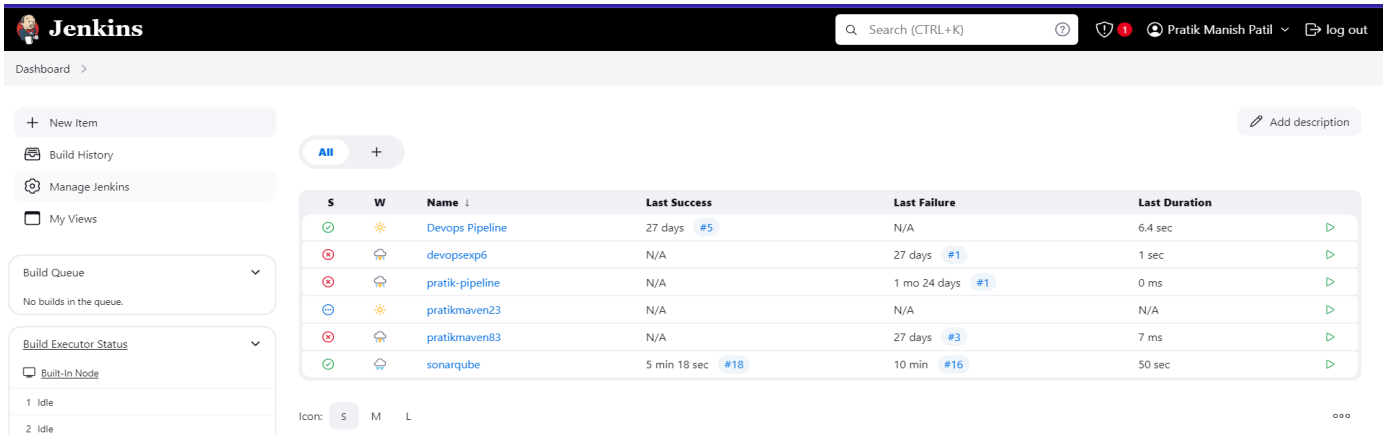
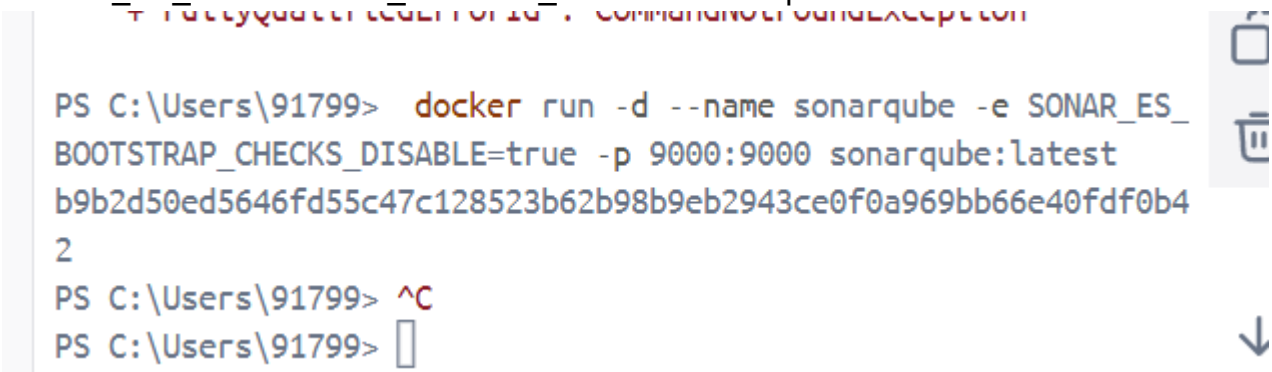
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.



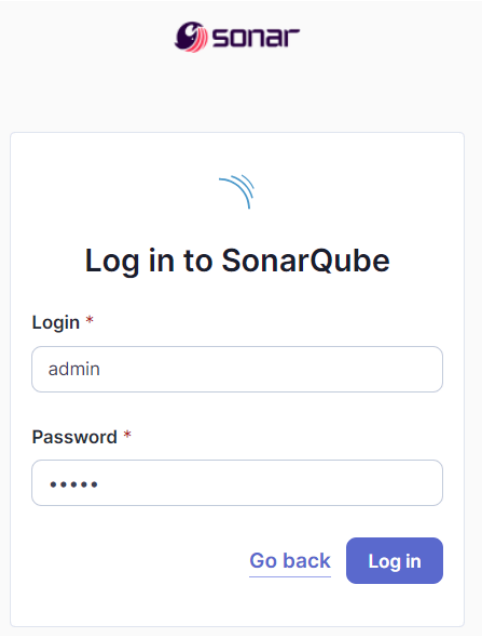
1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

