Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Theory :

1. Static Application Security Testing (SAST)

- **Definition**: SAST is a method of debugging by examining source code before a program is run. It identifies vulnerabilities early in the development lifecycle.
- **Key Features**:
  - **Early Detection**: Finds vulnerabilities during the coding phase, reducing remediation costs.
  - **Code Quality Analysis**: Beyond security, it also assesses code quality, maintainability, and adherence to coding standards.
  - **Integration**: Can be integrated into CI/CD pipelines for continuous security assessment.

2. SonarQube

- **Definition:** SonarQube is an open-source platform for continuous inspection of code quality, which includes detecting bugs, vulnerabilities, and code smells.
- **SAST Integration:** Supports SAST tools to analyze code and provide metrics and reports within the SonarQube dashboard.
- **Quality Gates:** Allows setting thresholds (quality gates) that must be met before code can proceed in the CI/CD pipeline.

3. Reporting and Remediation

- **Results Analysis**: After SAST scans, results are usually presented in a detailed report highlighting vulnerabilities, their severity, and remediation advice.
- **Feedback Loop**: Integrating SAST results into the development workflow helps create a feedback loop, encouraging developers to address vulnerabilities proactively.

4. Best Practices

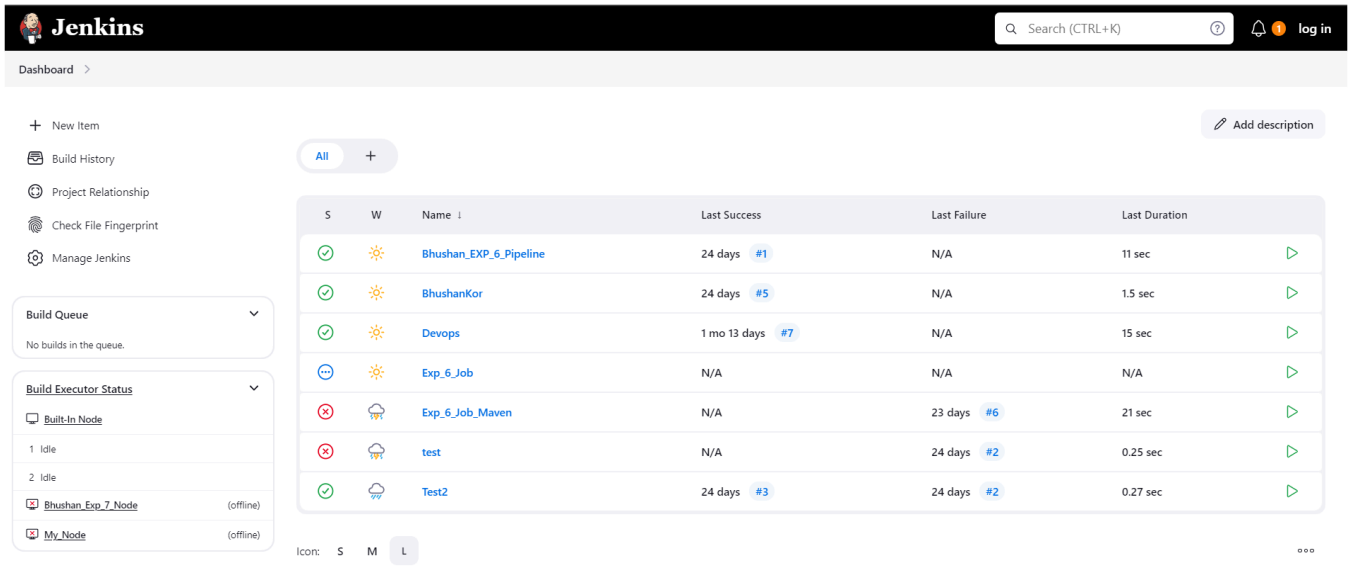
- **Regular Updates**: Keep SAST tools and configurations updated to recognize the latest vulnerabilities.
- **Customization**: Tailor SAST rules and configurations to suit the specific needs of the project and team.
- **Training**: Ensure developers are trained on security best practices to understand and mitigate vulnerabilities effectively.

Prerequisites:
- Jenkins installed
- Docker Installed (for SonarQube)

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.



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

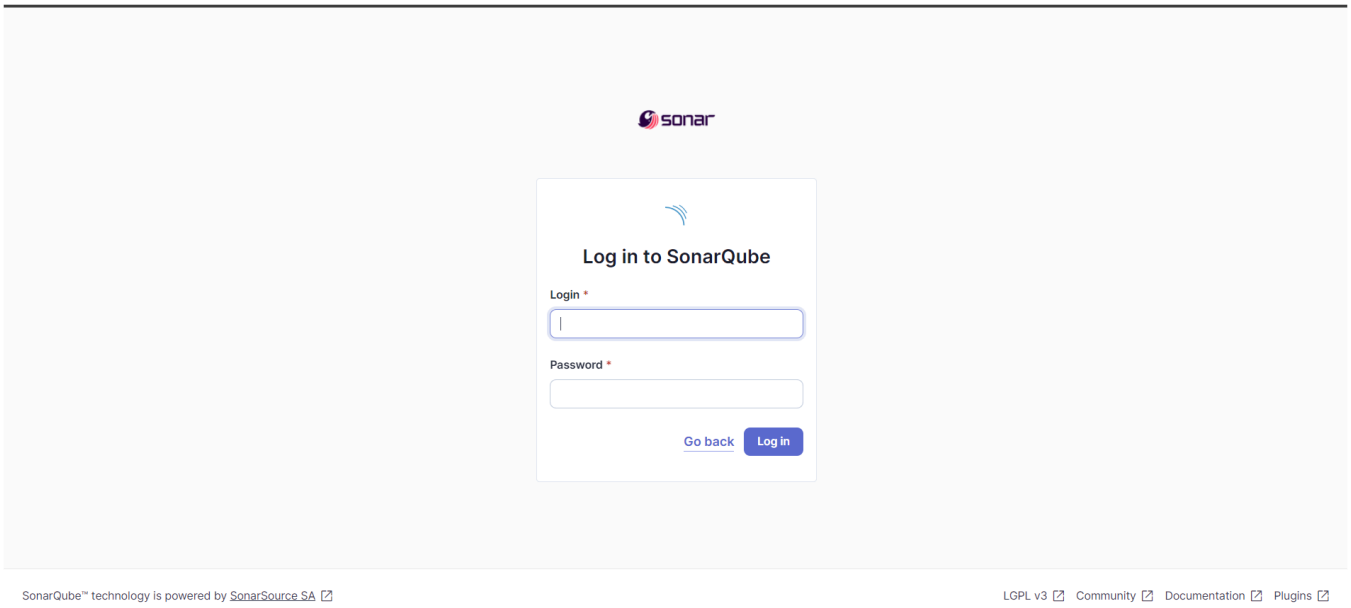
```
C:\Users\bhush\one drive 2\OneDrive\Desktop\Docker>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=t
rue -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
762bed f4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
dc00d2f31a229b2076f15c273ad750e45b74f871dd53b6d177b6f860d4fc8f0e

C:\Users\bhush\one drive 2\OneDrive\Desktop\Docker>
```

