

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

Prerequisites:

1) Docker

Run docker -v command.

Use this command to check if docker is installed and running on your system.

```
C:\Users\bhumi>docker -v
Docker version 27.2.0, build 3ab4256
```

2) Install SonarQube image Command:

docker pull sonarqube

This command helps you to install an image of SonarQube that can be used on the local system without actually installing the SonarQube installer.

```
C:\Users\bhumi>docker pull sonarqube
Using default tag: latest
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
docker.io/library/sonarqube:latest
```

3) Keep jenkins installed on your system.

Experiment Steps:

1. Run SonarQube in a Docker container using this command -

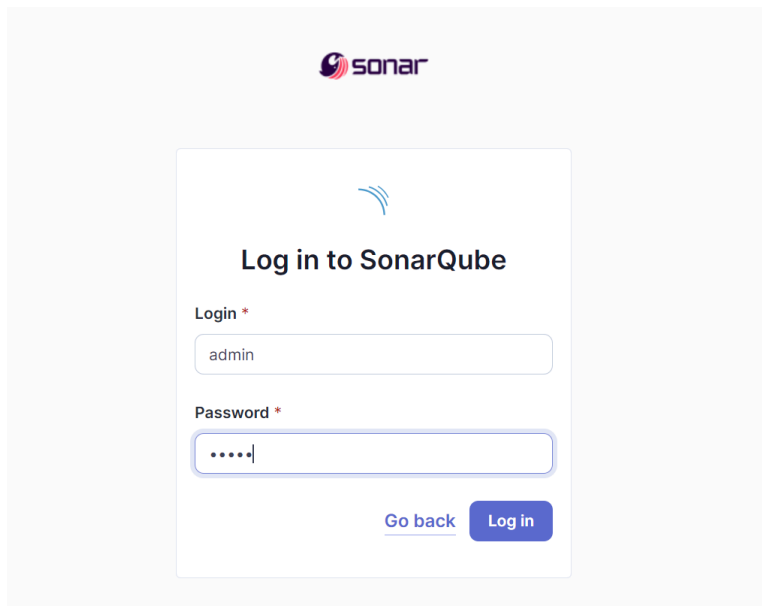
docker run -d --name sonarqube -e

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

```
C:\Users\bhumi>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
7401befce9e7a7248c0e5648a2913e99c3843cce08e91d403e5af3a1479a151e
```

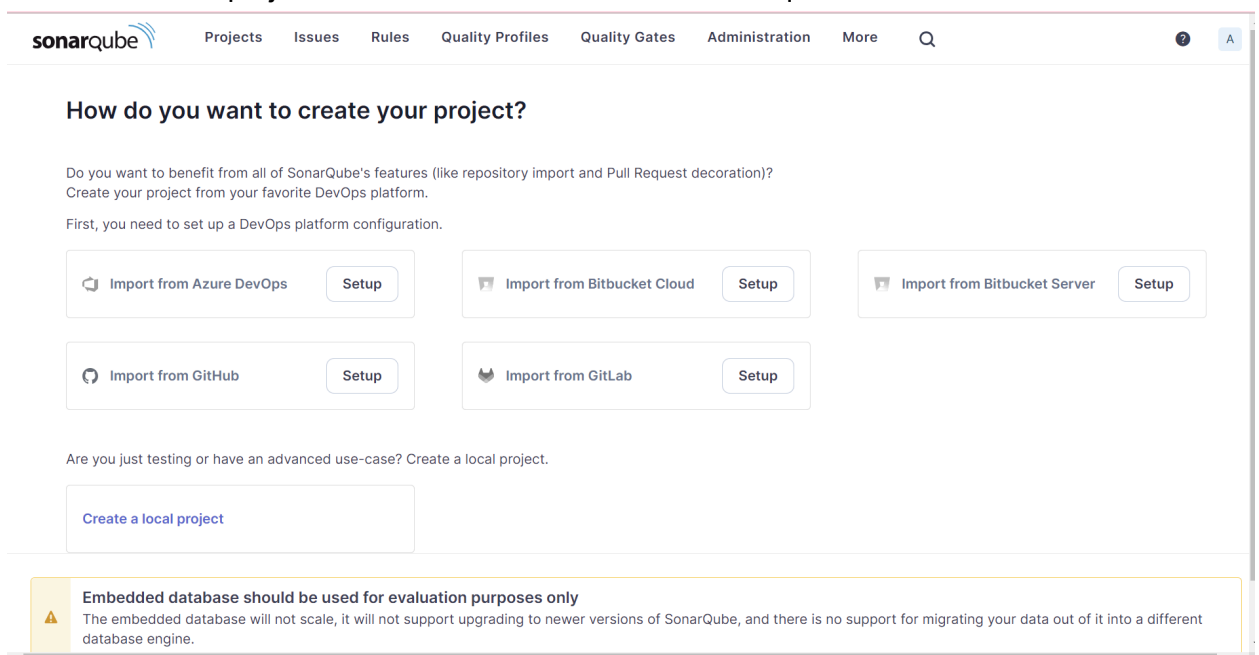
2. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.

