

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

Theory :

1. Static Application Security Testing (SAST)

- **Definition:** SAST is a method of debugging by examining source code before a program is run. It identifies vulnerabilities early in the development lifecycle.
- **Key Features:**
 - **Early Detection:** Finds vulnerabilities during the coding phase, reducing remediation costs.
 - **Code Quality Analysis:** Beyond security, it also assesses code quality, maintainability, and adherence to coding standards.
 - **Integration:** Can be integrated into CI/CD pipelines for continuous security assessment.

2. SonarQube

- **Definition:** SonarQube is an open-source platform for continuous inspection of code quality, which includes detecting bugs, vulnerabilities, and code smells.
- **SAST Integration:** Supports SAST tools to analyze code and provide metrics and reports within the SonarQube dashboard.
- **Quality Gates:** Allows setting thresholds (quality gates) that must be met before code can proceed in the CI/CD pipeline.

3. Reporting and Remediation

- **Results Analysis:** After SAST scans, results are usually presented in a detailed report highlighting vulnerabilities, their severity, and remediation advice.
- **Feedback Loop:** Integrating SAST results into the development workflow helps create a feedback loop, encouraging developers to address vulnerabilities proactively.

4. Best Practices

- **Regular Updates:** Keep SAST tools and configurations updated to recognize the latest vulnerabilities.
- **Customization:** Tailor SAST rules and configurations to suit the specific needs of the project and team.
- **Training:** Ensure developers are trained on security best practices to understand and mitigate vulnerabilities effectively.

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)

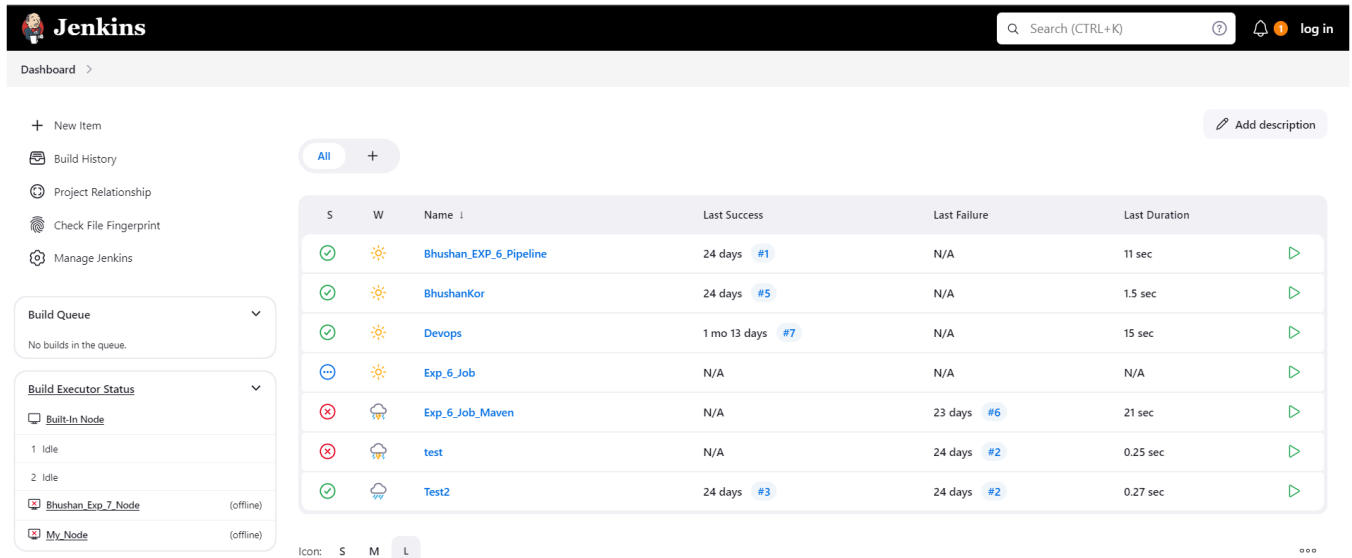
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

