

**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 / Java / Python application.

## **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?**

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term 'pipeline'?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.



Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

### **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.

### **Benefits of SonarQube**

- **Sustainability** - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
- **Increase productivity** - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

- **Quality code** - Code quality control is an inseparable part of the process of software development.
- **Detect Errors** - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency** - Determines where the code criteria are breached and enhances the quality
- **Business scaling** - No restriction on the number of projects to be evaluated
- **Enhance developer skills** - Regular feedback on quality problems helps developers to improve their coding skills

## Integrating Jenkins with SonarQube:

### Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

**Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST** Open up Jenkins Dashboard on localhost, port 8080

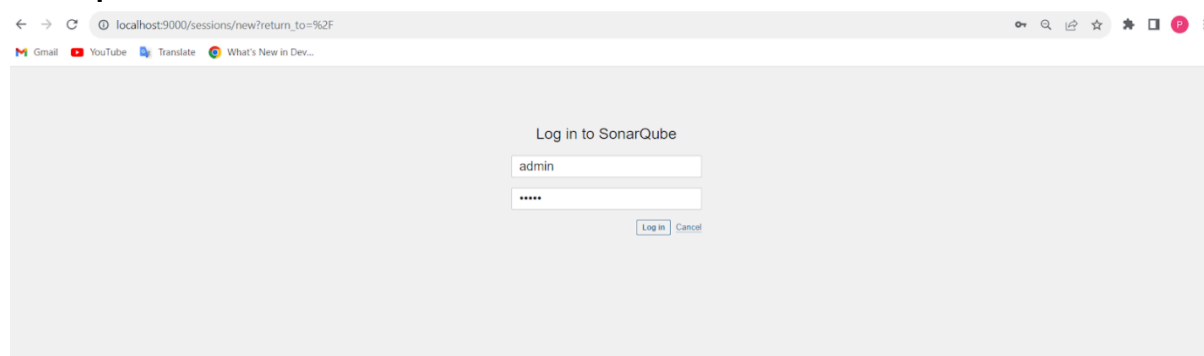
Run SonarQube in a Docker container using this command -

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

sonarqube:latest

```
PS D:\Desktop\DockerFile> 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
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
b795b715553d: Pull complete
c5244f6c9231: Pull complete
Digest: sha256:1fffd122cfb37ce982289dc7f5d38bb702ba05af7b5a50f7cb077ae25e60b5b9a
Status: Downloaded newer image for sonarqube:latest
1442c4e613b25aaedec05c060f020a00802b1c6dbaa27e8c5c0dad4ed8fc1f76
```

### sonarqube



**Create project manually**

## Create a project

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.

Main branch name \*

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

Next

The screenshot shows the SonarQube web interface. At the top, a green banner reads "Congratulations! Your project has been created". Below the navigation bar, the breadcrumb is "AdDevopsLab8 / main". The "Analysis Method" section is active, with the instruction: "Use this page to manage and set-up the way your analyses are performed." Under the heading "How do you want to analyze your repository?", there are several options: "With Jenkins", "With GitHub Actions", "With Bitbucket Pipelines", "With GitLab CI", "With Azure Pipelines", "Other CI" (with a note that SonarQube integrates with any CI tool), and "Locally" (with a note for testing or advanced use-cases). The "Locally" option is currently selected.

In Jenkins create a pipeline here named “SonarQube”

The screenshot shows the Jenkins "New Item" dialog. The "Enter an item name" field contains "SonarQube". Below this, a list of project types is shown: "Freestyle project", "Maven project", "Pipeline", "Multi-configuration project", "Folder", "Multibranch Pipeline", and "Organization Folder". The "Pipeline" option is highlighted with a blue border, indicating it is the selected choice.

Enter the following in pipeline script:

```
node {
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/PrajaktaUpadhye6/MSBuild_firstproject.git'
  }
  stage('SonarQube analysis') {
    withSonarQubeEnv('sonarqube') {
```

bat

"D:/sonar-scanner-cli-5.0.1.3006-windows/sonar-scanner-5.0.1.3006-windows/bin/sonar-scanner.bat \

```
-D sonar.login=admin \
-D sonar.password=abc \
-D sonar.projectKey=AdDevops \
-D sonar.exclusions=vendor/**,resources/**,/**/*.java \
-D sonar.host.url=http://127.0.0.1:9000/"
```

```
}
}
}
```

#### Definition

Pipeline script

#### Script ?

```
10
11 stage('SonarQube analysis') {
12   steps {
13     withSonarQubeEnv('SonarQube') {
14       bat '''
15       D:\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat ^
16       -Dsonar.login=admin ^
17       -Dsonar.password=9136591220 ^
18       -Dsonar.projectKey=sonarqube ^
19       -Dsonar.exclusions=vendor/**,resources/**,/**/*.java ^
20       -Dsonar.host.url=http://127.0.0.1:9000/
21       '''
22     }
23   }
24 }
25
26
```

☒ Use Groovy Sandbox ?

It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

Build and run:

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

🔄 GitHub

🔗 SonarQube

✎ Rename

📖 Pipeline Syntax

Build History

trend

Filter builds...

