

Adv DevOps Practical 7

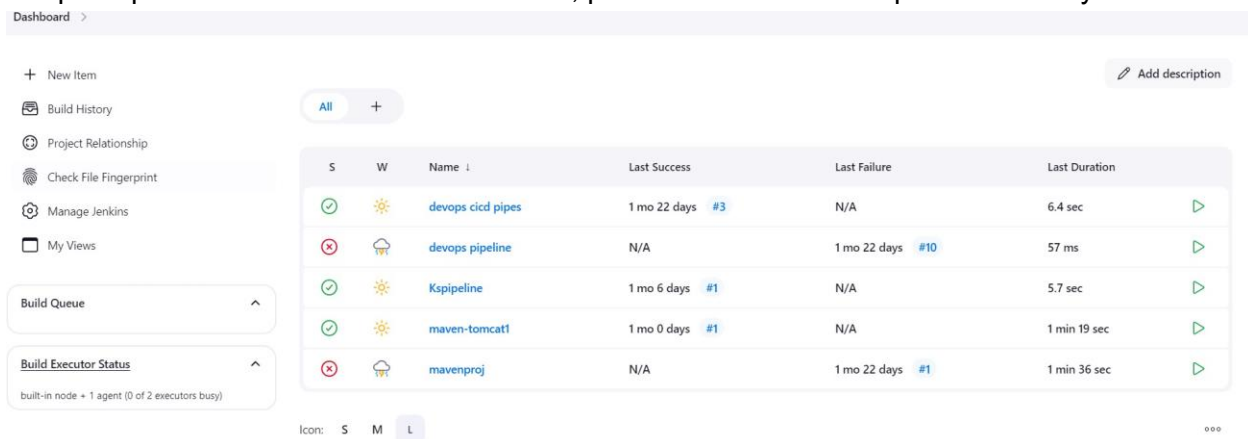
Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

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.



The screenshot shows the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. Below these are two expandable sections: 'Build Queue' and 'Build Executor Status'. The main area displays a table of build items. The table has columns for 'S' (Status), 'W' (Icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The table contains five rows of build items.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	devops cicd pipes	1 mo 22 days #3	N/A	6.4 sec
✗	☁	devops pipeline	N/A	1 mo 22 days #10	57 ms
✓	☀	Kspipeline	1 mo 6 days #1	N/A	5.7 sec
✓	☀	maven-tomcat1	1 mo 0 days #1	N/A	1 min 19 sec
✗	☁	mavenproj	N/A	1 mo 22 days #1	1 min 36 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

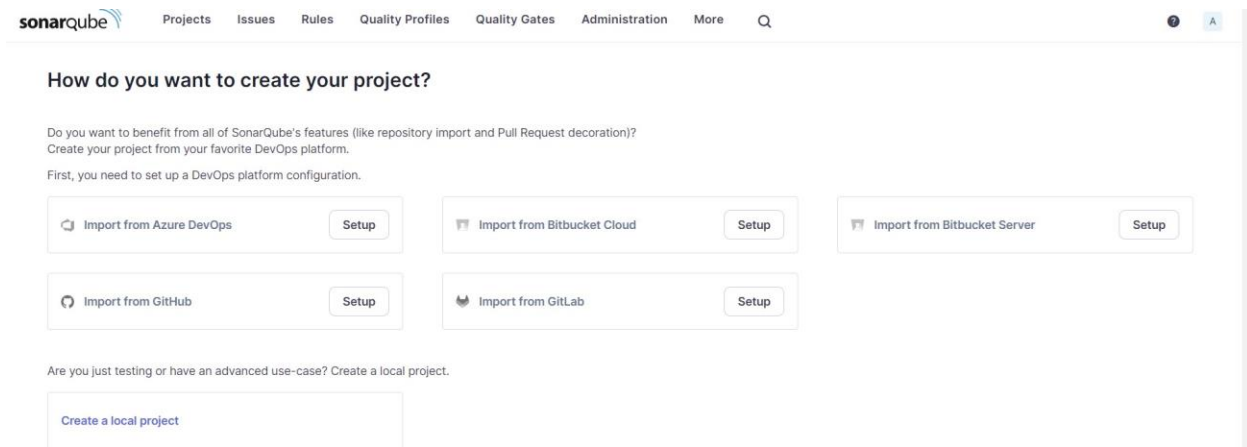
```
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull 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
77e678cded2ef5f989912d3d9e6991dd548eac03faa1eed68dd906614be53acc
PS C:\Users\91773\Desktop\College Resources\Exp7 adv devops>
```

