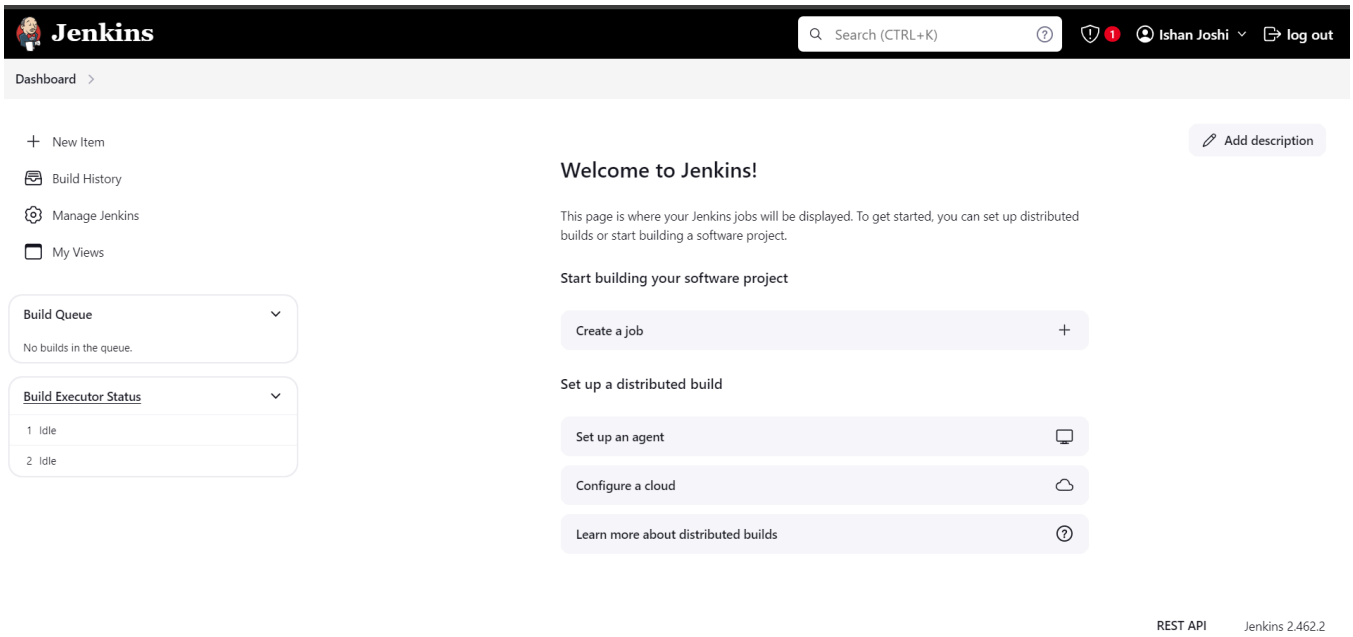
Name-Ishan Kiran Joshi DIV-D15C Roll NO-21 A.Y.-2024-25

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

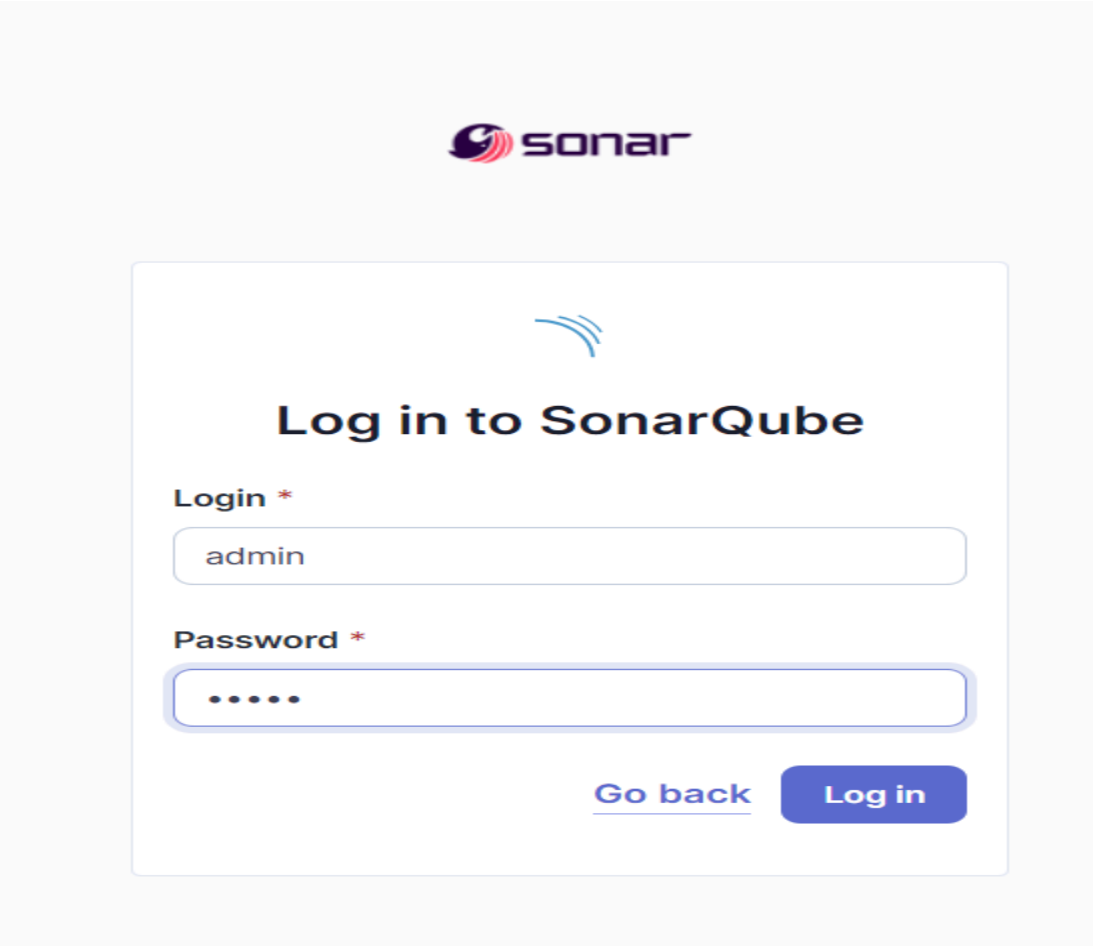
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



