

EXPERIMENT NO. 7

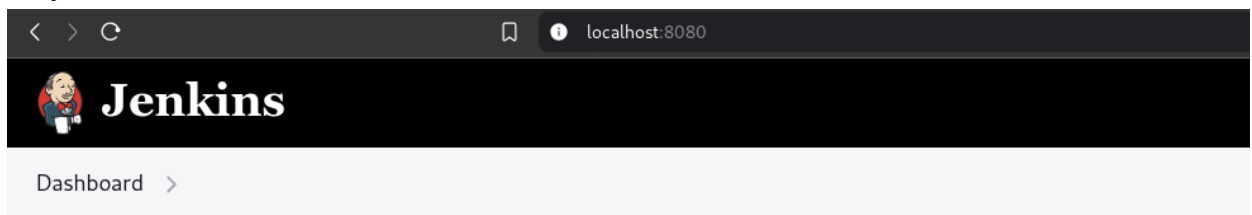
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Prerequisites:

- Jenkins installed (Java JDK required)
- Docker Installed (for SonarQube)

Steps to integrate Jenkins with SonarQube

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



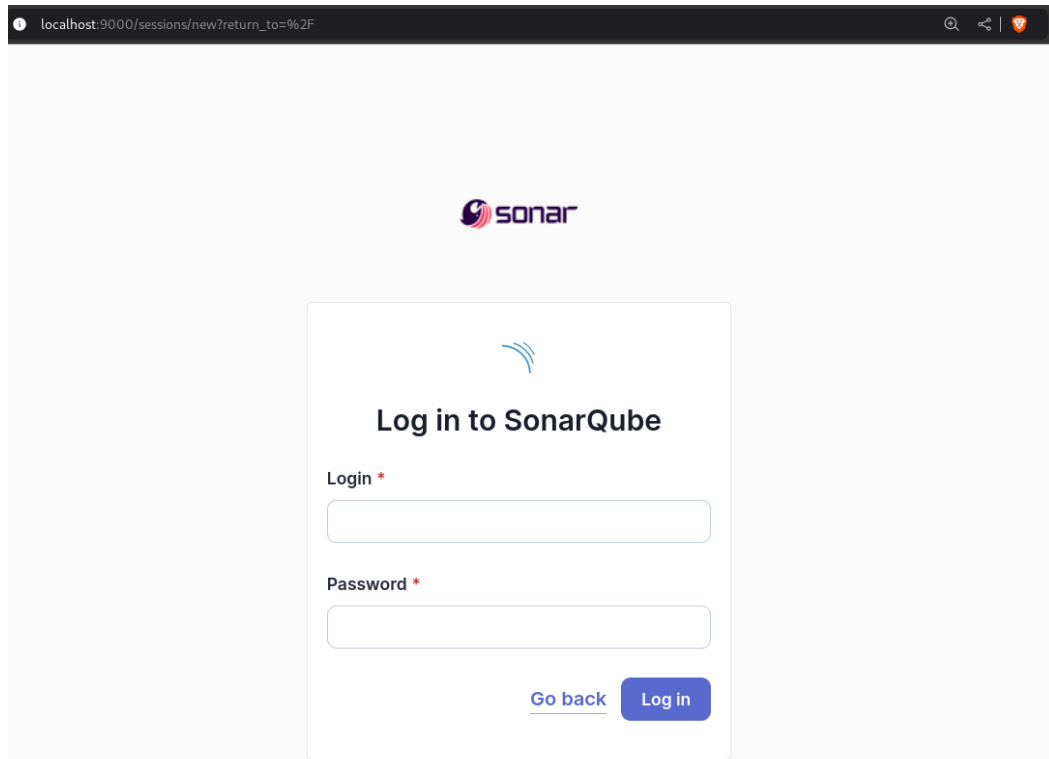
Our jenkins is running on port 8080

2. Run SonarQube in a Docker container using this command
`sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest`

Warning: run below command only once

```
quantum@machine ~$ sudo docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
[sudo] password for quantum:
f9c595308e368210e19e099256a47ec1fe44affdc778eb58ccb53174163ce057
```


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



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

localhost:9000/account/reset_password

Update your password

 This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

After logging, we have to change default password

5. Create a manual project in SonarQube with the name sonarqube Click on **create a local project** on dashboard

First, you need to set up a DevOps platform configuration.