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



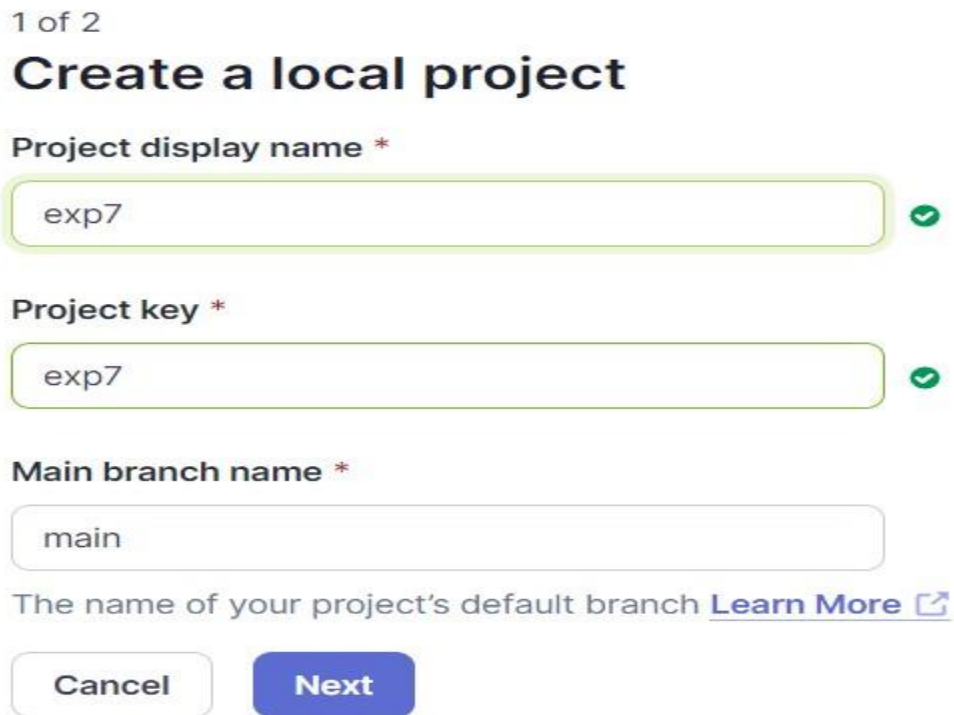
The image shows the SonarQube login page. At the top, there is the Sonar logo. Below it, the text "Log in to SonarQube" is displayed. There are two input fields: "Login *" with the value "admin" and "Password *" with masked characters ".....". At the bottom right, there are two buttons: "Go back" and "Log in".

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



The image shows the SonarQube web interface for creating a new project. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main heading is "How do you want to create your project?". Below this, there is a sub-heading "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." There are five buttons arranged in two rows: "Import from Azure DevOps", "Import from Bitbucket Cloud", "Import from Bitbucket Server", "Import from GitHub", and "Import from GitLab". Each button has a "Setup" link next to it. At the bottom, there is a link "Create a local project" under the heading "Are you just testing or have an advanced use-case? Create a local project."

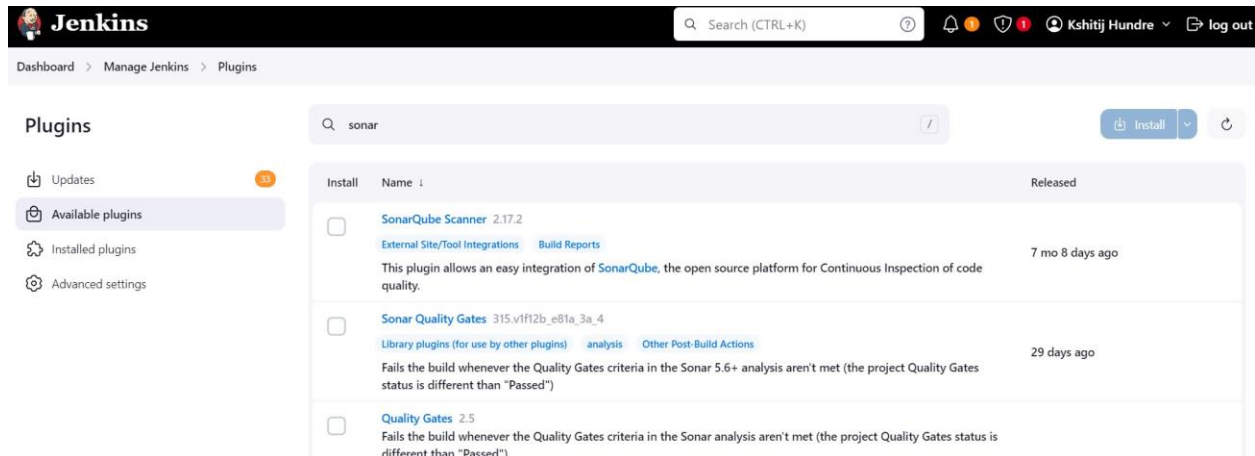
5. Create a manual project in SonarQube with the name sonarqube



The image shows the "Create a local project" form in SonarQube. The form has three input fields: "Project display name", "Project key", and "Main branch name". Each field has a green checkmark icon to its right, indicating that the input is valid. The "Project display name" field contains the text "exp7". The "Project key" field also contains the text "exp7". The "Main branch name" field contains the text "main". Below the input fields, there is a link "Learn More" with an external link icon. At the bottom of the form, there are two buttons: "Cancel" and "Next".

Setup the project and come back to Jenkins Dashboard.