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

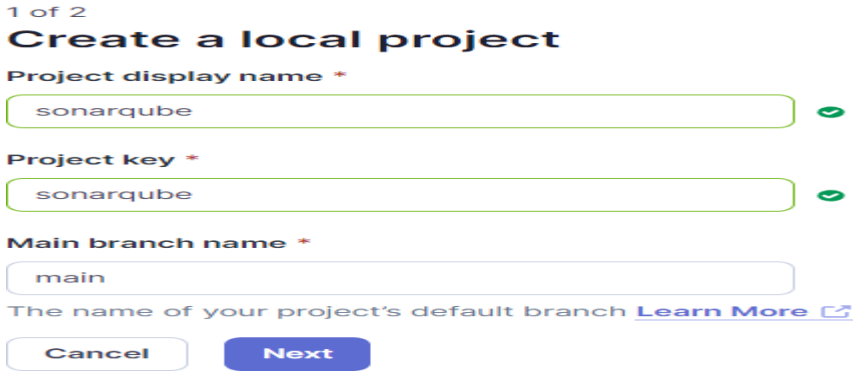


9000:9000
sonarqube:latest

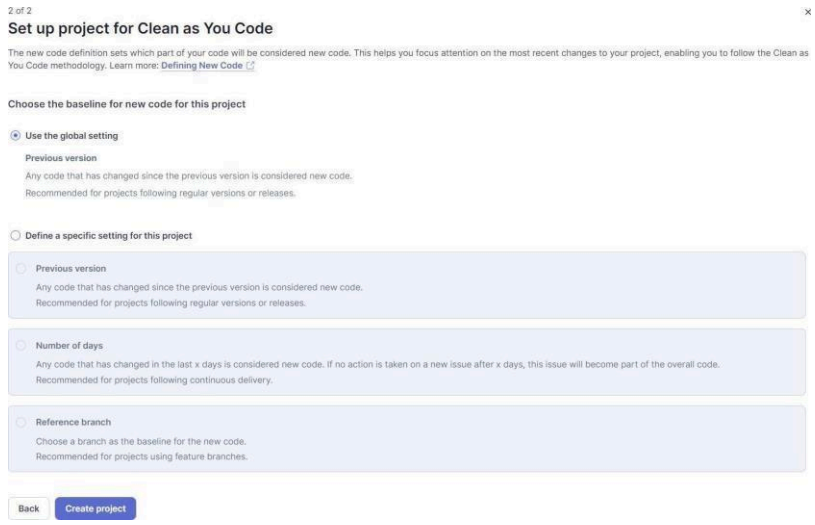
3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is “**aditya**”.



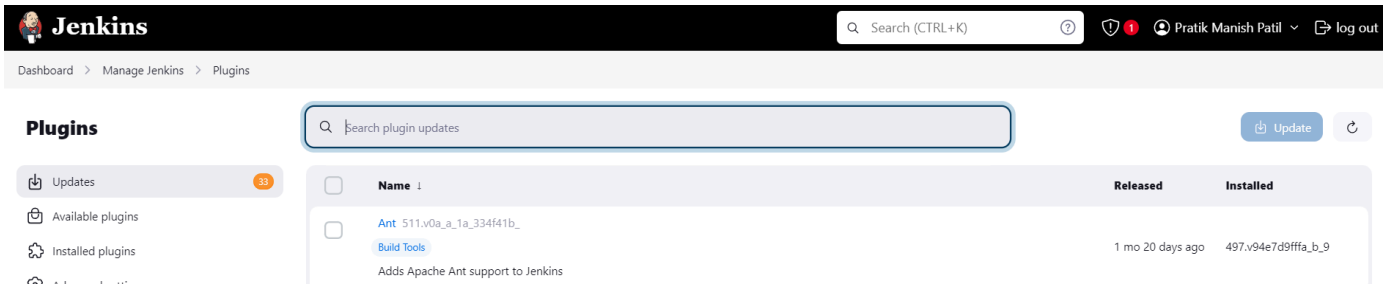
4. Create a local project in SonarQube with the name **sonarqube**



5.



