

## Experiment 7

### Aim:

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

### Integrating Jenkins with SonarQube:

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

Prerequisites: Make sure you have docker and jenkins installed.

Run **docker -v** to check the docker installation.

Run

1) Open up Jenkins Dashboard on localhost, port 8090 or whichever port it is at for you.

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' (showing 'No builds in the queue') and 'Build Executor Status' (showing 'Built-In Node' with 1 idle and 2 offline executors). The main area displays a table of build history for the 'sahil' job.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	sahil 7	24 days #2	N/A	96 ms
✓	☀	Sahil exp6	24 days #3	N/A	1 sec
⌛	☀	SahilExp6	N/A	N/A	N/A
✗	☁	sahiljob	N/A	24 days #1	1.5 sec

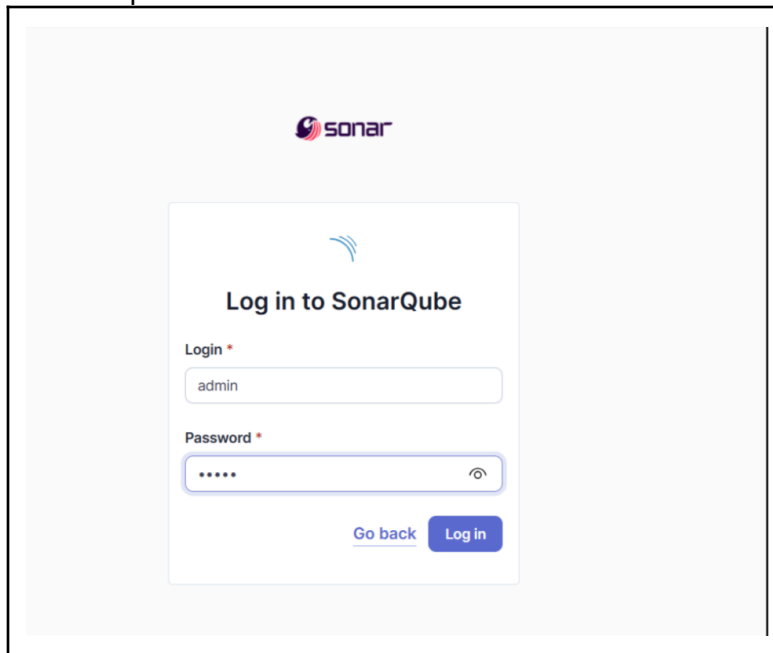
2) Run SonarQube in a Docker container using this command -

**docker run -d --name sonarqube -e SONAR\_ES\_BOOTSTRAP\_CHECKS\_DISABLE=true -p 9000:9000 sonarqube:latest**

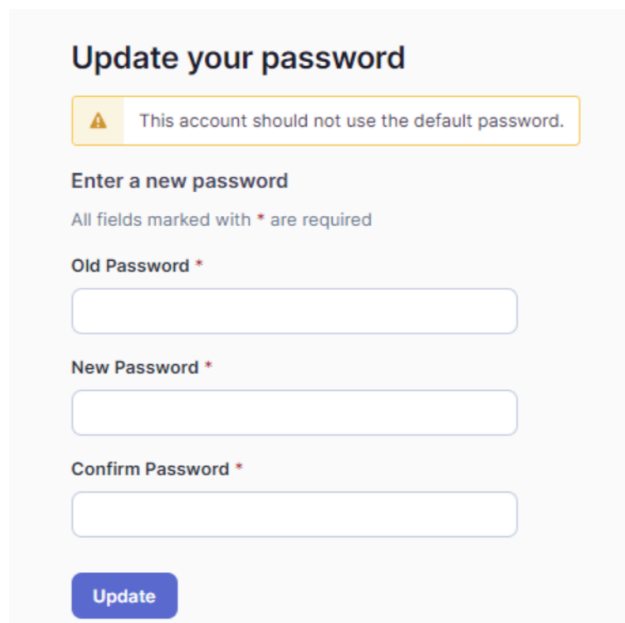
-----Warning: run below command only once

```
C:\Users\Lenovo>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
47f6db8dbf2ed99dbe384bc0ebdf47b9d4144c4e4add42055ba44ce231058272
```

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

The image shows the SonarQube login interface. At the top, there is the Sonar logo. Below it, a white box contains the title "Log in to SonarQube" and a Sonar icon. There are two input fields: "Login \*" with the text "admin" and "Password \*" with masked characters "\*\*\*\*\*". A "Go back" link and a "Log in" button are at the bottom of the form.