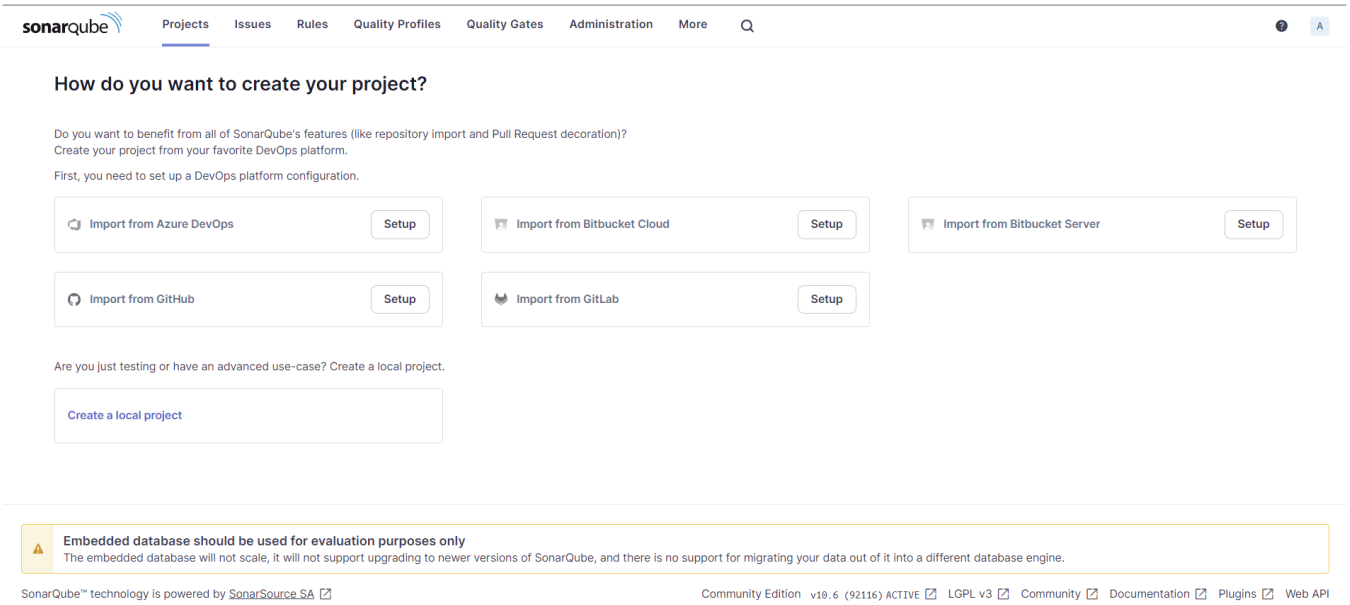
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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



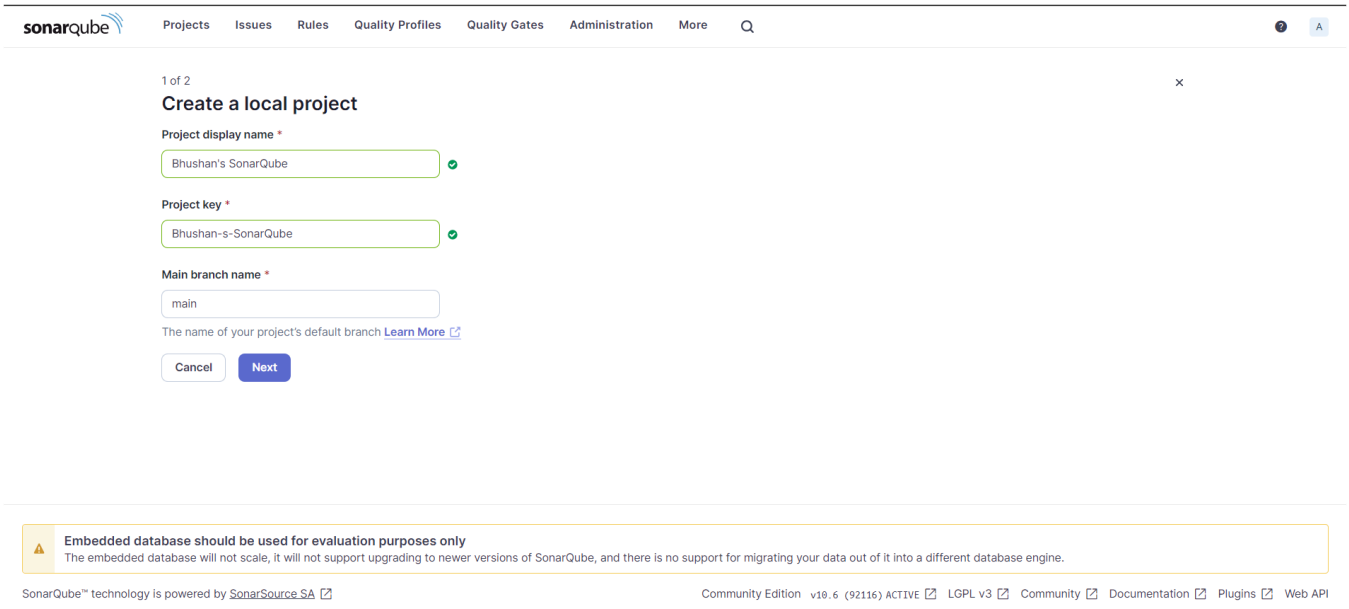
Start 4: Login to SonarQube using username admin and password admin.

