Adv DevOps Practical 7

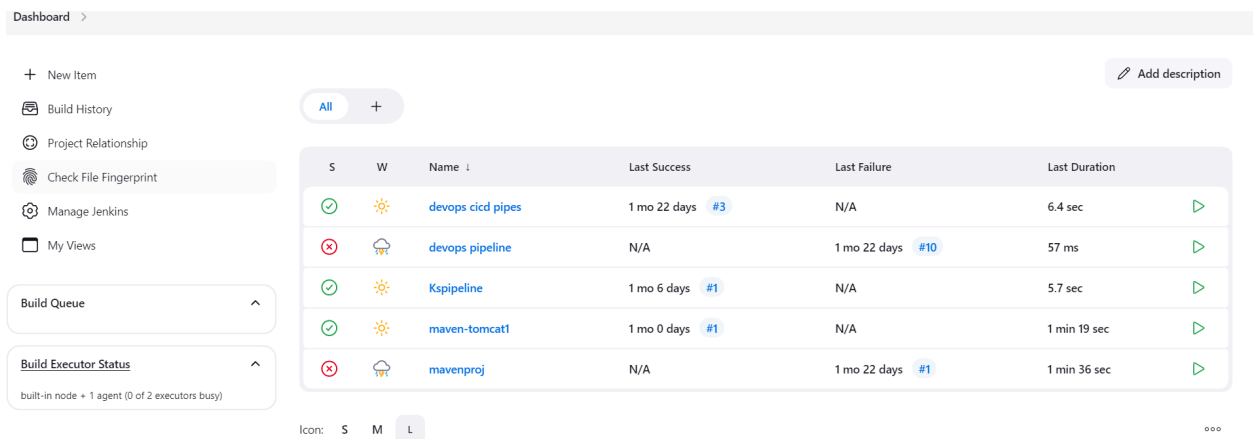**Aim:** To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Integrating Jenkins with SonarQube:

● Jenkins installed

● Docker Installed (for SonarQube)

● SonarQube Docker Image

Steps to integrate Jenkins with SonarQube

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

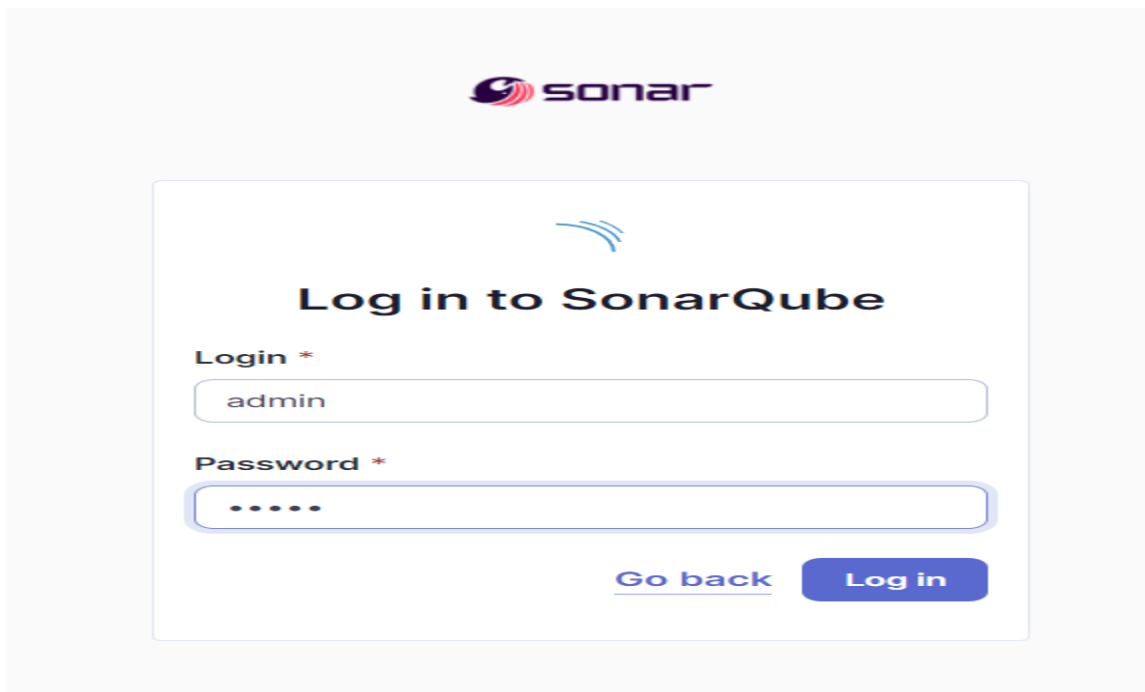2. Run SonarQube in a Docker container using this command -

