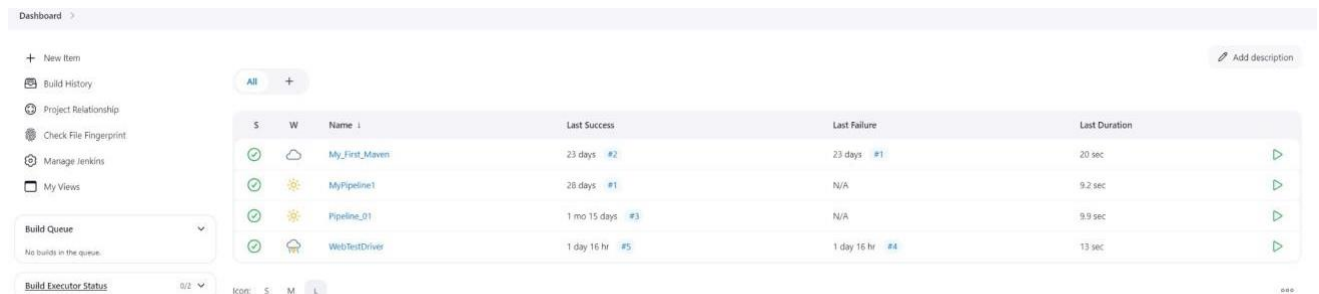


**Aim:** To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

### 1. Open up Jenkins on port 8080



### 2. In a Docker container run SonarQube using this command :

a] docker -v

b] docker pull sonarqube

c] docker run -d --name sonarqube -e

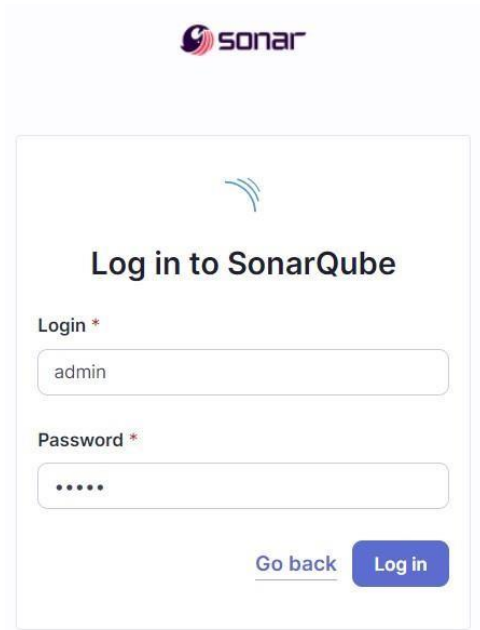
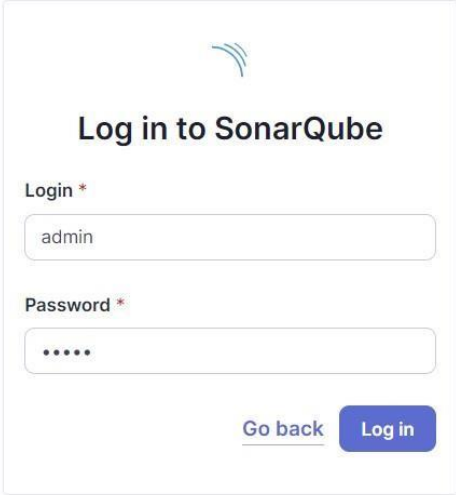
SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000