3. Login to SonarQube using username admin and password admin.



The image shows the SonarQube login page. At the top is the Sonar logo. Below it is a white box with the title "Log in to SonarQube". Inside the box, there are two input fields: "Login \*" with the value "admin" and "Password \*" with masked characters ".....". Below the password field are two buttons: "Go back" (a link) and "Log in" (a blue button).

4. Create a manual project in SonarQube with the name sonarqube.




The image shows the SonarQube project creation page. The header includes the SonarQube logo and navigation links: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main heading is "How do you want to create your project?". Below this, there is a paragraph: "Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration." There are five buttons arranged in two rows: "Import from Azure DevOps Setup", "Import from Bitbucket Cloud Setup", "Import from Bitbucket Server Setup", "Import from GitHub Setup", and "Import from GitLab Setup". Below these buttons is a link: "Create a local project". At the bottom, there is a yellow warning box with a triangle icon and the text: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine."

Click on Create a Local Project.

Here, I have given the name 'sonarqube'. This will be your project key as well. We'll use it later.

Keep the branch main only and click on next.

 [Projects](#) [Issues](#) [Rules](#) [Quality Profiles](#) [Quality](#)

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

Set up the project as required and click on create.

 [Projects](#) [Issues](#) [Rules](#) [Quality Profiles](#) [Quality Gates](#) [Administration](#) [More](#) [Q](#)

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. Open Jenkins on whichever port it is installed. (<http://localhost:8080>).

The screenshot shows the Jenkins Dashboard. On the left, there are links for 'New Item', 'Build History', and 'Manage Jenkins'. Below these are two panels: 'Build Queue' (empty) and 'Build Executor Status' (showing 2 idle executors and one offline slave named 'slave1'). The main area displays a table of recent builds:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	bhumi	27 days #2	N/A	2.7 sec
✓	☀	boompipeline	27 days #1	N/A	4.8 sec
✓	☁	web	27 days #7	27 days #6	0.36 sec

Below the table, there is a 'Icon' section with 'S', 'M', and 'L' buttons. At the bottom right, it says 'REST API' and 'Jenkins 2.462.1'.

6. Go to Manage jenkins → Plugins → Available Plugins  
Search for Sonarqube Scanner for Jenkins and install it.

The screenshot shows the Jenkins 'Manage Jenkins' → 'Plugins' page. The search bar contains 'sonarqube'. The results show three plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	<b>SonarQube Scanner</b> 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 9 days ago
<input type="checkbox"/>	<b>Sonar Gerrit</b> 388.v9b_f1cb_e42306 External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly.	3 mo 23 days ago
<input type="checkbox"/>	<b>SonarQube Generic Coverage</b> 1.0 TODO	5 yr 2 mo ago

7. Under Jenkins 'Configure System', look for SonarQube Servers and enter the details.  
I have named the server 'sonarqube' and added the server url for jenkins.  
Enter the Server Authentication token if needed.

