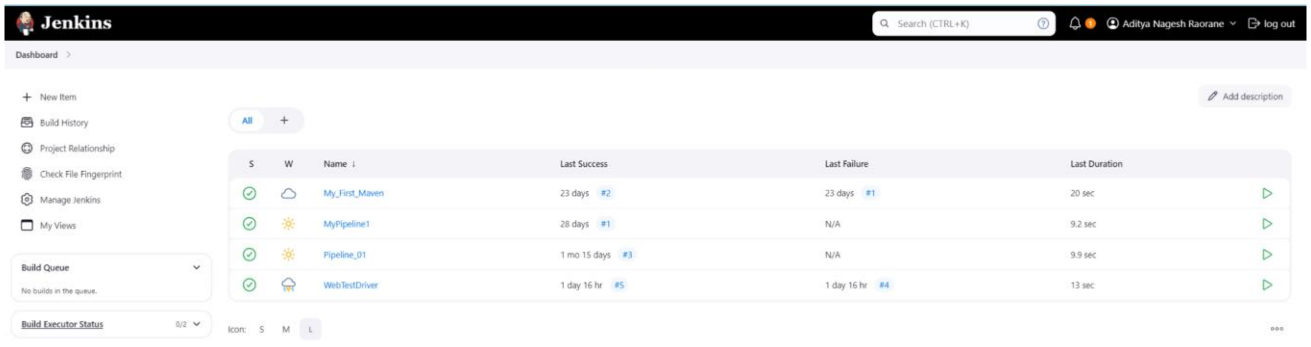
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

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

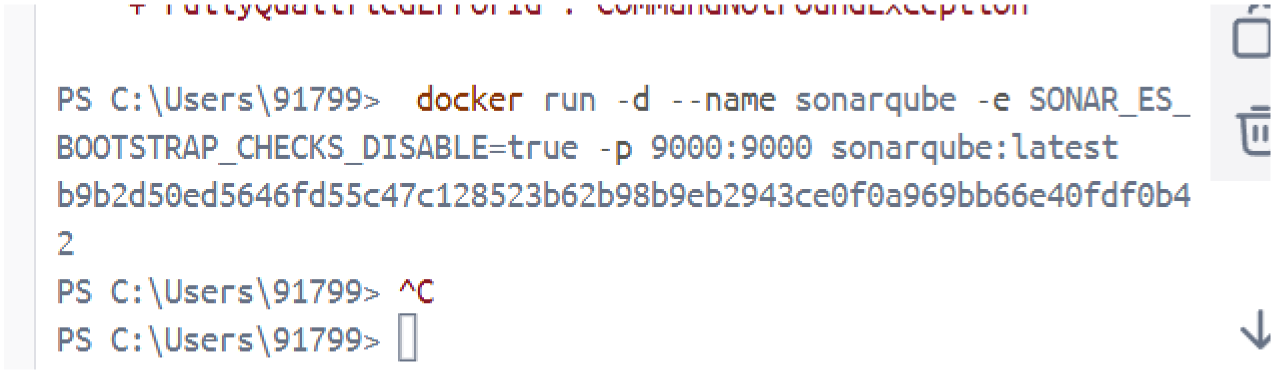


Jenkins | Search (CTRL+K) | Aditya Nagesh Raorane | log out

Dashboard

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		My_First_Maven	23 days #2	23 days #1	20 sec
✓		MyPipeline1	28 days #1	N/A	9.2 sec
✓		Pipeline_01	1 mo 15 days #3	N/A	9.9 sec
✓		WebTestDriver	1 day 16 hr #5	1 day 16 hr #4	13 sec

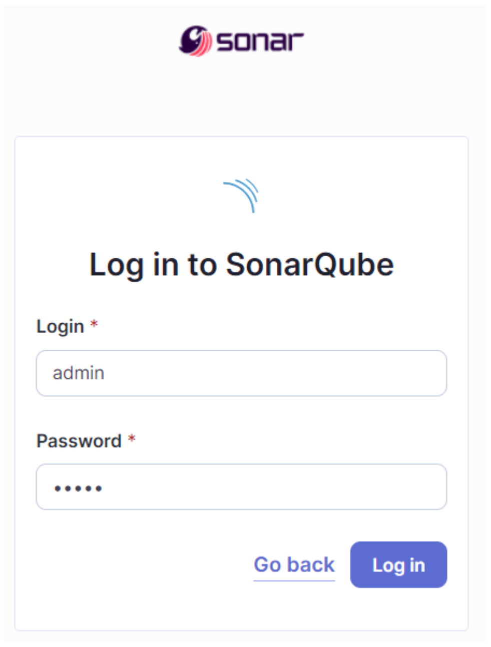
2. Run SonarQube in a Docker container using this command :- a] docker -v
b] docker pull sonarqube
c] docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000



```
PS C:\Users\91799>  docker run -d --name sonarqube -e SONAR_ES_
BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
b9b2d50ed5646fd55c47c128523b62b98b9eb2943ce0f0a969bb66e40fdf0b4
2
PS C:\Users\91799> ^C
PS C:\Users\91799>
```

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**".



