

Experiment No :7

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

PREREQUISITES:

1) Docker :

Run docker -v command. We use this command to check if docker is installed and running on your system.

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

2) Install SonarQube Image:

The command docker pull sonarqube downloads a SonarQube image from Docker's online repository. This image lets you run SonarQube on your system using Docker without needing to install the full SonarQube software manually. It's like getting a ready-to-use version of SonarQube that can be started with Docker.

```
C:\Users\praja>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) Make sure Jenkins is already installed on your system before starting the process. Jenkins will be used to automate tasks, like running SonarQube for code analysis. If Jenkins isn't installed yet, you can download and set it up from the official Jenkins website.

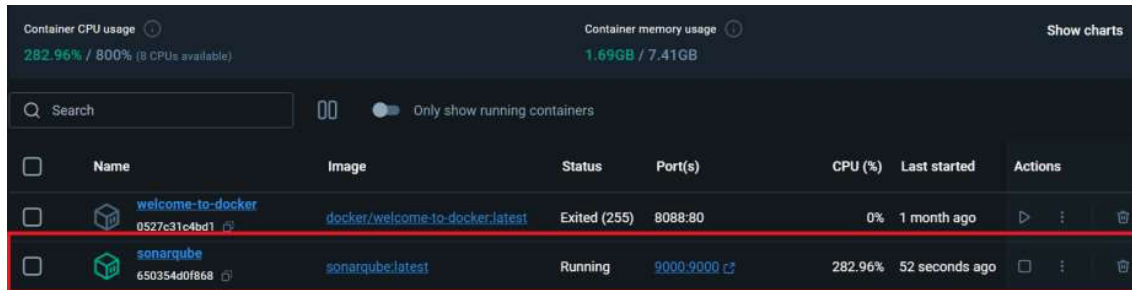
STEPS: Step1: The command docker run -d --name sonarqube -e

SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest starts

SonarQube in the background on port 9000 using Docker, allowing you to access it at

<http://localhost:9000>

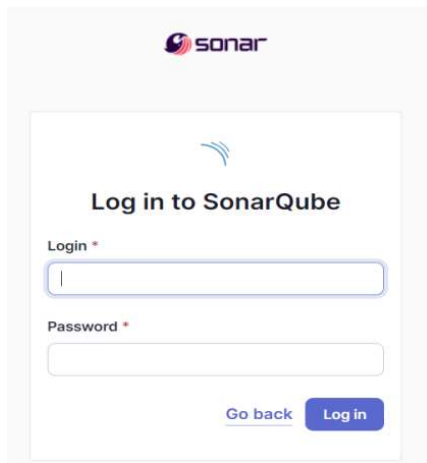
```
C:\Users\praja>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
650354d0f868ae4ad2d800426080076c604eb09f29b10d4a251aee70f51ce907
```



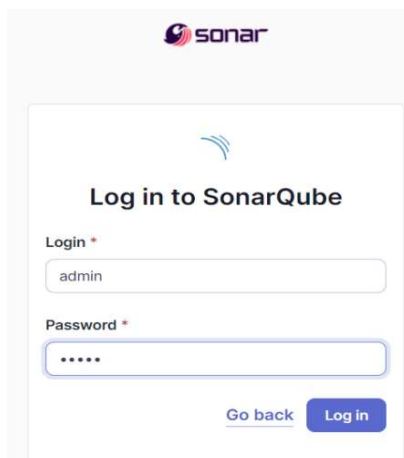
The screenshot shows the Docker Desktop interface. At the top, it displays 'Container CPU usage' at 282.96% / 800% (8 CPUs available) and 'Container memory usage' at 1.69GB / 7.41GB. Below this is a search bar and a toggle for 'Only show running containers'. A table lists the containers:

	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	welcome-to-docker 0527c31c4bd1	docker/welcome-to-docker:latest	Exited (255)	8088-80	0%	1 month ago	