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



The screenshot shows the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: New Item, Build History, Project Relationship, Check File Fingerprint, and Manage Jenkins. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status (Built-In Node, 1 Idle, 2 Idle, Bhushan_Exp_7_Node (offline), My_Node (offline)). The main area displays a table of jobs with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The jobs listed are Bhushan_EXP_6_Pipeline, BhushanKor, Devops, Exp_6_Job, Exp_6_Job_Maven, test, and Test2. Each job has a status icon (green checkmark for success, red X for failure, yellow sun for warning) and a link to its build history. The table also shows the last success and failure times and durations for each job.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	Bhushan_EXP_6_Pipeline	24 days #1	N/A	11 sec
✓	☀	BhushanKor	24 days #5	N/A	1.5 sec
✓	☀	Devops	1 mo 13 days #7	N/A	15 sec
⚠	☀	Exp_6_Job	N/A	N/A	N/A
✗	☁	Exp_6_Job_Maven	N/A	23 days #6	21 sec
✗	☁	test	N/A	24 days #2	0.25 sec
✓	☁	Test2	24 days #3	24 days #2	0.27 sec

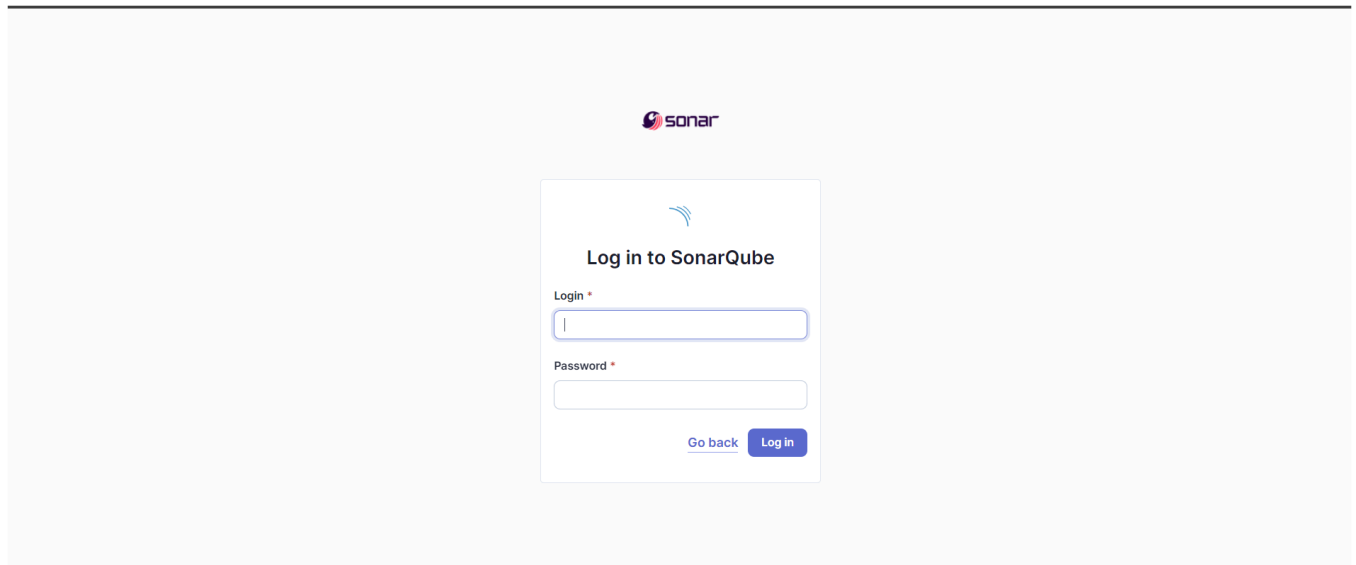
Step 2: Run SonarQube in a Docker container using this command

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

```
C:\Users\bhush\one drive 2\OneDrive\Desktop\Docker>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
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
dc00d2f31a229b2076f15c273ad750e45b74f871dd53b6d177b6f860d4fc8f0e

C:\Users\bhush\one drive 2\OneDrive\Desktop\Docker>
```