Log in to SonarQube

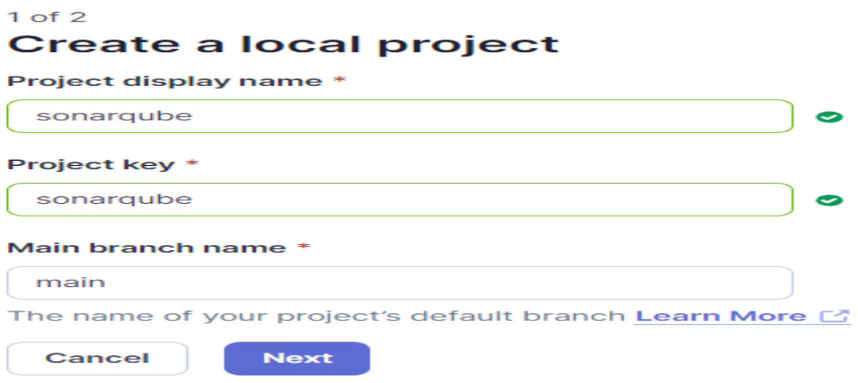
Login *

admin

Password *

•••••

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



1 of 2

Create a local project

Project display name *

sonarqube

Project key *

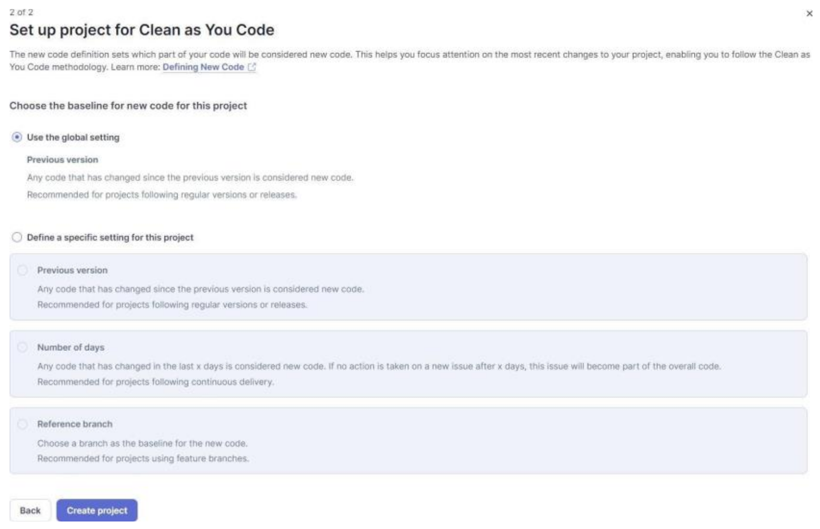
sonarqube

Main branch name *

main

The name of your project's default branch Learn More

5.


2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: Defining New Code

Choose the baseline for new code for this project

(●) Use the global setting

Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

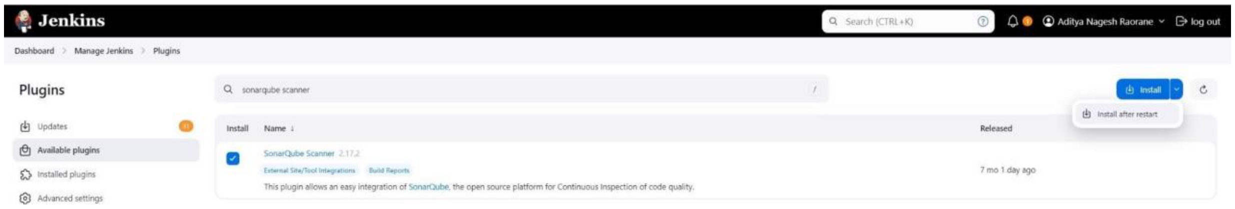
( ) Define a specific setting for this project

Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Number of days
Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