 Import from Azure DevOps	Setup	 Import from Bitbucket Cloud	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


1 of 2

Create a local project

Project display name *

SONARQUBE_PROJECT 

Project key *

SONARQUBE_PROJECT 

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

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 follow the Clean as You Code methodology. Learn more: [Defining New Code](#) 

Choose the baseline for new code for this project

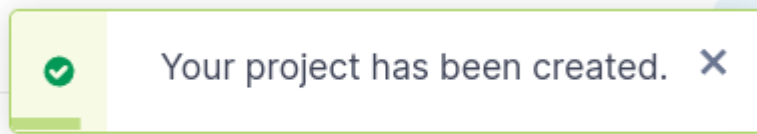
☒ Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

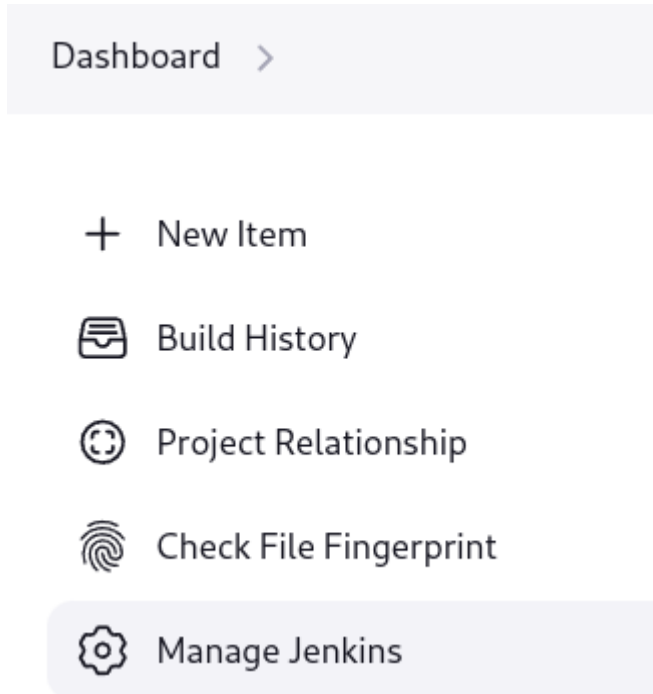
Recommended for projects following regular versions or releases.

We can either use new custom settings for the project or use global settings
Here I'm using global setting

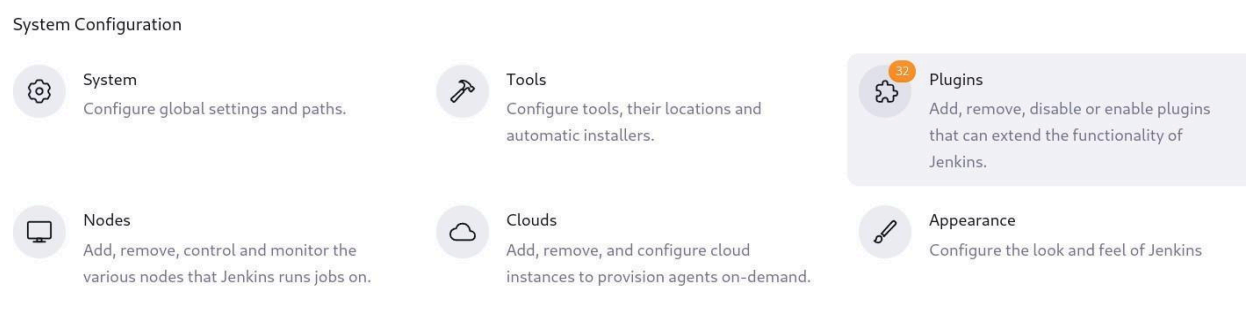


On successful creation of project, we a popup for the same

6. After setting project in sonarqube, go to **Jenkins Dashboard**



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



Install	Name ↓
<input checked="" type="checkbox"/>	<div><div>SonarQube Scanner 2.17.2</div><div>External Site/Tool Integrations Build Reports</div><div>This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.</div></div>
<input type="checkbox"/>	<div><div>Sonar Gerrit 388.v9b_f1cb_e42306</div><div>External Site/Tool Integrations</div><div>This plugin allows to submit issues from SonarQube to Gerrit as comments directly.</div></div>
<input type="checkbox"/>	<div><div>SonarQube Generic Coverage 1.0</div><div>TODO</div></div>

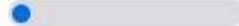
Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

SonarQube Scanner

Installing



Loading plugin extensions

Pending

→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→ ☐ Restart Jenkins when installation is complete and no jobs are running

Our installation is in progress wait for it to download and install packages

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

SonarQube Scanner

 Success

Loading plugin extensions

 Success

Plugin installed successfully

7. Under Jenkins dashboard 'Configure System', look for SonarQube Servers and enter the details.

System Configuration



System

Configure global settings and paths.