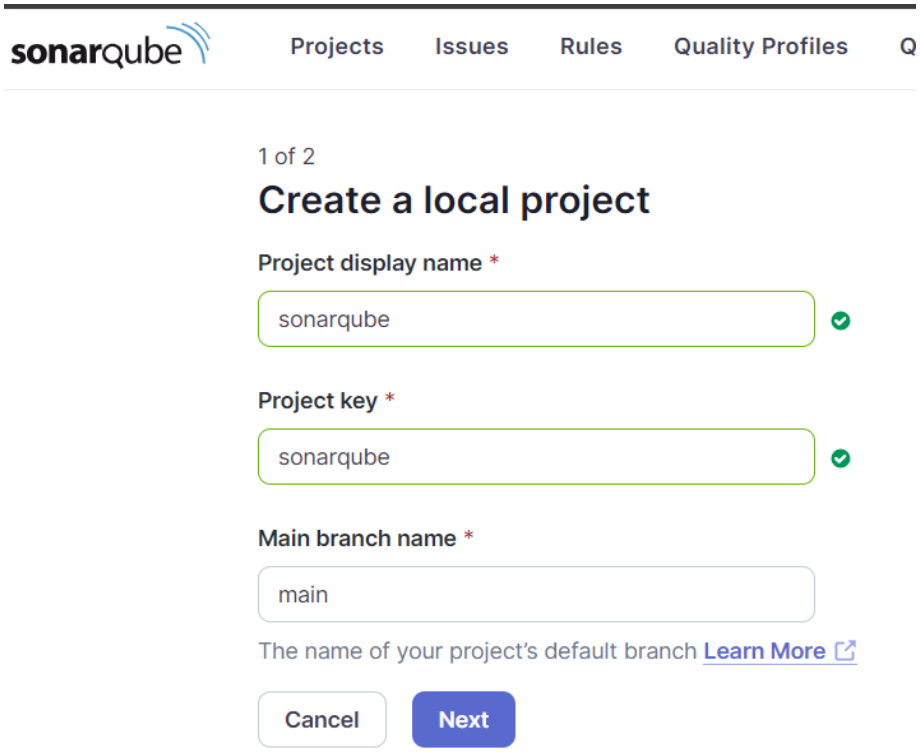
2. Run SonarQube in a Docker container using this command :- a] docker -v
b] docker pull sonarqube
c] docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000sonarqube:latest

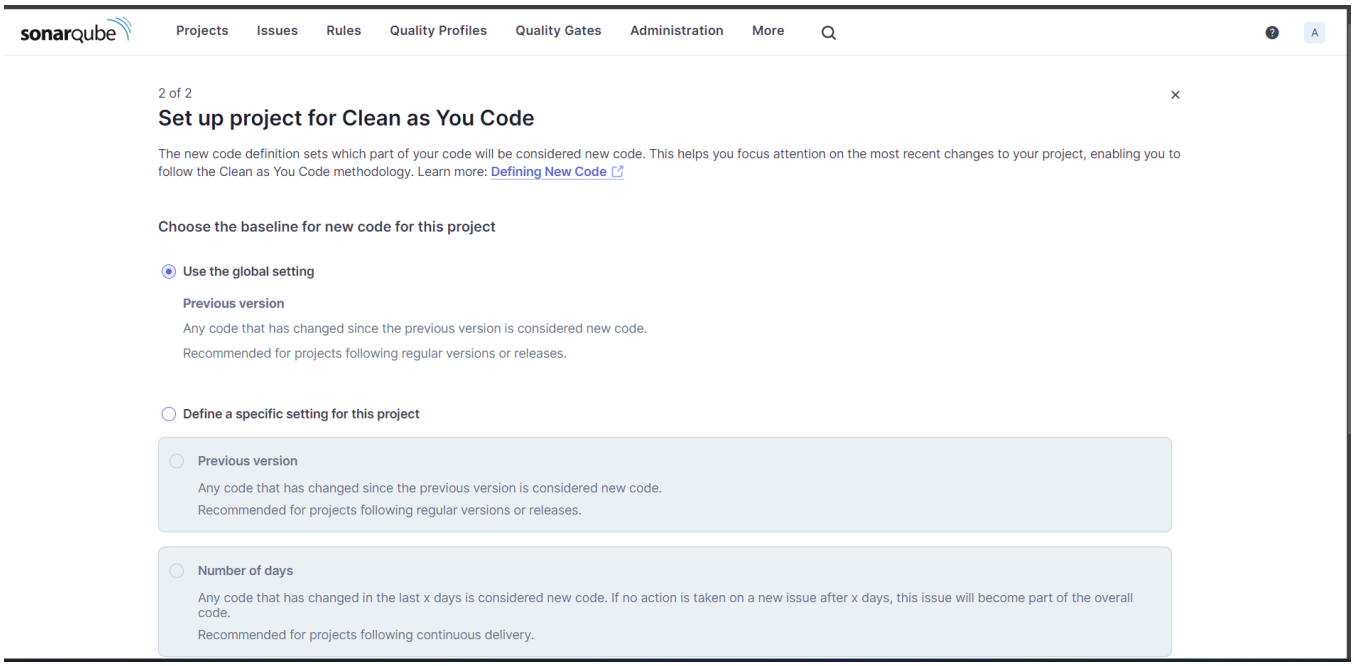
```
C:\Users\ishan>docker -v
Docker version 27.1.1, build 6312585

C:\Users\ishan>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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
23c14503da77ee785f6069bfaff714939ddb794a6b846124594503f6183b4c68
```