Reference branch
Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

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

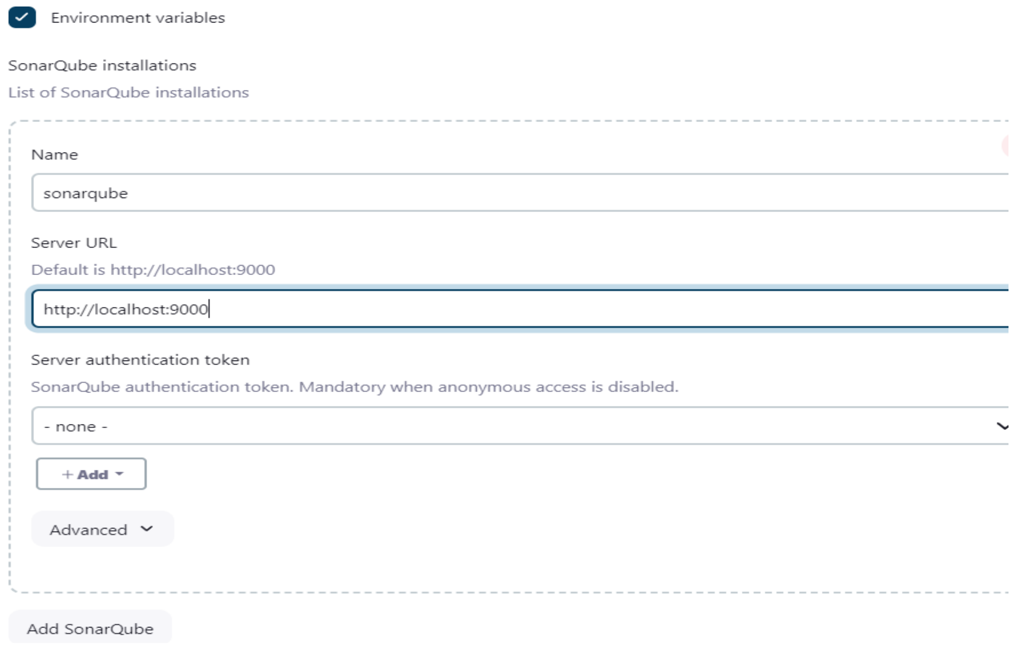


sonarqube scanner

Install	Name ↓	Released
✓	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	7 mo 1 day ago

install it.

