

Experiment 7

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

Theory:

Static Application Security Testing (SAST)

SAST is a method of security testing that analyzes source code to identify vulnerabilities without executing the program. It is also known as white-box testing. Here's a breakdown of the SAST process:

1. **Code Parsing:** The source code is parsed to create an abstract syntax tree (AST), which represents the code structure.
2. **Pattern Matching:** The AST is analyzed using predefined rules to detect patterns that may indicate security vulnerabilities.
3. **Data Flow Analysis:** This step examines how data moves through the code to identify potential security issues like SQL injection or cross-site scripting (XSS).
4. **Control Flow Analysis:** This involves analyzing the paths that the code execution might take to find logical errors or vulnerabilities.
5. **Reporting:** The tool generates a report highlighting the vulnerabilities found, their severity, and recommendations for fixing them.

Benefits of SAST

- **Early Detection:** Identifies vulnerabilities early in the development lifecycle, reducing the cost and effort required to fix them.
- **Comprehensive Coverage:** Can analyze 100% of the codebase, including all possible execution paths.
- **Automated and Scalable:** Suitable for large codebases and can be integrated into CI/CD pipelines for continuous monitoring.

SonarQube and SAST

SonarQube is a popular tool that provides static code analysis to detect bugs, code smells, and security vulnerabilities. Here's how SonarQube fits into the SAST process:

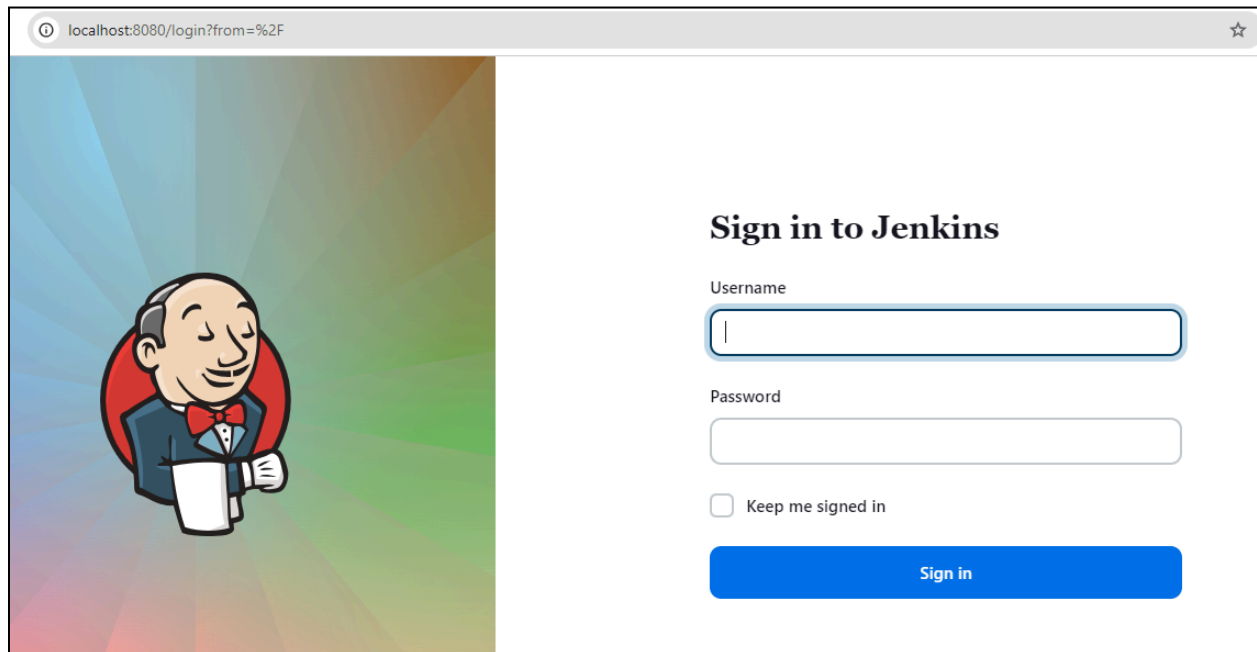
1. **Integration:** SonarQube can be integrated into your CI/CD pipeline to automatically analyze code every time it is committed.
2. **Rule Sets:** It uses a comprehensive set of rules to detect security vulnerabilities, coding standards violations, and code quality issues.
3. **Dashboards and Reports:** SonarQube provides detailed dashboards and reports that help developers understand and fix issues.
4. **Continuous Improvement:** By continuously analyzing code, SonarQube helps maintain high code quality and security standards over time.

Implementation:

Steps:

1. Open Jenkins Dashboard

- Access your Jenkins Dashboard by navigating to <http://localhost:8080> (or the port you have configured Jenkins to run on).