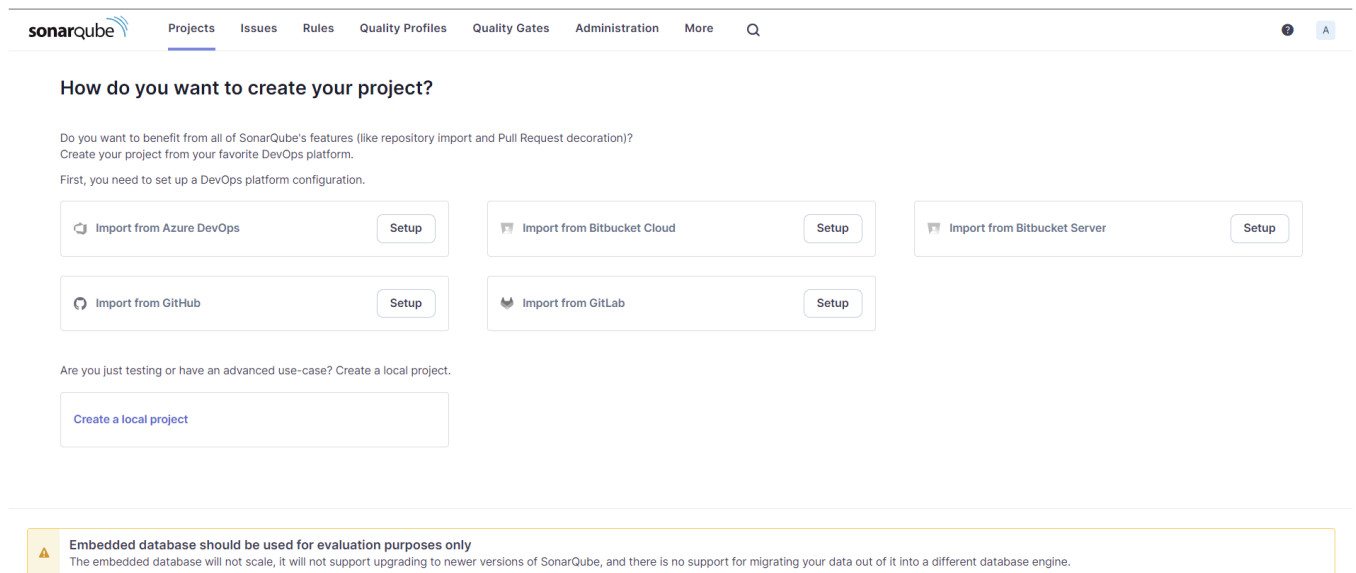
Step 3: Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



SonarQube™ technology is powered by [SonarSource SA](#)

[LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#)

Start 4: Login to SonarQube using username admin and password admin.



SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) [ACTIVE](#) [LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#) [Web API](#)

Step 5: Create a manual project in SonarQube with any Name

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

1 of 2

Create a local project

Project display name *

Bhushan's SonarQube

Project key *

Bhushan-s-SonarQube

Main branch name *

main

The name of your project's default branch [Learn More](#)

Cancel

Next

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.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) ACTIVE LGPL v3 Community Documentation Plugins Web API