#8

Sep 20, 2023, 12:36 PM

Atom feed for all

Atom feed for failures

Pipeline SonarQube

Lab 8

Stage View

Average stage times:  
(Average full run time: ~36s)

	Cloning the GitHub Repo	SonarQube analysis
	2s	33s
#8	2s	33s

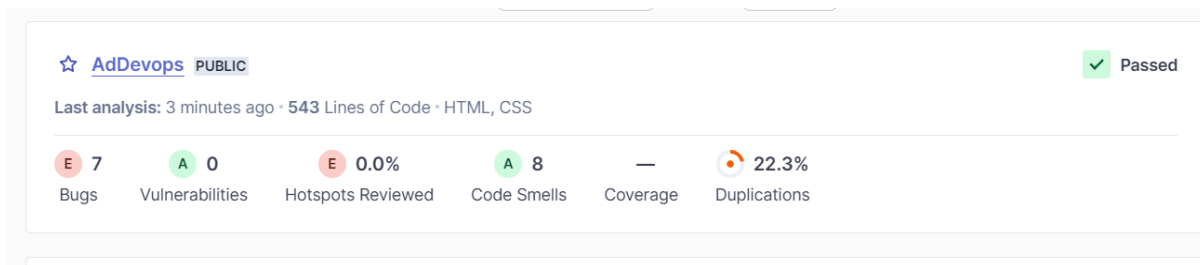
Permalinks

- Last build (#8), 1 min 44 sec ago
- Last stable build (#8), 1 min 44 sec ago
- Last successful build (#8), 1 min 44 sec ago
- Last completed build (#8), 1 min 44 sec ago