2. Run SonarQube in a Docker Container

- Open a terminal and run the following command to start SonarQube in a Docker container

Command -

```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

```
C:\Users\272241>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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31ae
cde
Status: Downloaded newer image for sonarqube:latest
8b62aeca4d09887a0db32d349116529581a639b59222a8b20987b42d8cec6ef3
```

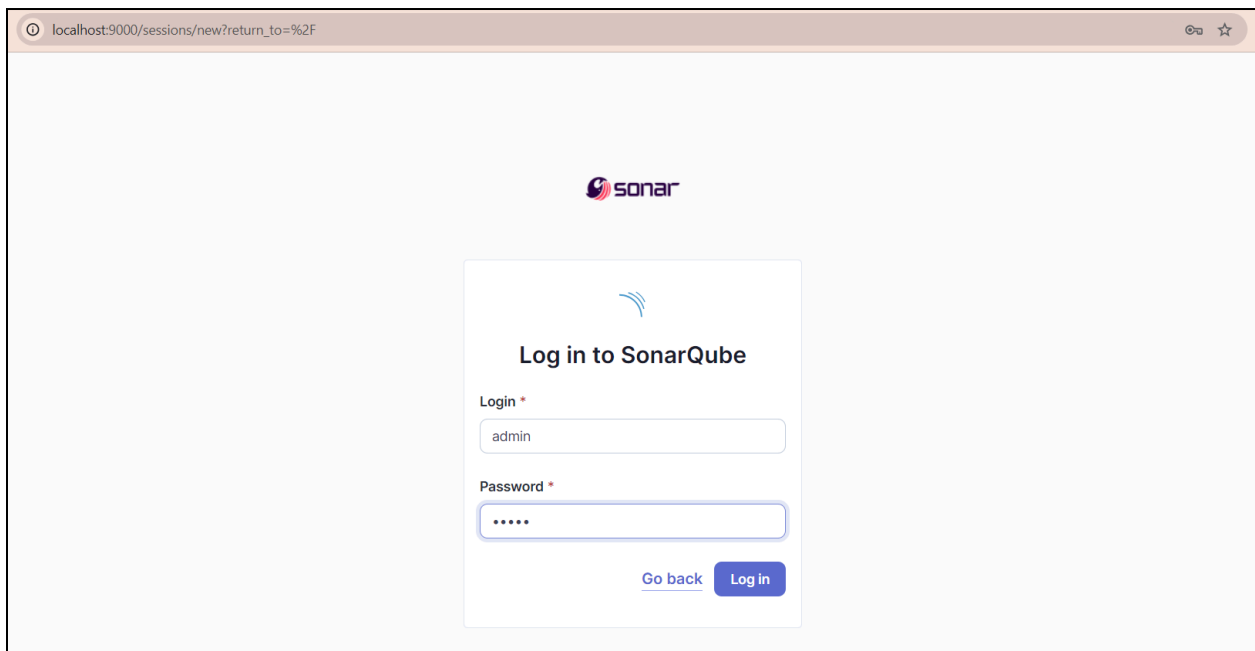
3. Check SonarQube Status

- Once the container is up and running, check the status of SonarQube by navigating to <http://localhost:9000>


4. Login to SonarQube


- Use the default credentials to log in:

- ☐ Login: admin
- ☐ Password: admin



localhost:9000/sessions/new?return_to=%2F





Log in to SonarQube

Login *

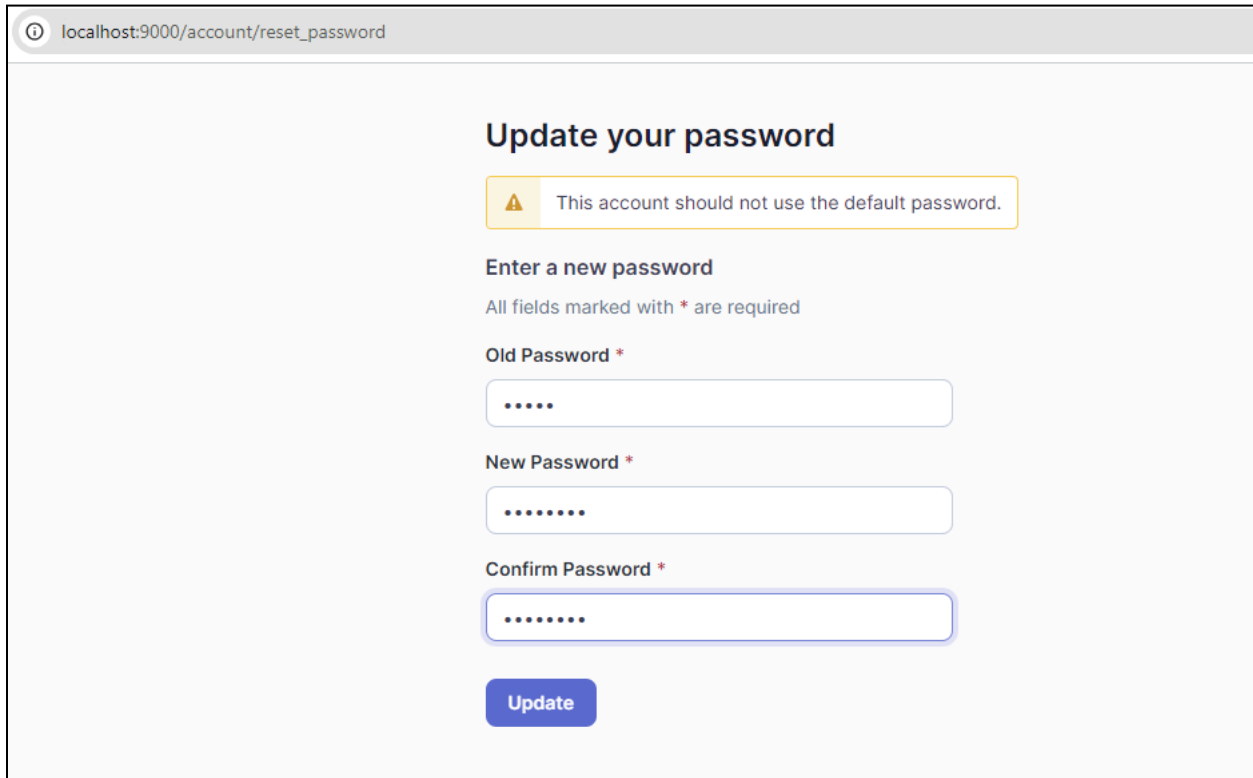
admin

Password *

.....

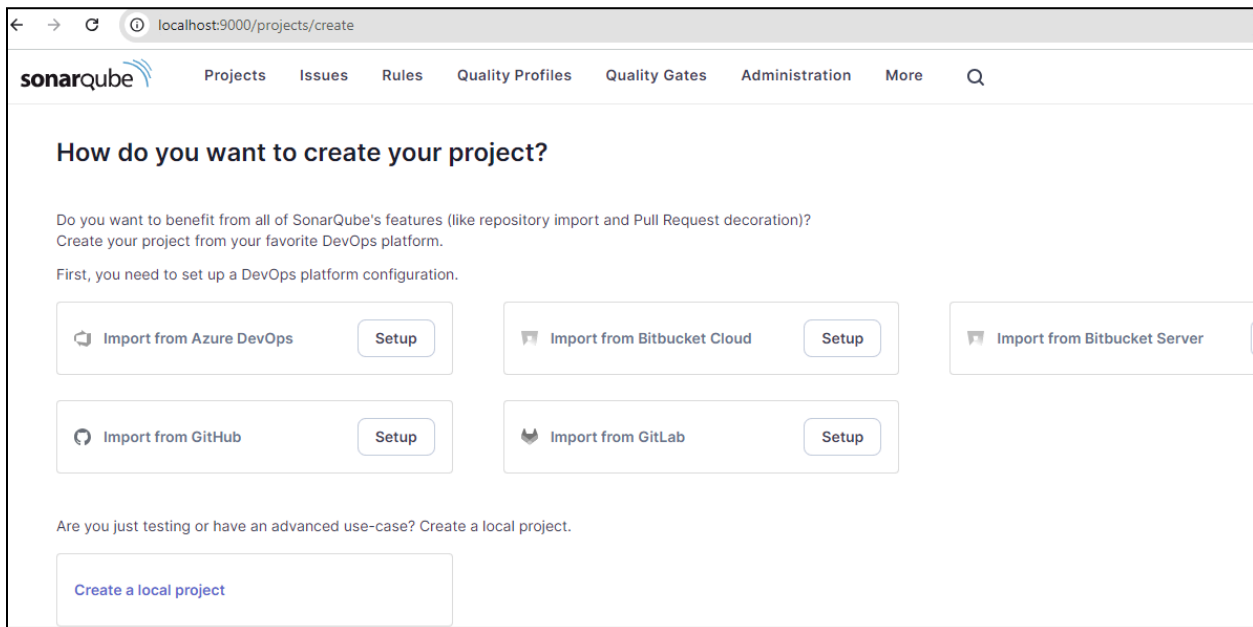
[Go back](#) [Log in](#)

5. Update the password:



The screenshot shows a web browser window with the address bar displaying 'localhost:9000/account/reset_password'. The main heading is 'Update your password'. Below it is a yellow warning box with a triangle icon and the text 'This account should not use the default password.' The section is titled 'Enter a new password' with a note 'All fields marked with * are required'. There are three input fields: 'Old Password *' with five dots, 'New Password *' with seven dots, and 'Confirm Password *' with seven dots. A blue 'Update' button is at the bottom.


6. After Logging in you will see this:



The screenshot shows the SonarQube web interface. The top navigation bar includes the SonarQube logo and links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main heading is 'How do you want to create your project?'. Below it 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' and 'Setup' (top left), 'Import from Bitbucket Cloud' and 'Setup' (top middle), 'Import from Bitbucket Server' (top right), 'Import from GitHub' and 'Setup' (bottom left), and 'Import from GitLab' and 'Setup' (bottom middle). At the bottom, there is a paragraph: 'Are you just testing or have an advanced use-case? Create a local project.' and a button labeled 'Create a local project'.

7. After that click on create a local project.


8. Then enter the name of the project:

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


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 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.

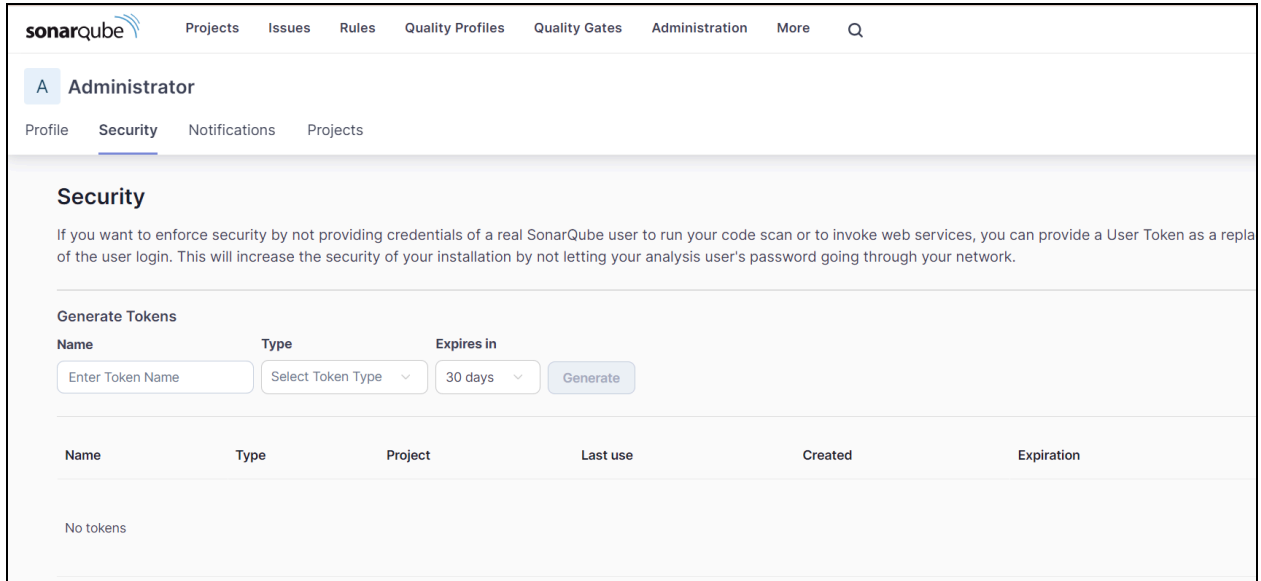
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

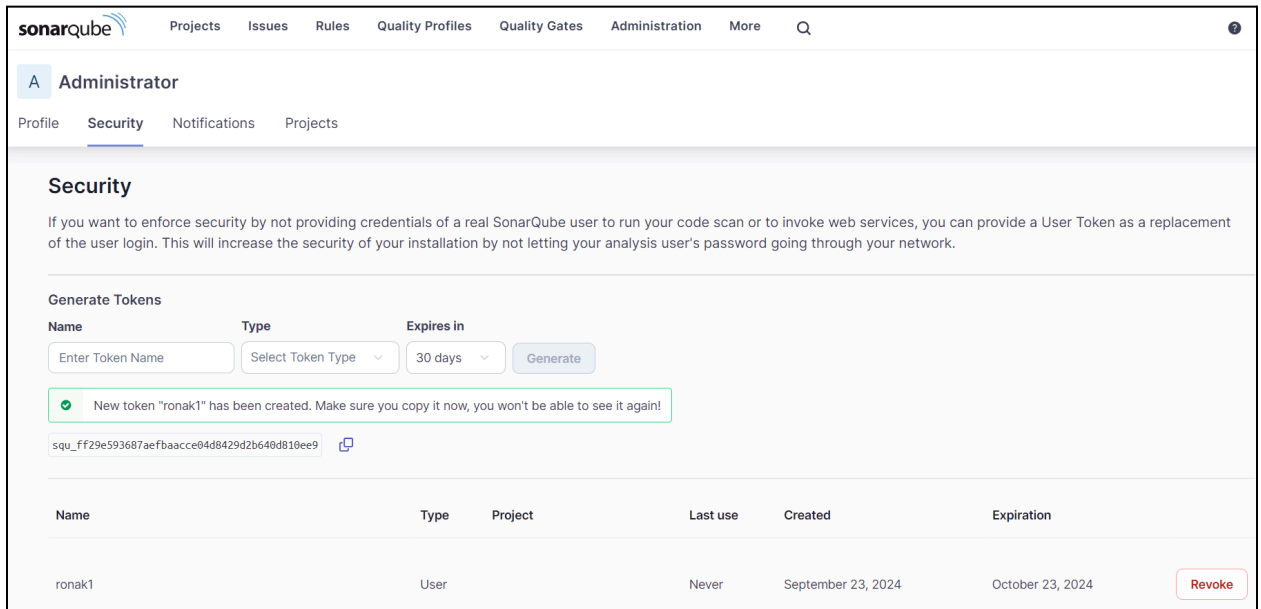
☐ Previous version

9. Then click on the 'My Account' tab:

- Under that select security.
- Then name a token you want to generate.
- Make sure you save it in notepad or somewhere else.

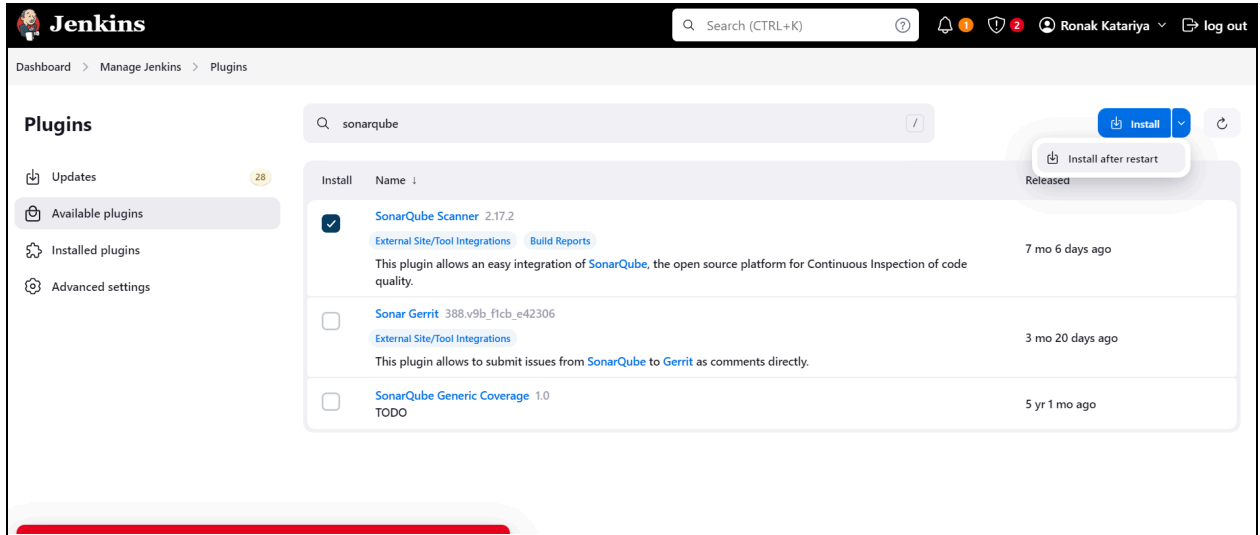


10. After adding the token you can see the list of tokens and the token you just generated:

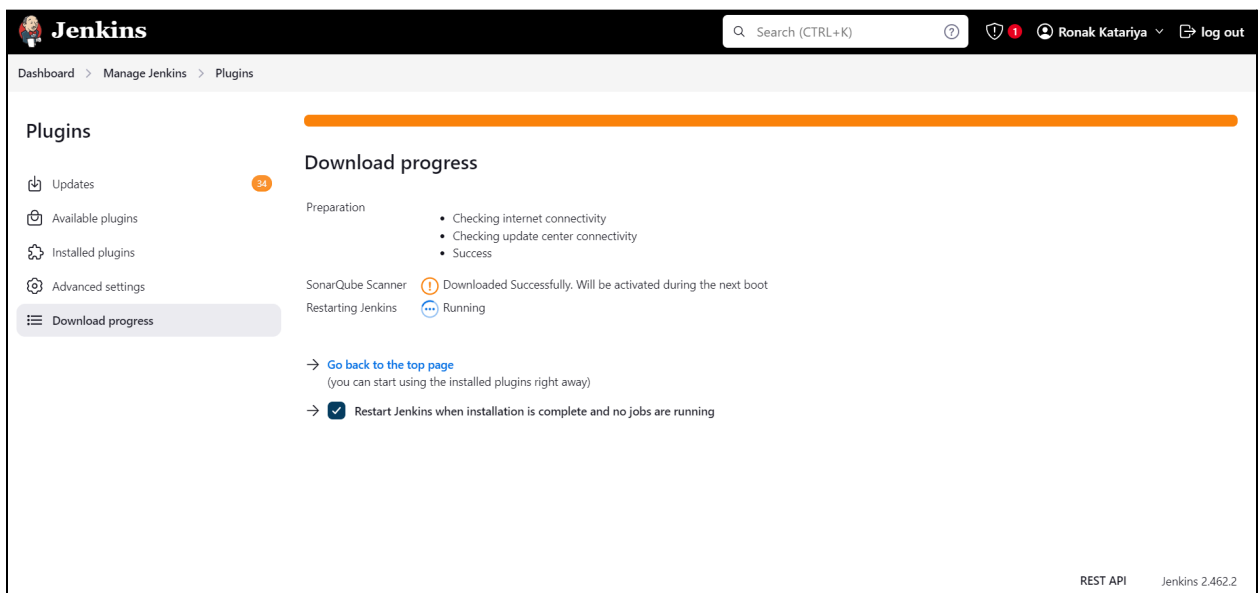


11. Install SonarQube Scanner for Jenkins

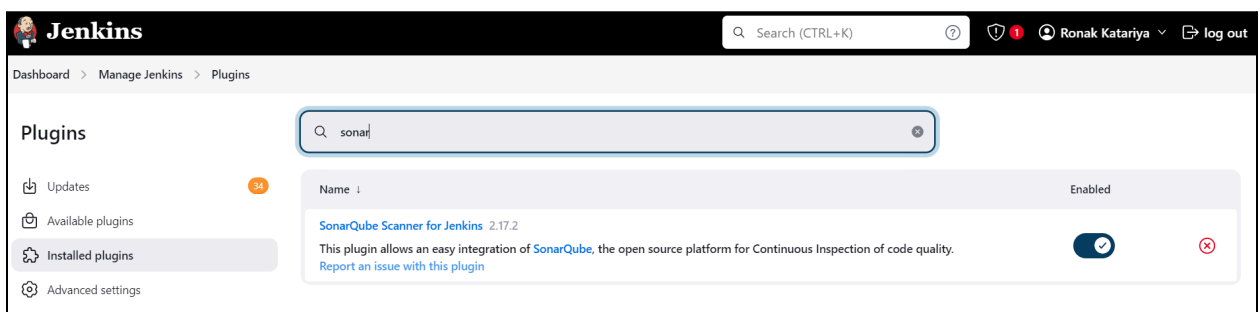
- Go back to the Jenkins Dashboard.
- Navigate to Manage Jenkins > Manage Plugins.
- Search for SonarQube Scanner for Jenkins and install it.



12. You will see such window :



