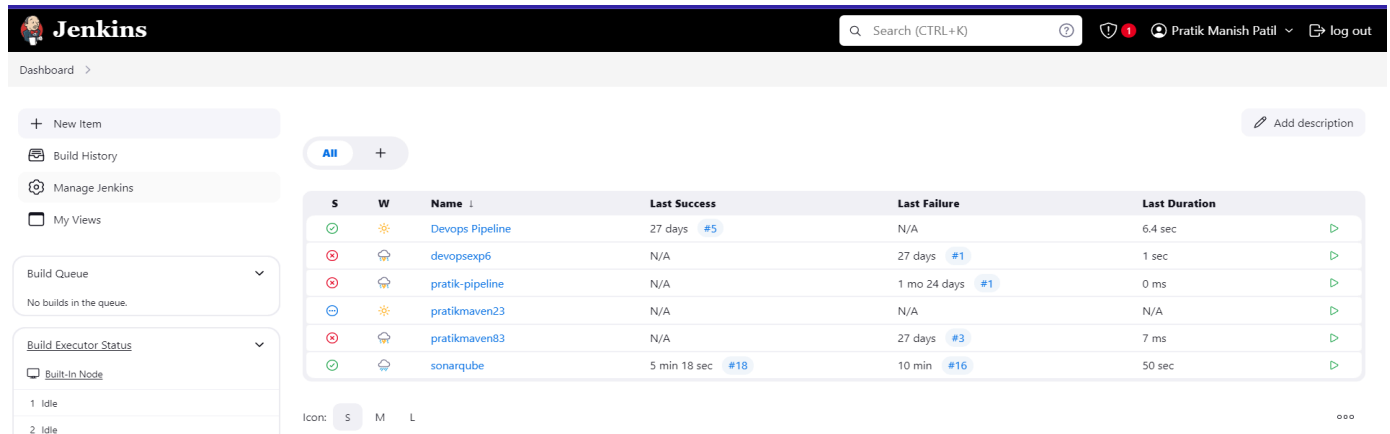


Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.



The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links: New Item, Build History, Manage Jenkins, and My Views. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status (2 Idle). The main area displays a table of builds with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists several builds, including 'Devops Pipeline', 'devopsexp6', 'pratik-pipeline', 'pratikmaven23', 'pratikmaven83', and 'sonarqube'. The 'sonarqube' build is highlighted with a green status icon and shows a last success of 5 min 18 sec and a last failure of 10 min.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	Devops Pipeline	27 days #5	N/A	6.4 sec
✗	☀	devopsexp6	N/A	27 days #1	1 sec
✗	☀	pratik-pipeline	N/A	1 mo 24 days #1	0 ms
✗	☀	pratikmaven23	N/A	N/A	N/A
✗	☀	pratikmaven83	N/A	27 days #3	7 ms
✓	☀	sonarqube	5 min 18 sec #18	10 min #16	50 sec

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

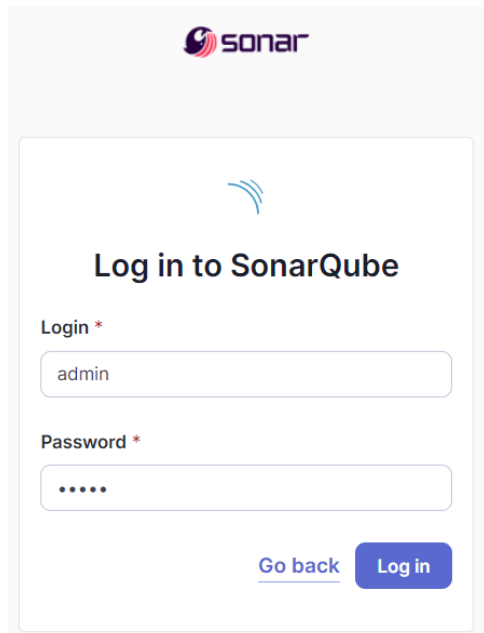
SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p

```
PS C:\Users\91799> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest b9b2d50ed5646fd55c47c128523b62b98b9eb2943ce0f0a969bb66e40fdf0b42
```

9000:9000

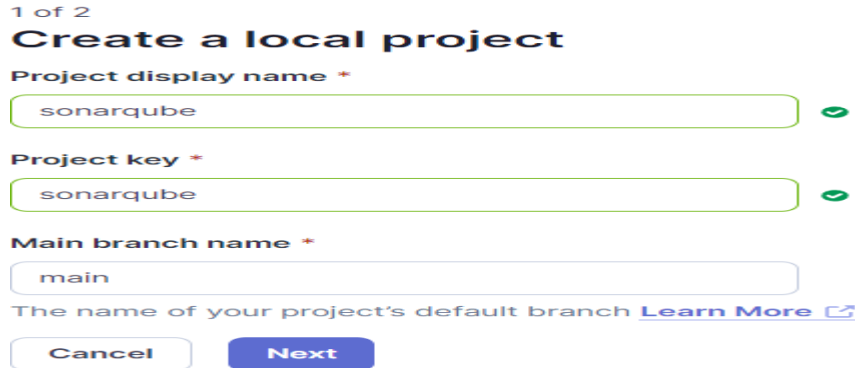
sonarqube:latest

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**”.



