

EXPERIMENT 8 - ADVANCE DEVOPS

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web /

Theory:

What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that

analyzes source code to find security vulnerabilities that make your organization's applications

susceptible to attack. SAST scans an application before the code is compiled. It's also known as

white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not

require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues

without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they

pass the code to the next phase of the SDLC. This prevents security-related issues from being

considered an afterthought. SAST tools also provide graphical representations of the issues

found, from source to sink. These help you navigate the code easier. Some tools point out the

exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth

guidance on how to fix issues and the best place in the code to fix them, without requiring deep

security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as

during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

What is a CI/CD Pipeline?

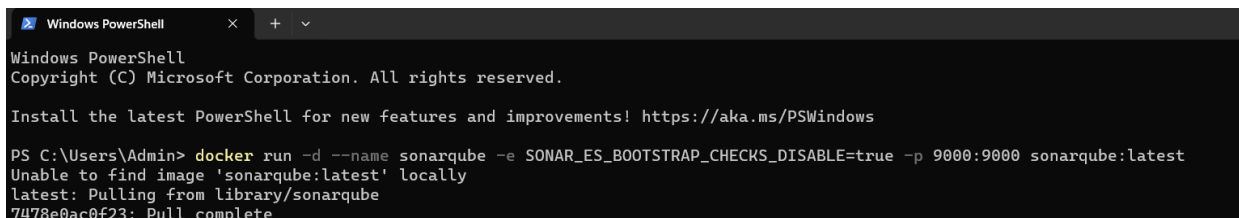
What is SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

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 -

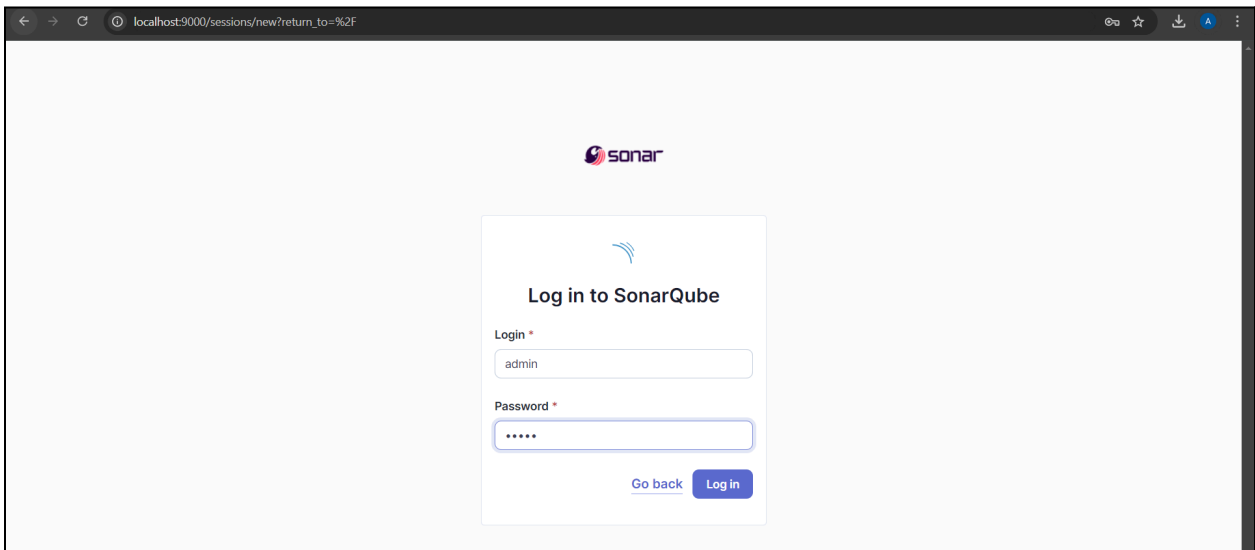
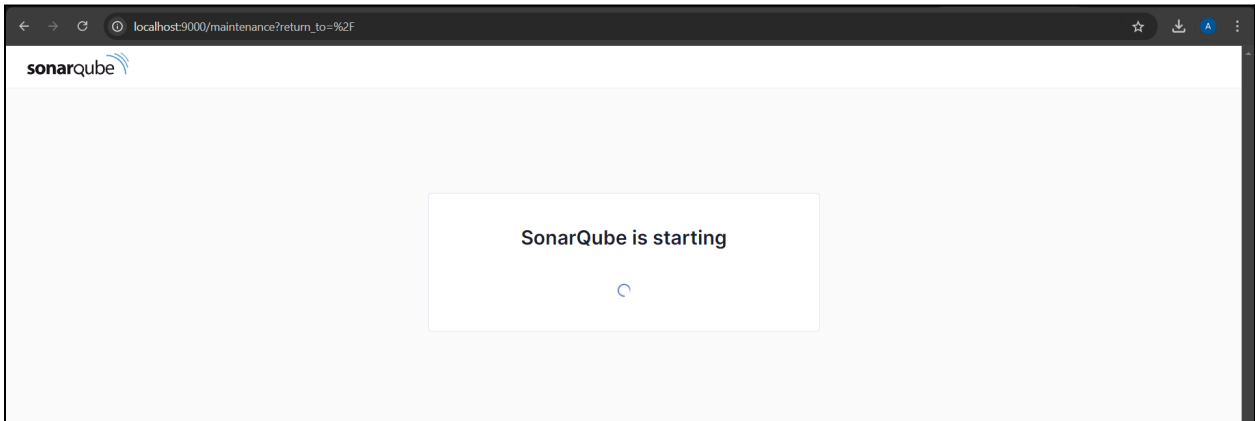