Dashboard > Manage Jenkins > System >

### SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

#### SonarQube installations

List of SonarQube installations

**Name**

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

**Server authentication token**  
SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

Advanced

Save

Apply

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

### SonarQube Scanner installations

Add SonarQube Scanner

☰

**SonarQube Scanner**

×

**Name**

☒ Install automatically ?

☰

**Install from Maven Central**

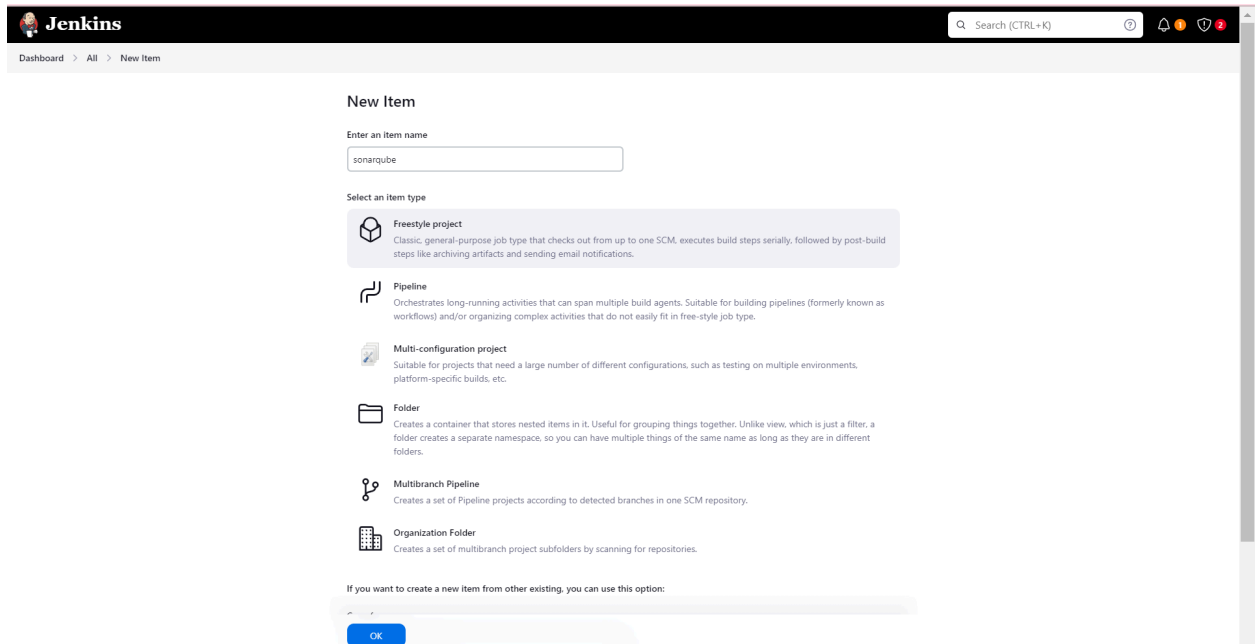
×

**Version**

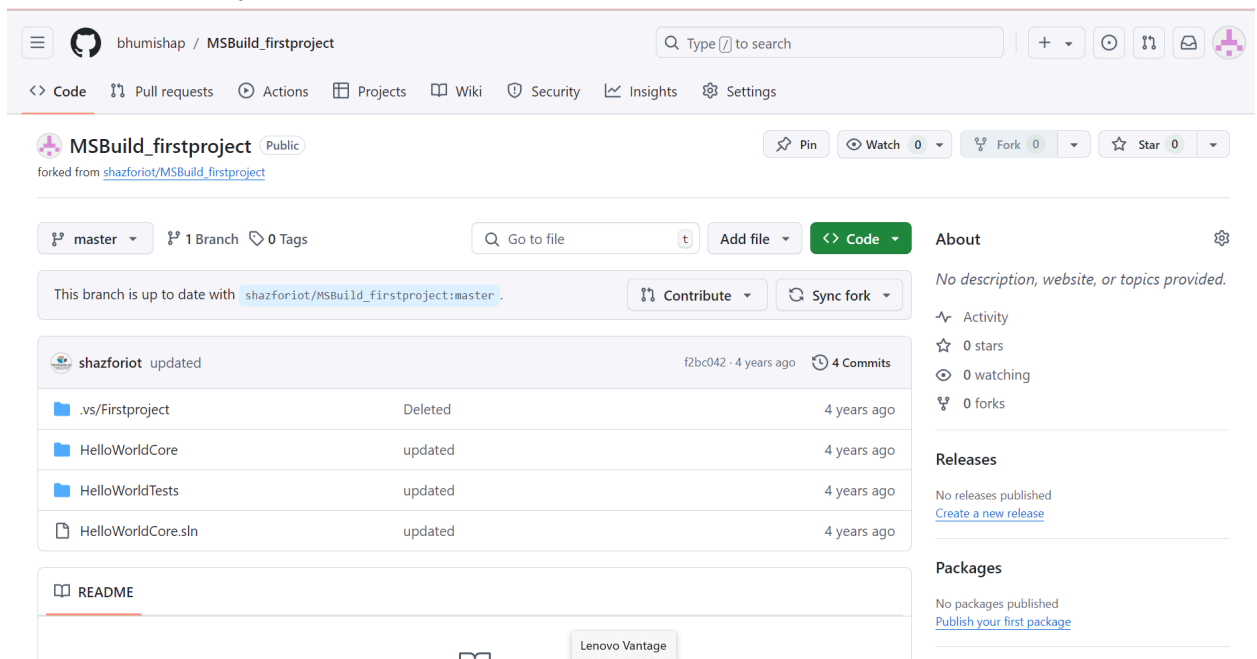
Sonarqube Scanner 6.1.0.4477

Add Installer

9. After configuration, create a New Item → choose a freestyle project and name your project. Here, I have given the 'sonarqube', then click on OK.



10. Choose the github repository [https://github.com/shazforiot/MSBuild\\_firstproject](https://github.com/shazforiot/MSBuild_firstproject). It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.  
Fork this repository.



## 11. Enable git and add the repository you forked.

Dashboard > bhumisha > Configuration

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

#### Repositories ?

Repository URL ?

https://github.com/bhumishap/MSBuild\_firstproject

Credentials ?

- none -

+ Add

Advanced

Add Repository

#### Branches to build ?

Branch Specifier (blank for 'any') ?