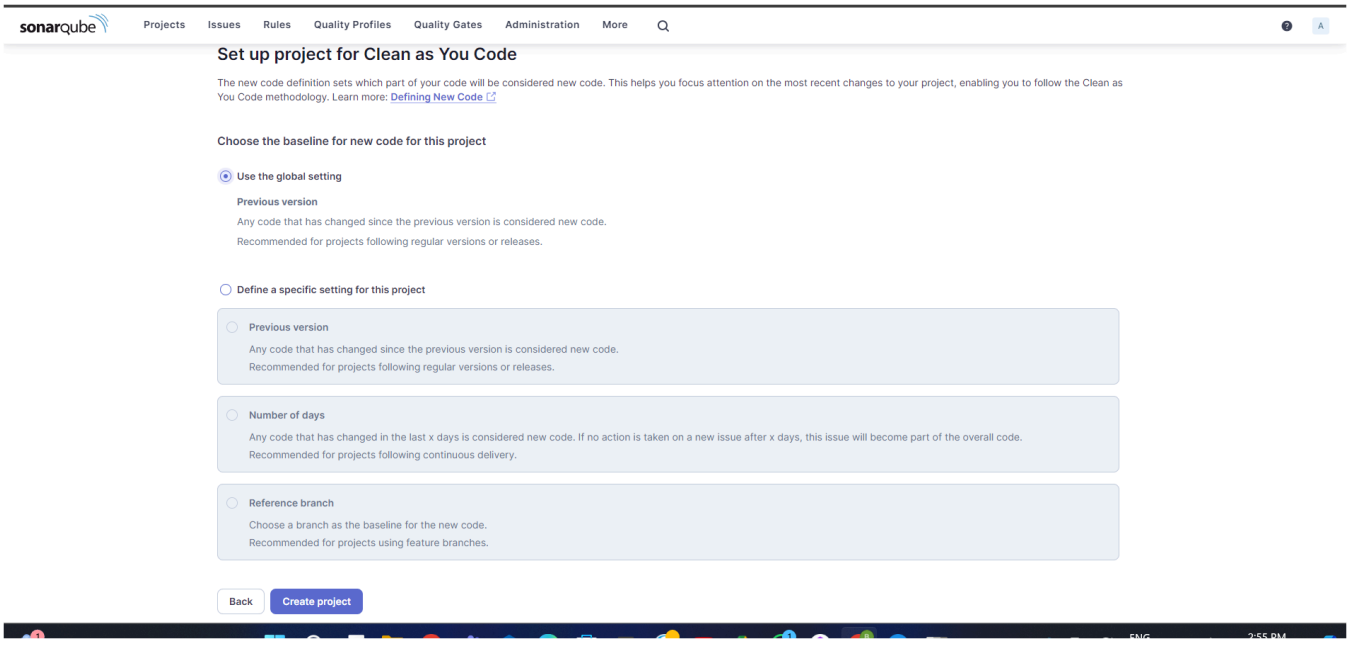


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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