sonarqube:latest

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

C:\Users\adity>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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecd
Status: Downloaded newer image for sonarqube:latest
4a6e73f4472de892b1ddead1abe77372a85a7b09408cce3a0abd37c5ab6b49a4
```

### 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 **“shweta”**.

**Log in to SonarQube**

Login \*

admin

Password \*

.....

[Go back](#) [Log in](#)

#### 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](#)

Cancel

Next

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.

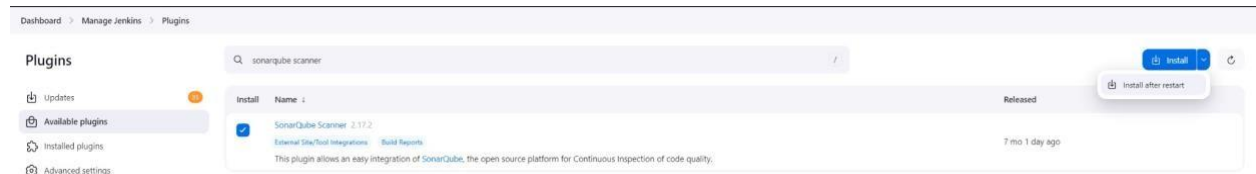
☐ Reference branch

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

Back

Create project

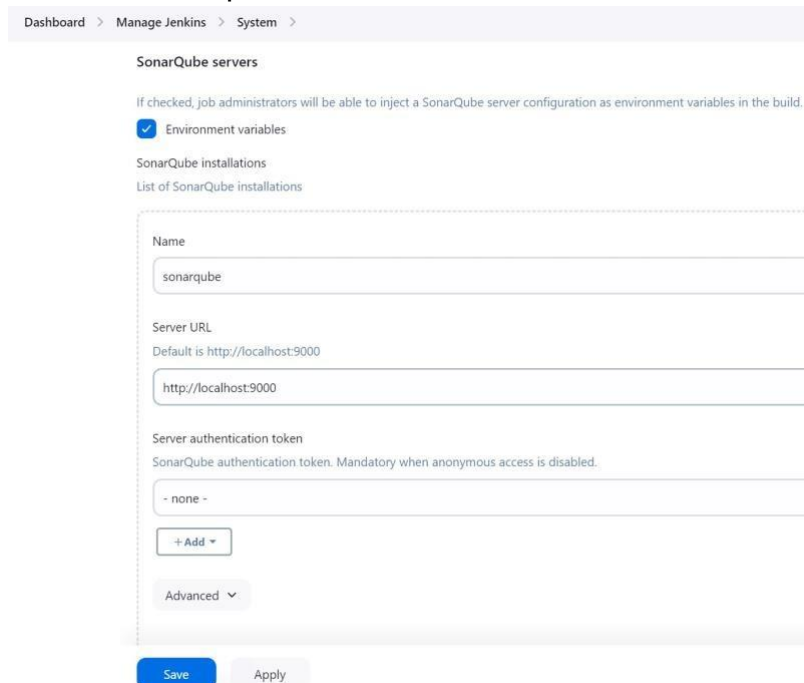
5. Setup the project and come back to Jenkins Dashboard. Go to **Manage Jenkins** → **Plugins** and search for **SonarQube Scanner** in **Available Plugins** and install it.



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**

The screenshot shows the Jenkins configuration page for 'Tools'. The breadcrumb trail is 'Dashboard > Manage Jenkins > Tools'. Under 'SonarQube Scanner installations', there is a button 'Add SonarQube Scanner'. Below it, a form for 'SonarQube Scanner' is shown with the following fields: 'Name' (sonarqube), 'Install automatically' (checked), and 'Install from Maven Central' (expanded) with 'Version' (SonarQube Scanner 6.2.0.4584). There is an 'Add Installer' dropdown button. At the bottom, there is a 'Save' button and an 'Apply' button.

8. After the configuration, create a **New Item** in Jenkins, choose a **freestyle project** named **sonarqube**.

The screenshot shows the 'New Item' dialog in Jenkins. The 'Enter an item name' field contains 'sonarqube'. Under 'Select an item type', the 'Freestyle project' option is selected. The other options are 'Maven project', 'Pipeline', 'Multi-configuration project', and 'Folder'. The 'OK' button is at the bottom.

9. Choose this GitHub repository in **Source Code Management**.

[https://github.com/shazforiot/MSBuild\\_firstproject.git](https://github.com/shazforiot/MSBuild_firstproject.git)

It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

The screenshot shows the SonarQube Configuration page, specifically the Source Code Management section. The left sidebar lists various configuration categories: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main content area is titled 'Source Code Management' and includes a radio button for 'None' and a selected radio button for 'Git'. Below this, there is a 'Repositories' section with a text input for 'Repository URL' containing 'https://github.com/shazforiot/MSBuild\_firstproject.git'. There is also a 'Credentials' dropdown menu set to '- none -' and an '+ Add' button. An 'Advanced' dropdown is visible. At the bottom of the configuration area, there are 'Add Repository' and 'Branches to build' sections. The page concludes with 'Save' and 'Apply' buttons.

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=shweta

sonar.sources=.

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