13. You can see the plugins has been downloaded:



14. Configure SonarQube in Jenkins

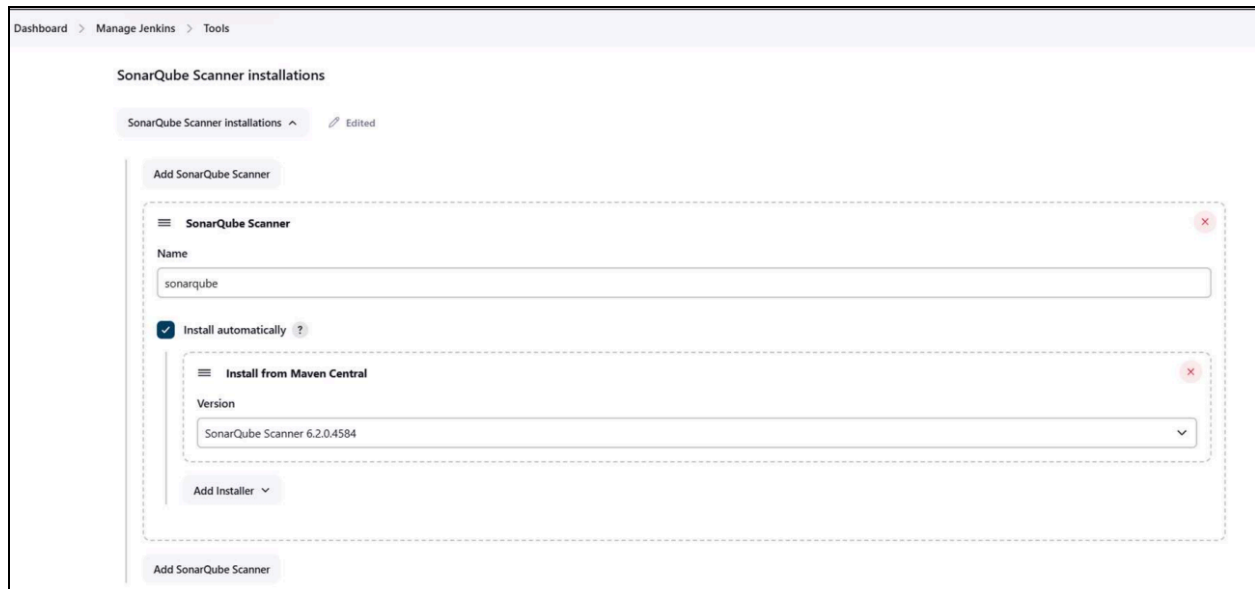
- Go to Manage Jenkins > Configure System.
- Scroll down to the SonarQube Servers section and enter the required details:
 - Name: Any name you prefer.
 - Server URL: `http://localhost:9000`
 - Server Authentication Token: (Generate this token in SonarQube under My Account > Security > Generate Tokens).
 - Add Jenkins: Select Kind - Secret Text > Secret (Paste Generated Token)



The screenshot shows the 'SonarQube servers' configuration page in Jenkins. At the top, there is a checkbox for 'Environment variables' with a note: 'If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.' Below this is the 'SonarQube installations' section, which contains a list of installations. The first installation is shown with the following details: Name: 'sonarqube', Server URL: 'http://localhost:9000' (with a default note), and Server authentication token: '- none -' (with a note that it is mandatory when anonymous access is disabled). There is an '+ Add' button at the bottom of the list.

15. Configure SonarQube Scanner in Jenkins

- Go to Manage Jenkins > Global Tool Configuration.
- Scroll down to SonarQube Scanner.
- Choose the latest version and select Install automatically




16. Create a New Jenkins Job

- In Jenkins, create a new item and select Freestyle project.
- Under Source Code Management, choose Git and enter the repository URL:
- https://github.com/shazforiot/MSBuild_firstproject.git


Enter an item name

» Required field




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 folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Branch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

17. Configure Build Steps

- Under the Build section, add a build step to Execute SonarQube Scanner.
- Enter the following analysis properties:
 - sonar.projectKey=my_project_name
 - sonar.login=your_generated_token
 - sonar.sources=HelloWorldCore
 - 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=squ_1fa56ee0a0d333885d02c288f69057eafed9adeb
sonar.sources=HelloWorldCore
sonar.host.url=http://localhost:9000
```