3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is “**ishan**”.



4. Create a local project in SonarQube with the name **sonarqube**





5. Setup the project and come back to Jenkins Dashboard. Go to **Manage Jenkins → Plugins** and search for **SonarQube Scanner** in **Available Plugins**

and install it.

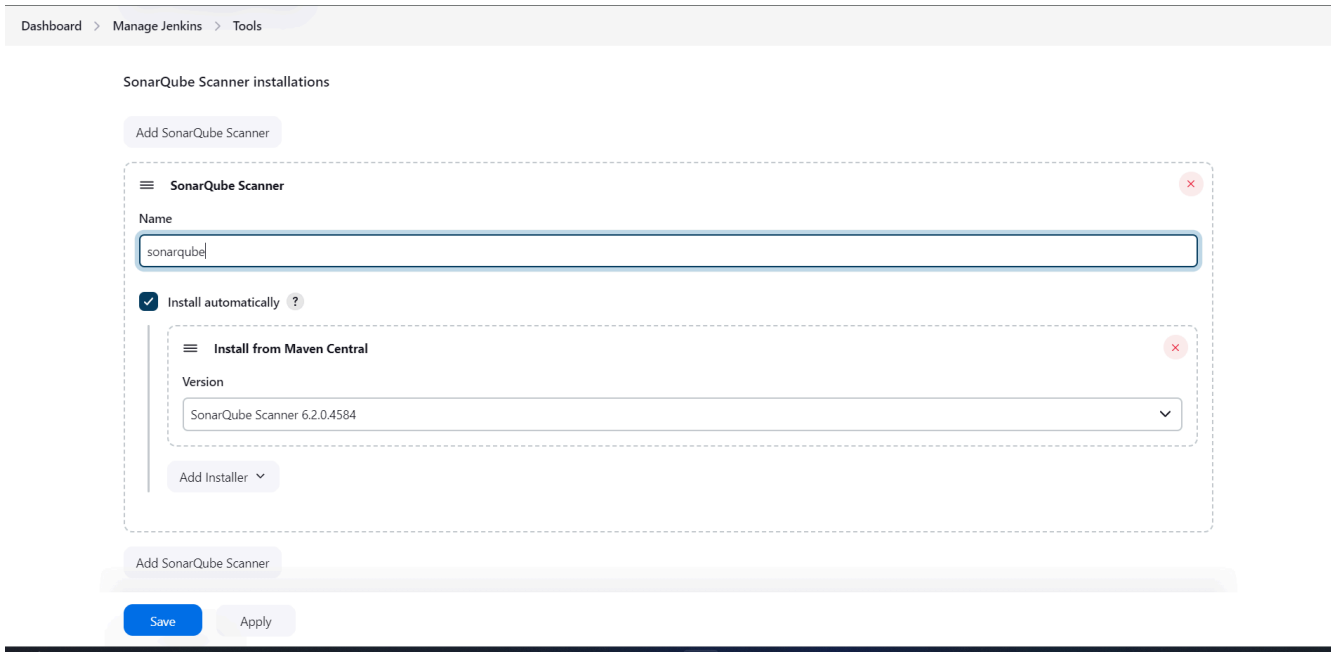
6. Under '**Manage Jenkins → System**', look for **SonarQube Servers** and enter these details.