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

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

Name: Bhushan Mukund Kor


Academic Year: 2024-2025

Division: D15C

Roll No: 28

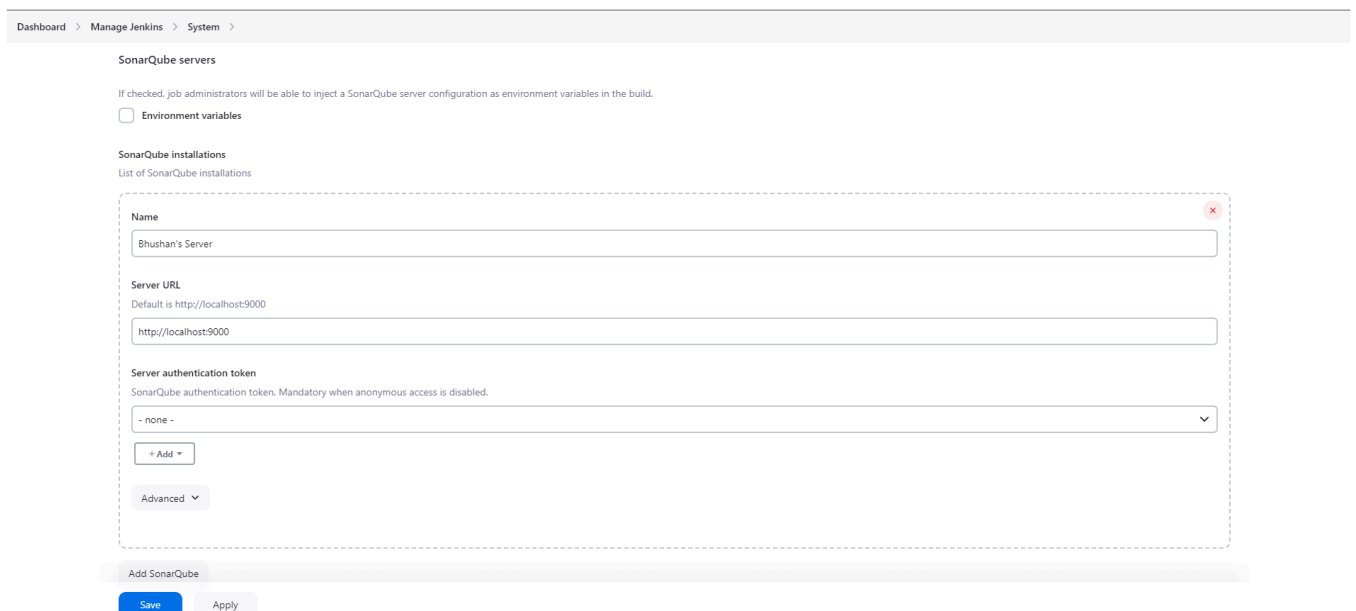
Step 6: Setup the project and come back to Jenkins Dashboard.

Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



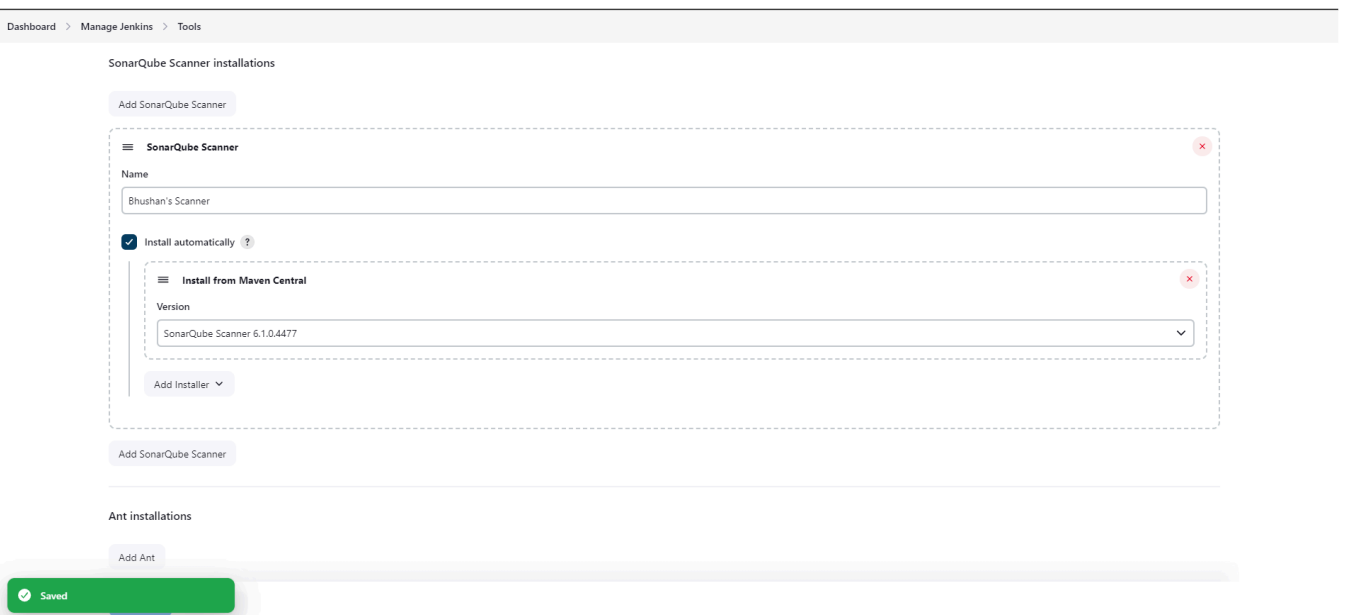
The screenshot shows the Jenkins 'Manage Jenkins' > 'Plugins' page. A search bar at the top contains 'sonar'. On the left, the 'Installed plugins' tab is selected. The main area displays a table with one entry: 'SonarQube Scanner for Jenkins' version 2.17.2. The 'Enabled' column for this plugin has a toggle switch that is turned on (blue).

