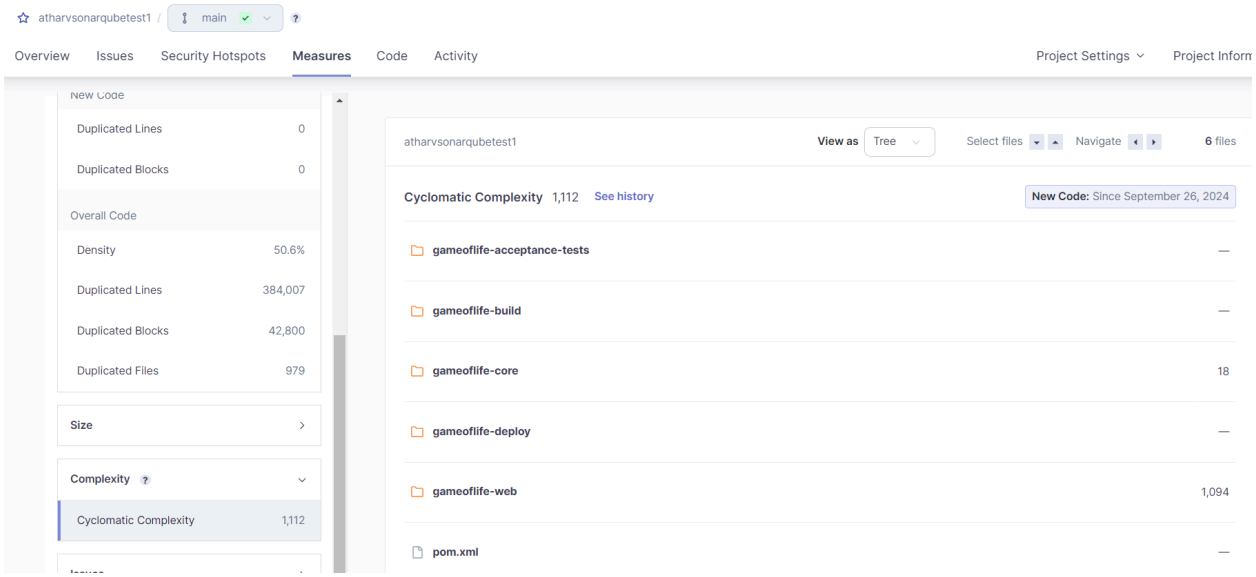


Name:Pratik Patil **Div:D15C** **Roll No:40**

- Duplications



- Cyclomatic Complexities



Name:Pratik Patil **Div:D15C** **Roll No:40**

Conclusion:

This experiment allowed us to integrate Jenkins and SonarQube to set up a CI/CD pipeline capable of performing static analysis on Java code. Through this process, we automated the detection of common code issues such as bugs, code smells, and duplications. By leveraging Docker for SonarQube and the Jenkins pipeline, we streamlined the code scanning process, ensuring any issues were highlighted during the build phase. This integration demonstrates the importance of automated code quality checks in a continuous delivery environment