<input type="checkbox"/>	sonarqube 650354d0f868	sonarqube:latest	Running	9000-9000	282.96%	52 seconds ago	

Step2: After starting the SonarQube image, open your browser and go to <http://localhost:9000> to access SonarQube.

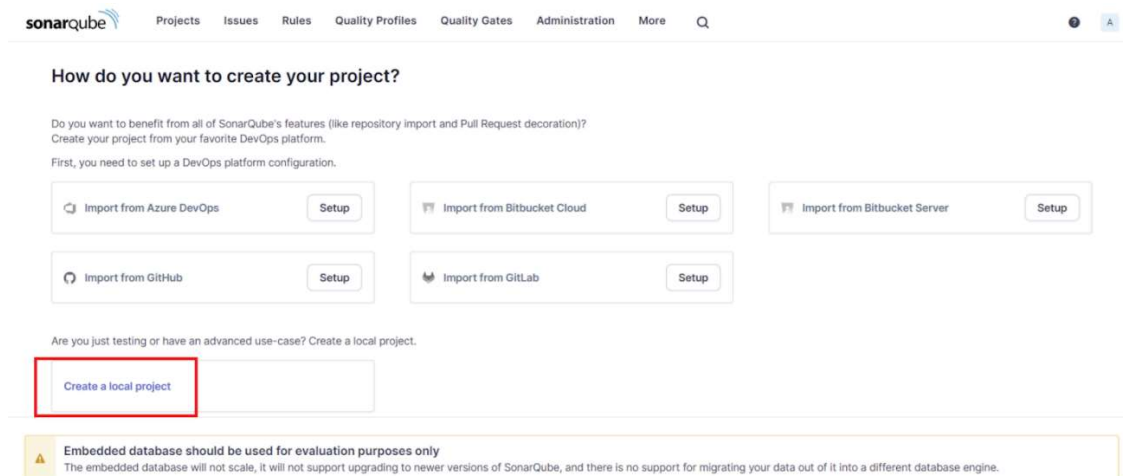


Step 3: On the SonarQube login page, use the default credentials: Username: admin , Password: admin. After logging in, you'll be prompted to change the password. Set a new password and make sure to remember it.



Click on Log in
Click on Update .

Step 4: After changing the password, you will be directed to this screen. Click on Create a Local Project.



The image shows the SonarQube project creation wizard. At the top, there's a navigation bar with links to Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. Below this, the main heading is "How do you want to create your project?". A sub-heading asks if the user wants to benefit from all of SonarQube's features (like repository import and Pull Request decoration) and suggests creating a project from a favorite DevOps platform. It then lists five options: Import from Azure DevOps, Import from Bitbucket Cloud, Import from Bitbucket Server, Import from GitHub, and Import from GitLab. Each option has a "Setup" button. Below these options, there's a section for "Are you just testing or have an advanced use-case? Create a local project." with a "Create a local project" button highlighted by a red rectangle. At the bottom, there's a warning message: "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

Projects Issues Rules Quality Profiles Quality Gates Administration More

How do you want to create your project?

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.

Import from Azure DevOps Setup

Import from Bitbucket Cloud Setup

Import from Bitbucket Server Setup

Import from GitHub Setup

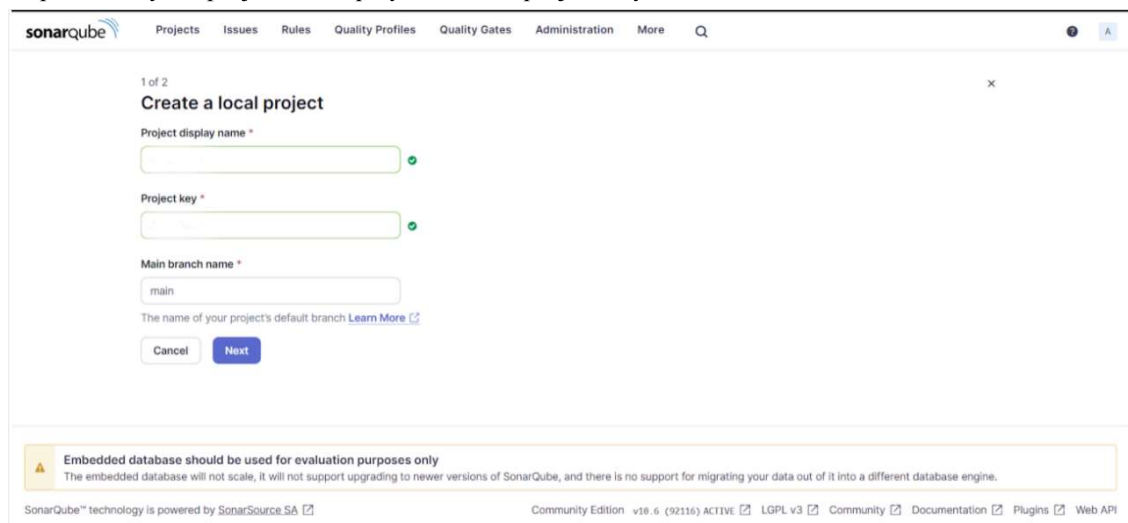
Import from GitLab Setup