Step 7: Under Jenkins 'Configure System', look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.



The screenshot shows the 'Dashboard' > 'Manage Jenkins' > 'System' page. Under the 'SonarQube servers' section, there is a checkbox for 'Environment variables' which is unchecked. Below this, the 'SonarQube installations' section is visible, showing a list of installations. One installation is configured with the following details: Name: 'Bhushan's Server', Server URL: 'http://localhost:9000', and Server authentication token: '- none -'. At the bottom, there are 'Save' and 'Apply' buttons.

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



The screenshot shows the 'Dashboard' > 'Manage Jenkins' > 'Tools' page. Under 'SonarQube Scanner installations', there is a button 'Add SonarQube Scanner'. Below it, a configuration box for 'SonarQube Scanner' is shown. The 'Name' field is 'Bhushan's Scanner'. The 'Install automatically' checkbox is checked. Under the 'Install from Maven Central' section, the 'Version' is set to 'SonarQube Scanner 6.1.0.4477'. At the bottom, there is a green 'Saved' notification bar.

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

Step 9: After the configuration, create a New Item in Jenkins, and choose a freestyle project.

Jenkins

Dashboard > All > New Item

New Item

Enter an item name

Bhushan's 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.
- 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.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Step 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 integration.

Dashboard > Bhushan's SonarQube > Configuration

Configure

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

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

https://github.com/BhushanVesitLabs/MSBuild_firstproject.git

Credentials

none

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Save Apply

Step 11: Under Build select Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

Dashboard > Bhushan's SonarQube > Configuration

☐ Poll SCM ?

Configure

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

Filter

Execute SonarQube Scanner

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

Run with timeout

Set build status to "pending" on GitHub commit

SonarScanner for MSBuild - Begin Analysis

SonarScanner for MSBuild - End Analysis

Add build step ^

Post-build Actions

Add post-build action v

Save Apply

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=Bhushan-s-SonarQube
sonar.login=admin
sonar.password=
sonar.sources=.
sonar.host.url=http://localhost:9000

Additional arguments ?

JVM Options ?

Add build step v

Save Apply

sonar.projectKey=<Your Project Key>
sonar.login=<User Name>
sonar.password=<Password>
sonar.sources=.
sonar.host.url=http://localhost:9000

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

Step 12: In the SonarQube go to Security then for Administrator allow Administer system and Execute analysis.

The screenshot shows the SonarQube Administration interface, specifically the Security section. The 'Users' tab is selected, and a search bar is present. The table below lists the permissions for various users and groups.

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
sonar-administrators 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 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 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 checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

4 of 4 shown

Step 13: See the Console Output.