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


✓ Environment variables

SonarQube installations
List of SonarQube installations

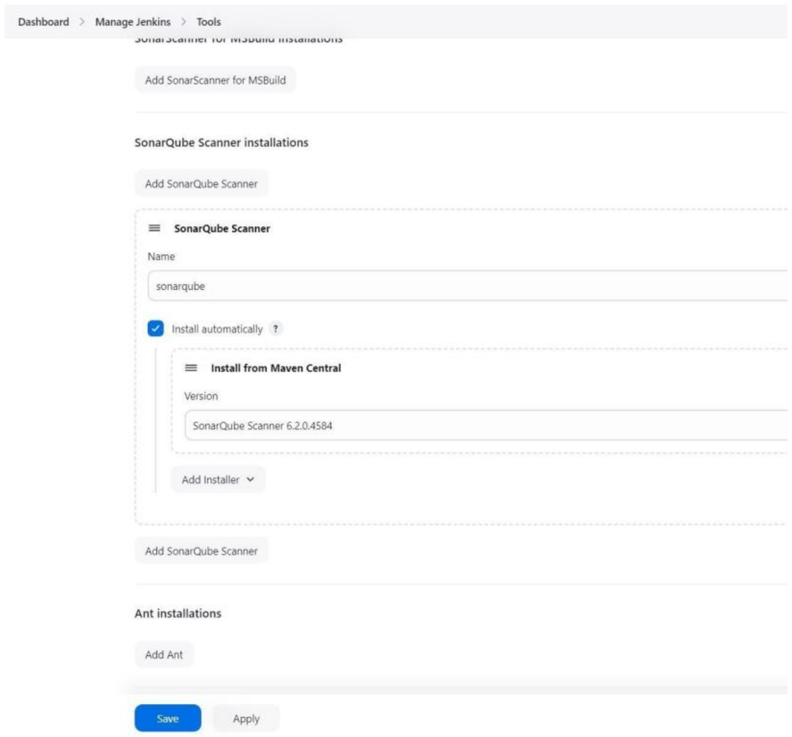
Name
sonarqube

Server URL
Default is http://localhost:9000
http://localhost:9000

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -

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

Manage Jeknins → Tools → SonarQube Scanner Installation


Dashboard > Manage Jenkins > Tools

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name: sonarqube

Install automatically ✓

Install from Maven Central

Version: SonarQube Scanner 6.2.0.4584

Add Installer

Add SonarQube Scanner

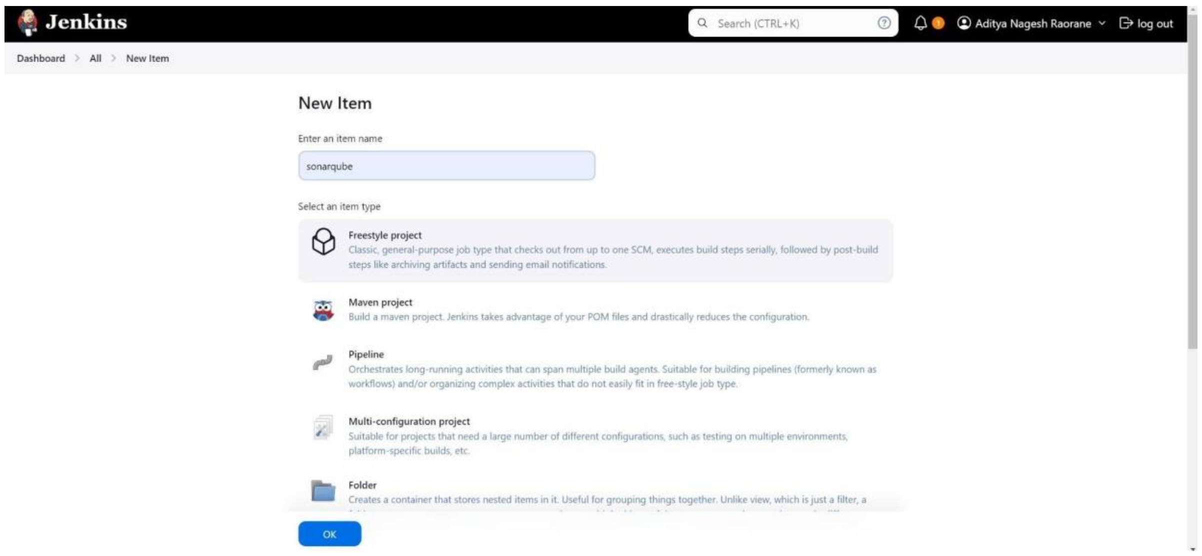
Ant installations

Add Ant

Save Apply

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

freestyle project named **sonarqube**.


Jenkins — Search (CTRL+K) — Aditya Nagesh Raorane — log out

Dashboard > All > New Item

New Item

Enter an item name: sonarqube

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

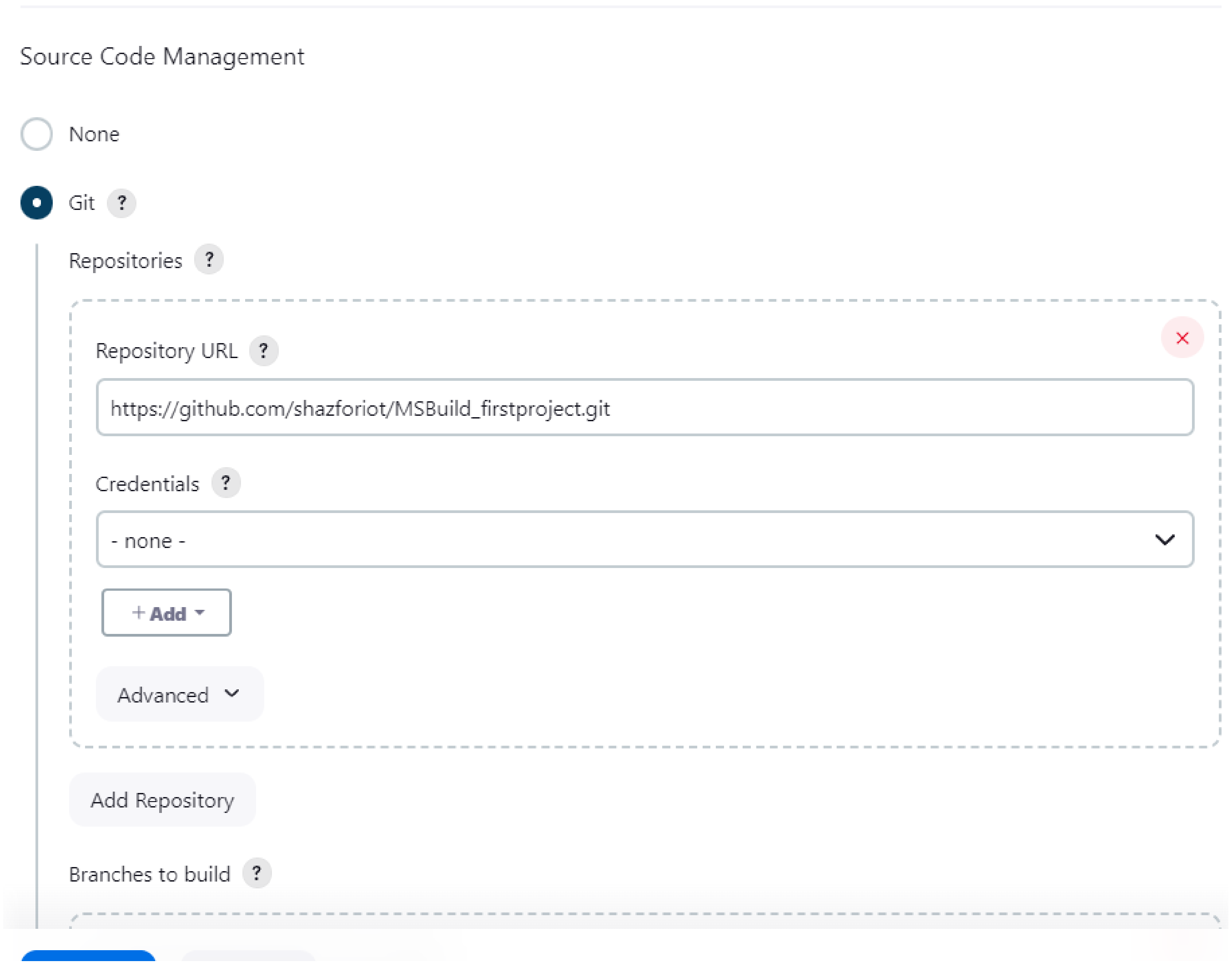
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

OK

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.