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> 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
```

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-test

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

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 you focus attention on the most recent changes to your project, enabling you to 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

☐ Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall

Jenkins

Search (CTRL+K)


Anushka Shahane log out

Dashboard > All


Enter an item name

sonarqube-test


> Required field

 **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.

 **Folder**

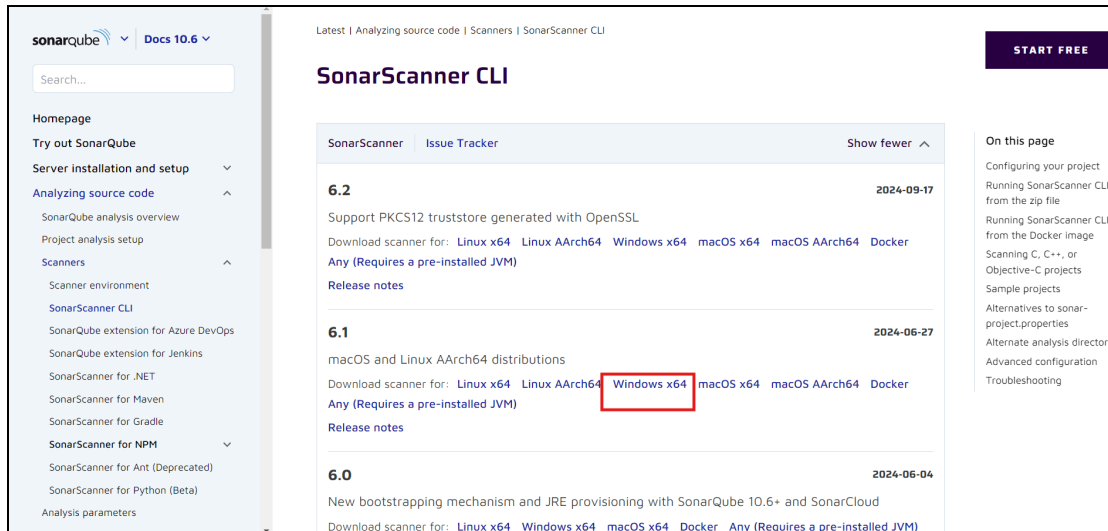
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

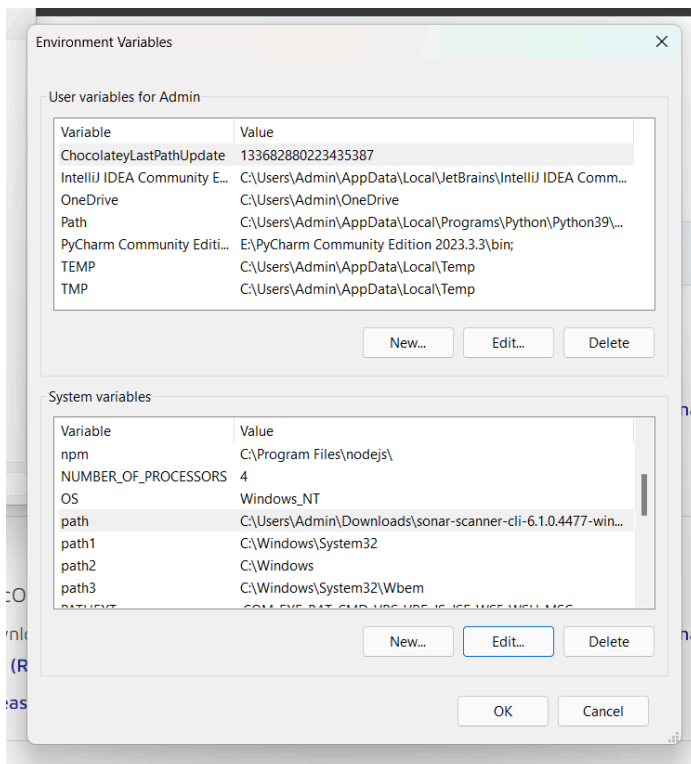
Cancel

Choose Pipeline

Step 6 :Go to [download_sonarscanner](#) to download sonar scanner



Step 7: After the download is complete, extract the file and copy the path to bin folder
Go to environment variables, system variables and click on path
Add a new path, paste the path copied earlier.



Step 8 : Save the pipeline and build it.

Pipeline

Definition

Pipeline script

Script ?