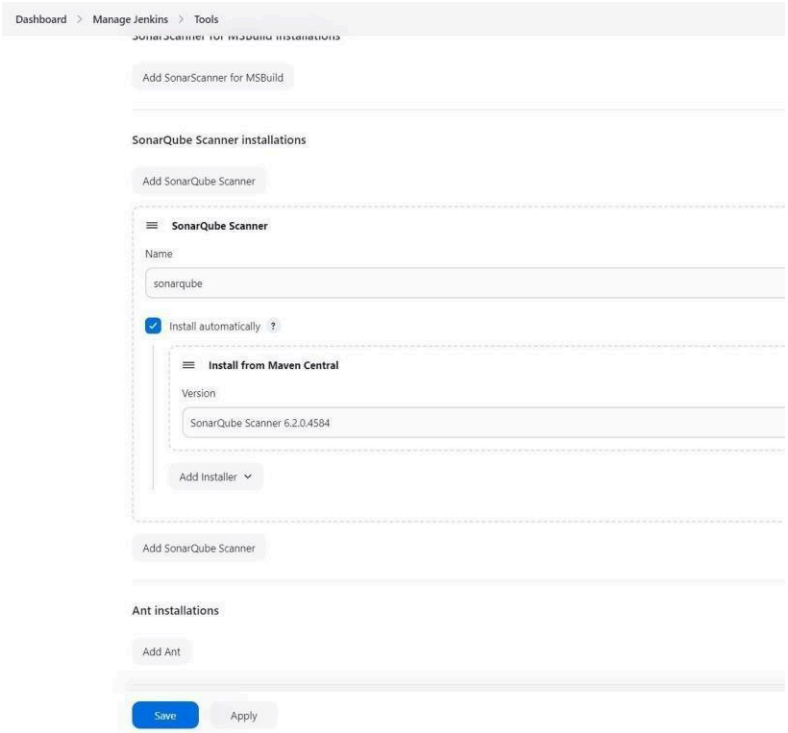
6. Setup the project and come back to Jenkins Dashboard. Go to **Manage Jenkins → Plugins** and search for **SonarQube Scanner** in **Available Plugins** and
install it.



7. Under '**Manage Jenkins → System**', look for **SonarQube Servers** and enter these details.
Name : sonarqube
Server URL : http://localhost:9000

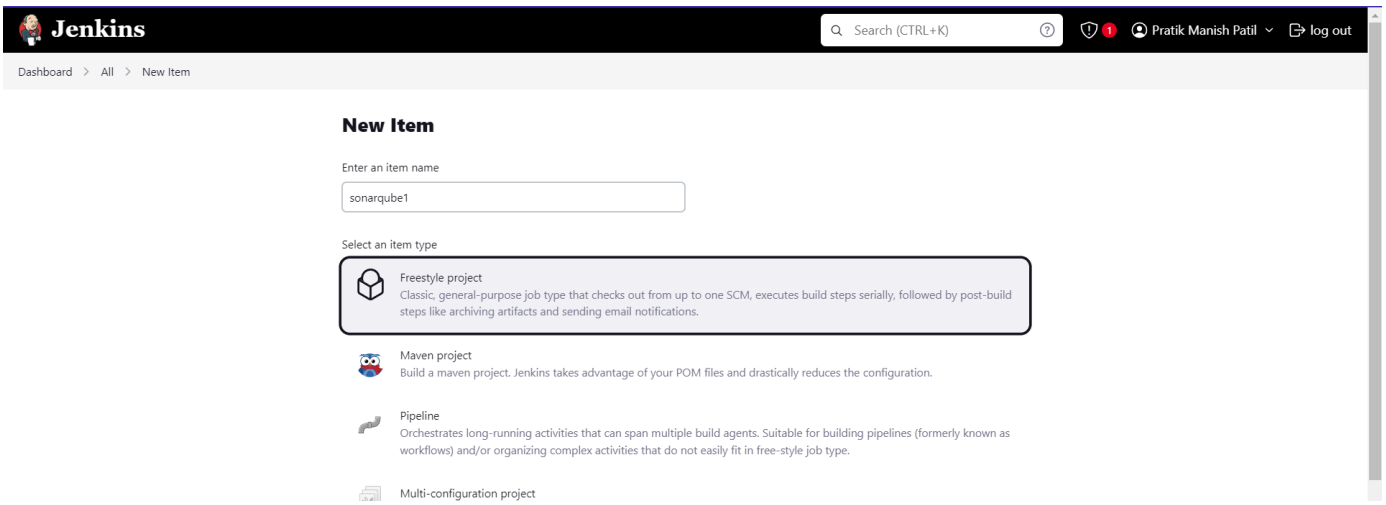
8. Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.