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

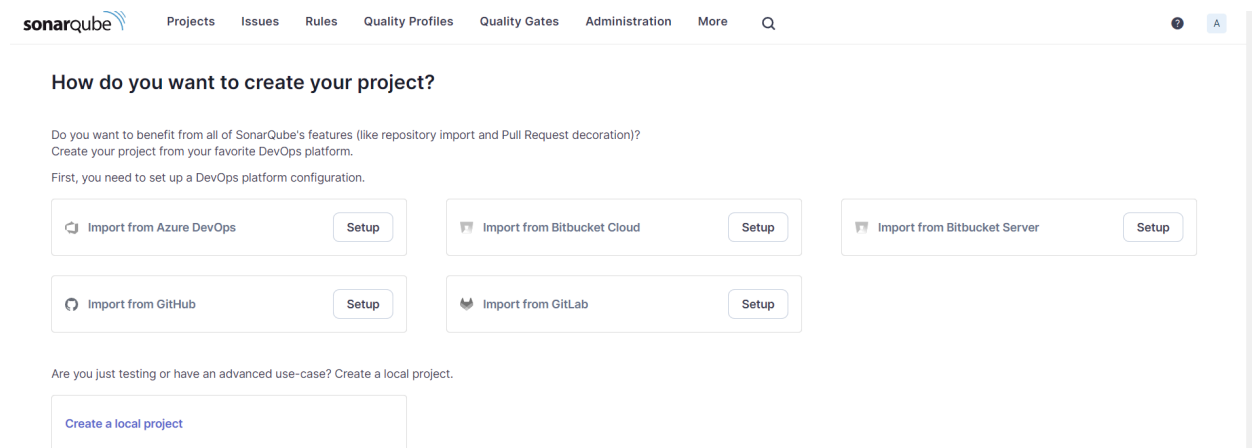**-------------------Warning: run below command only once**

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
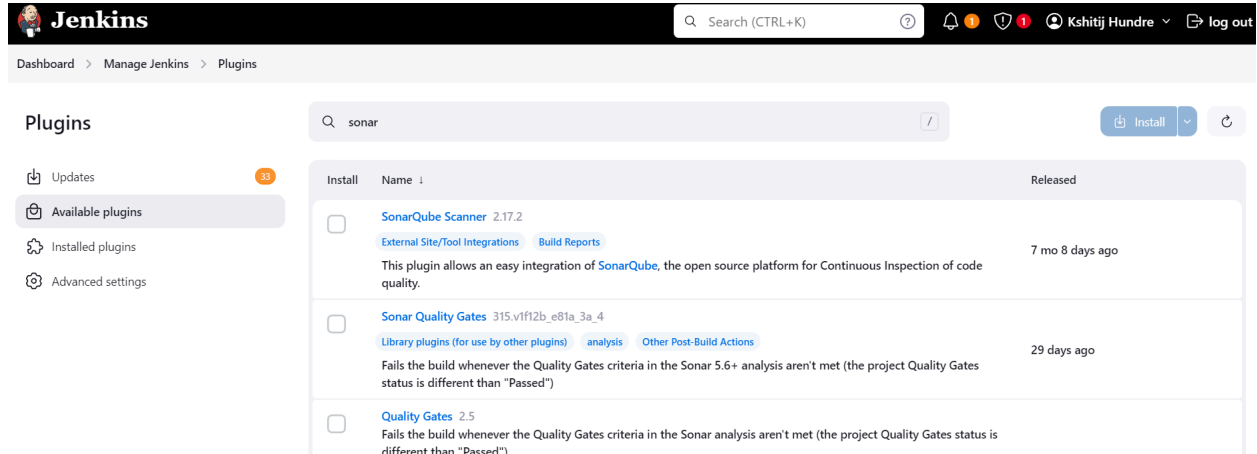
Setup the project and come back to Jenkins Dashboard.

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

6. Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for **SonarQube Servers** and enter the details.

Enter the Server Authentication token if needed.

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

sahilexp7

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

7. Search for SonarQube Scanner under Global Tool Configuration. Choose the

latest configuration and choose Install automatically.

Dashboard   >   Manage Jenkins   >   Tools

Check the "Install automatically" option.   →   Under name any name as identifier   →   Check the "Install automatically" option.

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

9. 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.