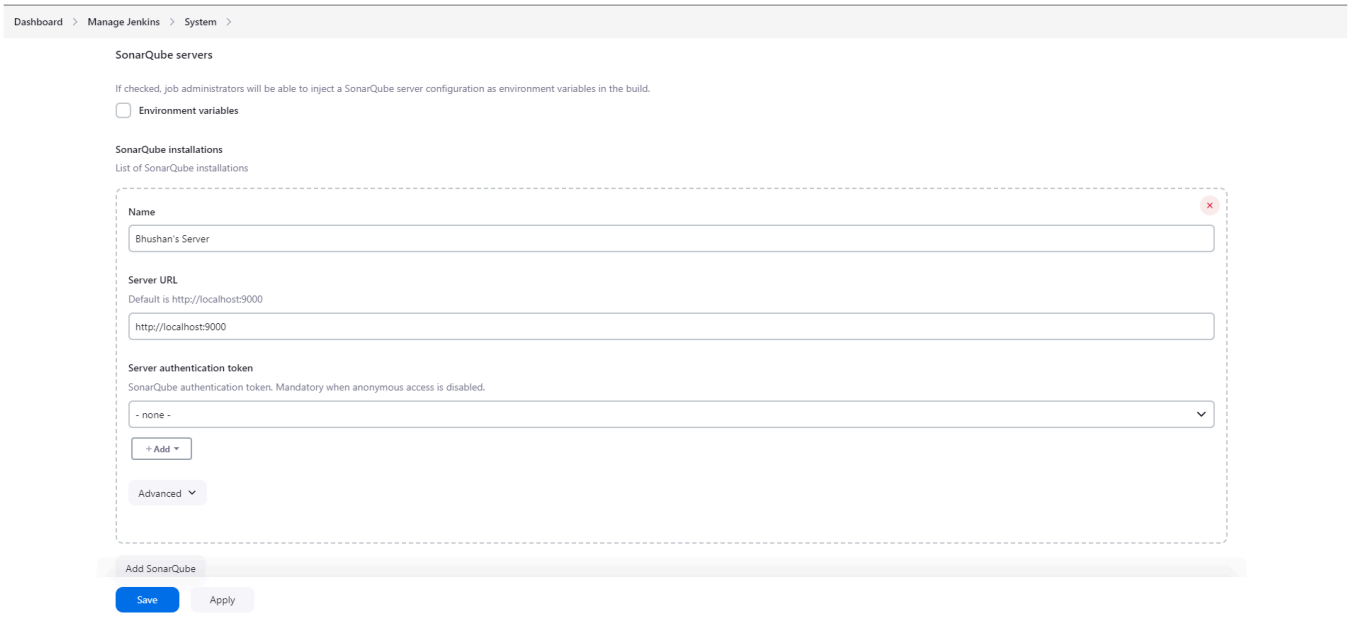


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

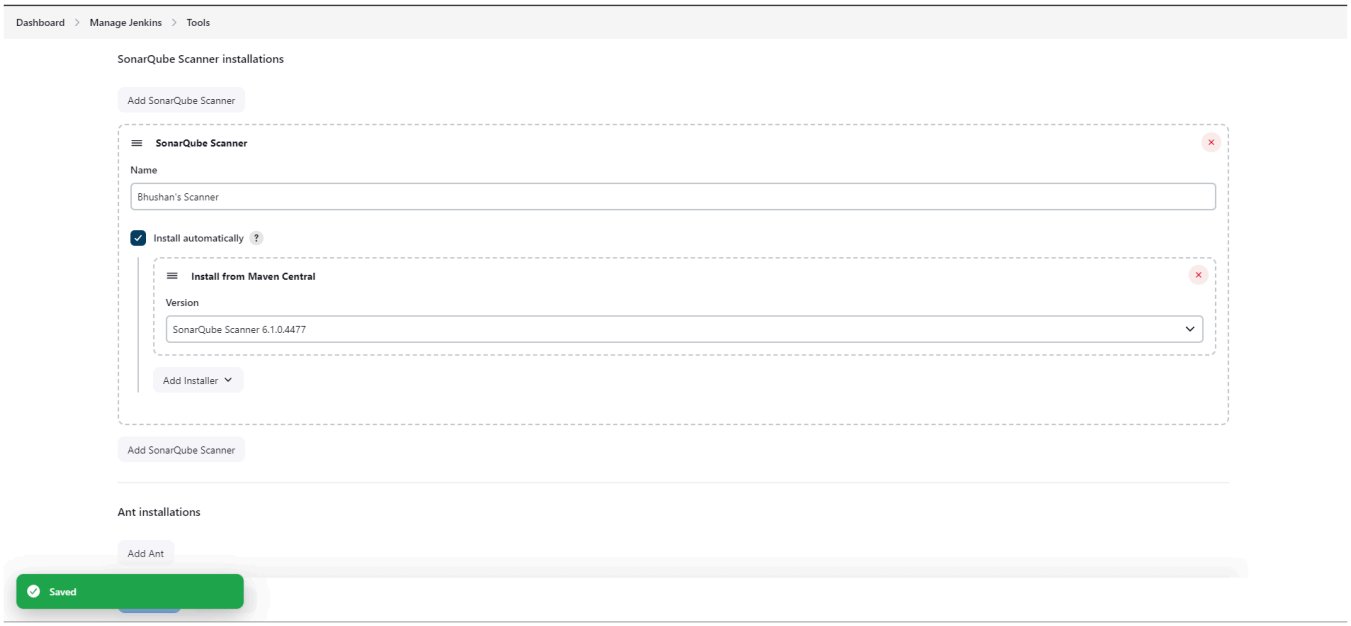
Division: D15C **Roll No: 28**

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.

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



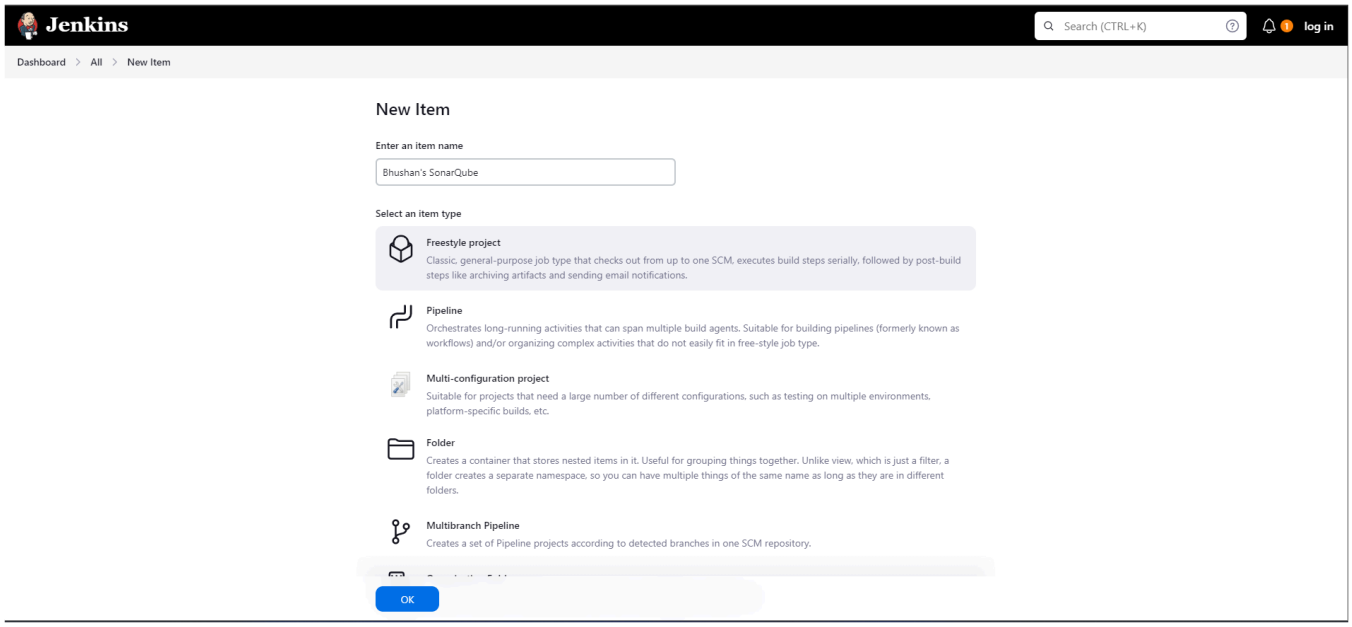
Step 8: Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