Manage Jeknins → Tools → SonarQube Scanner Installation

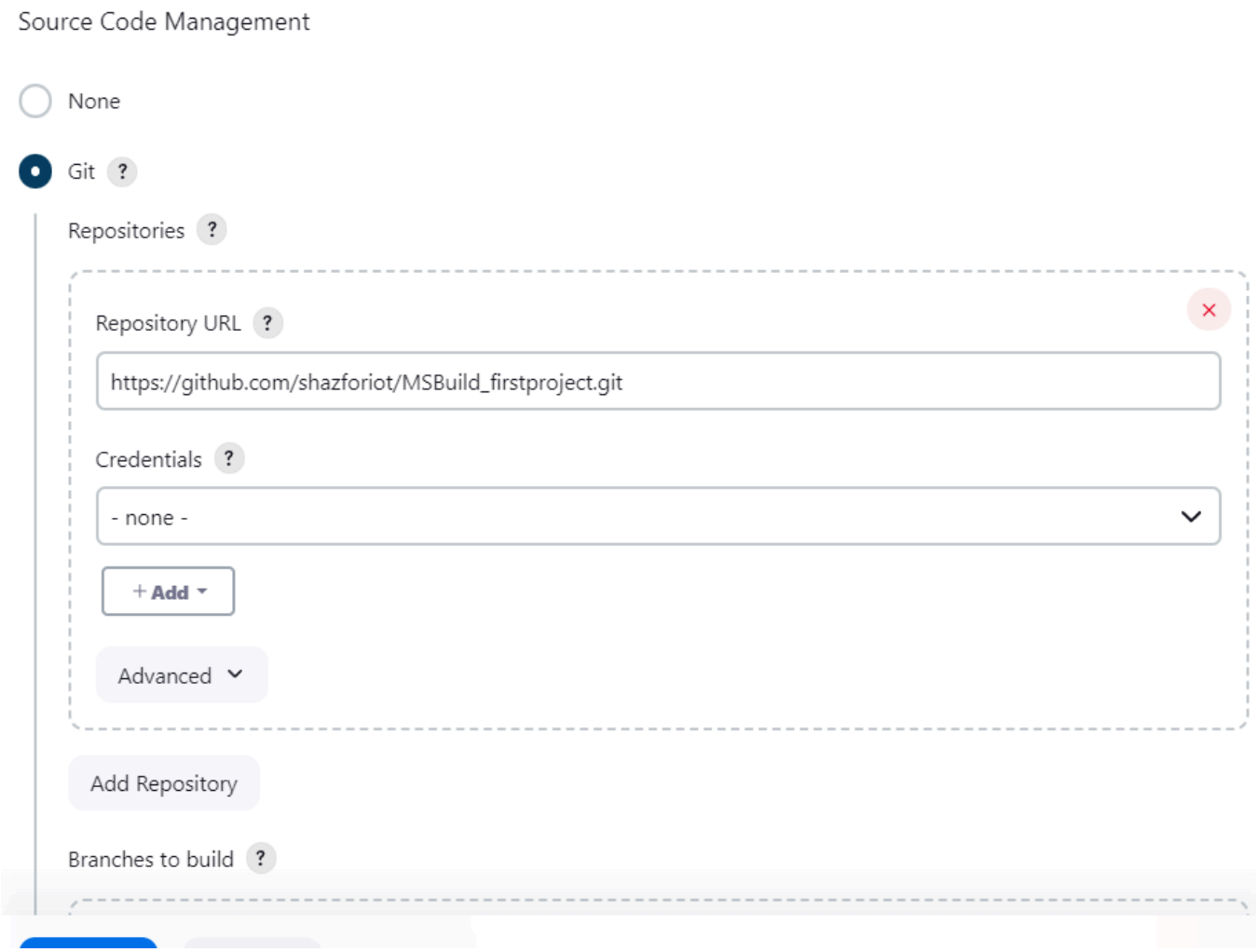


9. After the configuration, create a **New Item** in Jenkins, choose a

freestyle project named **sonarqube**.



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 the integration.



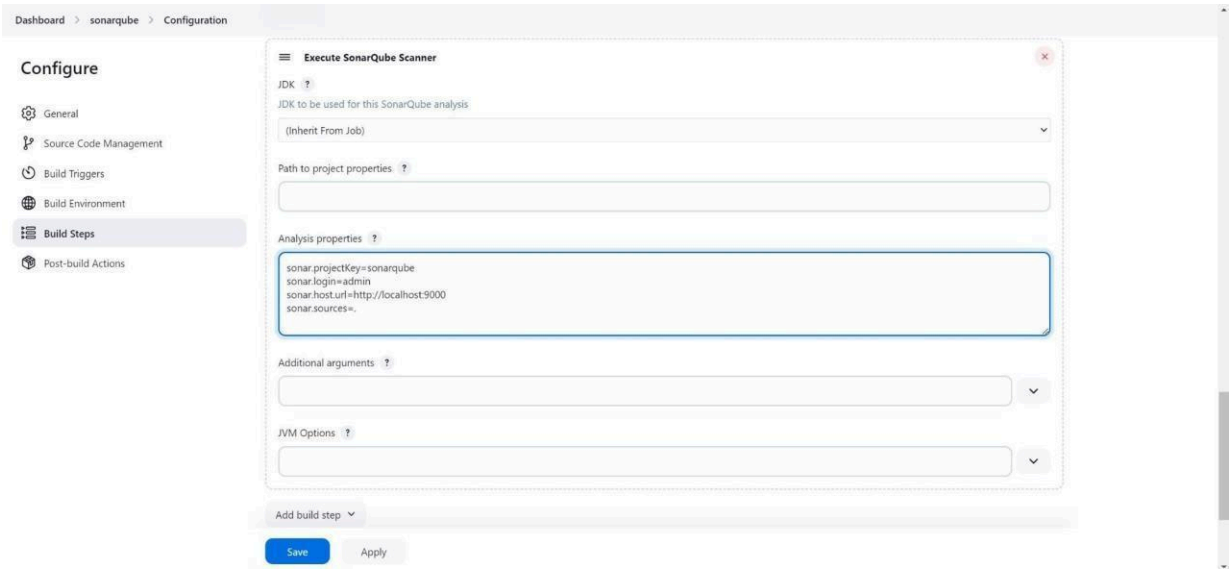
11. Under **Build-> Execute SonarQube Scanner**, enter these **Analysis Properties**. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