Name : sonarqube

Server URL : http://localhost:9000

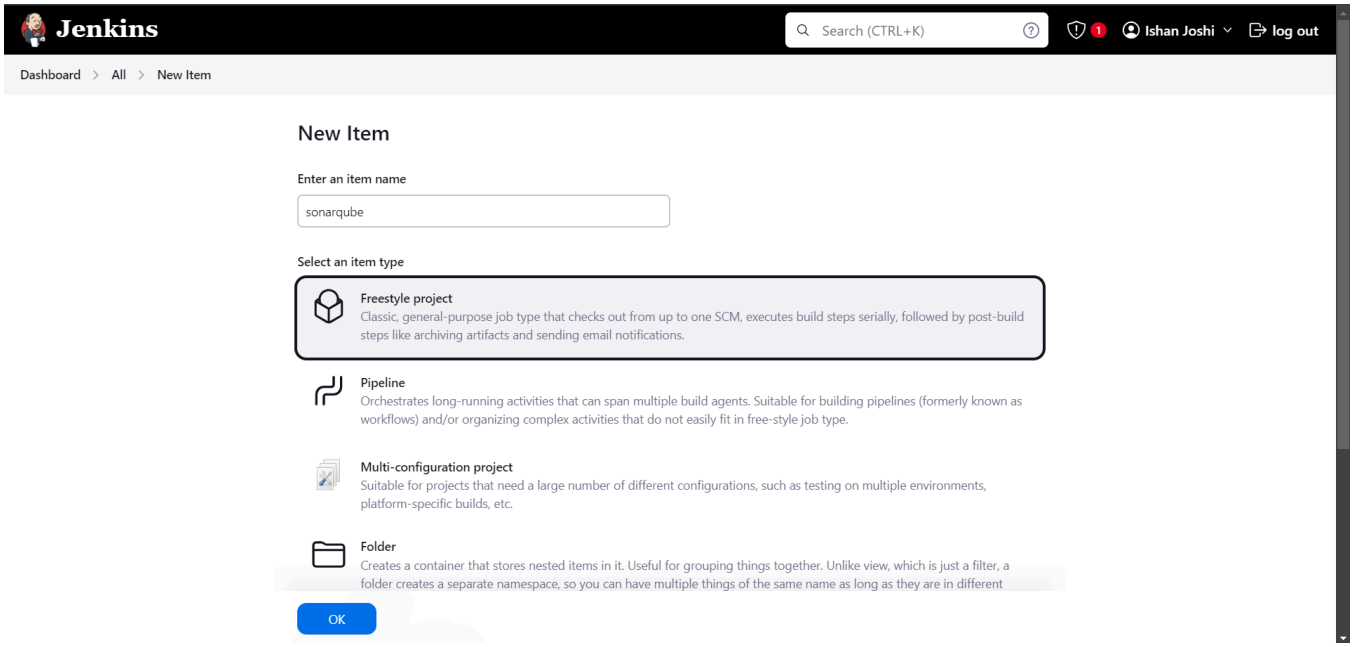
7. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Manage Jeknins → Tools → SonarQube Scanner Installation

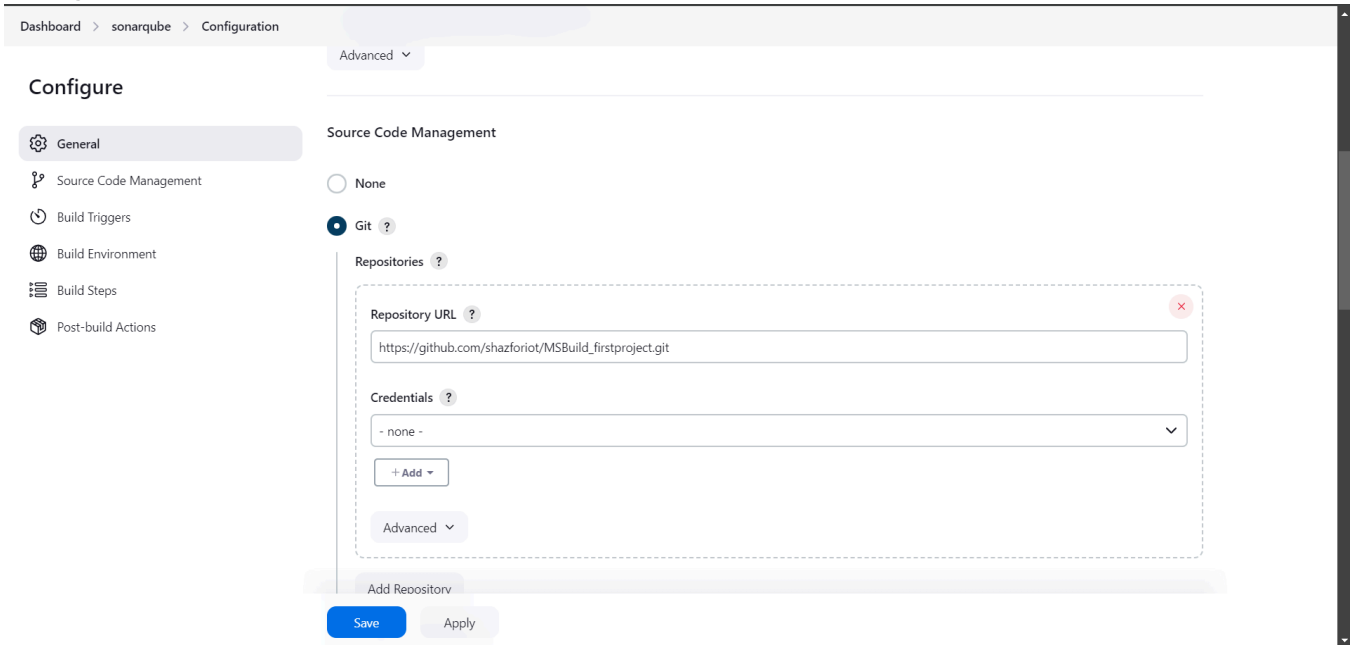


8. After the configuration, create a **New Item** in Jenkins, choose a

freestyle project named **sonarqube**.