4) Login to SonarQube using username - *admin* and password - *admin*.  
(do change the password as you cannot use the default one)

The image shows the "Update your password" page. It features a warning message: "This account should not use the default password." Below this, the heading "Enter a new password" is followed by the instruction "All fields marked with \* are required". There are three input fields: "Old Password \*", "New Password \*", and "Confirm Password \*". An "Update" button is located at the bottom.

## 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

- 5) Create a manual project in SonarQube with the name sonarqube  
(Click on create local project)

## Create a local project

Project display name \*

exp7



Project key \*

exp7



Main branch name \*

main

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

Cancel

Next

- 6) Setup the project and come back to Jenkins Dashboard.

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

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 pa

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

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

Jenkins

Search (CTRL+K)

Shubham Jha

Dashboard > Manage Jenkins > Plugins

Plugins

Search SonarQu

Install

Install

Name

Released

☒

SonarQube Scanner

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 8 days ago

☐

Sonar Gerrit

388.v9b\_1f1cb\_e42306

External Site/Tool Integrations

This plugin allows to submit issues from SonarQube to Gerrit as comments directly.

3 mo 22 days ago

☐

SonarQube Generic Coverage

1.0

TODO

5 yr 1 mo ago

8) Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and click on **add SonarQube** and then enter the details.

Enter the Server Authentication token if needed.(I didn't do it)

In SonarQube installations: Under **Name** add <project name of sonarqube> for me its sonarqube\_exp7

In **Server URL** Default is <http://localhost:9000>

## 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\_exp7

#### 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 ▾

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

**Dashboard > Manage Jenkins > Tools**

Dashboard > Manage Jenkins > Tools

Gradle installations

Add Gradle

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild


SonarQube Scanner installations

Add SonarQube Scanner

Ant installations

Add Ant

Maven installations

Maven installations ▾  Edited

Save Apply

Click on **Add SonarQube Scanner** .

Check the “Install automatically” option. → Under name write any name as identifier →

Check the “Install automatically” option.

## SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

## SonarQube Scanner installations

Add SonarQube Scanner

☰

**SonarQube Scanner**

**Name**

sonarqube\_scanner\_exp7

☒ **Install automatically** ?

☰

**Install from Maven Central**

**Version**

SonarQube Scanner 6.2.0.4584

Add Installer ▼

Add SonarQube Scanner


10) After the configuration, create a New Item in Jenkins, choose a freestyle project.

Dashboard > All >

**Enter an item name**

exp7

= 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.

11) Choose this GitHub repository in Source Code Management.

[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.

The screenshot shows the 'Configure' page for 'Source Code Management' in SonarQube. The left sidebar lists configuration categories: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main content area is titled 'Source Code Management' and has two radio buttons: 'None' and 'Git' (selected). Below the 'Git' button is a 'Repositories' section with a 'Repository URL' field containing 'https://github.com/shazforiot/MSBuild\_firstproject.git'. There is also a 'Credentials' dropdown menu set to '- none -' and an 'Add Repository' button. At the bottom, there is a 'Branches to build' field and 'Save' and 'Apply' buttons.

12) Under **Select project** → **Configuration** → **Build steps** → **Execute SonarQube Scanner**, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

The screenshot shows the 'Configure' page for 'Build Steps' in SonarQube. The left sidebar lists configuration categories: General, Source Code Management, Build Triggers, Build Environment, Build Steps (selected), and Post-build Actions. The main content area is titled 'Build Steps' and has a 'Filter' dropdown menu. Below the filter is a list of build steps: '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', and 'SonarScanner for MSBuild - End Analysis'. There is an 'Add build step' button at the bottom. Below the build steps list is a 'Post-build Actions' section with an 'Add post-build action' dropdown menu and 'Save' and 'Apply' buttons.



Following window will open -

The screenshot shows the 'Configure' page for the 'Execute SonarQube Scanner' build step. On the left is a sidebar with navigation links: General, Source Code Management, Build Triggers, Build Environment, Build Steps (selected), and Post-build Actions. The main area contains the configuration for the scanner. It includes a 'JDK' dropdown menu set to '(Inherit From Job)', a 'Path to project properties' text input, an 'Analysis properties' text area, an 'Additional arguments' dropdown menu, and a 'JVM Options' dropdown menu. At the bottom of the configuration box is an 'Add build step' button.

Open sonarQube again and go to Project Information appearing in the right side. Click on it and you can copy the project key from About the Project Section.

The screenshot shows the 'Project Information' page for a project named 'exp7'. The left sidebar contains sections: 'About this Project' (Quality Gate used: Default, Sonar way; Project Key: exp7; Visibility: Public; Description: No description added for this project; Tags: No tags), 'Notifications' (A notification is never sent to the author of the event; Send me an email for: Background tasks in failure, Changes in issues/hotspots assigned to me, Quality gate changes, Issues resolved as false positive or accepted, New Issues, My new issues), and 'Badges'. The right sidebar contains 'Project Settings' and 'Project Information' (selected).