Additional arguments ?

JVM Options ?

Add build step

18. Set Permissions in SonarQube

- Navigate to <http://localhost:9000/permissions>.
- 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, executing analysis, and performing global system administration.

All Users Groups 🔍 Search for users or groups...

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
sonar-administrators <i>R</i> 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
sonar-users <i>R</i> 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
Anyone DEPRECATED <i>R</i> 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
Administrator admin	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

4 of 4 shown

19. Run the Build

- Go back to Jenkins and run the build.
- Check the console output for any errors or issues.

Failed output:

Dashboard > sonarqube > #4 > Console Output

Status

Changes

Console Output

Edit Build Information

Delete build '#4'

Timings

Git Build Data

Previous Build

Console Output

Download

Copy

View as plain text

Started by user [Ronak Katariya](#)
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 2.43.0.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 f2bc042c04c6e72427c380bcae6d6fee7b49adf (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "updated"
> git.exe rev-list --no-walk f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
[sonarqube] \$ C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=<http://localhost:9000> -Dproject.settings=C:\Users\272241\AppData\Local\Programs\Eclipse Adoptium\jdk-21.0.4.7-hotspot" -
Dsonar.projectKey=sonarqube -Dsonar.login=squ_e1bac8ee4a480693b1a8e9408e247b8ed0b2323d -Dsonar.host.url=<http://localhost:9000> -
Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\sonarqube

ERROR: JAVA_HOME not found in your environment, and no Java
executable present in the PATH.
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation, or add "java.exe" to the PATH

WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
ERROR: SonarQube scanner exited with non-zero code: 1
Finished: FAILURE

Successful Output:

