

**Name: SHREYAS NAIK**

**Roll No:41**

**Div:D15B**

## **AdvDevOps-Exp 7**

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

### **Integrating Jenkins with SonarQube:**

Windows installation

Step 1 Install JDK 1.8

Step 2 download and install jenkins

<https://www.blazemeter.com/blog/how-to-install-jenkins-on-windows>

Ubuntu installation

<https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-20-04#installing-the-default-jre-jdk>

Step 1 Install JDK 1.8

sudo apt-get install openjdk-8-jre

sudo apt install default-jre

<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04>

[Open SSH](#)

### **Prerequisites:**

- [Jenkins installed](#)
- [Docker Installed](#) (for SonarQube)

(sudo apt-get install docker-ce=5:20.10.15~3-0~ubuntu-jammy docker-ce-cli=5:20.10.15~3-0~ubuntu-jammy containerd.io docker-compose-plugin)

- SonarQube Docker Image






### **Steps to integrate Jenkins with SonarQube**

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.
2. Run SonarQube in a Docker container using this command -

```
Microsoft Windows [Version 10.0.22621.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Akshad>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: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
17d3306b83f949e36c78e546922218bc5a5516806112e1b352e657add724f226

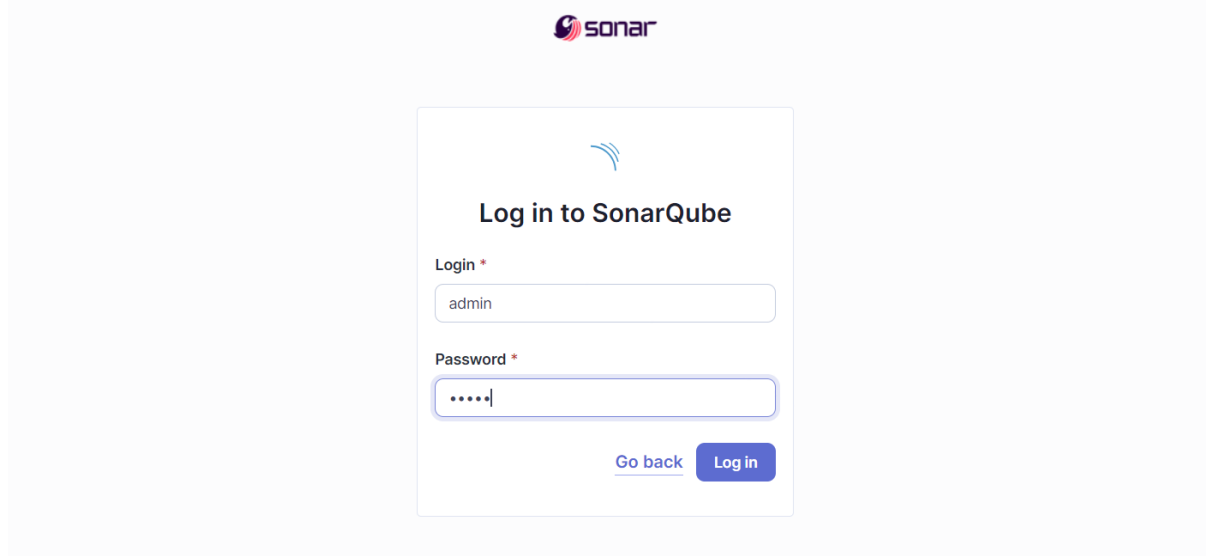
C:\Users\Akshad>
```

|   |   |                                  |         |                           |                      |   |   |   |
|---|---|----------------------------------|---------|---------------------------|----------------------|---|---|---|
|  |  <a href="#">sonarqube</a><br>17d3306b83f9 | <a href="#">sonarqube:latest</a> | Running | <a href="#">9000:9000</a> | 1.73% 41 minutes ago |  |  |  |
|---|---|----------------------------------|---------|---------------------------|----------------------|---|---|---|