10. 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.
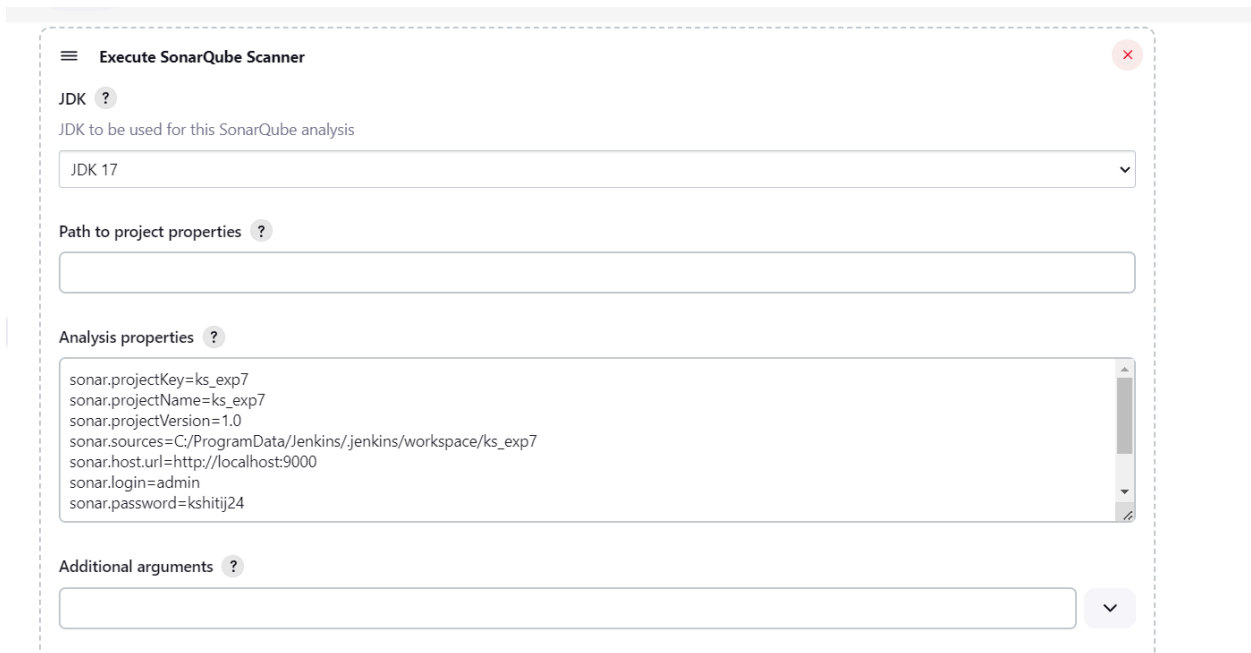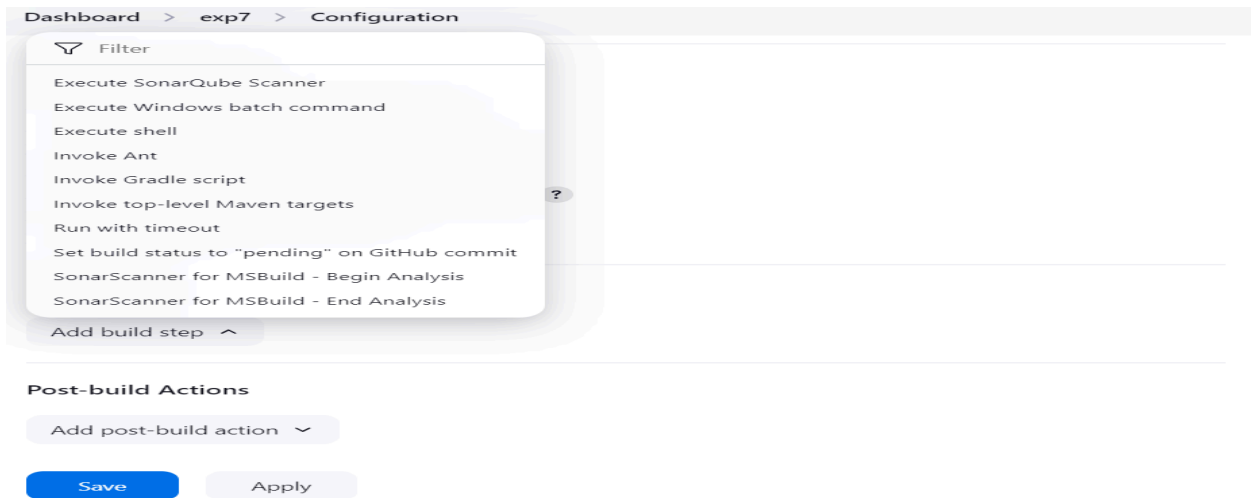
Dashboard > ks_exp7 >

📄 Status	✓ **ks_exp7**
</> Changes	
📁 Workspace	SonarQube
▷ Build Now	**Permalinks**
⚙ Configure	• **Last build (#7), 4 min 55 sec ago**
🗑 Delete Project	• **Last stable build (#7), 4 min 55 sec ago**
〰 SonarQube	• **Last successful build (#7), 4 min 55 sec ago**
✎ Rename	• **Last failed build (#6), 17 min ago**
	• **Last unsuccessful build (#6), 17 min ago**
	• **Last completed build (#7), 4 min 55 sec ago**