Dashboard > sonarqube > #17 > Console Output

Status

Changes

Console Output

Edit Build Information

Timings

Git Build Data

Previous Build

Console Output

Download

Copy

View as plain text

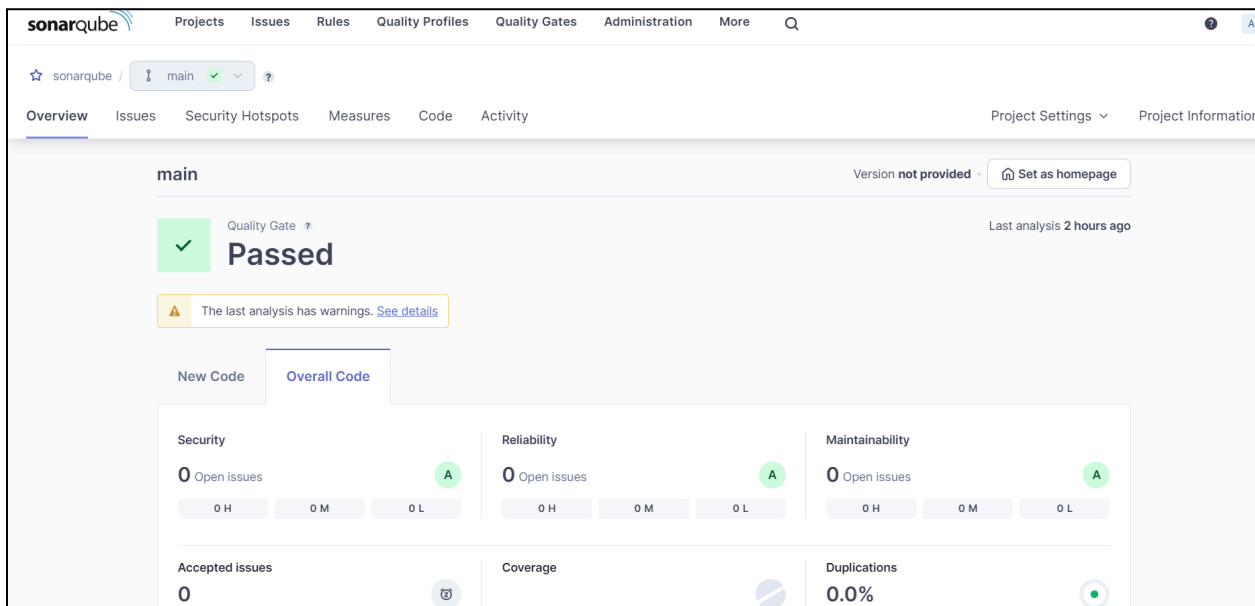
Started by user [Ronak Katariya](#)
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\sonarqube
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\sonarqube\.git # 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 2.43.0.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 f2bc042c04c6e72427c380bcae6d6fee7b49adf (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "updated"
> git.exe rev-list --no-walk f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
[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=squ_e1bac8ee4a480693b1a8e9408e247b8ed0b2323d -
Dsonar.host.url=<http://localhost:9000> -Dsonar.sources=HelloWorldCore -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\sonarqube
11:38:36.992 WARN Property 'sonar.host.url' with value '<http://localhost:9000>' is overridden with value '<http://localhost:9000>'
11:38:37.000 INFO Scanner configuration file:
C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
11:38:37.001 INFO Project root configuration file: NONE
11:38:37.018 INFO SonarScanner CLI 6.2.0.4584
11:38:37.018 INFO Java 21.0.4 Eclipse Adoptium (64-bit)
11:38:37.018 INFO Windows 11 10.0 amd64
11:38:37.035 INFO User cache: C:\WINDOWS\system32\config\systemprofile\.sonar\cache
11:38:37.600 INFO JRE provisioning: os[windows], arch[amd64]
11:38:40.300 INFO Communicating with SonarQube Server 10.6.0.92116
11:38:40.686 INFO Starting SonarScanner Engine...
11:38:40.686 INFO Java 17.0.11 Eclipse Adoptium (64-bit)

```
Dashboard > sonarqube > #17 > Console Output

SonarScanner for .NET 5.x or higher, see https://redirect.sonarsource.com/doc/install-configure-scanner-msbuild.html
11:38:54.130 INFO Sensor C# [csharp] (done) | time=0ms
11:38:54.130 INFO Sensor Analysis Warnings import [csharp]
11:38:54.130 INFO Sensor Analysis Warnings import [csharp] (done) | time=0ms
11:38:54.146 INFO Sensor C# File Caching Sensor [csharp]
11:38:54.146 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
11:38:54.146 INFO Sensor C# File Caching Sensor [csharp] (done) | time=16ms
11:38:54.146 INFO Sensor Zero Coverage Sensor
11:38:54.146 INFO Sensor Zero Coverage Sensor (done) | time=0ms
11:38:54.146 INFO SCM Publisher SCM provider for this project is: git
11:38:54.146 INFO SCM Publisher 2 source files to be analyzed
11:38:54.565 INFO SCM Publisher 2/2 source files have been analyzed (done) | time=419ms
11:38:54.565 INFO CPD Executor Calculating CPD for 0 files
11:38:54.565 INFO CPD Executor CPD calculation finished (done) | time=0ms
11:38:54.579 INFO SCM revision ID 'f2bc042c04c6e72427c380bcaee6d6fee7b49adf'
11:38:54.705 INFO Analysis report generated in 125ms, dir size=199.0 kB
11:38:54.749 INFO Analysis report compressed in 28ms, zip size=20.6 kB
11:38:54.799 INFO Analysis report uploaded in 50ms
11:38:54.799 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
11:38:54.799 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
11:38:54.799 INFO More about the report processing at http://localhost:9000/api/ce/task?id=63391c49-f9c6-45ff-8ca1-cd1dfed1c1ef
11:38:54.813 INFO Analysis total time: 12.635 s
11:38:54.813 INFO SonarScanner Engine completed successfully
11:38:54.868 INFO EXECUTION SUCCESS
11:38:54.868 INFO Total time: 17.871s
Finished: SUCCESS
```

20. Verify in SonarQube

- Once the build is complete, check the project in SonarQube to see the analysis results.



Conclusion:

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