sonar.projectKey=sonarqube

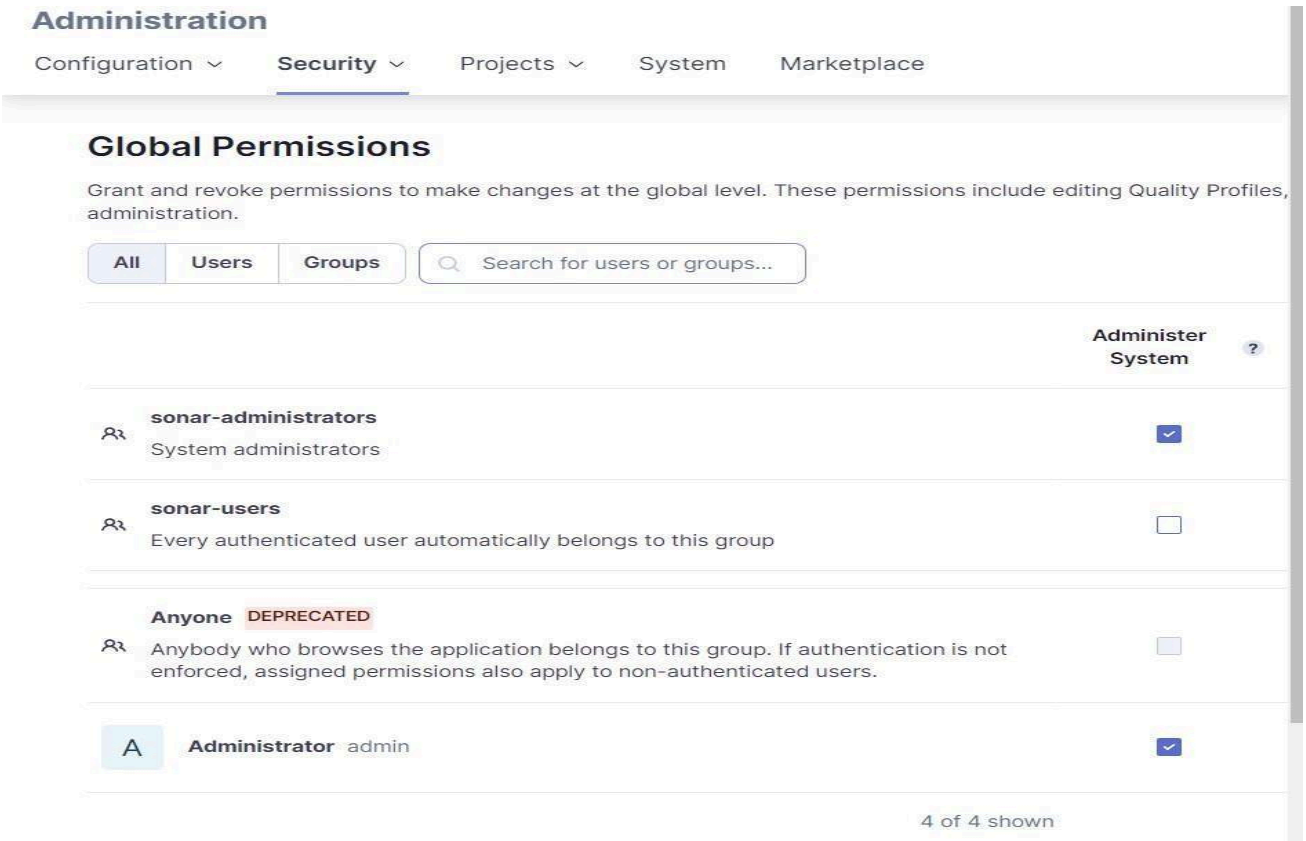
sonar.login=admin

sonar.password=pratik sonar.sources=.