Build History   trend ⌄

🔍 Filter...   /

✓ **#7**
  Sep 25, 2024, 3:09 PM

✓ **Console Output**   ⬇ Download   📋 Copy   View as plai

```
Started by user Kshitij Hundre
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7\.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/* # timeou
 > 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
[ks_exp7] $ C:\ProgramData\Jenkins\.jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube1_exp7\bin\sonar-scanner.bat -
Dsonar.host.url=http://localhost:9000 -Dsonar.projectKey=ks_exp7 -Dsonar.projectName=ks_exp7 -Dsonar.host.url=http://localhost:9000 -
Dsonar.login=admin -Dsonar.projectVersion=1.0 -Dsonar.sources=C:/ProgramData/Jenkins/.jenkins/workspace/ks_exp7 -Dsonar.password=kshitij24 -
Dsonar.projectBaseDir=C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7
15:09:08.473 WARN  Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
```
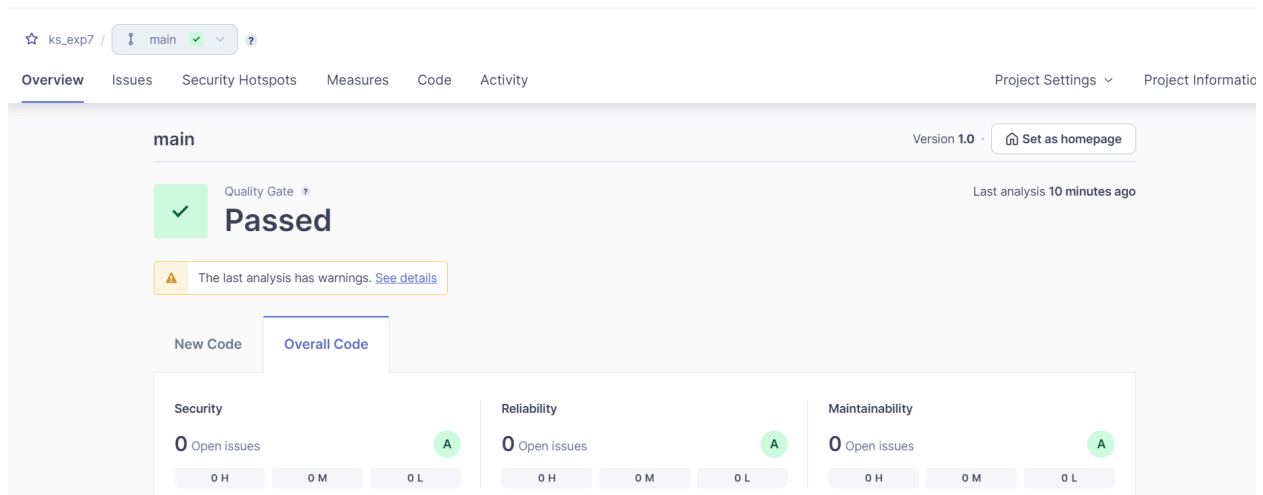
11. Go to http://localhost:9000/<user_name>/permissions and allow Execute Permissions to the Admin user.

	Administer System ?	Administer ?	Execute Analysis ?	Create ?
sonar-administrators System administrators	☑	☑ Quality Gates ☑ Quality Profiles	☐	☑ Projects
sonar-users Every authenticated user automatically belongs to this group	☐	☐ Quality Gates ☐ Quality Profiles	☑	☑ Projects
A **Administrator** admin	☑	☐ Quality Gates ☐ Quality Profiles	☑	☐ 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.	☐	☐ Quality Gates ☐ Quality Profiles	☐	☐ Projects

4 of 4 shown

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

☆ ks_exp7 / ↕ main ✓ ▾ ?

Overview　Issues　Security Hotspots　Measures　Code　Activity　　　　　Project Settings ▾　Project Informatio

main　　　　　　　　　　　　　　　Version **1.0** ·　🏠 Set as homepage

Quality Gate ?
✓ **Passed**　　　　　　　　　　Last analysis **10 minutes ago**

⚠ The last analysis has warnings. See details

New Code　**Overall Code**

Security			Reliability			Maintainability		
0 Open issues		Ⓐ	**0** Open issues		Ⓐ	**0** Open issues		Ⓐ
0 H	0 M	0 L	0 H	0 M	0 L	0 H	0 M	0 L

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

Conclusion:

In this project, we integrated Jenkins with SonarQube for automated static application security testing (SAST). We set up SonarQube using Docker, configured Jenkins with the necessary plugins and authentication, and linked it to a GitHub repository. The SonarQube scanner was added as a build step, enabling continuous code analysis for vulnerabilities, code smells, and quality issues, ensuring automated reporting and continuous code quality improvement.