The image shows the SonarQube login interface. At the top is the Sonar logo. Below it is a loading spinner. The main heading is "Log in to SonarQube". There are two input fields: "Login *" with the value "admin" and "Password *" with masked characters ".....". At the bottom right are two buttons: "Go back" (a link) and "Log in" (a solid blue button).

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



The image shows the "Create a local project" form in SonarQube. It is labeled "1 of 2". The form has three required fields: "Project display name *" with the value "sonarqube", "Project key *" with the value "sonarqube", and "Main branch name *" with the value "main". Each of the first two fields has a green checkmark icon to its right. Below the fields is a note: "The name of your project's default branch" followed by a link "Learn More" and an external link icon. At the bottom are two buttons: "Cancel" and "Next".

- 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.

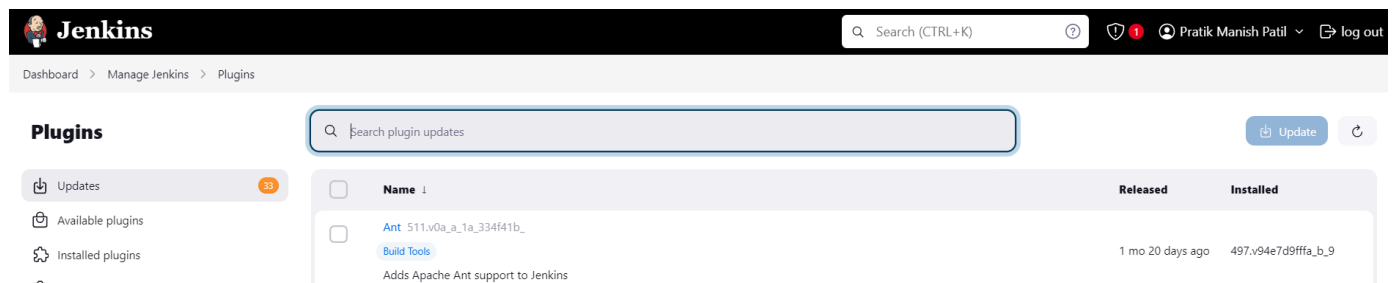
☐ Reference branch

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

[Back](#)

[Create project](#)

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

☒ 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 -

+ Add

Advanced

Add SonarQube

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

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)

Pratik Manish Patil log out

Dashboard > All > New Item

New Item

Enter an item name
sonarqube1

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

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 ?

Credentials ?

+ Add ▾

Advanced ▾

Add Repository

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

Dashboard > sonarqube > Configuration

Configure

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

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

Additional arguments [?]
 ▼

JVM Options [?]
 ▼

Add build step ▼

Save Apply

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

Administration

Configuration ▼ **Security** ▼ Projects ▼ System Marketplace

Global Permissions


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

All **Users** **Groups**

		Administer System [?]
	sonar-administrators System administrators	<input checked="" type="checkbox"/>
	sonar-users Every authenticated user automatically belongs to this group	<input type="checkbox"/>
	Anyone DEPRECATED 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"/>
	Administrator admin	<input checked="" type="checkbox"/>

4 of 4 shown

13. Run The Build and check the console output.

**Jenkins**

Search (CTRL+K) ? 1 Pratik Manish Patil log out

Dashboard > sonarqube >

Status

</> Changes

Workspace

Build Now


Configure

Delete Project

SonarQube

Rename

sonarqube

 SonarQube

Permalinks

- Last build (#18), 7 min 12 sec ago
- Last stable build (#18), 7 min 12 sec ago
- Last successful build (#18), 7 min 12 sec ago
- Last failed build (#16), 12 min ago
- Last unsuccessful build (#16), 12 min ago
- Last completed build (#18), 7 min 12 sec ago


Build History

trend

Filter...

#18

Sep 26, 2024, 11:57 PM

**Jenkins**

Search (CTRL+K) ? 1 Pratik Manish Patil log out

Dashboard > sonarqube > #18 > Console Output

Status

</> Changes

Console Output

Edit Build Information

Delete build '#18'

Timings

Git Build Data

Previous Build

Console Output

Download Copy View as plain text

Started by user Pratik Manish Patil

Running as SYSTEM

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=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"

> git.exe rev-list --no-walk f2bc042c04c6e72427c380bcaee6d6fee7b49adf # timeout=10

[sonarqube] \$ C:\ProgramData\Jenkins\jenkins\tools\udson.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=pratik -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\sonarqube

23:57:19.010 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'

23:57:19.031 INFO Scanner configuration file: C:\ProgramData\Jenkins\jenkins\tools\udson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\.\conf\sonar-scanner.properties

23:57:19.031 INFO Project root configuration file: NONE

```

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.

The screenshot shows the SonarQube web interface in a browser. The address bar displays 'localhost:9000/projects?gate=OK'. The interface includes a top navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. On the left, there is a sidebar with filters for 'My Favorites' (All), 'Filters' (Clear All Filters), 'Quality Gate' (Passed: 1, Failed: 0), 'Reliability' (A: 1, B: 0, C: 0, D: 0, E: 0), and 'Security'. The main content area shows a search bar, a 'Perspective' dropdown, an 'Overall Status' dropdown, and a 'Sort by' dropdown. A single project is listed: 'sonarqube' (PUBLIC), with a status of 'Passed' and a last analysis time of '8 minutes ago'. The project description states 'The main branch of this project is empty.' The bottom right corner indicates '1 of 1 shown'.

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.