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



The screenshot shows the Jenkins 'Manage Jenkins' > 'Plugins' page. A search bar at the top contains 'sonar'. The left sidebar shows 'Available plugins' selected. The main area displays a table of search results:

Install	Name	Released
<input type="checkbox"/>	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
<input type="checkbox"/>	Sonar Quality Gates 315.v1f12b_e81a_3a_4 Library plugins (for use by other plugins) analysis Other Post-Build Actions Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")	29 days ago
<input type="checkbox"/>	Quality Gates 2.5 Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")	

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>



The screenshot shows the 'SonarQube servers' configuration page in Jenkins. It includes a checkbox for 'Environment variables' which is checked. Below is a section for 'SonarQube installations' with a list of installations. One installation is shown with the following details:

- Name:** exp7
- Server URL:** http://localhost:9000 (Default is http://localhost:9000)
- Server authentication token:** - none - (Mandatory when anonymous access is disabled)

At the bottom, there is an '+ Add' button and an 'Advanced' dropdown menu.

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

Dashboard > Manage Jenkins > Tools

Dashboard > Manage Jenkins > Tools

Add Git ▾

Gradle installations

Add Gradle

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

Ant installations

Add Ant

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

☰ **SonarQube Scanner** ✕

Name

sonarqube_exp7

☒ Install automatically ?

☰ **Install from Maven Central** ✕

Version

SonarQube Scanner 6.2.0.4584 ▾

Add Installer ▾

Add SonarQube Scanner

Ant installations

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

New Item

Enter an item name

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

OK

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.

Dashboard > exp7 > Configuration

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add

Advanced

Add Repository

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 > exp7 > Configuration

Filter

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

SonarScanner for MSBuild - End Analysis

Add build step ^

Post-build Actions

Add post-build action v

Save Apply

Execute SonarQube Scanner


JDK ?
JDK to be used for this SonarQube analysis
JDK 17 v


Path to project properties ?


Analysis properties ?
sonar.projectKey=ks_exp7
sonar.projectName=ks_exp7
sonar.projectVersion=1.0
sonar.sources=C:/ProgramData/Jenkins/jenkins/workspace/ks_exp7
sonar.host.url=http://localhost:9000
sonar.login=admin
sonar.password=kshitij24


Additional arguments ?


Dashboard > ks_exp7 >


 Status


 Changes


 Workspace


 Build Now

 Configure


 Delete Project

 SonarQube

 Rename




ks_exp7



Permalinks

- [Last build \(#7\), 4 min 55 sec ago](#)
- [Last stable build \(#7\), 4 min 55 sec ago](#)
- [Last successful build \(#7\), 4 min 55 sec ago](#)
- [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 plain](#)

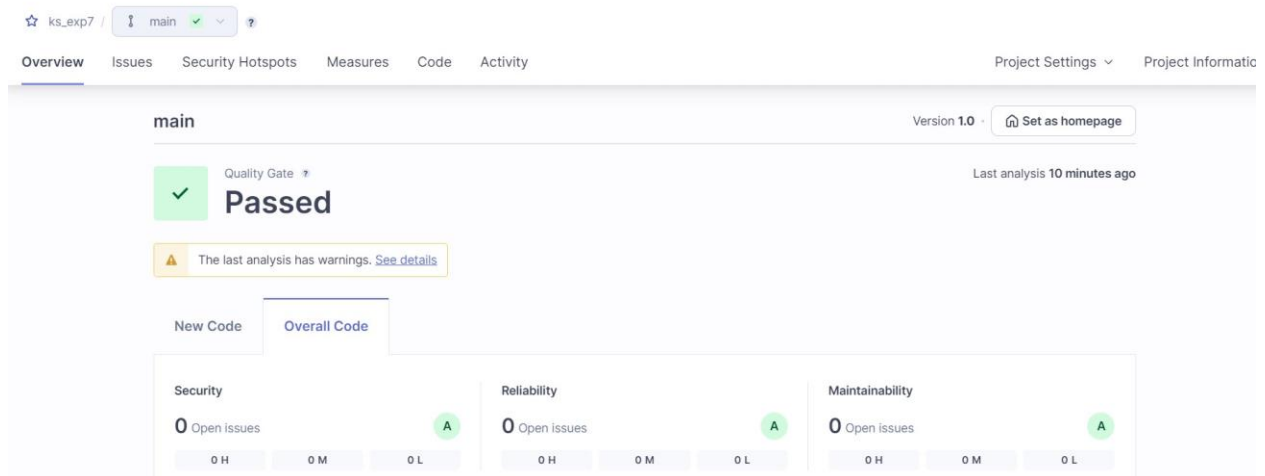
```
Started by user Kshitij Hundre
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\ks_exp7\.git # timeout=10
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/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # 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
[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	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
 sonar-users 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
 Administrator admin	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> 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.	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> 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 Information


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 0 Open issues  0 H 0 M 0 L

Reliability 0 Open issues  0 H 0 M 0 L

Maintainability 0 Open issues  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.