Are you just testing or have an advanced use-case? Create a local project.

Create a local project

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.

Step 5: Give your project , a display name and project key



The image shows the "Create a local project" form in SonarQube. It's a multi-step process, currently on step 1 of 2. The form has three main input fields: "Project display name" with a green checkmark, "Project key" with a green checkmark, and "Main branch name" with the value "main". Below the "Main branch name" field, there's a link to "Learn More" about the default branch. At the bottom of the form, there are "Cancel" and "Next" buttons. At the very bottom, there's a warning message: "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." The footer of the page shows "SonarQube™ technology is powered by SonarSource SA" and "Community Edition v10.6 (92116) ACTIVE" with links to LGPL v3, Community, Documentation, Plugins, and Web API.

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

1 of 2

Create a local project

Project display name *

Project key *

Main branch name *

main

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

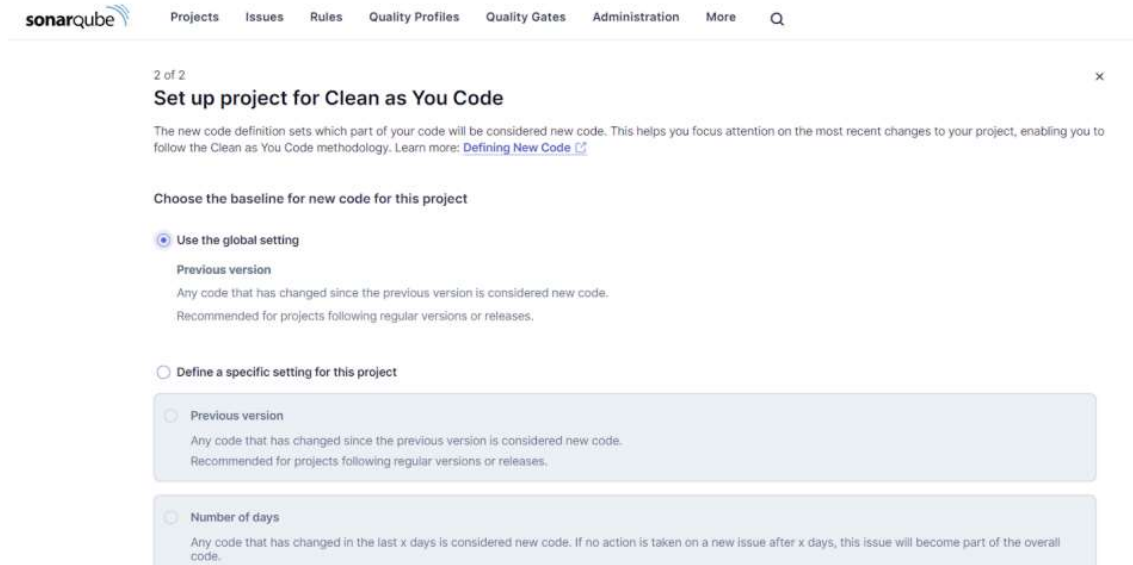
Cancel Next

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

Step 6: Configure the project by providing the necessary settings like choosing the baseline for the new code for the project , then click Create to finalize the setup.

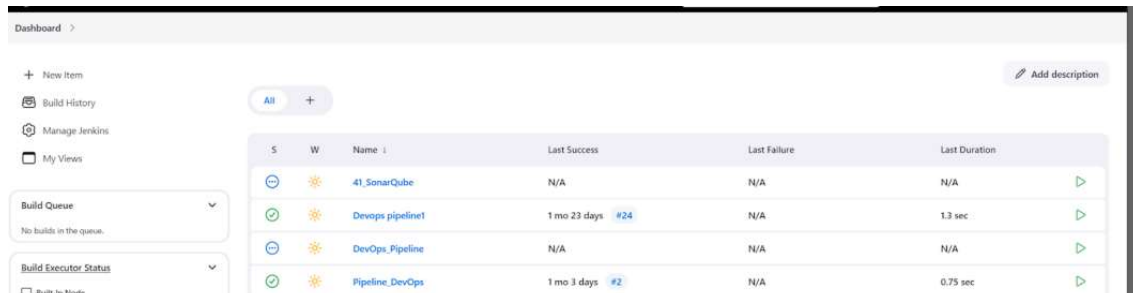


The image shows the SonarQube web interface for configuring a project. The page title is "Set up project for Clean as You Code". It explains that the new code definition sets which part of your code will be considered new code. The page is divided into two main sections: "Choose the baseline for new code for this project". The first section, "Use the global setting", is selected. It includes a "Previous version" option