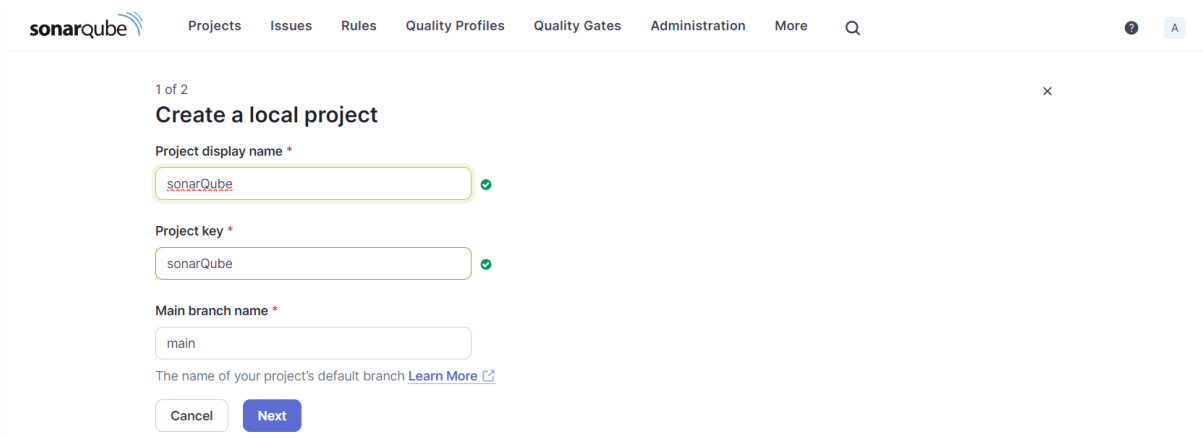
Warning: run below command only once

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

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*.
5. Create a manual project in SonarQube with the name **sonarqube**



1 of 2

**Create a local project**

Project display name \*

sonarQube

Project key \*

sonarQube

Main branch name \*

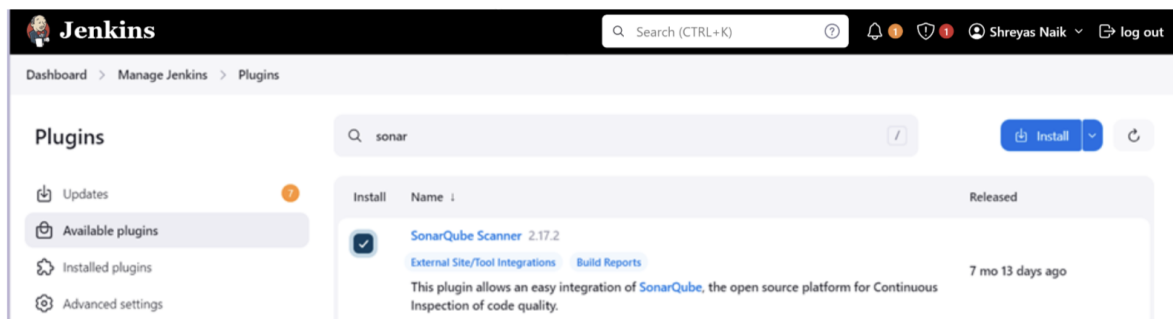
main

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

Cancel Next

Setup the project and come back to Jenkins Dashboard.