```
1 node {
2   stage('Cloning the GitHub Repo') {
3     git 'https://github.com/shazforiot/GOL.git'
4   }
5
6   stage('SonarQube analysis') {
7     withSonarQubeEnv('sonarqube') {
8       bat
9       C:/Users/Admin/Downloads/sonar-scanner-cli-6.1.0.4477-windows-x64/sonar-scanner-6.1.0.4477-windows-x64/bin/sonar-scanner.bat ^
10      -D sonar.login=admin ^
11      -D sonar.password=Anushka32 ^
12      -D sonar.projectKey=sonarqube-test ^
13      -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
14      -D sonar.host.url=http://localhost:9000/
15      .....
16    }
17  }
18 }
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save **Apply**

Output :

Jenkins Search (CTRL+K) Anushka Shahane log out

Dashboard > sonarqube-test >

Status **sonarqube-test** Add description Disable Project

Changes Build Now Configure Delete Pipeline Full Stage View Stages Rename Pipeline Syntax

Stage View

Average stage times: 16s 5s
(Average full run time: ~45s)

	Cloning the GitHub Repo	SonarQube analysis
#7 Sep 27 09:56 No Changes	38s	468ms
#6 Sep 27 09:44 No Changes	6s	3s failed
#5 Sep 27 01:54 No Changes	15s	4s failed

Build History trend

#7 Sep 27, 2024, 9:56 AM

#6

The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and user information for 'Anushka Shahane'. The breadcrumb trail indicates the current view is 'Dashboard > sonarqube-test > #7'. On the left sidebar, the 'Console Output' tab is selected. The main content area displays the console output for build #7, which is a pipeline job. The output shows the pipeline starting, cloning a repository from GitHub, and checking out a specific revision. The build is currently in a 'Waiting for resources' state, indicated by a green checkmark icon.

```
Started by user Anushka Shahane
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the Github Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> C:\Program Files\Git\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> git --version # 'git version 2.46.2.windows.1'
> C:\Program Files\Git\bin\git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse --refs/remotes/origin/master^(commit) # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
> C:\Program Files\Git\bin\git.exe branch -a -v --no-abbrev # timeout=10
```

Under different tabs, check all different issues with the code.