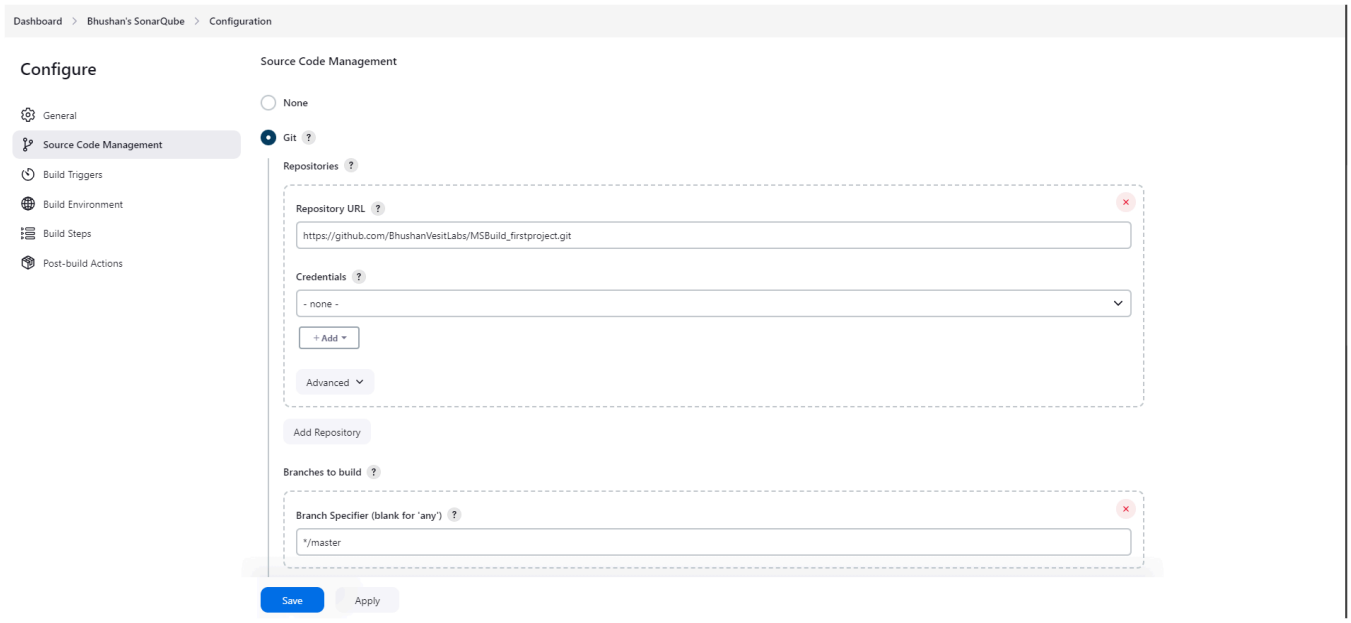
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 9: After the configuration, create a New Item in Jenkins, and choose a freestyle project.



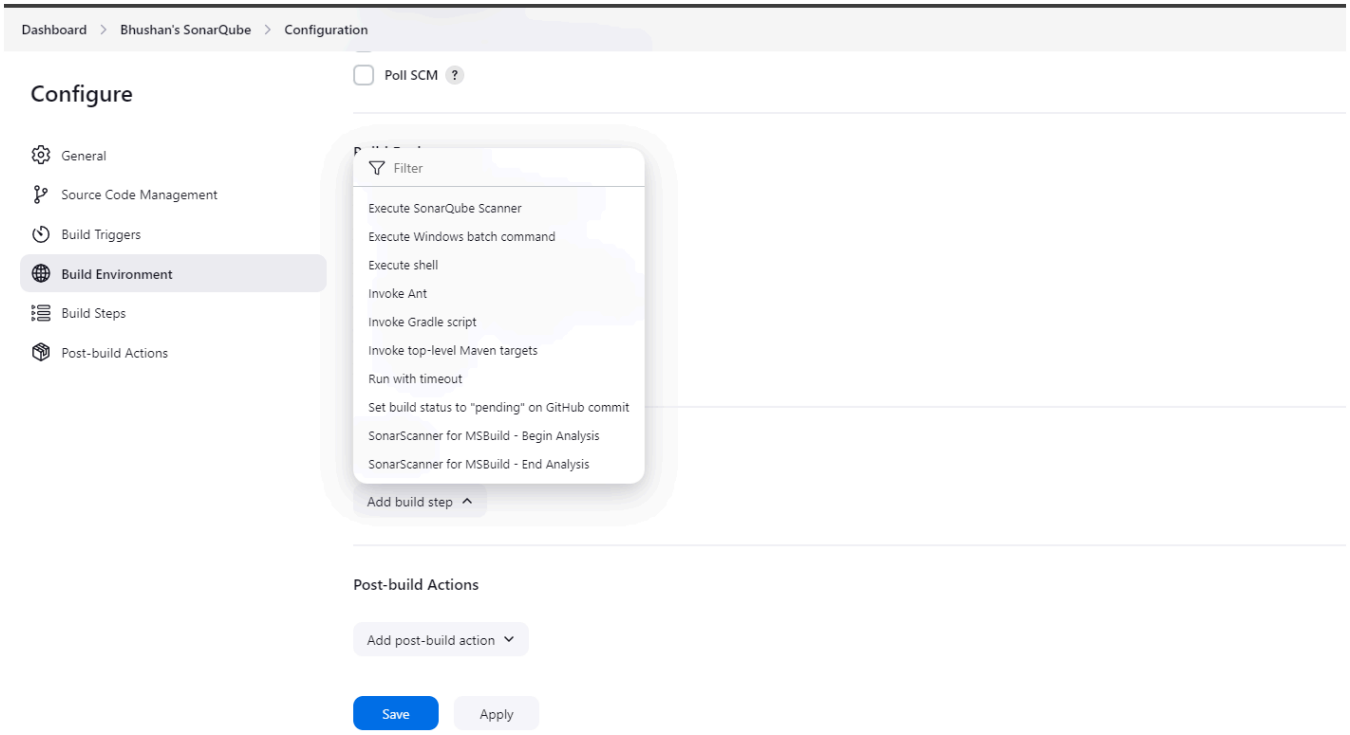
Step 10: Choose this GitHub repository in Source Code Management. https://github.com/shazforiot/MSBuild_firstproject.git .It is a sample hello-world project with no vulnerabilities and issues, just to test integration.