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



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

#### 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

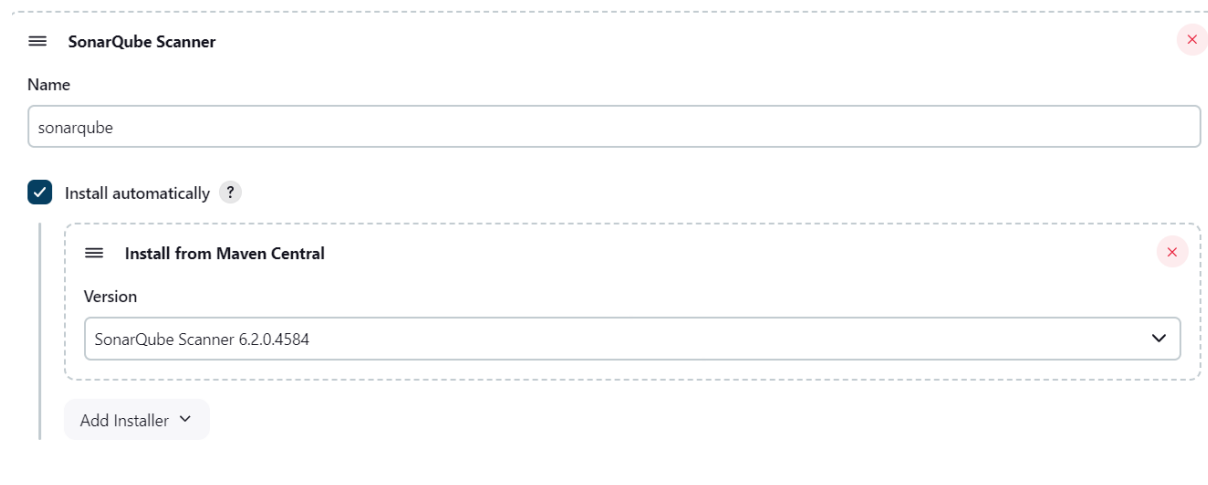
Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

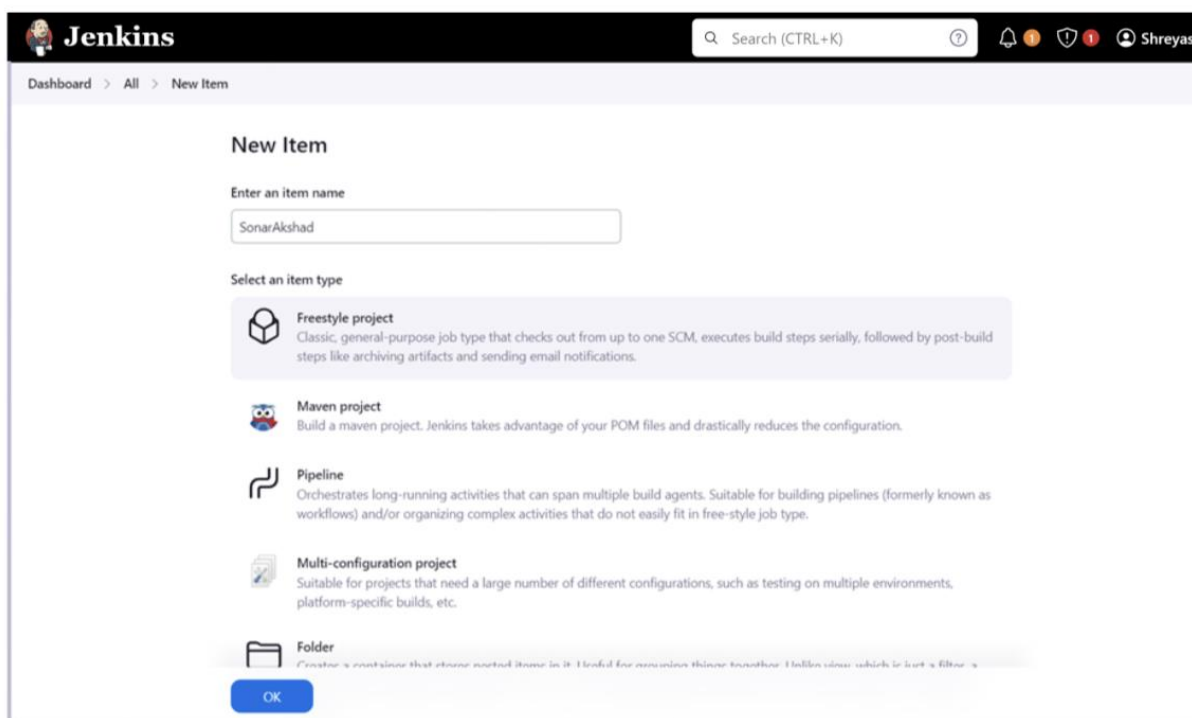
+ Add

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



The screenshot shows the 'SonarQube Scanner' configuration page in Jenkins. At the top, there is a menu icon and the title 'SonarQube Scanner'. Below it, the 'Name' field is set to 'sonarqube'. A checkbox labeled 'Install automatically' is checked. Below this, there is a section titled 'Install from Maven Central' which contains a 'Version' dropdown menu set to 'SonarQube Scanner 6.2.0.4584'. At the bottom of this section is an 'Add Installer' button.