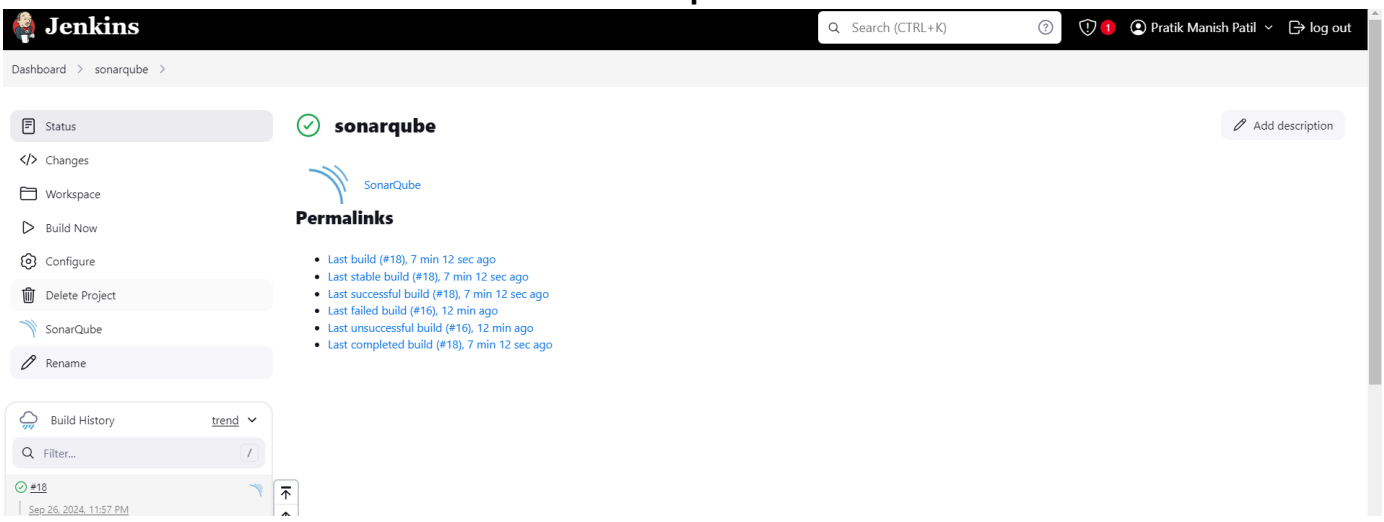
sonar.host.url=http://localhost:9000

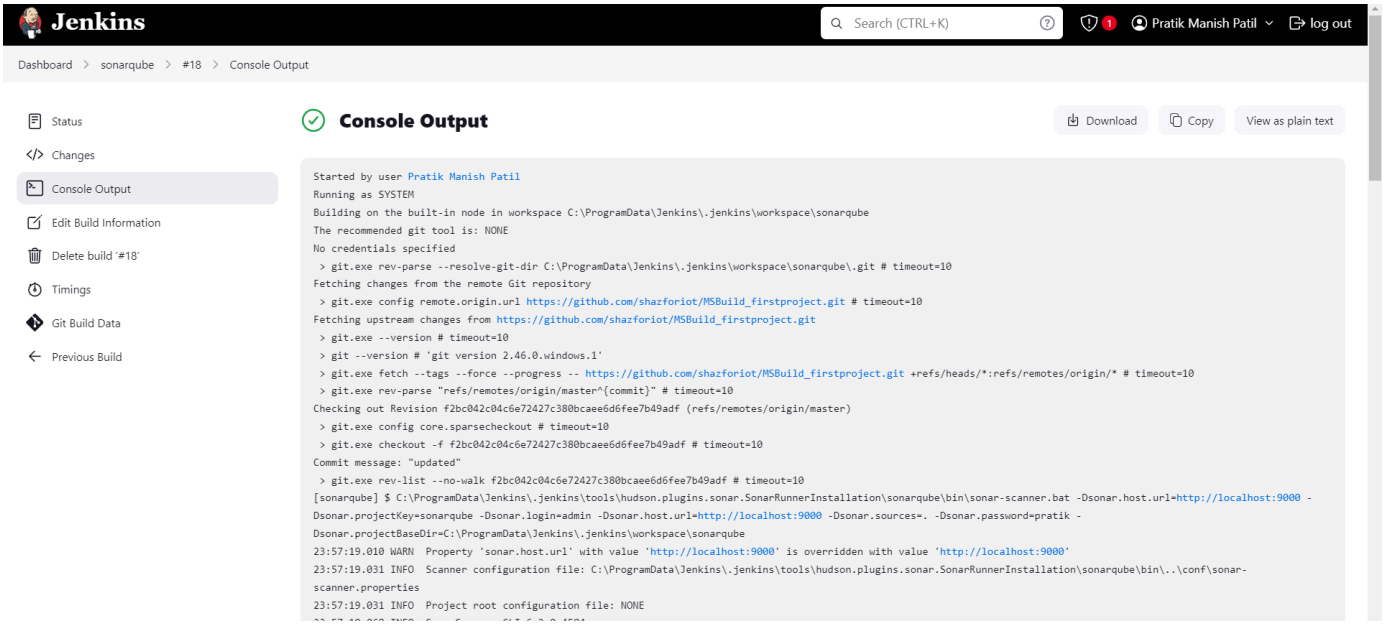


12. Go to http://localhost:9000/admin/permissions