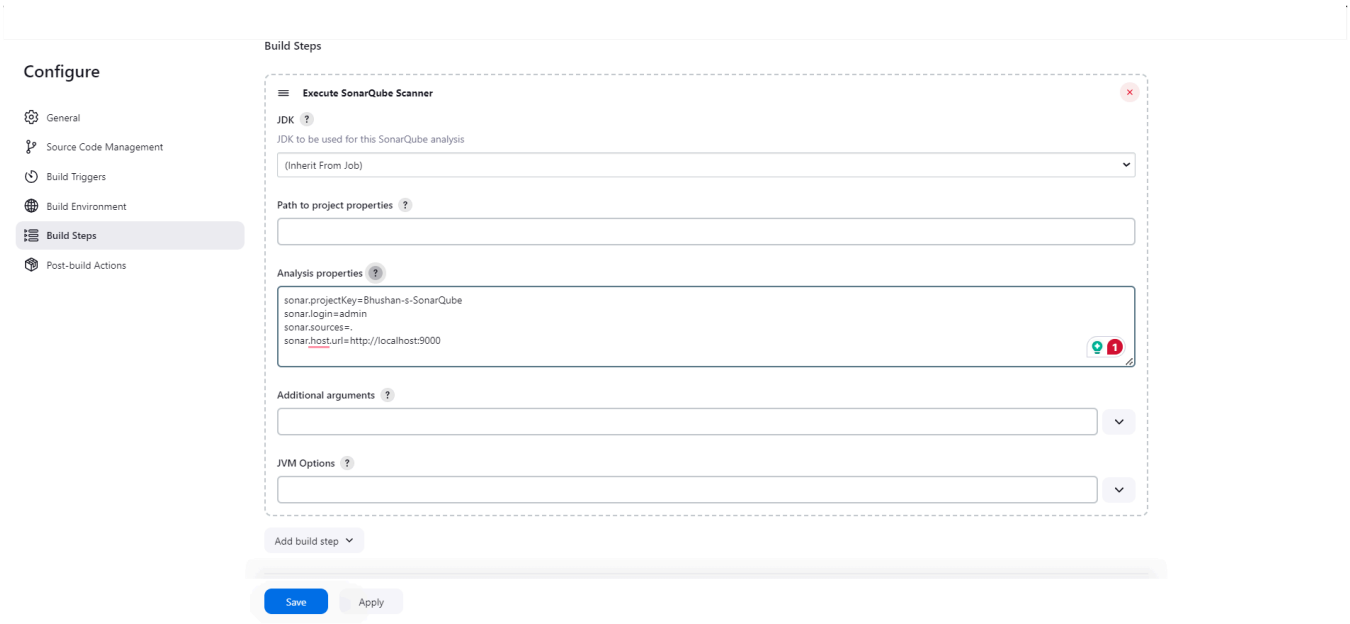


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 11: Under Build select Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.