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



The screenshot shows the 'New Item' page in Jenkins. The top navigation bar includes the Jenkins logo, a search bar, and the user name 'Shreyas'. The breadcrumb trail is 'Dashboard > All > New Item'. The main heading is 'New Item'. Below it, there is a text field for 'Enter an item name' with the value 'SonarAkshad'. Under the heading 'Select an item type', there are five options: 'Freestyle project' (Classic, general-purpose job type), 'Maven project' (Build a maven project), 'Pipeline' (Orchestrates long-running activities), 'Multi-configuration project' (Suitable for projects that need a large number of different configurations), and 'Folder' (Creates a container that stores related items). The 'Freestyle project' option is highlighted. At the bottom, there is an 'OK' button.

7. Choose this GitHub repository in Source Code Management.  
[https://github.com/shazforiot/MSBuild\\_firstproject.git](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 ?

https://github.com/shazforiot/MSBuild\_firstproject.git

Credentials ?

- none -

+ Add ▾

8. Under Build-> Execute SonarQube Scanner, enter these Analysis properties. Mention the SonarQube Project Key, Login, Password, Source path and Host URL.

9. Go to [http://localhost:9000/<user\\_name>/permissions](http://localhost:9000/<user_name>/permissions) and allow Execute Permissions to the Admin user.

## 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 ?                          |
|-----------------------|--------------------------|---|--------------------------|-----------------------------------|
| A Administrator admin | <input type="checkbox"/> | <input type="checkbox"/> Quality Gates<br><input type="checkbox"/> Quality Profiles | <input type="checkbox"/> | <input type="checkbox"/> Projects |

1 of 1 shown

11. Run The Build.



Status



Changes



Workspace



Build Now



Configure



Delete Project



GitHub



SonarQube



Rename

## SonarAkshad

This is SonarQube Experiment Adv DevOps



SonarQube

## Permalinks



Build History

trend ▾



Filter...



✓ #1



Sep 29, 2024, 8:48 PM



[Atom feed for all](#)



[Atom feed for failures](#)

Check the console output.

## ✓ Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Shreyas Naik `
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\SonarAkshad
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/shazforiot/MSBuild_firstproject.git
> git.exe init C:\ProgramData\Jenkins\jenkins\workspace\SonarAkshad # 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 config remote.origin.url https://github.com/shazforiot/MSBuild_firstproject.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> 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"
First time build. Skipping changelog.
Injecting SonarQube environment variables using the configuration: sonarqube
WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
Finished: SUCCESS
```

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

The screenshot displays the SonarQube web interface. At the top, there's a navigation bar with tabs: Overview, Issues, Security Hotspots, Measures, Code, and Activity. Below this, the 'Overview' tab is active, showing the 'QUALITY GATE STATUS' as 'Passed' with a green bar and the text 'All conditions passed'. To the right, under 'MEASURES', there are two tabs: 'New Code' and 'Overall Code'. The 'Overall Code' tab is selected, showing a grid of metrics: Bugs (0), Vulnerabilities (0), Security Hotspots (0), Debt (0), Code Smells (0), Reliability (A), Security (A), Security Review (A), and Maintainability (A). Below these, there are two more metrics: Duplications (0.0%) and Duplicated Blocks (0). At the bottom, the 'ACTIVITY' section shows a 'Passed' status with the SonarQube logo and a note: 'The main branch of this project is empty.' The last analysis was performed 7 hours ago.

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

## Conclusion

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