with a description: "Any code that has changed since the previous version is considered new code. Recommended for projects following regular versions or releases." The second section, "Define a specific setting for this project", is unselected. It includes two options: "Previous version" (with the same description as above) and "Number of days" (with a description: "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.").

Scroll Down

Click on Create project

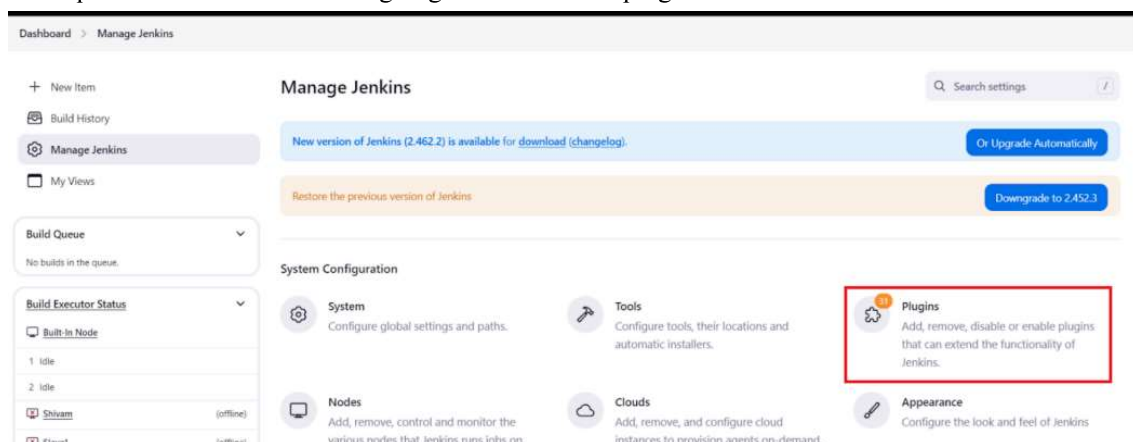
Step 7: Open Jenkins by going to <http://localhost:8080> in your browser, replacing with the specific port Jenkins is running on.



The image shows the Jenkins Dashboard. On the left, there is 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" (Built-in Node). The main area displays a table of builds. The table has columns: "S", "W", "Name", "Last Success", "Last Failure", and "Last Duration".

S	W	Name	Last Success	Last Failure	Last Duration
🔄	🔴	41_SonarQube	N/A	N/A	N/A
✅	🔴	DevOps pipeline1	1 mo 23 days #24	N/A	1.3 sec
🔄	🔴	DevOps Pipeline	N/A	N/A	N/A
✅	🔴	Pipeline_DevOps	1 mo 1 days #2	N/A	0.75 sec

Step 8: Now go to Manage Jenkin then go for Plugins followed by Available plugins search for Sonarqube Scanner where we are going to install it as a plugin



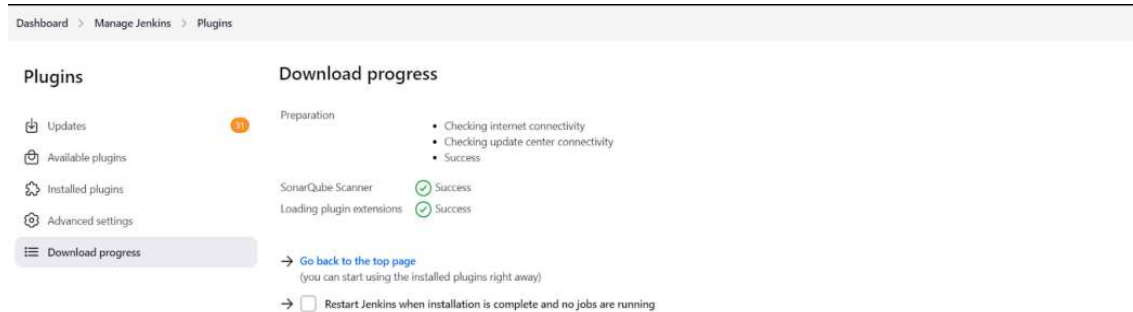
The image shows the Jenkins "Manage Jenkins" page. It includes a sidebar with links: "New Item", "Build History", "Manage Jenkins", and "My Views". Below these are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (Built-in Node). The main area displays the "Manage Jenkins" page. It includes a "System Configuration" section with links for "System", "Tools", "Nodes", and "Clouds". The "Plugins" link is highlighted with a red box. Below the "Plugins" link, there is a description: "Add, remove, disable or enable plugins that can extend the functionality of Jenkins." There is also a "Search settings" input field and a "New version of Jenkins (2.462.2) is available for download (changelog)." notification.