The screenshot shows the SonarQube Console Output for a build. The output includes the following information:

```
Started by user unknown or anonymous
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\Bhushan's SonarQube
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\Bhushan's SonarQube\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/BhushanVestilabs/MSBuild_firstproject.git # timeout=10
Fetching upstream changes from https://github.com/BhushanVestilabs/MSBuild_firstproject.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.46.0.windows.1'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/BhushanVestilabs/MSBuild_firstproject.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse --refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision f2bc042c04c6e72427c380bc04e6d6fee7b49adf (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f f2bc042c04c6e72427c380bc04e6d6fee7b49adf # timeout=10
Commit message: "updated"
> C:\Program Files\Git\bin\git.exe rev-list --no-walk f2bc042c04c6e72427c380bc04e6d6fee7b49adf # timeout=10
[Bhushan's SonarQube] $ C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\Bhushan_s_Scanner\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=Bhushan-s-SonarQube -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -Dsonar.password=Bhushan -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\Bhushan's SonarQube
21:01:37.186 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
21:01:37.224 INFO Scanner configuration file: C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\Bhushan_s_Scanner\bin\conf\sonar-scanner.properties
21:01:37.231 INFO Project root configuration file: NONE
21:01:37.316 INFO SonarScanner CLI 6.1.0.4477
21:01:37.316 INFO Java 22.0.1 Oracle Corporation (64-bit)
21:01:37.333 INFO Windows 11 10.0 amd64
21:01:37.447 INFO User cache: C:\WINDOWS\system32\config\systemprofile\.sonar\cache
21:01:42.385 INFO JRE provisioning: os[Windows], arch[amd64]
21:03:06.440 INFO Communicating with SonarQube Server 10.6.0.92116
21:03:08.434 INFO Starting SonarScanner Engine...
21:03:08.434 INFO Java 17.0.11 Eclipse Adoptium (64-bit)
21:03:11.123 INFO Load global settings
21:03:16.628 INFO Load global settings (done) | time=5473ms
21:03:16.634 INFO Server id: 1478411E-A21XphtfuaC1GjP00pN
21:03:16.640 INFO Loading required plugins
```


Dashboard > Bhushan's SonarQube > #2 > Console Output

* Any directory in the file path has a name ending in "test" or "tests"

21:09:00.357 INFO Using git CLI to retrieve untracked files

21:09:00.653 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git

21:09:00.889 INFO 14 source files to be analyzed

21:09:03.146 INFO 14/14 source files have been analyzed

21:09:03.151 INFO Sensor TextAndSecretsSensor [text] (done) | time=5027ms

21:09:03.165 INFO ----- Run sensors on project

21:09:03.364 INFO Sensor C# [csharp]

21:09:03.370 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>

21:09:03.370 INFO Sensor C# [csharp] (done) | time=0ms

21:09:03.370 INFO Sensor Analysis Warnings Import [csharp]

21:09:03.374 INFO Sensor Analysis Warnings Import [csharp] (done) | time=9ms

21:09:03.376 INFO Sensor C# File Caching Sensor [csharp]

21:09:03.376 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir' property.

21:09:03.376 INFO Sensor C# File Caching Sensor [csharp] (done) | time=1ms

21:09:03.376 INFO Sensor Zero Coverage Sensor

21:09:03.400 INFO Sensor Zero Coverage Sensor (done) | time=26ms

21:09:03.406 INFO SCM Publisher SCM provider for this project is: git

21:09:03.408 INFO SCM Publisher 4 source files to be analyzed

21:09:06.271 INFO SCM Publisher 4/4 source files have been analyzed (done) | time=2859ms

21:09:06.271 INFO CPD Executor Calculating CPD for 0 files

21:09:06.275 INFO CPD Executor CPD calculation finished (done) | time=0ms

21:09:06.302 INFO SCM revision ID 'f2bc042c04c6e72427c380bcae6dd6fee7b49adf'

21:09:12.497 INFO Analysis report generated in 494ms, dir size=201.1 kB

21:09:12.727 INFO Analysis report compressed in 60ms, zip size=22.3 kB

21:09:23.622 INFO Analysis report uploaded in 10889ms

21:09:23.626 INFO ANALYSIS SUCCESSFUL, you can find the results at: <http://localhost:9000/dashboard?id=Bhushan-s-SonarQube>

21:09:23.631 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report

21:09:23.631 INFO More about the report processing at <http://localhost:9000/api/ce/task?id=a6525be8-4189-4cdd-Baca-a6d0b9bb7551>

21:09:23.663 INFO Analysis total time: 5:02.251 s

21:09:23.676 INFO SonarScanner Engine completed successfully

21:09:23.925 INFO EXECUTION SUCCESS

21:09:24.066 INFO Total time: 7:46.713s

Finished: SUCCESS

Jenkins

Search (CTRL+K)

log in

Dashboard > Bhushan's SonarQube >

Status

Changes

Workspace

Build Now

Configure

Delete Project

SonarQube

Rename

Bhushan's SonarQube

SonarQube

Permalinks

Last build (#2), 20 min ago

Last stable build (#2), 20 min ago

Last successful build (#2), 20 min ago

Last failed build (#1), 29 min ago

Last unsuccessful build (#1), 29 min ago

Last completed build (#2), 20 min ago

Build History

trend

Filter...