9. 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 the integration.



10. Under **Build-> Execute SonarQube Scanner**, enter these **Analysis Properties**. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

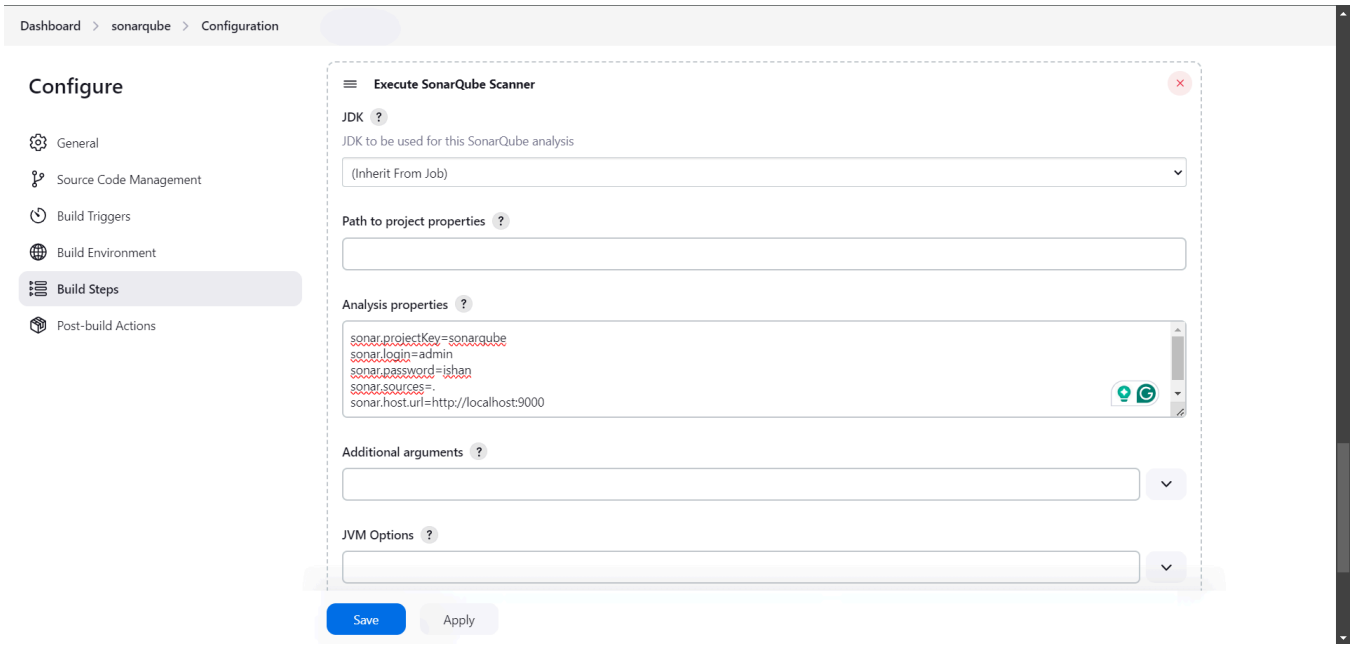
sonar.projectKey=sonarqube

sonar.login=admin

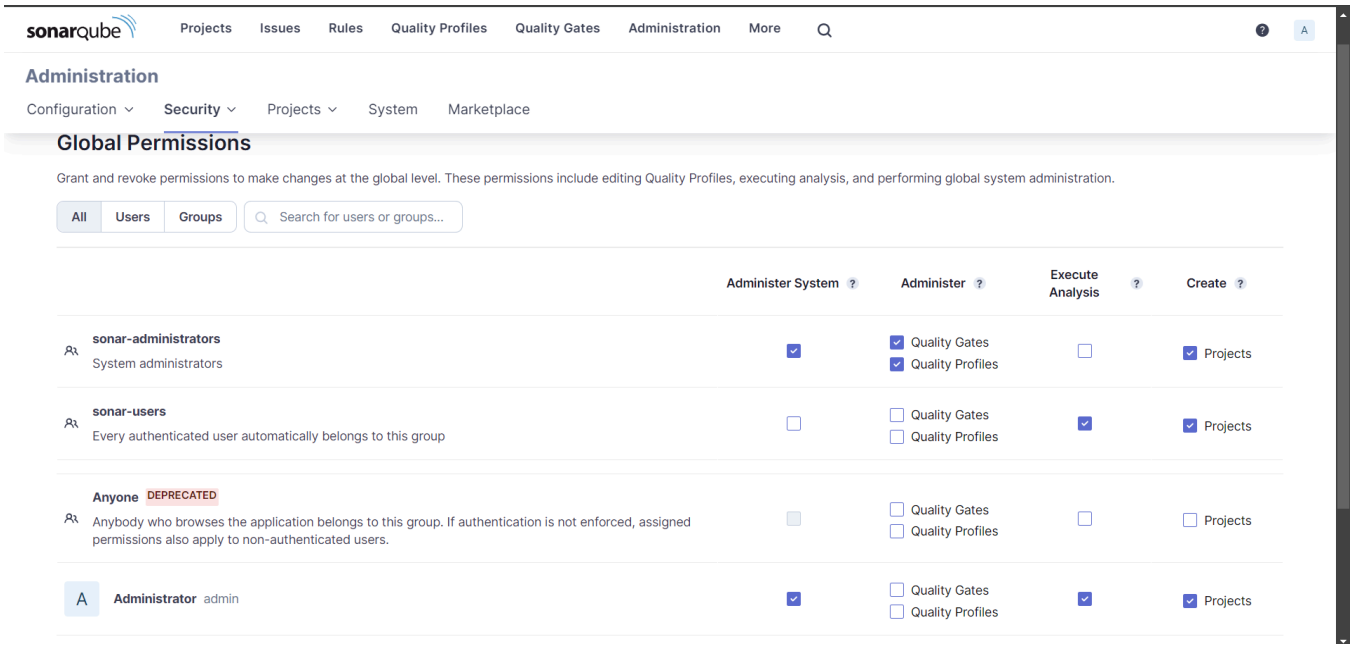
sonar.password=aditya

sonar.sources=.