Click on Available Plugins.

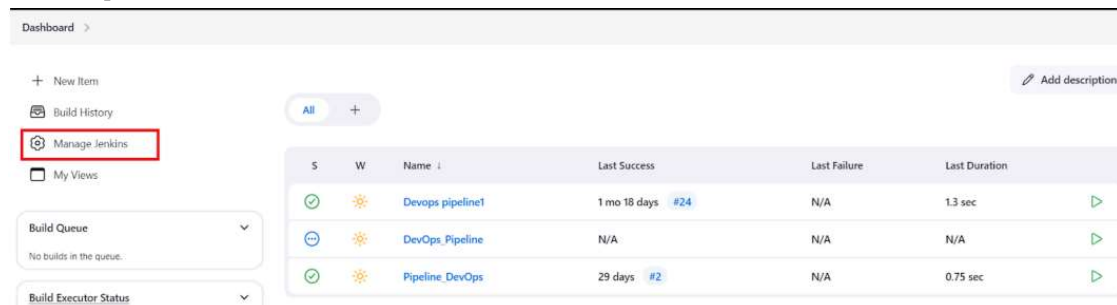


Search in the Search bar the required Plugin Name and click on Install.



Plugin Installed Successfully

Step 9: In Jenkins, go to Manage Jenkins → System, then find SonarQube servers. Add a new server, and if required, include the authentication token for secure access



Go to system

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left sidebar, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing two idle executors). The main content area is titled 'Manage Jenkins' and includes a search bar. It features two prominent banners: a blue one for upgrading to Jenkins 2.452.2 and an orange one for restoring the previous version. Below these is the 'System Configuration' section, which contains three cards: 'System' (highlighted with a red box), 'Tools', and 'Plugins'. The 'System' card is labeled 'Configure global settings and paths.'

The screenshot shows the 'System' configuration page in Jenkins. The breadcrumb trail at the top reads 'Dashboard > Manage Jenkins > System'. The page is divided into several sections: 'SonarQube servers' with a checkbox for 'Environment variables' (which is checked), 'SonarQube installations' with an 'Add SonarQube' button, 'Metrics' with an 'Add new access key' button, and 'Pipeline Speed / Durability' with a dropdown menu set to 'None: use pipeline default (Maximum survivability/durability but slowest)'. At the bottom, there are 'Save' and 'Apply' buttons.

Select Environment Variable and Click on Add Sonar Qube button in order to Add SonarQube Server to Jenkin

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:

[Add SonarQube](#)

Metrics

Access keys ?

[Add new access key](#)

Pipeline Speed / Durability