## 12. Under Build Steps, select Execute Sonarqube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Dashboard > bhumisha > Configuration

### Build Steps

#### Execute SonarQube Scanner

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.password=password
sonar.host.url=http://localhost:9000
sonar.sources=.
```

Additional arguments ?

Save Apply

13. Then click on save. Go to [http://localhost:9000/<user\\_name>/permissions](http://localhost:9000/<user_name>/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, executing analysis, and performing global system administration.

All	Users	Groups	Search for users or 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 type="checkbox"/> Projects

14. Go back to Jenkins. Go to the job you had just built and click on Build Now.

Dashboard > sonarqube >

Status
 Changes
 Workspace
 Build Now
 Configure
 Delete Project
 SonarQube
 Rename

**sonarqube**

### Permalinks

- [Last build \(#1\), 2 hr 53 min ago](#)
- [Last stable build \(#1\), 2 hr 53 min ago](#)
- [Last successful build \(#1\), 2 hr 53 min ago](#)
- [Last completed build \(#1\), 2 hr 53 min ago](#)

Build History trend ▾

15. Check the console Output.



Dashboard > sonarqube > #1 > Console Output

Status  
Changes  
Console Output  
Set Build Information  
Delete build #1  
Timings  
Git Build Data

Console Output

Download Copy View as plain text