Source Code Management

None

[✓ Git]

Repositories

Repository URL
https://github.com/shazforiot/MSBuild_firstproject.git

Credentials
- none -

Branches to build

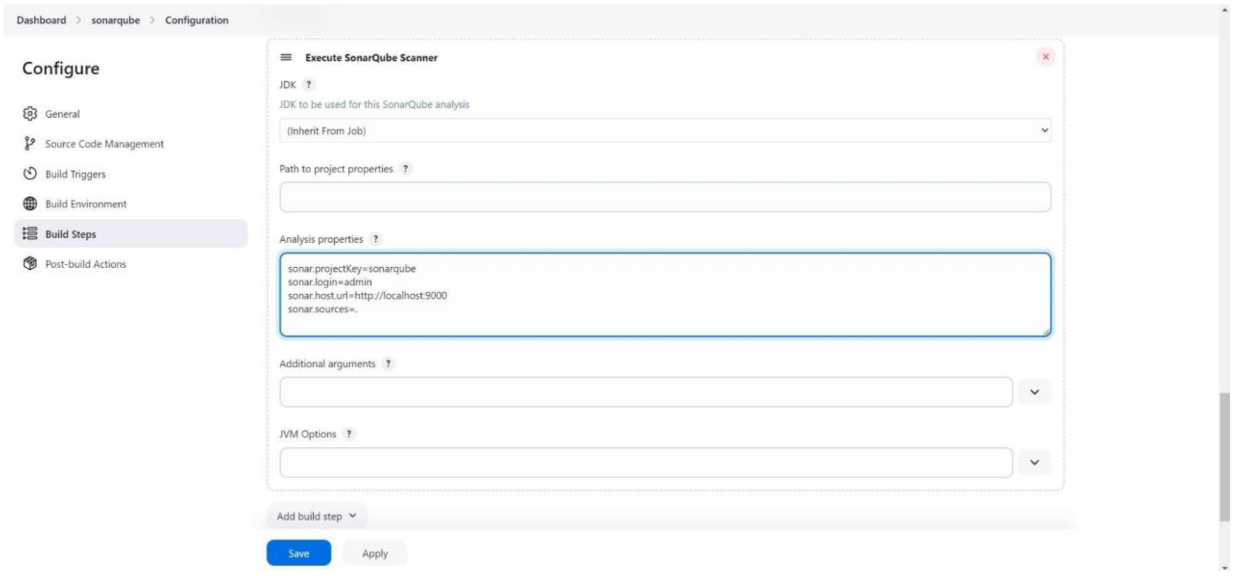
11. Under **Build-> Execute SonarQube Scanner**, enter these **Analysis Properties**. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

sonar.projectKey=sonarqube

sonar.login=admin sonar.password=pratik

sonar.sources=.

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



JDK

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties

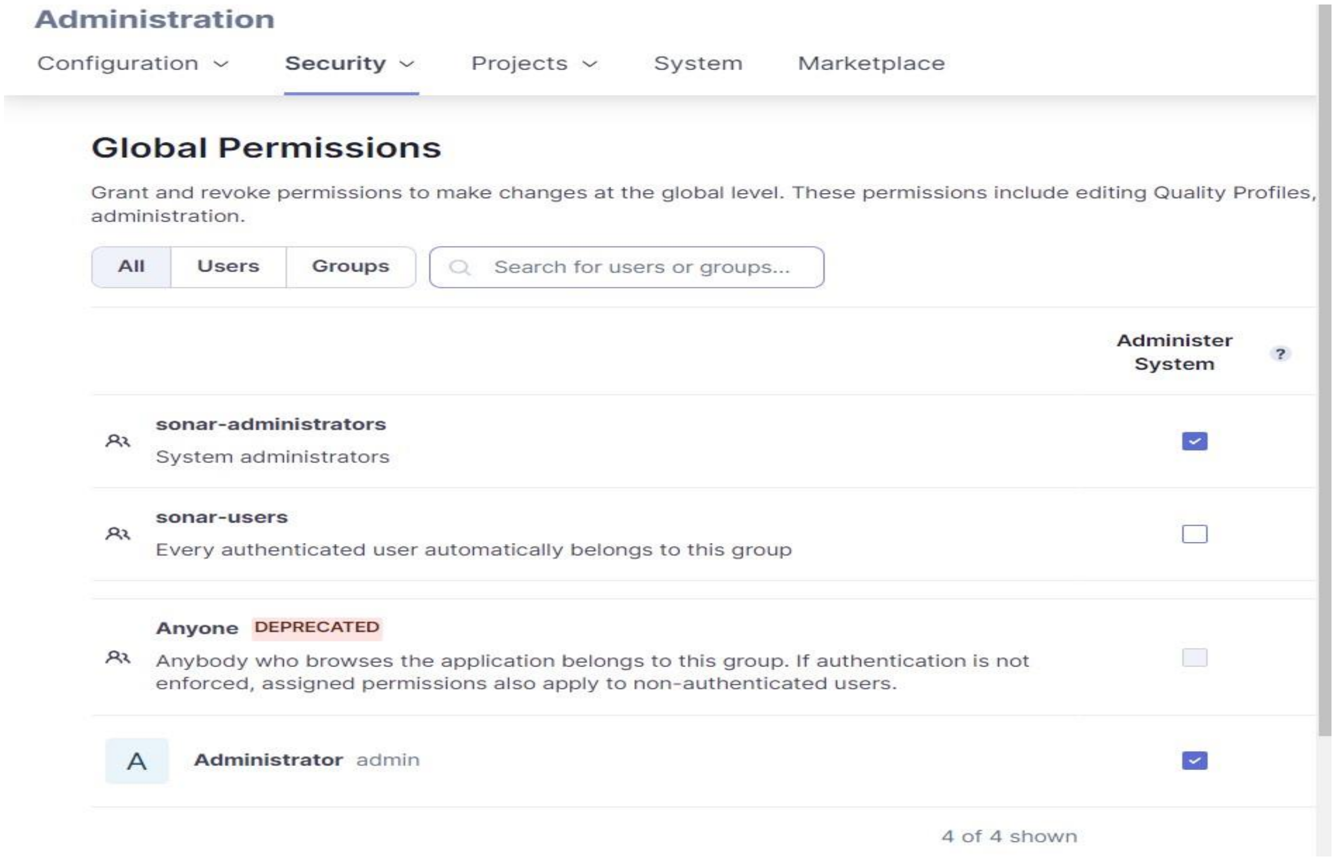
Analysis properties

```
sonar.projectKey=sonarqube
sonar.login=admin
sonar.host.url=http://localhost:9000
sonar.sources=.
```