Do the required entries as shown below

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

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](#)

[Save](#) [Apply](#)

Click on save





Step 10: After configuration, create a New Item → choose a freestyle project

Dashboard > All > New Item

New Item

Enter an item name

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**

OK

Step 11: Use this github repository in Source Code Management.

https://github.com/shazforiot/MSBuild_firstproject. It is a sample hello-world project with no vulnerabilities.

Configure

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

Source Code Management

☐ None

☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

- none -

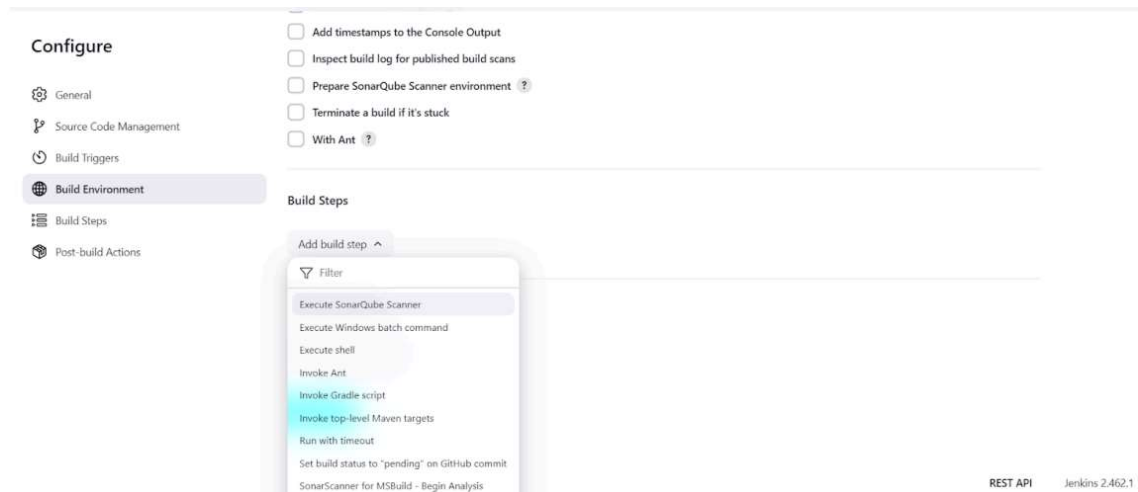
+ Add

Advanced

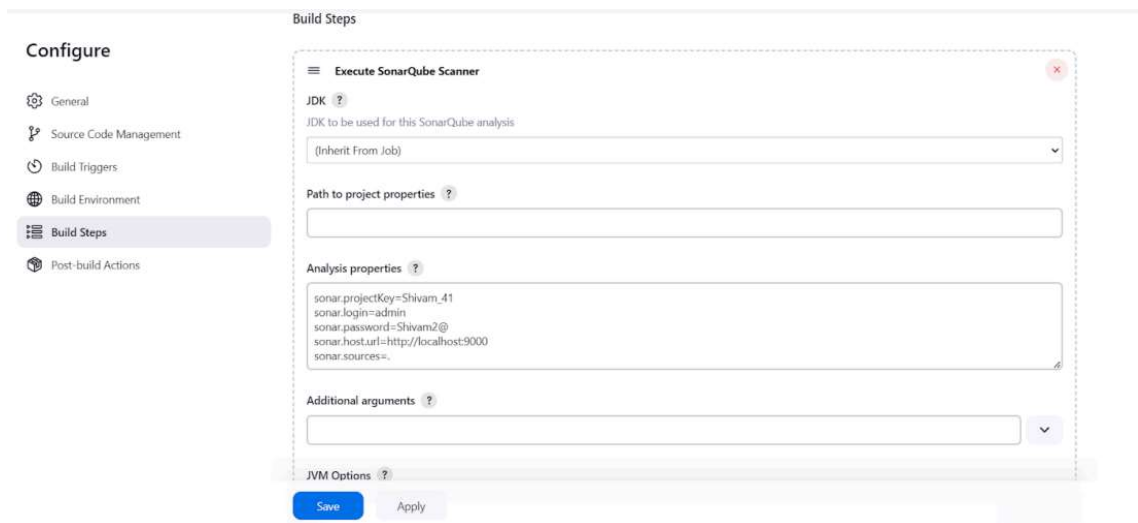
Add Repository

Save Apply

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



Click on execute sonar scanner



Step 13: Now, you need to grant the local user (here admin user) permissions to Execute the Analysis stage on SonarQube. For this, go to <http://localhost:admin/permissions> and check the 'Execute Analysis' checkbox under Administrator.

Administration				
Configuration Security Projects System Marketplace				
	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
Administrator admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input 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

Step 14: Go back to Jenkins. Go to the job you had just built and click on Build Now and Check the Console Output

The screenshot shows the Jenkins console output for a build named 'Sonarqube_lab7_41'. The output starts with 'Started by user Shyam Prajapati' and shows the execution of various commands to fetch changes from a remote Git repository, checkout the code, and run the SonarScanner. The build is successful, and the console output ends with 'Finished: SUCCESS'.

