

Name: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

Sub: Advanced DevOps

Experiment No: 7

Date: 31/07/2022

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

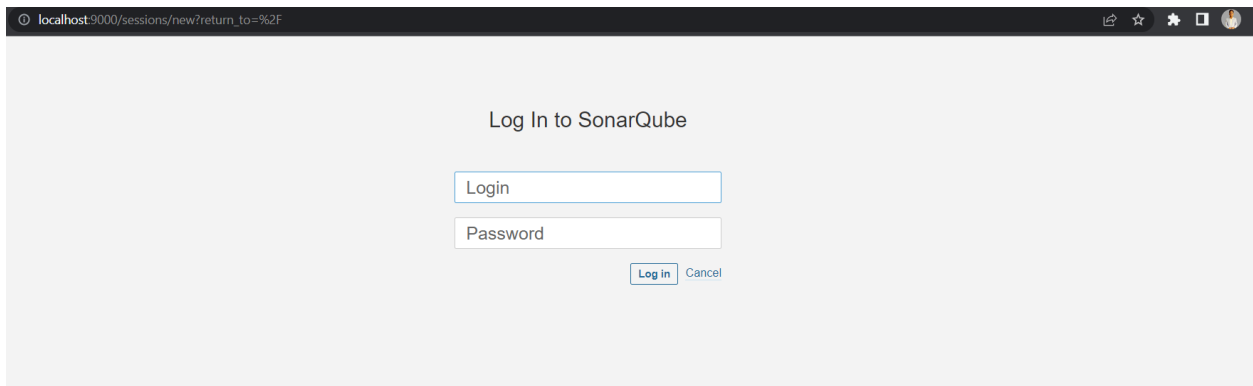
### Steps to integrate Jenkins with SonarQube

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 -

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

```
C:\Users\acer>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest  
fac04ce1b97fadf955ddf9cc9c4f041b2ccac89196d5f93a0bf8199a71c404d2  
C:\Users\acer>
```


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.


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.




From Azure DevOps

Set up global configuration




From Bitbucket

Set up global configuration



From GitHub


Set up global configuration



From GitLab

Set up global configuration

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




Manually

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

## Create a project


All fields marked with \* are required

**Project display name \***

Up to 255 characters. Some scanners might override the value you provide.

**Project key \***

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.

**Set Up**

Setup the project and come back to Jenkins Dashboard.

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

## Plugin Manager

Updates Available Installed Advanced

Name ↓	Enabled
<b>SonarQube Scanner for Jenkins</b> 2.14 This plugin allows an easy integration of <a href="#">SonarQube</a> , the open source platform for Continuous Inspection of code quality. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/>

7. 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** Enable injection of SonarQube server configuration as build environment variables

### SonarQube installations

List of SonarQube installations

**Name**

**Server URL**

Default is `http://localhost:9000`

**Server authentication token**

SonarQube authentication token. Mandatory when anonymous access is disabled.

Save

Apply

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

## SonarQube Scanner

### SonarQube Scanner installations

List of SonarQube Scanner installations on this system

Add SonarQube Scanner

☰

SonarQube Scanner

✕

Name

sonarqube

☒ Install automatically ?

☰

Install from Maven Central

✕

Version

SonarQube Scanner 4.7.0.2747

▼

Add Installer ▼

Save

Apply

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

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

## Source Code Management

☐ None

☒ Git ?

### Repositories ?

#### Repository URL ?

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

#### Credentials ?

- none -

+ Add

Advanced...

It is a sample hello-world project with no vulnerabilities and issues, just to test the integration.

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

```
sonar.projectKey = sonarqube
sonar.login = admin
sonar.password = <password>
sonar.sources =
C:\\WINDOWS\\config\\systemprofile\\AppData\\Local\\Jenkins\\Jenkins\\.Jenkins\\workspace\\SonarQube
sonar.host.url = https://127.0.0.1:9000
```

### Analysis properties ?

```
sonar.projectKey = sonarqube
sonar.login = admin
sonar.password = 
sonar.sources = C:\\WINDOWS\\config\\systemprofile\\AppData\\Local\\Jenkins\\Jenkins\\Jenkins\\workspace\\SonarQube
sonar.host.url = https://127.0.0.1:9000
```

## 12. Go to <http://localhost:9000//permissions> and allow Execute Permissions to the Admin user

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

### 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	Q Search for users or groups...	Administer System ⓘ	Administer ⓘ	Execute Analysis ⓘ	Create ⓘ
A	Administrator	admin		<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

1 of 1 shown

## 13. Run The Build

```
Started by user Hrishikesh Kumbhar
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\sonarqube
The recommended git tool is: NONE
using credential c74d8266-8c25-4ce0-b641-349af906b0fa
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\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 2.37.3.windows.1'
using GIT_ASKPASS to set credentials
> 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
[sonarqube] $ C:\ProgramData\Jenkins\jenkins\tools\udson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=sonarqube -Dsonar.login=admin -Dsonar.host.url=http://localhost:9000 -Dsonar.password=hrishi156
-Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\sonarqube
WARN: Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
INFO: Scanner configuration file: C:\ProgramData\Jenkins\jenkins\tools\udson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-
```

```
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
INFO: Sensor C# [csharp] (done) | time=2ms
INFO: Sensor Analysis Warnings import [csharp]
INFO: Sensor Analysis Warnings import [csharp] (done) | time=2ms
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=31ms
INFO: SCM Publisher SCM provider for this project is: git
INFO: SCM Publisher 4 source files to be analyzed
INFO: SCM Publisher 4/4 source files have been analyzed (done) | time=1231ms
INFO: CPD Executor Calculating CPD for 0 files
INFO: CPD Executor CPD calculation finished (done) | time=0ms
INFO: Analysis report generated in 214ms, dir size=119.2 kB
INFO: Analysis report compressed in 73ms, zip size=17.1 kB
INFO: Analysis report uploaded in 450ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=sonarqube
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AYOciLPsEPLnNonNJ0eh
INFO: Analysis total time: 31.291 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 56.365s
INFO: Final Memory: 15M/60M
INFO: -----
Finished: SUCCESS
```

## Conclusion:

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