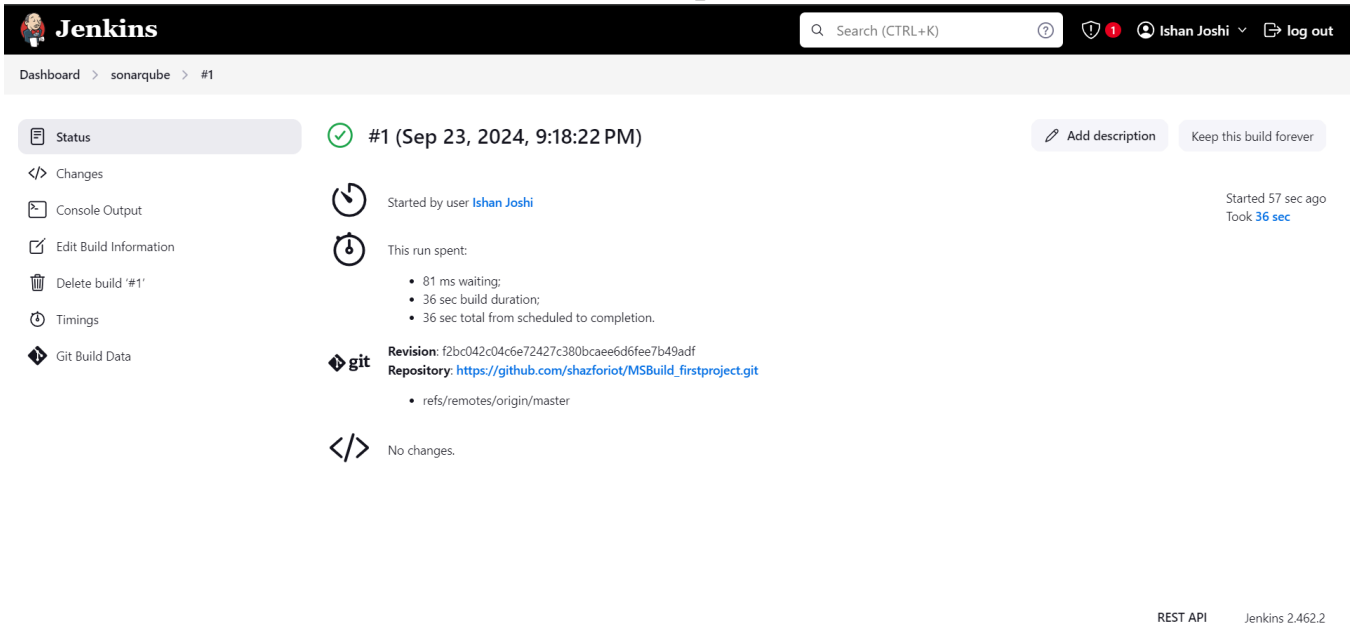
sonar.host.url=http://localhost:9000

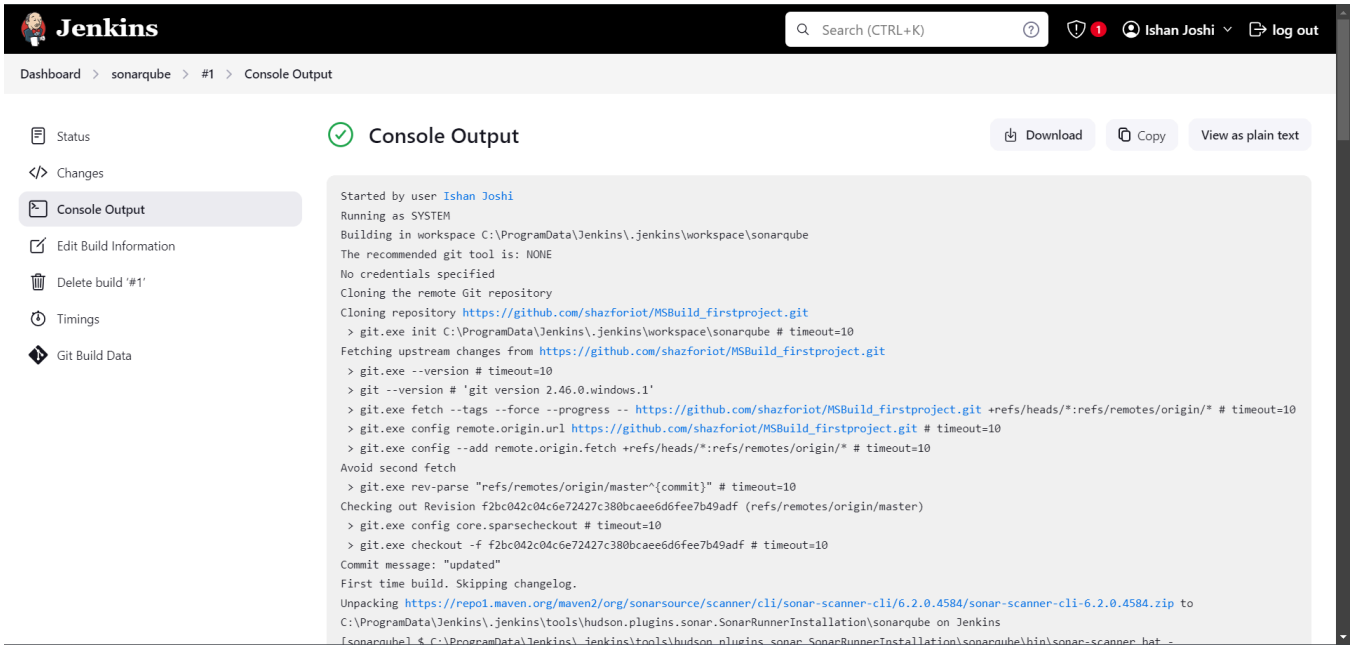


11. Go to http://localhost:9000/admin/permissions and allow Execute Permissions to the Admin user.



12. Run The **Build** and check the **console output**.

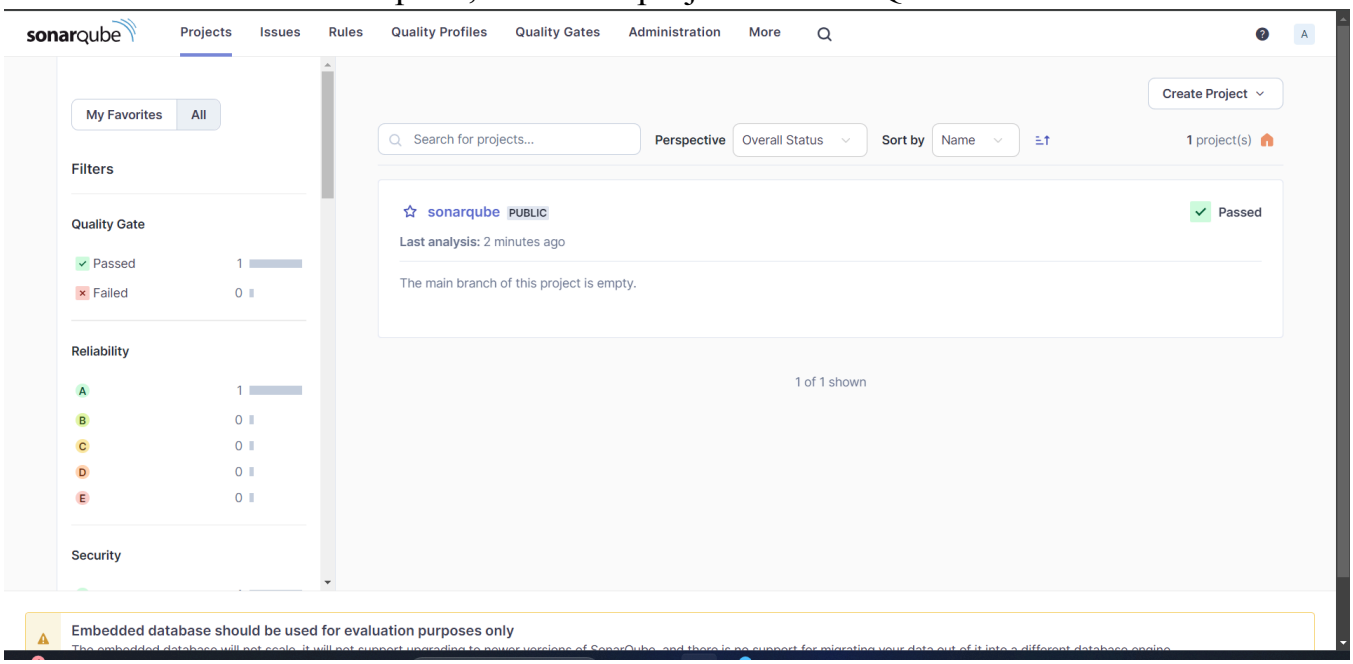


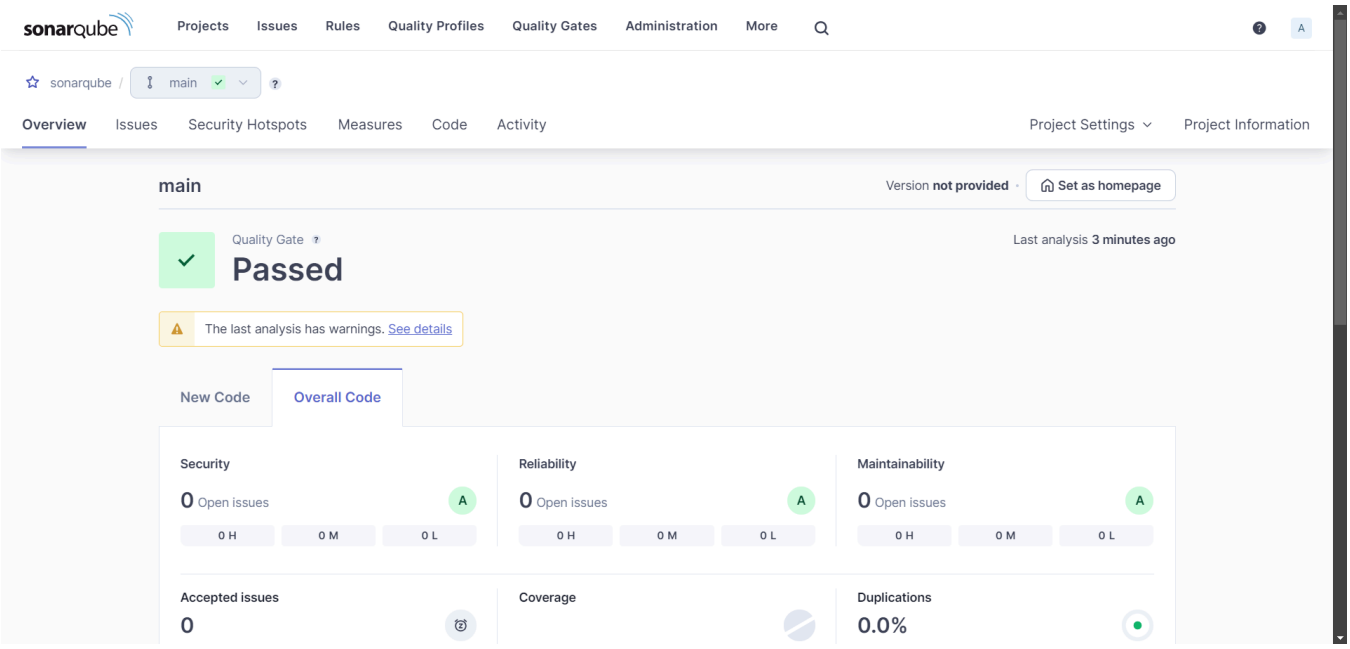


Dashboard > sonarqube > #1 > Console Output

```
21:18:56.871 INFO  ------------- Run sensors on project
21:18:57.082 INFO  Sensor C# [csharp]
21:18:57.082 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
21:18:57.082 INFO  Sensor C# [csharp] (done) | time=0ms
21:18:57.082 INFO  Sensor Analysis Warnings import [csharp]
21:18:57.088 INFO  Sensor Analysis Warnings import [csharp] (done) | time=6ms
21:18:57.088 INFO  Sensor C# File Caching Sensor [csharp]
21:18:57.088 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
21:18:57.089 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=0ms