#2

Sep 19, 2024, 9:01 PM

#1

Sep 19, 2024, 8:52 PM

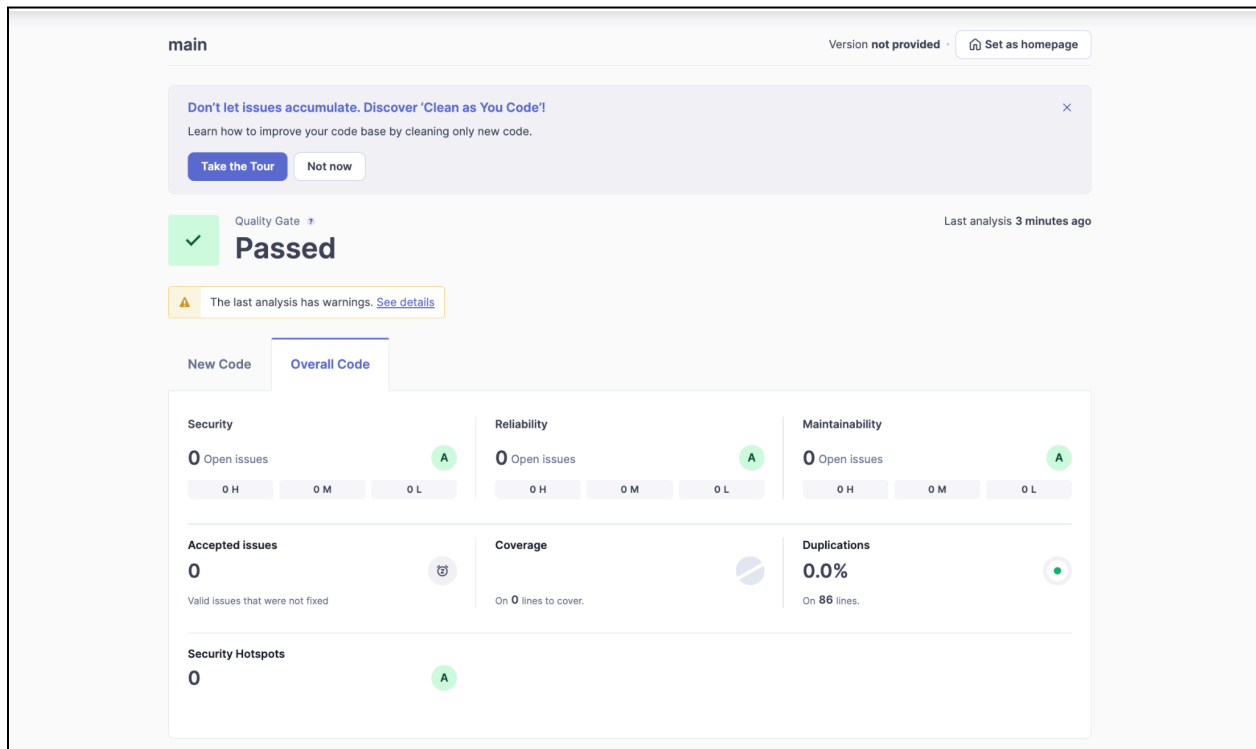
Atom feed for all

Atom feed for failures

REST API

Jenkins 2.462.1

Step 14: Now See the SonarQube Project.



Conclusion :

In this experiment, we have learned how to perform Jenkins SAST using SonarQube. For this, we used a docker image of SonarQube to not install it locally on our system. After installing the required configurations on Jenkins, using a code from a GitHub repository, we analyze its code using SonarQube. Once we build the project, we can see that the SonarQube project displays that the code has no errors.