## Console output:

```
17:49:39.413 INFO CPD Executor CPD calculation finished (done) | time=159837ms
17:49:39.428 INFO SCM revision ID 'ba799ba7e1b576f04a4612322b0412c5e6e1e5e4'
17:52:16.075 INFO Analysis report generated in 4781ms, dir size=127.2 MB
17:52:33.696 INFO Analysis report compressed in 17599ms, zip size=29.6 MB
17:52:34.987 INFO Analysis report uploaded in 1290ms
17:52:34.989 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=sonarqube
17:52:34.989 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
17:52:34.989 INFO More about the report processing at http://127.0.0.1:9000/api/ce/task?id=8c04859e-050f-41e5-b320-ee32d44d4259
17:52:43.609 INFO Analysis total time: 16:49.875 s
17:52:43.612 INFO SonarScanner Engine completed successfully
17:52:44.416 INFO EXECUTION SUCCESS
17:52:44.419 INFO Total time: 17:04.839s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline completed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## sonarqube:



AdDevops / main

The last analysis has warnings. [See details](#) Version not provided


OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

Quality Gate Status

✓

Quality Gate Passed



Enjoy your sparkling clean code!

Measures

New CodeOverall Code

Reliability

7 Bugs

E

Maintainability

8 Code Smells

A

Security

0 Vulnerabilities

A

Security Review

6 Security Hotspots

E

Duplications

22.3% Duplications

## Reliability:

AdDevops / main

The last analysis has warnings. [See details](#) Version not provided

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

Project Overview

Reliability

Overview

New Code

Bugs0

RatingA

Remediation Effort0

Overall Code

Bugs7

AdDevops

View asList

Select files

Navigate

15 files

Reliability Rating E

[See history](#)

New Code: Since September 20, 2023

scroll.cssE

trial.cssC

verticaltable.htmlC

demo.htmlB

index.htmlB

payment.htmlD

There are 9 hidden components with a score of A.

Show Them

## Bugs:

AdDevops / main

The last analysis has warnings. [See details](#) Version not provided

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

Bugs0

RatingA

Remediation Effort0

Overall Code

Bugs7

RatingE

Remediation Effort28min

Security

Security Review

Bugs 7

[See history](#)

New Code: Since September 20, 2023

box.html0

class.css0

demo.html1

element.css0

float property.html0

home.html0

id.css0

index.html1

payment.html1

scroll.css1

## Maintanaibility:

AdDevops / main ? The last analysis has warnings. [See details](#) Version not provided 🔗

Overview Issues Security Hotspots **Measures** Code Activity Project Settings Project Information

Project Overview
Reliability ?
Security ?
Security Review ?
Maintainability ?

Overview
New Code
Code Smells 0

AdDevops
View as List Select files Navigate 15 files

**Maintainability Rating** A [See history](#) New Code: Since September 20, 2023

box.html	A
class.css	A
demo.html	A
element.css	A
float property.html	A
home.html	A
id.css	A
index.html	A

## Security:

AdDevops / main ? The last analysis has warnings. [See details](#) Version not provided 🔗

Overview Issues Security Hotspots **Measures** Code Activity Project Settings Project Information

Project Overview
Reliability ?
Security ?
Security Review ?

New Code
Security Hotspots 0
Rating A
Overall Code
Security Hotspots 6

AdDevops
View as List Select files Navigate 15 files

**Security Review Rating** E [See history](#) New Code: Since September 20, 2023

demo.html	E
home.html	E
index.html	E

There are 12 hidden components with a score of A. [Show Them](#)

## Duplications:

AdDevops / main ? The last analysis has warnings. [See details](#) Version not provided 🔗

Overview Issues Security Hotspots **Measures** Code Activity Project Settings Project Information

Project Overview
Reliability ?
Security ?
Security Review ?
Maintainability ?
Coverage
Duplications
Overview

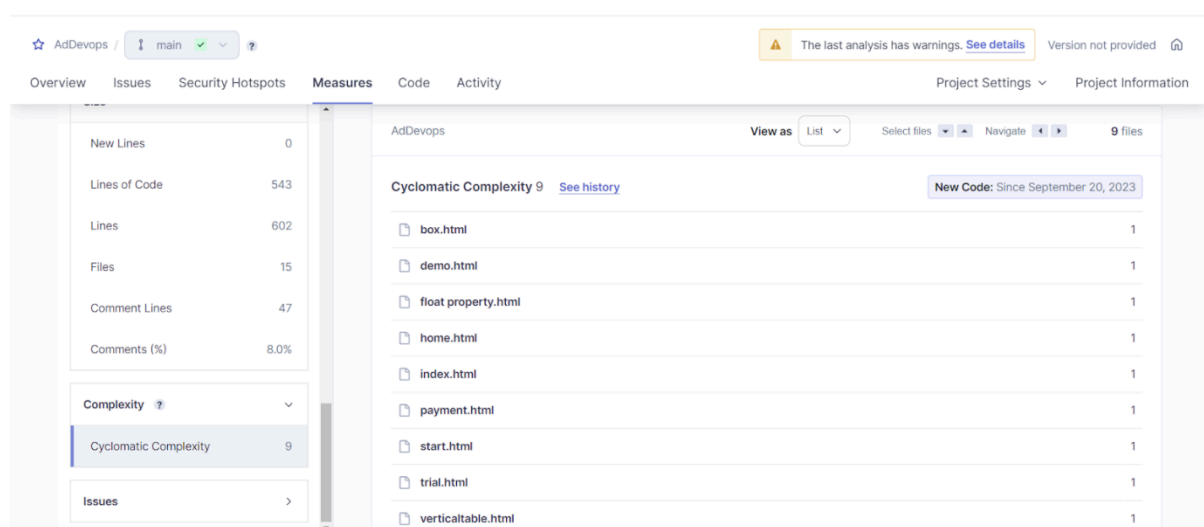
AdDevops
View as List Select files Navigate 15 files

**Duplicated Lines (%)** 22.3% [See history](#) New Code: Since September 20, 2023

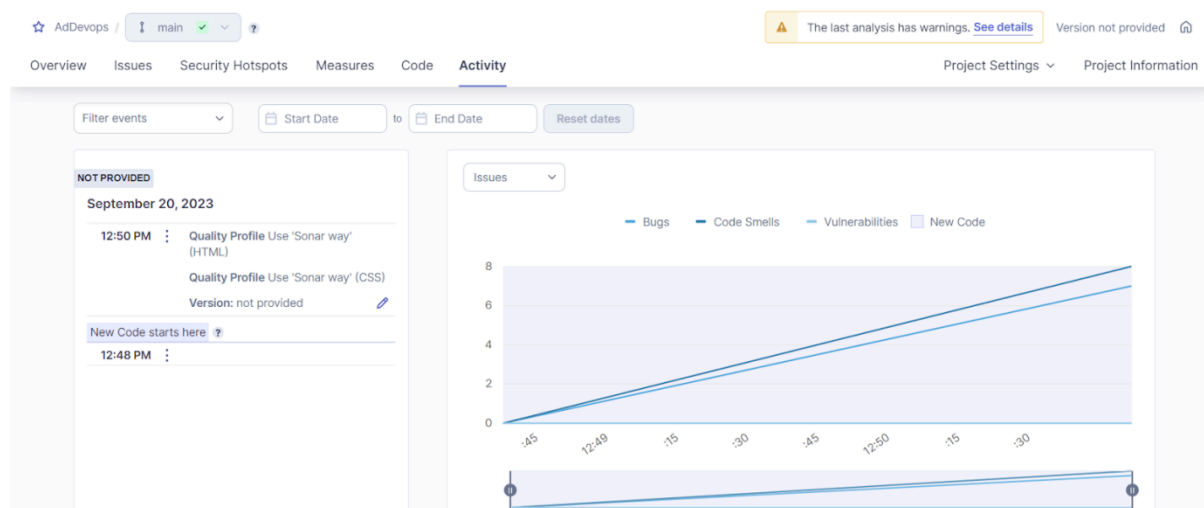
	Duplicated Lines (%)	Duplicated Lines
demo.html	100%	67
index.html	100%	67
box.html	0.0%	0
class.css	0.0%	0
element.css	0.0%	0
float property.html	0.0%	0
home.html	0.0%	0

## Cyclomatic complexity:





## Activity:



**Conclusion:** Thus, we have successfully integrated Jenkins with SonarQube.