Tools

Configure tools, their locations and automatic installers.



Nodes

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Clouds

Add, remove, and configure cloud instances to provision agents on-demand.

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

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


System Configuration




System
Configure global settings and paths.



Tools
Configure tools, their locations and automatic installers.



Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

Click on **Add SonarQube Scanner**

SonarQube Scanner installations

Add SonarQube Scanner

☰

SonarQube Scanner

✕

Name

sonarqube installer

☒ Install automatically ?

☰

Install from Maven Central

✕

Version

SonarQube Scanner 6.2.0.4584

▼

Add Installer ▼

Select Latest version and save configuration

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

New Item

Enter an item name

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.

**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.

OK

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 ?

- none -

+ Add ▾

Advanced ▾

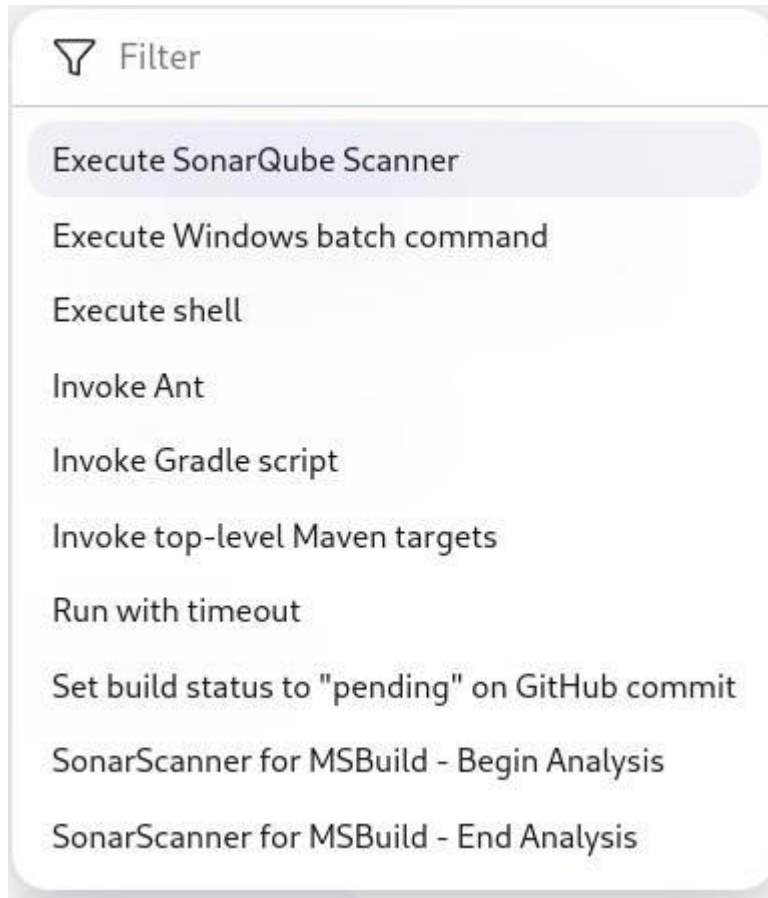
Add Repository

Branches to build ?