Additional arguments

JVM Options

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



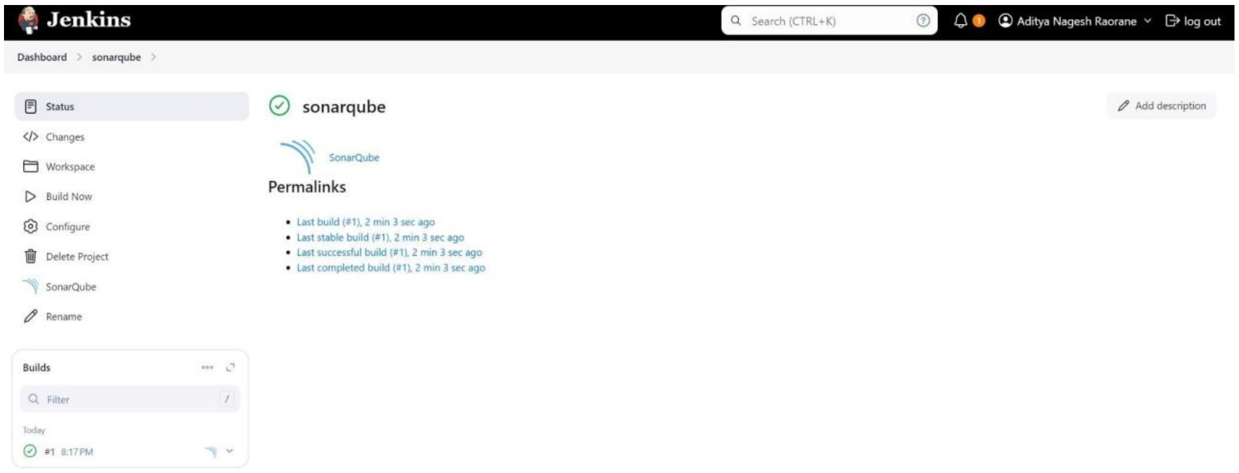
Global Permissions

Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, administration.

	Administer System
sonar-administrators System administrators	✓
sonar-users Every authenticated user automatically belongs to this group	☐
Anyone DEPRECATED Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.	☐
A Administrator admin	✓