```
Started by user unknown or anonymous
Running as 381016
Building on the built-in node 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=30
Fetching changes from the remote git repository
> git.exe config remote.origin.url https://github.com/Bhumisha/HBBuild_FirstProject # timeout=30
Fetching upstream changes from https://github.com/Bhumisha/HBBuild_FirstProject
> git.exe --version # timeout=30
> git --version # 'git version 2.46.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/Bhumisha/HBBuild_FirstProject refs/heads/*:refs/remotes/origin/* # timeout=30
> git.exe rev-parse --refs/remotes/origin/master (commit) # timeout=30
Checking out revision f2b28d208d6e7d27c380c8e0d8f6e7b0d0f (refs/remotes/origin/master)
> git.exe config core.sshCommand # timeout=30
> git.exe checkout -f f2b28d208d6e7d27c380c8e0d8f6e7b0d0f # timeout=30
Commit message: "update"
First time build. Skipping changelog.
Unpacking https://repl.maven.org/default/org/sonarsource/scanner/cli/sonar-scanner-cli/6.1.0.4677/sonar-scanner-cli-6.1.0.4677.zip to C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\sonar-scanner
[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=.
23:05:55.587 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
23:05:55.589 INFO Scanner configuration file: C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\conf\sonar-scanner.properties
23:05:55.600 INFO SonarScanner cli 6.1.0.4677
23:05:55.612 INFO Java 21 Oracle Corporation (64-bit)
23:05:55.622 INFO Windows 11 10.0 amd64
23:05:55.648 INFO User cache: C:\Windows\system32\config\systemprofile\jenkins\sonar\cache
23:05:56.923 INFO JRE provisioning: os[windows], arch[amd64]
23:05:04.216 INFO Communicating with SonarQube Server 10.6.0.92316
23:05:05.896 INFO Starting SonarScanner Engine.
23:05:05.896 INFO Java 17.0.13 Eclipse Adoptium (64-bit)
23:05:06.839 INFO Load global settings
23:05:07.202 INFO Load global settings (done) | time=300ms
23:05:07.226 INFO Server id: 16794118-A21VrQtcn308VC3js
23:05:07.242 INFO Loading required plugins
23:05:07.242 INFO Load plugin index
23:05:07.516 INFO Load plugins index (done) | time=292ms
23:05:07.534 INFO Load/download plugins
23:05:56.792 INFO Load/download plugin (done) | time=1248ms
23:05:11.812 INFO Process project properties
23:05:11.837 INFO Process project properties (done) | time=20ms
23:05:11.886 INFO Project key: sonarqube
23:05:11.886 INFO Base dir: C:\ProgramData\Jenkins\jenkins\workspace\sonarqube
23:05:11.886 INFO Working dir: C:\ProgramData\Jenkins\jenkins\workspace\sonarqube\scannerwork
23:05:11.942 INFO Load project settings for component key: 'sonarqube'
23:05:12.106 INFO Load project settings for component key: 'sonarqube' (done) | time=145ms
23:05:12.436 INFO Load quality profiles
23:05:12.904 INFO Load quality profiles (done) | time=488ms
23:05:12.933 INFO Auto-configuring with CI 'Jenkins'
23:05:40.280 INFO Sensor Iac CloudFormation Sensor [Iac]
23:05:40.418 INFO 0 source files to be analyzed
23:05:40.467 INFO 0 source files have been analyzed
23:05:40.467 INFO Sensor Iac CloudFormation Sensor [Iac] (done) | time=106ms
23:05:40.467 INFO Sensor Iac AzureResourceManager Sensor [Iac]
23:05:40.484 INFO 0 source files to be analyzed
23:05:40.712 INFO 0 source files have been analyzed
23:05:40.712 INFO Sensor Iac AzureResourceManager Sensor [Iac] (done) | time=246ms
23:05:40.712 INFO Sensor Java Config Sensor [Iac]
23:05:40.728 INFO 0 source files to be analyzed
23:05:40.736 INFO 0 source files have been analyzed
23:05:40.736 INFO Sensor Java Config Sensor [Iac] (done) | time=24ms
23:05:40.736 INFO Sensor Iac Docker Sensor [Iac]
23:05:40.744 INFO 0 source files to be analyzed
23:05:40.819 INFO 0 source files have been analyzed
23:05:40.819 INFO Sensor Iac Docker Sensor [Iac] (done) | time=83ms
23:05:40.819 INFO Sensor TextAndSecretsSensor [Text]
23:05:40.819 INFO Available preprocessors: 11
23:05:40.819 INFO Using 12 threads for analysis.