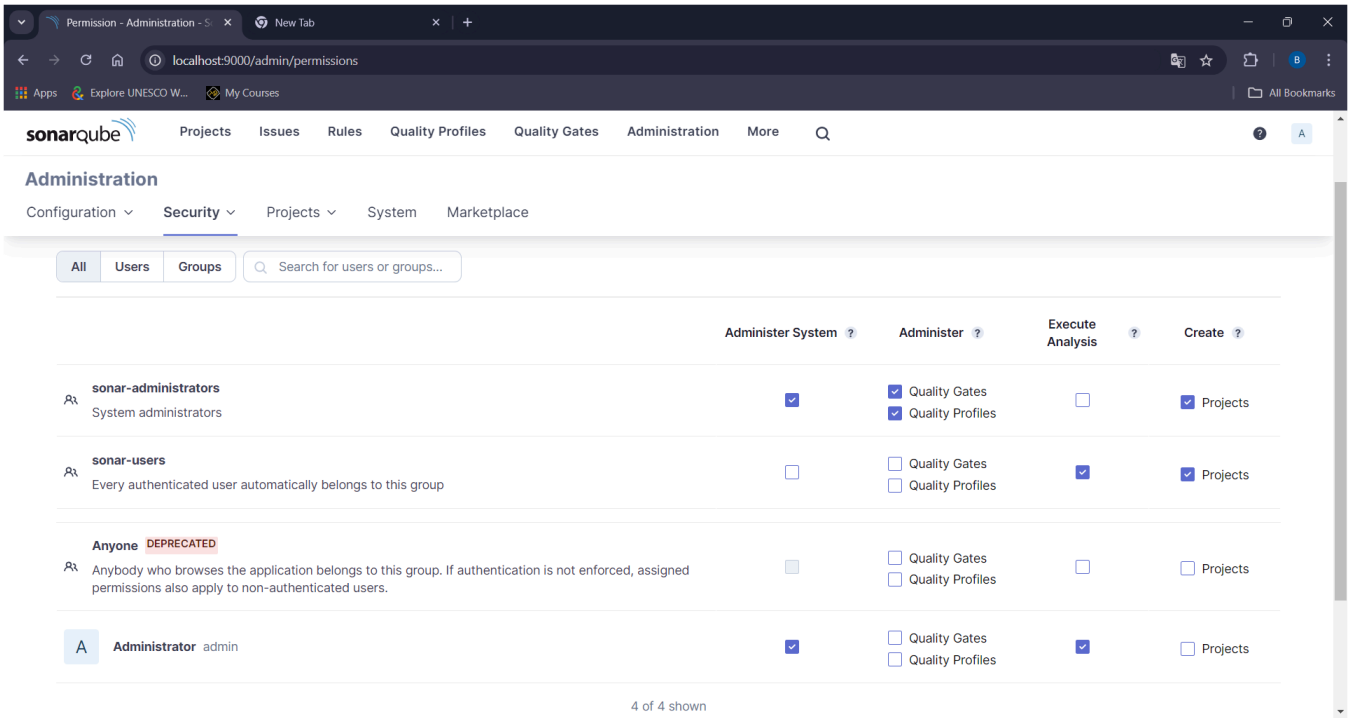


sonar.projectKey=<Your Project Key>
sonar.login=<User Name>
sonar.password=<Password>
sonar.sources=.
sonar.host.url=http://localhost:9000

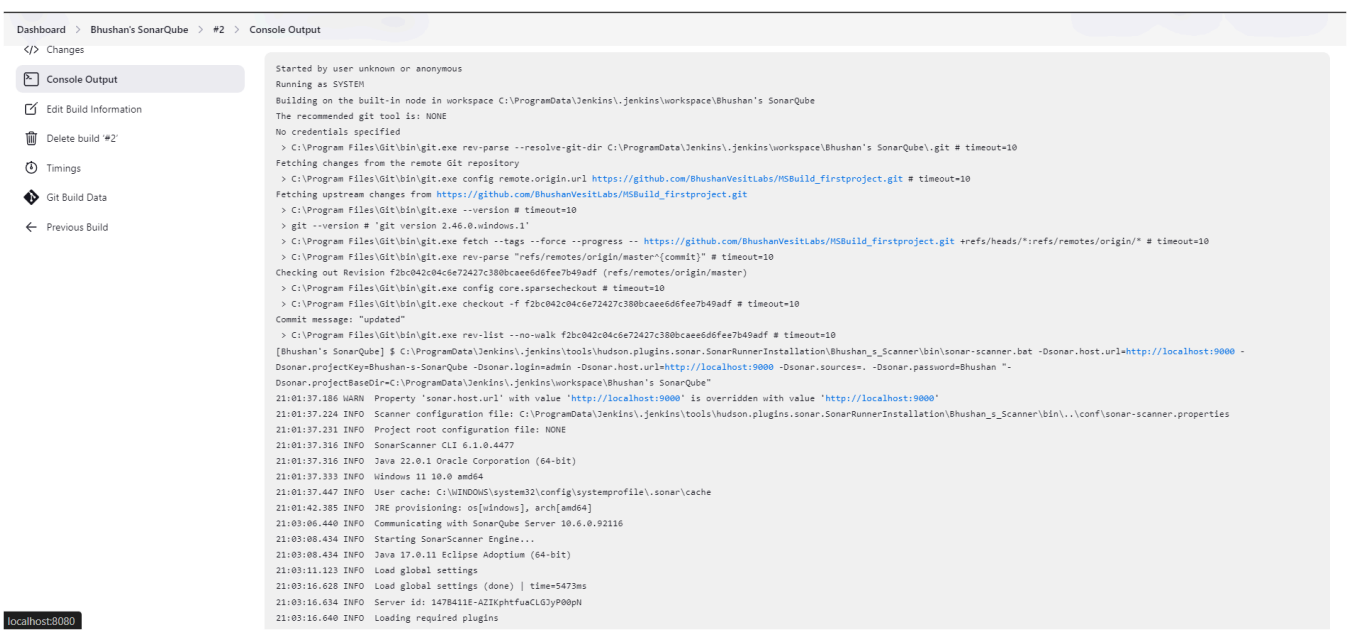
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 12: In the SonarQube go to Security then for Administrator allow Administer system and Execute analysis.



Step 13: See the Console Output.



Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Dashboard > Bhushan's SonarQube > #2 > Console Output

```
  * Any directory in the file path has a name ending in "test" or "tests"

21:09:00.357 INFO  Using git CLI to retrieve untracked files
21:09:00.653 INFO  Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
21:09:00.889 INFO  14 source files to be analyzed
21:09:03.146 INFO  14/14 source files have been analyzed
21:09:03.151 INFO  Sensor TextAndSecretsSensor [text] (done) | time=5027ms
21:09:03.165 INFO  ------------- Run sensors on project
21:09:03.364 INFO  Sensor C# [csharp]
21:09:03.370 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for .NET 5.x or higher, see
https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
21:09:03.370 INFO  Sensor C# [csharp] (done) | time=0ms
21:09:03.370 INFO  Sensor Analysis Warnings import [csharp]
21:09:03.374 INFO  Sensor Analysis Warnings import [csharp] (done) | time=9ms
21:09:03.376 INFO  Sensor C# File Caching Sensor [csharp]
21:09:03.376 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
21:09:03.376 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=1ms
21:09:03.376 INFO  Sensor Zero Coverage Sensor
21:09:03.400 INFO  Sensor Zero Coverage Sensor (done) | time=26ms
21:09:03.406 INFO  SCM Publisher SCM provider for this project is: git
21:09:03.408 INFO  SCM Publisher 4 source files to be analyzed
21:09:06.271 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=2859ms
21:09:06.271 INFO  CPD Executor Calculating CPD for 0 files
21:09:06.275 INFO  CPD Executor CPD calculation finished (done) | time=0ms
21:09:06.302 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
21:09:12.497 INFO  Analysis report generated in 494ms, dir size=201.1 kB
21:09:12.727 INFO  Analysis report compressed in 60ms, zip size=22.3 kB
21:09:23.622 INFO  Analysis report uploaded in 10889ms
21:09:23.629 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=Bhushan-s-SonarQube
21:09:23.631 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:09:23.631 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=a6525be8-4189-4c6d-8aca-a6d0b9bb7551
21:09:23.663 INFO  Analysis total time: 5:02.251 s
21:09:23.676 INFO  SonarScanner Engine completed successfully
21:09:23.925 INFO  EXECUTION SUCCESS
21:09:24.066 INFO  Total time: 7:46.713s
Finished: SUCCESS
```

Jenkins

Search (CTRL+K) log in

Dashboard > Bhushan's SonarQube >

- Status
- Changes
- Workspace
- Build Now
- Configure
- Delete Project
- SonarQube
- Rename

Build History trend

Filter... /

#2 Sep 19, 2024, 9:01 PM

#1 Sep 19, 2024, 8:52 PM

Atom feed for all Atom feed for failures

Bhushan's SonarQube

Add description

SonarQube

Permalinks

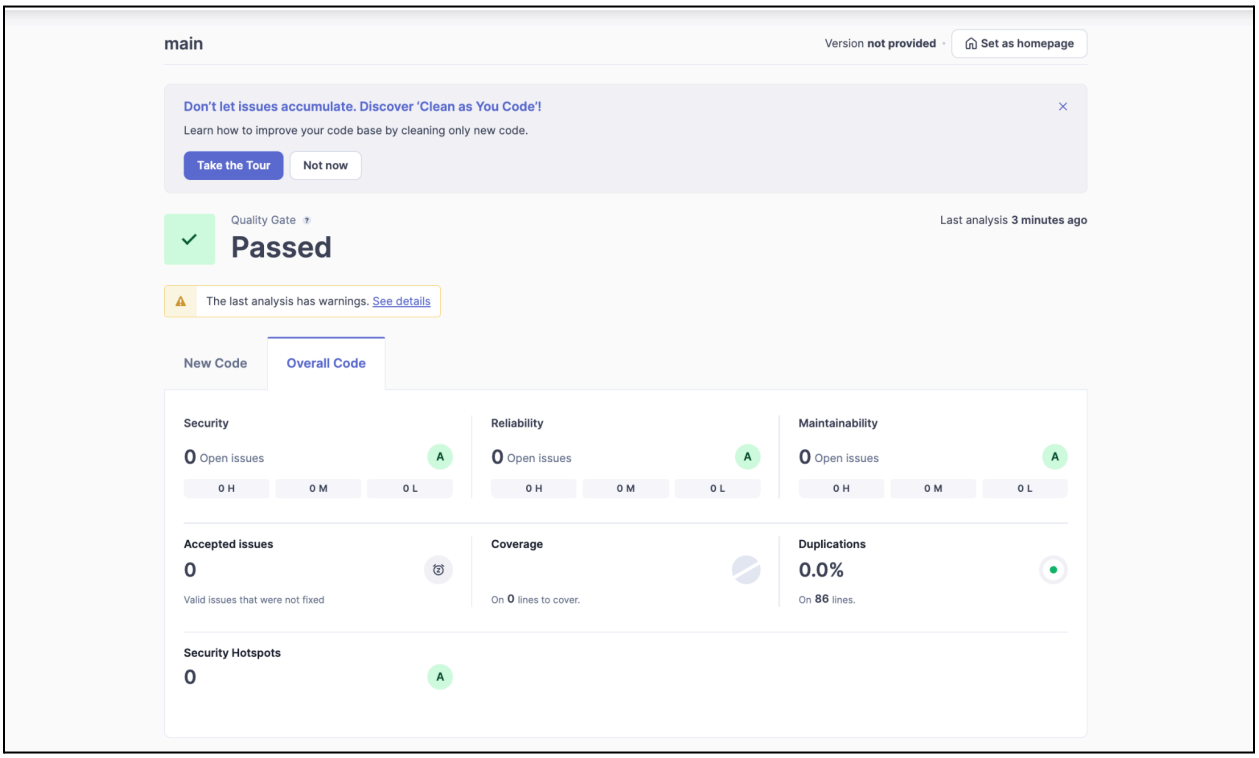
- Last build (#2), 20 min ago
- Last stable build (#2), 20 min ago
- Last successful build (#2), 20 min ago
- Last failed build (#1), 29 min ago
- Last unsuccessful build (#1), 29 min ago
- Last completed build (#2), 20 min ago

REST API Jenkins 2.462.1

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 14: Now See the SonarQube Project.



Conclusion :

In this experiment, we have learned how to perform Jenkins SAST using SonarQube. For this, we used a docker image of SonarQube to not install it locally on our system. After installing the required configurations on Jenkins, using a code from a GitHub repository, we analyze its code using SonarQube. Once we build the project, we can see that the SonarQube project displays that the code has no errors.