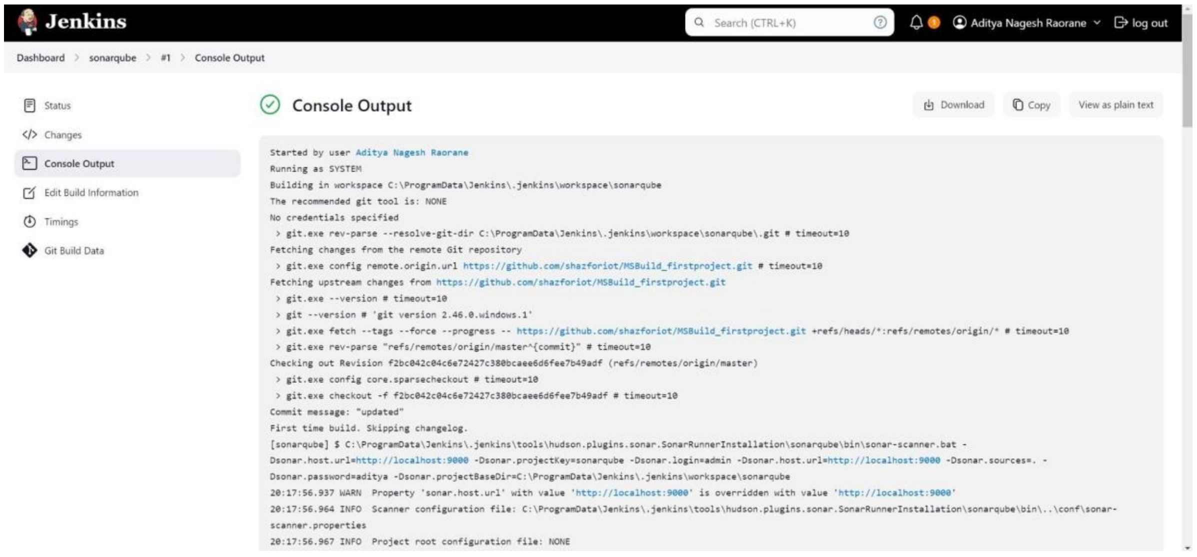
4 of 4 shown

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

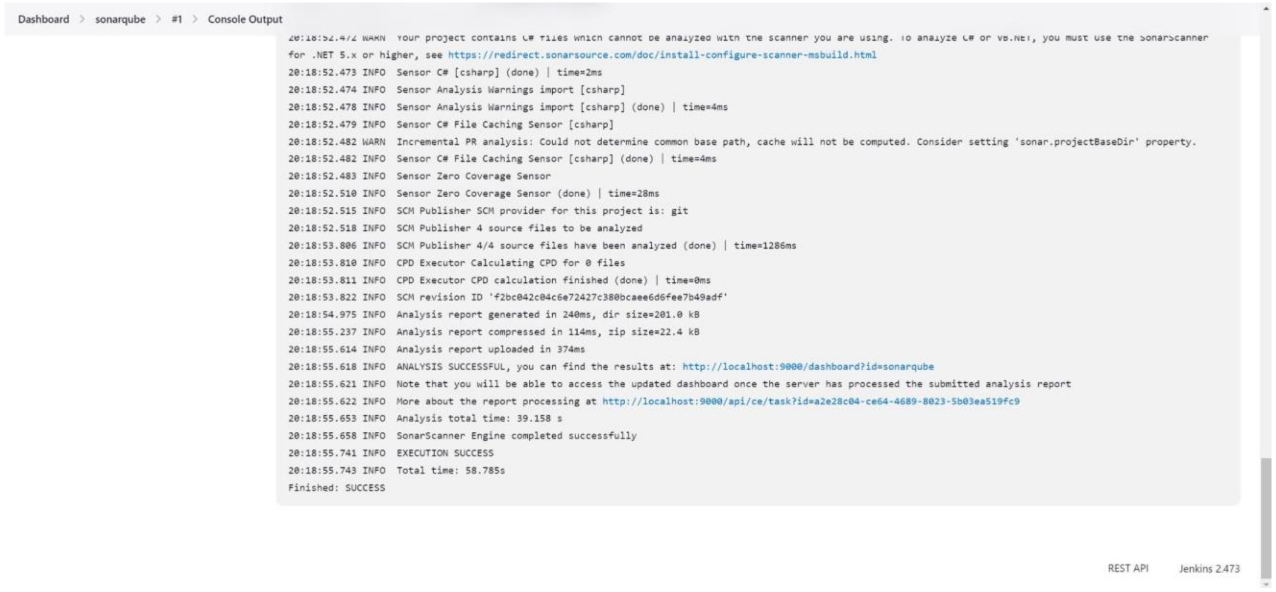


Permalinks

- Last build (#1), 2 min 3 sec ago
- Last stable build (#1), 2 min 3 sec ago
- Last successful build (#1), 2 min 3 sec ago
- Last completed build (#1), 2 min 3 sec ago