The screenshot shows the SonarQube Configuration page, specifically the Build Steps section. The left sidebar lists various configuration categories: General, Source Code Management, Build Triggers, Build Environment, Build Steps (selected), and Post-build Actions. The main content area is titled 'Execute SonarQube Scanner' and includes a 'JDK' dropdown menu set to '(Inherit From Job)'. Below this is a 'Path to project properties' text input. The 'Analysis properties' section contains a text area with the following properties: sonar.projectKey=sonarqube, sonar.login=admin, sonar.host.url=http://localhost:9000, and sonar.sources=.. Below the analysis properties, there are 'Additional arguments' and 'JVM Options' dropdown menus. At the bottom of the configuration area, there is an 'Add build step' button. The page concludes with 'Save' and 'Apply' buttons.

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

The screenshot shows the SonarQube Administration interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The 'Administration' section is active, with sub-tabs for 'Configuration', 'Security', 'Projects', 'System', and 'Marketplace'. The 'Global Permissions' page is displayed, with a search bar and tabs for 'All', 'Users', and 'Groups'. A table lists permissions for various groups:

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
<b>sonar-administrators</b> System administrators	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<b>sonar-users</b> Every authenticated user automatically belongs to this group	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<b>Anyone</b> <span>DEPRECATED</span> Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects
<b>Administrator</b> admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects

4 of 4 shown

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

The screenshot shows the SonarQube Dashboard. The top navigation bar includes 'Dashboard', 'MyPipeline1', and 'Configuration'. The 'Configuration' page is active, showing the 'Builds' section. The 'Builds' section displays a list of builds with a filter bar and a 'Filter' button. The 'Builds' list shows a single build with a status of 'Success' and a time of '8:17PM'. The 'Permalinks' section lists the following links:

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

Dashboard > sonarqube > #1 > Console Output

Status

Changes

Console Output

Edit Build Information

Timings

Git Build Data

Console Output

Started by user Shweta Wadhwa

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 f2bc842c84c6e72427c380bc6eed6d6fee7b49adf (refs/remotes/origin/master)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f f2bc842c84c6e72427c380bc6eed6d6fee7b49adf # 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.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

20:18:34.474 WARN Your project contains LW files which cannot be analyzed with the scanner you are using. To analyze LW or VS.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 8 files

20:18:53.811 INFO CPD Executor CPD calculation finished (done) | time=0ms

20:18:53.822 INFO SCM revision ID 'f2bc842c84c6e72427c380bc6eed6d6fee7b49adf'

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-ce64-4689-8823-5b03ea519fc9>

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

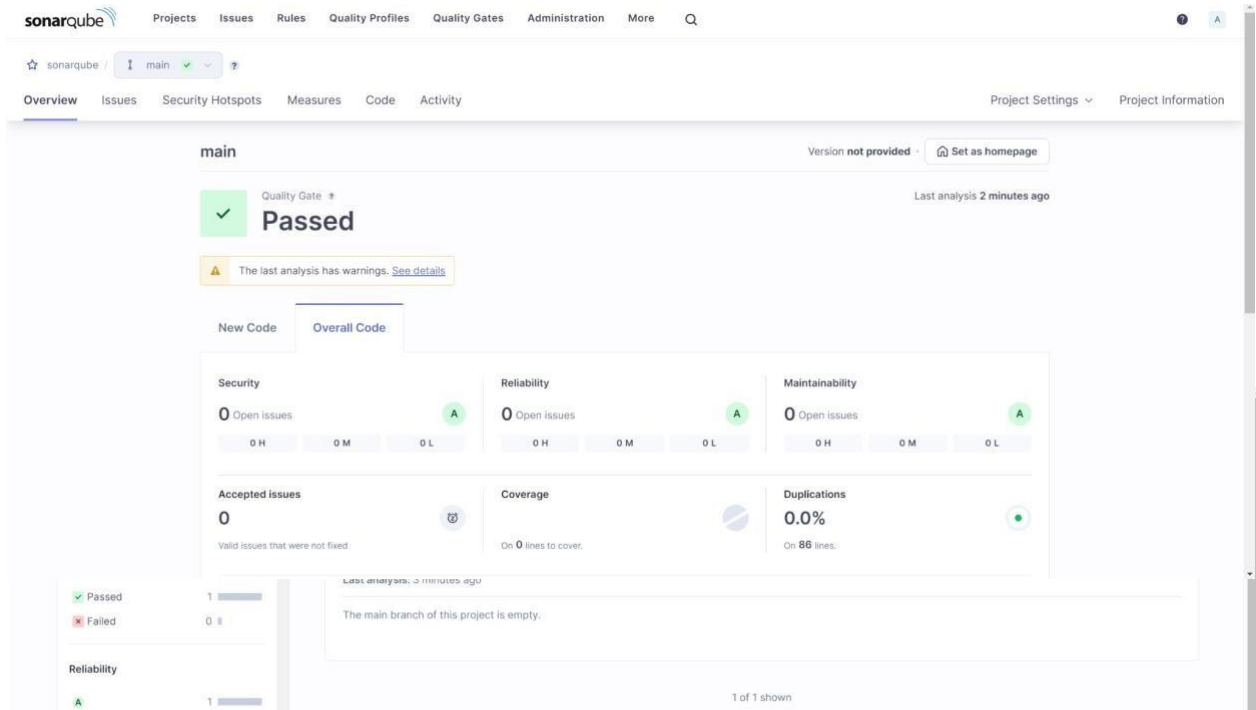
20:18:55.743 INFO Total time: 58.785s

Finished: SUCCESS

Dashboard > sonarqube > #1 > Console Output

REST API Jenkins 2.473

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



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

## **Conclusion:**

In this experiment, we have understood the importance of SAST and have successfully integrated Jenkins with SonarQube for Static Analysis and Code Testing.