21:18:57.089 INFO  Sensor Zero Coverage Sensor
21:18:57.099 INFO  Sensor Zero Coverage Sensor (done) | time=11ms
21:18:57.099 INFO  SCM Publisher SCM provider for this project is: git
21:18:57.108 INFO  SCM Publisher 4 source files to be analyzed
21:18:57.578 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=463ms
21:18:57.579 INFO  CPD Executor Calculating CPD for 0 files
21:18:57.579 INFO  CPD Executor CPD calculation finished (done) | time=0ms
21:18:57.588 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
21:18:57.812 INFO  Analysis report generated in 175ms, dir size=201.0 kB
21:18:57.878 INFO  Analysis report compressed in 66ms, zip size=22.4 kB
21:18:58.055 INFO  Analysis report uploaded in 174ms
21:18:58.058 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
21:18:58.058 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
21:18:58.058 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=60f78a61-ae1b-4623-aa2e-0601bb2f5292
21:18:58.060 INFO  Analysis total time: 16.021 s
21:18:58.060 INFO  SonarScanner Engine completed successfully
21:18:58.150 INFO  EXECUTION SUCCESS
21:18:58.151 INFO  Total time: 25.472s
Finished: SUCCESS
```

13. Once the build is complete, check the project in SonarQube.





By following these steps, we have successfully integrated Jenkins with SonarQube for performing SAST.

Conclusion:

In this integration process, we have successfully connected Jenkins with SonarQube for Static Application Security Testing (SAST), emphasizing the importance of proactive code analysis. This setup automates the detection of security vulnerabilities and code quality issues throughout the development cycle. By implementing SAST, teams can identify potential flaws early, leading to more secure and reliable applications. Ultimately, this integration fosters a culture of continuous improvement, ensuring adherence to security best practices in software development.