and allow Execute Permissions to the Admin user.



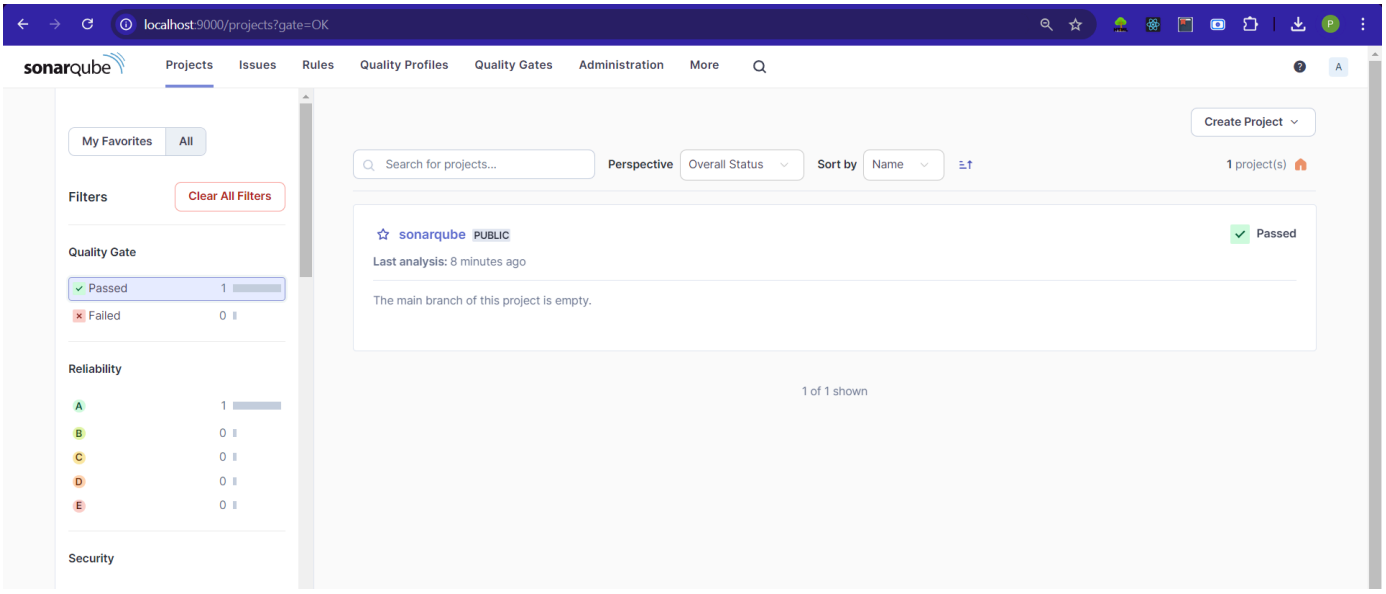
13. Run The **Build** and check the **console output**.