Use this key in place of <projectKey> in the following code

`sonar.projectKey =<projectKey>`

`sonar.login =admin`

`sonar.password =<yourpassword for sonar qube>`

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

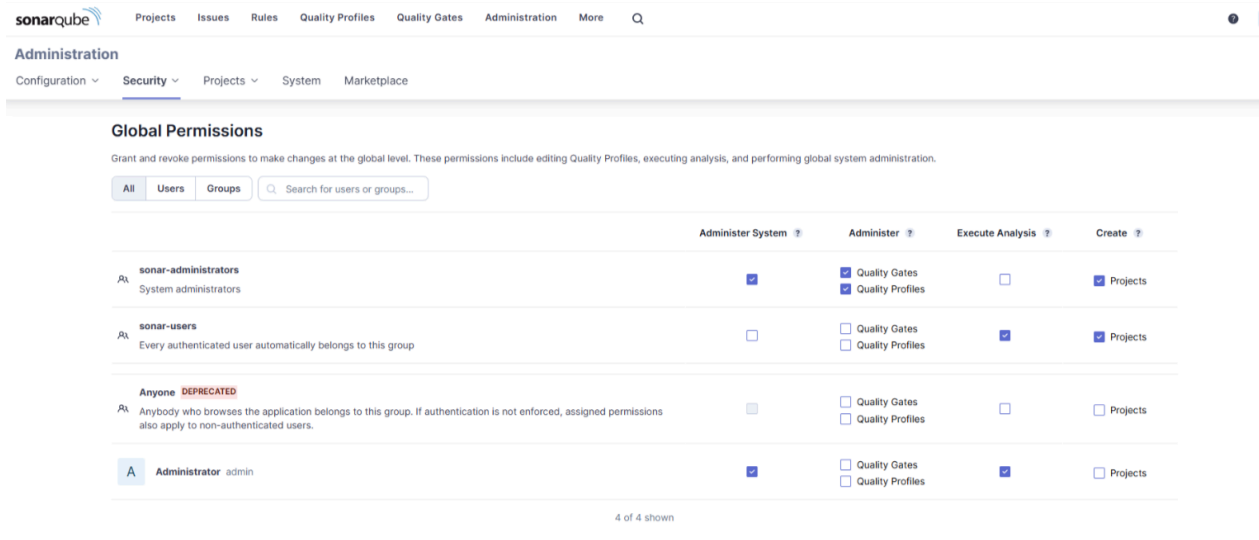
`sonar.sources =`

This screenshot is a closer view of the 'Configure' page, specifically the 'Analysis properties' text area. The text area contains the following configuration code:  
`sonar.projectKey =exp7  
sonar.login =admin  
sonar.password =2923  
sonar.host.url =http://localhost:9000  
sonar.sources =`  
The other fields (JDK, Path to project properties, Additional arguments, JVM Options) are visible but not the focus of this image.

Apply and save.

13) Go to sonarQube and go to administration → Security (dropdown) → Global Permissions.

See the administrator below and check the boxes as checked below..