Build Steps

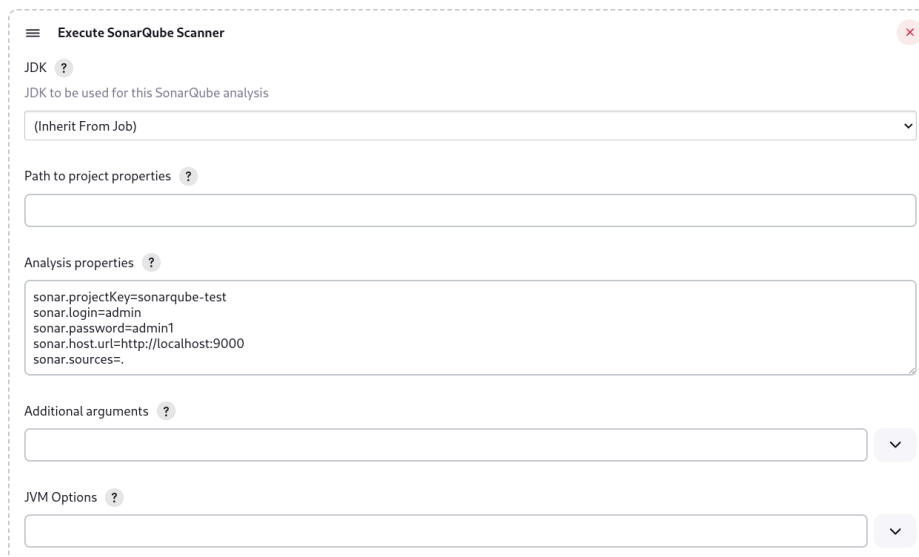
Add build step ▾

Click on add build steps



Then click on **Execute SonarQube Scanner**

12. Mention the SonarQube Project Key, Login, Password, Source path and Host URL in Analysis properties



13. Go to http://localhost:9000/project_roles?id=<project_key> and allow Execute Permissions to the Admin user.

All	Users	Groups	Search for users or groups...	Administer Issues ?	Administer Security Hotspots ?	Administer ?	Execute Analysis ?
A	Administrator	admin		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

1 of 1 shown

14. Run The Build.

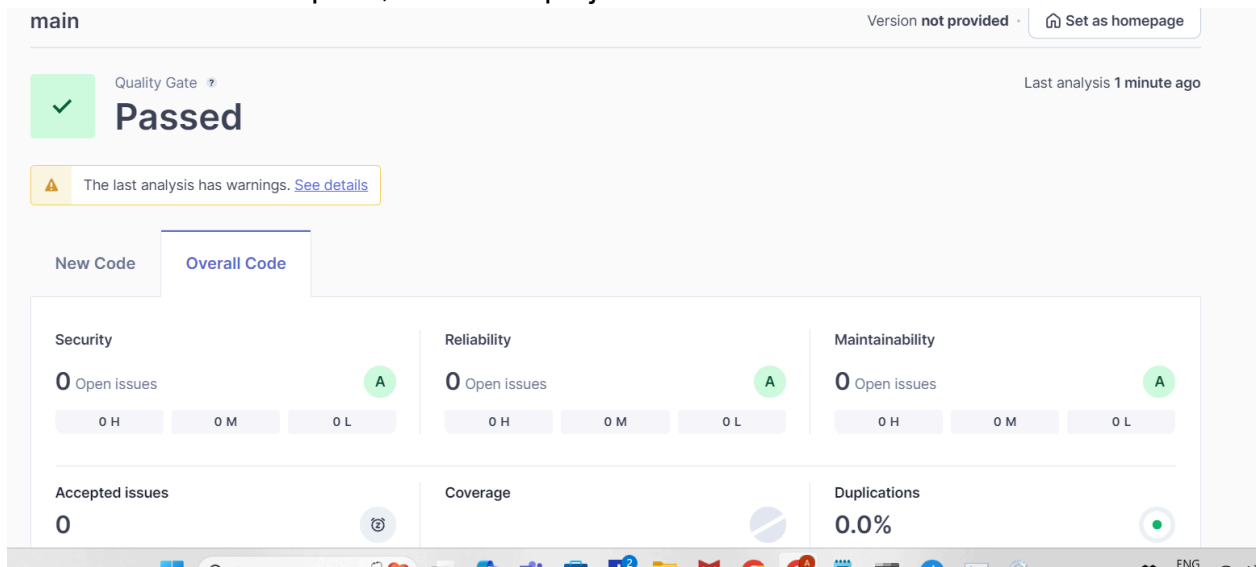
- Status
- Changes
- Workspace
- Build Now
- Configure
- Delete Project
- SonarQube
- Rename

15. Check the console output

```
Running as SYSTEM
Building on the built-in node in workspace C:\Users\ADITYA DUBEY\.jenkins\workspace\Sonarqube
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\Users\ADITYA DUBEY\.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
```

```
10:30:32.234 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube-test
10:30:32.234 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
10:30:32.235 INFO More about the report processing at http://localhost:9000/api/ce/task?id=96795973-9667-4456-a15c-3311dbc9d067
10:30:32.244 INFO Analysis total time: 9.533 s
10:30:32.245 INFO SonarScanner Engine completed successfully
10:30:32.300 INFO EXECUTION SUCCESS
10:30:32.301 INFO Total time: 14.090s
Finished: SUCCESS
```

16. Once the build is complete, check the project in SonarQube.



In this way, we have integrated Jenkins with SonarQube for SAST.

Conclusion: We began the experiment with installation of SonarQube Docker Image followed by setting up a new project in SonarQube. Then we installed the SonarQube scanner plugin and then created a new freestyle project in Jenkins with a Git repository for code analysis. Then we configured the Jenkins with appropriate settings to work with SonarQube. Gave permissions to Jenkins to perform code analysis. It is essential to provide correct properties in **Analysis Properties** for Jenkins to run correctly. The Jenkins project ran successfully with all tests passed in SonarQube.