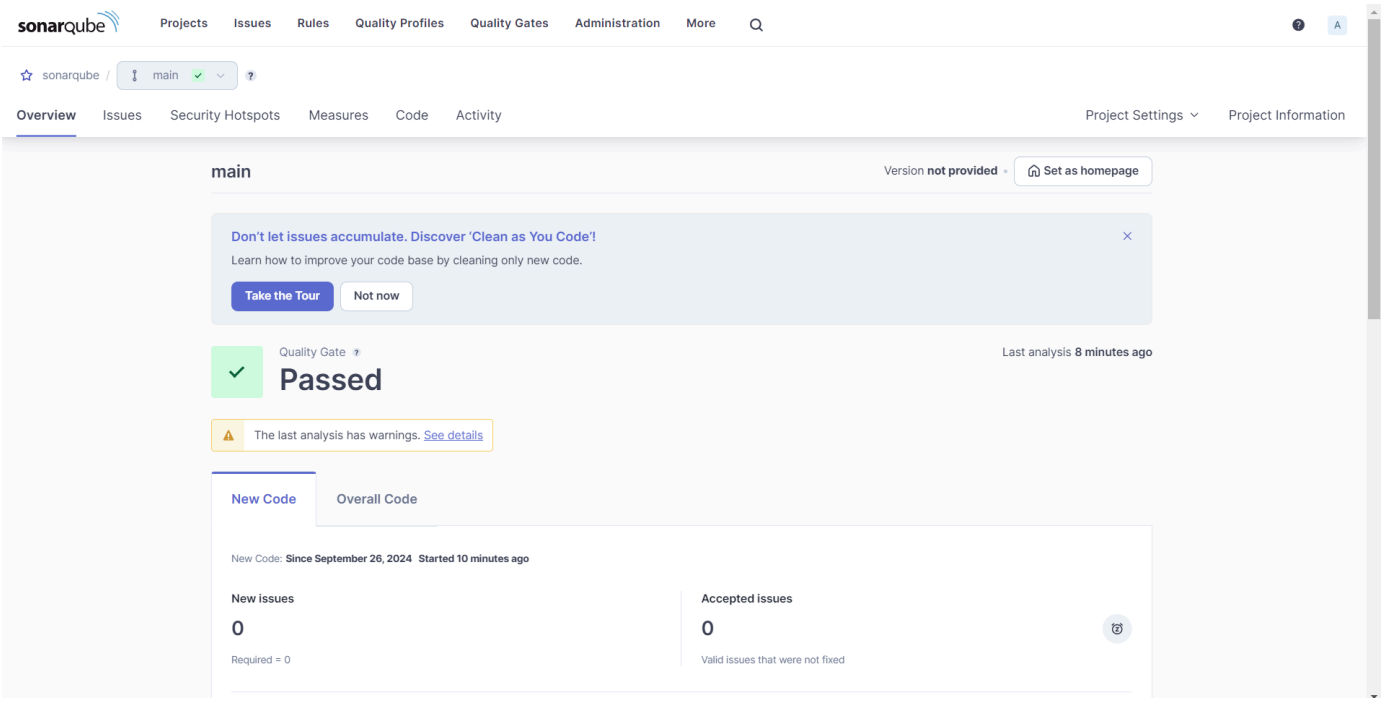




```
23:58:03.598 WARN  Your project contains C# files which cannot be analyzed with the scanner you are using. To analyze C# or VB.NET, you must use the SonarScanner for
5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
23:58:03.598 INFO  Sensor C# [csharp] (done) | time=3ms
23:58:03.598 INFO  Sensor Analysis Warnings import [csharp]
23:58:03.603 INFO  Sensor Analysis Warnings import [csharp] (done) | time=5ms
23:58:03.605 INFO  Sensor C# File Caching Sensor [csharp]
23:58:03.605 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
23:58:03.606 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=2ms
23:58:03.606 INFO  Sensor Zero Coverage Sensor
23:58:03.629 INFO  Sensor Zero Coverage Sensor (done) | time=24ms
23:58:03.715 INFO  CPD Executor Calculating CPD for 0 files
23:58:03.715 INFO  CPD Executor CPD calculation finished (done) | time=0ms
23:58:03.725 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
23:58:04.048 INFO  Analysis report generated in 199ms, dir size=200.5 kB
23:58:04.123 INFO  Analysis report compressed in 58ms, zip size=21.9 kB
23:58:04.383 INFO  Analysis report uploaded in 256ms
23:58:04.397 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
23:58:04.398 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
23:58:04.398 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=e1f59b99-8e2c-4e18-bf46-89b93df6a637
23:58:04.430 INFO  Analysis total time: 34.976 s
23:58:04.432 INFO  SonarScanner Engine completed successfully
23:58:04.504 INFO  EXECUTION SUCCESS
23:58:04.504 INFO  Total time: 45.485s
Finished: SUCCESS
```

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




sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
sonarqube /
main
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
main
Version not provided
Set as homepage
Don't let issues accumulate. Discover 'Clean as You Code'!
Learn how to improve your code base by cleaning only new code.
Take the Tour
Not now
Quality Gate
Passed
Last analysis 8 minutes ago
The last analysis has warnings. See details
New Code
Overall Code
New Code: Since September 26, 2024 Started 10 minutes ago
New issues
0
Required = 0
Accepted issues
0
Valid issues that were not fixed

In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion:

The goal of the Static Application Security Testing (SAST) process is to identify security vulnerabilities early in the development cycle by analyzing source code without executing it. Integrating SAST tools like SonarQube with Jenkins or GitLab enables automated code scanning during the CI/CD pipeline, ensuring that potential security flaws are detected and addressed before code deployment.

By using Jenkins, you can set up a pipeline to run SAST tools like SonarQube after each commit, generating security reports and enforcing quality gates. This helps ensure continuous security checks and improves the overall code quality and security posture of the application.