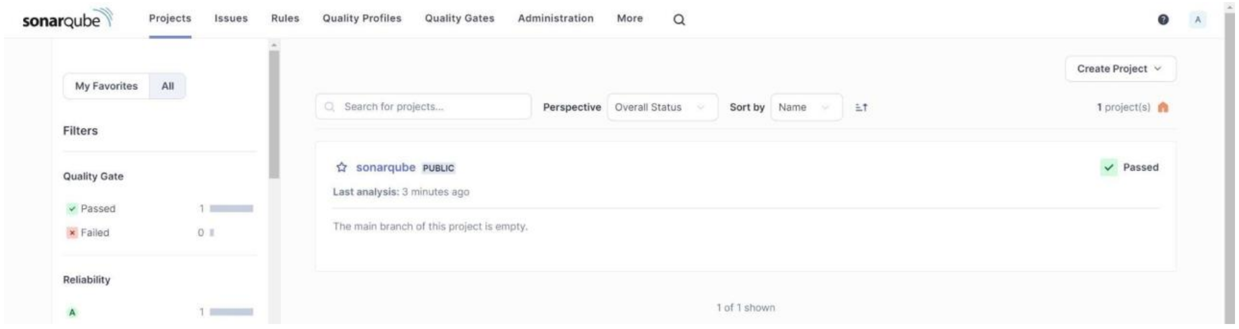


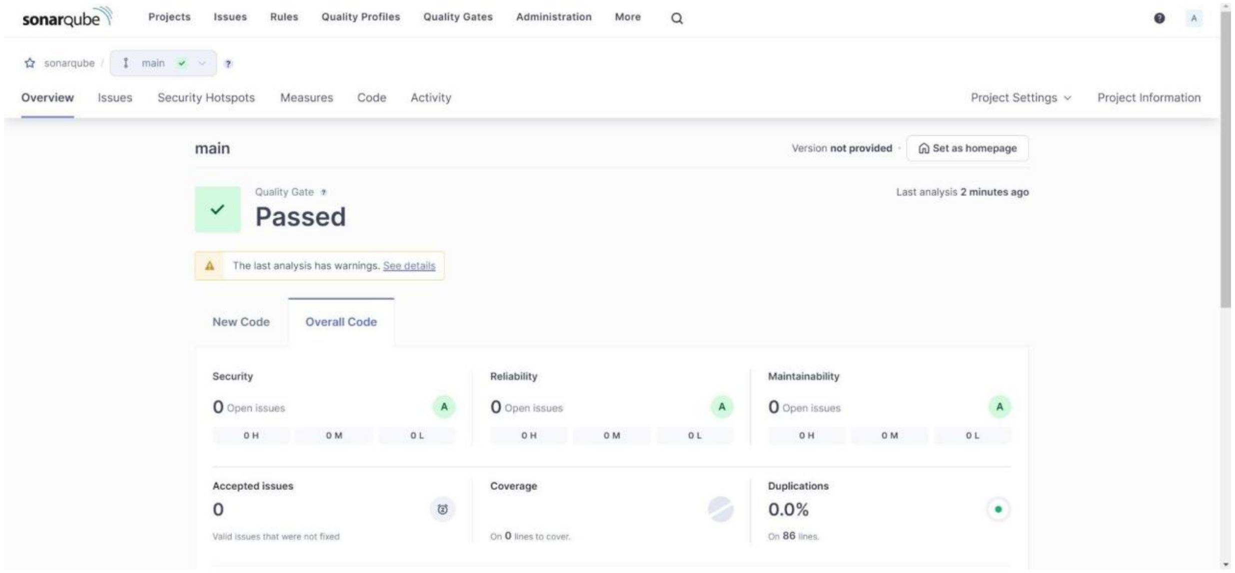
```
Started by user Aditya Nagesh Raorane
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_firstproject.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/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision f2bc042c04c6e72427c380bcaee6d6fee7b49adf (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10
Commit message: "updated"
First time build. Skipping changelog.
[sonarqube] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -
Dsonar.password=aditya -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube
20:17:56.937 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
20:17:56.964 INFO  Scanner configuration file: C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-
scanner.properties
20:17:56.967 INFO  Project root configuration file: NONE
```



```
20:18:52.472 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
20:18:52.473 INFO  Sensor C# [csharp] (done) | time=2ms
20:18:52.474 INFO  Sensor Analysis Warnings import [csharp]
20:18:52.478 INFO  Sensor Analysis Warnings import [csharp] (done) | time=4ms
20:18:52.479 INFO  Sensor C# File Caching Sensor [csharp]
20:18:52.482 WARN  Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.
20:18:52.482 INFO  Sensor C# File Caching Sensor [csharp] (done) | time=4ms
20:18:52.483 INFO  Sensor Zero Coverage Sensor
20:18:52.510 INFO  Sensor Zero Coverage Sensor (done) | time=28ms
20:18:52.515 INFO  SCM Publisher SCM provider for this project is: git
20:18:52.518 INFO  SCM Publisher 4 source files to be analyzed
20:18:53.806 INFO  SCM Publisher 4/4 source files have been analyzed (done) | time=1286ms
20:18:53.810 INFO  CPD Executor Calculating CPD for 0 files
20:18:53.811 INFO  CPD Executor CPD calculation finished (done) | time=0ms
20:18:53.822 INFO  SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
20:18:54.975 INFO  Analysis report generated in 240ms, dir size=201.0 kB
20:18:55.237 INFO  Analysis report compressed in 114ms, zip size=22.4 kB
20:18:55.614 INFO  Analysis report uploaded in 374ms
20:18:55.618 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
20:18:55.621 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:18:55.622 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=a2e28c04-ce64-4689-8023-5b03ea519fc9
20:18:55.653 INFO  Analysis total time: 39.158 s
20:18:55.658 INFO  SonarScanner Engine completed successfully
20:18:55.741 INFO  EXECUTION SUCCESS
20:18:55.743 INFO  Total time: 58.785s
Finished: SUCCESS
```

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





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.