```
Dashboard > Sonarqube_lab7_41 > #41 > Console Output

Status
Changes
Console Output
Edit Build Information
Delete build '#41'
Timings
Git Build Data
Previous Build

Console Output

Started by user Shyam Prajapati
Running as SVSTER
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\Sonarqube_lab7_41
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\Sonarqube_lab7_41\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/MSBuild_FirstProject # timeout=10
Fetching upstream changes from https://github.com/shazforiot/MSBuild_FirstProject
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.44.0.windows.1'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/MSBuild_FirstProject +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 f2bc042c04c6e72427c380bcae6d6fee7b49adf (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 f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
Commit message: "updated"
> C:\Program Files\Git\bin\git.exe rev-list --no-walk f2bc042c04c6e72427c380bcae6d6fee7b49adf # timeout=10
[Sonarqube_lab7_41] $ C:\ProgramData\Jenkins\jenkins\tools\udson\plugins\sonar\SonarRunner\Installation\Sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=Shyam_41 -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -
Dsonar.password=Shyam2@ -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\Sonarqube_lab7_41

08:32:55.264 INFO Sensor Analysis Warnings import [csharp] (done) | time=4ms
08:32:55.264 INFO Sensor C# File Caching Sensor [csharp]
08:32:55.264 WARN Incremental PR analysis: Could not determine common base path, cache will not be computed. Consider setting 'sonar.projectBaseDir'
property.
08:32:55.264 INFO Sensor C# File Caching Sensor [csharp] (done) | time=0ms
08:32:55.264 INFO Sensor Zero Coverage Sensor
08:32:55.279 INFO Sensor Zero Coverage Sensor (done) | time=15ms
08:32:55.288 INFO SCM Publisher SCM provider for this project is: git
08:32:55.290 INFO SCM Publisher 4 source files to be analyzed
08:32:55.515 INFO SCM Publisher 4/4 source files have been analyzed (done) | time=225ms
08:32:55.522 INFO CPD Executor Calculating CPD for 0 files
08:32:55.523 INFO CPD Executor CPD calculation finished (done) | time=0ms
08:32:55.524 INFO SCM revision ID 'f2bc042c04c6e72427c380bcae6d6fee7b49adf'
08:32:55.809 INFO Analysis report generated in 125ms, dir size=201.0 kB
08:32:55.858 INFO Analysis report compressed in 40ms, zip size=22.5 kB
08:32:56.061 INFO Analysis report uploaded in 201ms
08:32:56.062 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=Shyam_41
08:32:56.063 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
08:32:56.063 INFO More about the report processing at http://localhost:9000/api/ce/task?id=94824f79-1689-41ea-99d7-abfee5815e63
08:32:56.077 INFO Analysis total time: 28.732 s
08:32:56.078 INFO SonarScanner Engine completed successfully
08:32:56.158 INFO EXECUTION SUCCESS
08:32:56.158 INFO Total time: 38.798s
Finished: SUCCESS

REST API Jenkins 2.462.1
```

Step 15: Once the build is complete, go back to SonarQube and check the project linked

The screenshot shows the SonarQube project overview page for 'Shyam_41'. The page displays a 'Passed' status for the Quality Gate. Below this, there are several metrics: 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 page also includes a 'New Code' tab and an 'Overall Code' tab.

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Quality Gate * Passed Last analysis 4 minutes ago

The last analysis has warnings. [See details](#)

New Code Overall Code

Security 0 Open Issues 0 H 0 M 0 L

Reliability 0 Open Issues 0 H 0 M 0 L

Maintainability 0 Open Issues 0 H 0 M 0 L

Accepted Issues 0 0 Valid issues that were not fixed

Coverage 0.0% On 0 lines to cover.

Duplications 0.0% On 86 lines.

Security Hotspots 0

CONCLUSION :

In this experiment, we successfully integrated Jenkins with SonarQube to perform static application security testing (SAST) on a project. We used Docker to run SonarQube without installing it directly on the system, simplifying the setup process. After configuring Jenkins with the SonarQube Scanner plugin and connecting it to a SonarQube server, we analysed a sample project from GitHub. The analysis demonstrated that the project had no vulnerabilities. This experiment helped us understand how to automate code analysis using Jenkins and SonarQube to ensure the security and quality of the code.