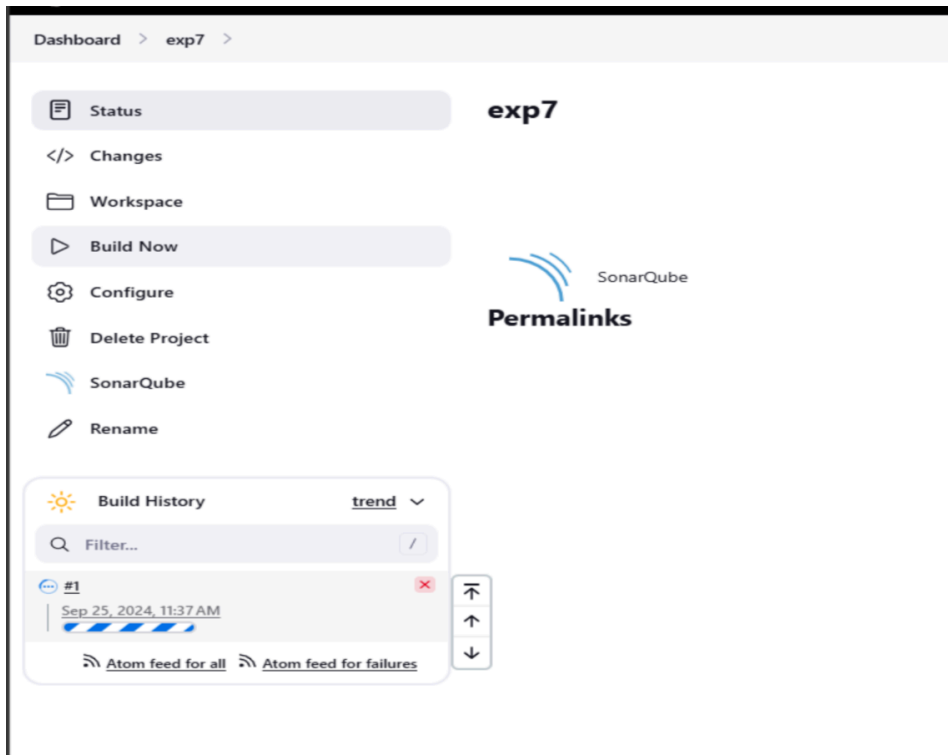


The screenshot shows the SonarQube Administration interface, specifically the 'Global Permissions' section under 'Security'. The page title is 'Global Permissions' and it includes a subtitle: 'Grant and revoke permissions to make changes at the global level. These permissions include editing Quality Profiles, executing analysis, and performing global system administration.' Below the subtitle are tabs for 'All', 'Users', and 'Groups', with a search bar 'Search for users or groups...'. The main content is a table listing permissions for different groups. The table has columns for the group name, 'Administer System', 'Administer', 'Execute Analysis', and 'Create'. The 'Administer' column is further detailed with checkboxes for 'Quality Gates' and 'Quality Profiles'. The groups listed are 'sonar-administrators', 'sonar-users', 'Anyone DEPRECATED', and 'Administrator admin'. The 'Administrator admin' group has all permissions checked.

	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 DEPRECATED</b> 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 admin</b>	<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

12. Go to jenkins and click build:



The screenshot shows the Jenkins web interface for a project named 'exp7'. The top navigation bar includes 'Dashboard' and 'exp7'. On the left, there is a sidebar with icons and labels for 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'SonarQube', and 'Rename'. The main content area displays the 'exp7' project name, the SonarQube logo, and the text 'Permalinks'. Below this, there is a 'Build History' section with a 'trend' dropdown and a 'Filter...' input. The build history shows a single build labeled '#1' with a timestamp of 'Sep 25, 2024, 11:37 AM' and a status icon. At the bottom of the build history, there are links for 'Atom feed for all' and 'Atom feed for failures'. On the right side of the build history, there are three vertical arrows (up, middle, down) for navigation.

Status

&lt;/&gt; Changes

Console Output

View as plain text

Edit Build Information

Delete build '#5'

Timings

Git Build Data

Previous Build

Console Output

```

Started by user Shubham Jha
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\exp7
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\exp7\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shafarior/MSBuild_firstproject # timeout=10
Fetching upstream changes from https://github.com/shafarior/MSBuild_firstproject
> git.exe --version # timeout=10
> git --version # 'git version 2.45.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shafarior/MSBuild_firstproject +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master" # 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
[exp7] $ C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunner\Installation\sonarqube_scanner_exp7\bin\sonar-scanner.bat -Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=exp7
-Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.sources=. -Dsonar.password=2923 -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\exp7
13:47:35.366 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
13:47:35.382 INFO Project root configuration file: NONE
13:47:35.483 INFO SonarScanner CLI 6.2.0.4584
13:47:35.483 INFO Java 21.0.4 Oracle Corporation (64-bit)
13:47:35.413 INFO Windows 11 10.0 amd64
13:47:35.429 INFO User cache: C:\Windows\system32\config\systemprofile\.sonar\cache
13:47:37.015 INFO 3RE provisioning: os[windows], arch[amd64]
13:47:47.097 INFO Communicating with SonarQube Server 10.6.0.92116
13:47:47.665 INFO Starting SonarScanner Engine...

```

## Conclusion:

In this project, we successfully integrated Jenkins with SonarQube to establish a robust automated static application security testing (SAST) pipeline. The setup involved deploying SonarQube using Docker, ensuring smooth container orchestration and efficient resource management. A key component was configuring Jenkins with the appropriate SonarQube plugins, authentication mechanisms, and linking it to a GitHub repository for continuous integration.

One of the challenges was configuring Docker on the Jenkins environment, which required resolving networking issues between the Docker containers and ensuring that the SonarQube server was reachable from Jenkins. Additionally, setting up secure authentication between Jenkins and SonarQube involved troubleshooting token-based authentication and resolving environment path issues, particularly with the **JAVA\_HOME** setup for the SonarQube scanner.

After overcoming these obstacles, I integrated the SonarQube scanner as a build step, allowing for continuous code analysis. This setup provided automated detection of code vulnerabilities, code smells, and quality issues. It helped ensure that any new commits triggered immediate analysis, generating detailed reports and promoting continuous improvement in code quality.