23:05:42.014 INFO The property "sonar.tests" is not set. To improve the analysis accuracy, we categorize a file as a test file if any of the following is true:
  * The filename starts with "test"
  * The filename contains "test." or "tests."
  * Any directory in the file path is named: "doc", "docs", "test" or "tests"
  * Any directory in the file path has a name ending in "test" or "tests"
23:05:42.263 INFO Using git CLI to retrieve untracked files
23:05:42.303 INFO Analyzing language associated files and files included via "sonar.test.inclusions" that are tracked by git
23:05:42.531 INFO 14 source files to be analyzed
23:05:43.087 INFO 14/14 source files have been analyzed
23:05:43.088 INFO Sensor TextAndSecretsSensor [Text] (done) | time=2209ms
23:05:43.112 INFO Run sensors on project
23:05:43.421 INFO Sensor Cx [csharp]
23:05:43.421 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
23:05:43.429 INFO Sensor Cx [csharp] (done) | time=9ms
23:05:43.429 INFO Sensor Analysis Warnings Import [csharp]
23:05:43.429 INFO Sensor Analysis Warnings Import [csharp] (done) | time=8ms
23:05:43.429 INFO Sensor Cx File Caching Sensor [csharp]
23:05:43.429 INFO Sensor Cx File Caching Sensor [csharp] (done) | time=8ms
23:05:43.429 INFO Sensor Zero Coverage Sensor
23:05:43.446 INFO Sensor Zero Coverage Sensor (done) | time=17ms
23:05:43.454 INFO SCM Publisher SCM provider for this project is: git
23:05:43.454 INFO SCM Publisher 4 source files to be analyzed
23:05:44.112 INFO SCM Publisher 4/4 source files have been analyzed (done) | time=658ms
23:05:44.119 INFO CPD Executor Calculating CPD for 0 files
23:05:44.119 INFO CPD Executor CPD calculation finished (done) | time=8ms
23:05:44.127 INFO SCM Revision ID "f2b28d208d6e7d27c380c8e0d8f6e7b0d0f"
23:05:44.623 INFO Analysis report generated in 2036s, dir size=201.9 KB
23:05:44.710 INFO Analysis report compressed in 66ms, zip size=22.3 KB
23:05:45.062 INFO Analysis report uploaded in 348ms
23:05:45.067 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
23:05:45.067 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
23:05:45.068 INFO More about the report processing at http://localhost:9000/api/cx/task?id=1b3b1b0ef-8e11-4295-5286-83a74f4e6412
23:05:45.084 INFO Analysis total time: 34.059 s
23:05:45.084 INFO SonarScanner Engine completed successfully
23:05:45.165 INFO EXECUTION SUCCESS
23:05:45.165 INFO Total time: 49.576s
Execution success
```

16. Once the build is complete, go back to SonarQube and check the project linked.

The image displays two screenshots. The top screenshot is the SonarQube web interface for a project named 'main'. It shows a 'Passed' status for the Quality Gate, with a warning that the last analysis has warnings. The 'Overall Code' tab is selected, showing metrics for Security (0 Open Issues), Reliability (0 Open Issues), Maintainability (0 Open Issues), Accepted issues (0), Coverage (0.0%), Duplications (0.0%), and Security Hotspots (0). The bottom screenshot is the Jenkins dashboard for the 'sonarqube' job. It shows the job status as 'Completed' and provides a list of build history links, including 'Last build (#1), 2 hr 57 min ago', 'Last stable build (#1), 2 hr 57 min ago', 'Last successful build (#1), 2 hr 57 min ago', and 'Last completed build (#1), 2 hr 57 min ago'. A 'Build History' widget is also visible, showing a filter and a list of builds.

### Conclusion:

In this experiment, we explored how to perform Static Application Security Testing (SAST) in Jenkins using SonarQube. Instead of installing SonarQube directly on the local system, we utilized a Docker image for a more streamlined setup. After configuring Jenkins with the necessary SonarQube settings, we integrated it with a GitHub repository containing code for analysis. Upon building the project, SonarQube successfully scanned the code, confirming that no errors were present. This process demonstrated how Jenkins and SonarQube can be effectively used together for automated code quality and security analysis.