



# Vivekanand Education Society's

## Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

### Department of Information Technology

A.Y. 2024-25

## Advance DevOps Lab

### Experiment 08

Aim: To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.

Roll No.	22
Name	Sarthak Harade
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements.  LO4: To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.
Grade:	

- **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.

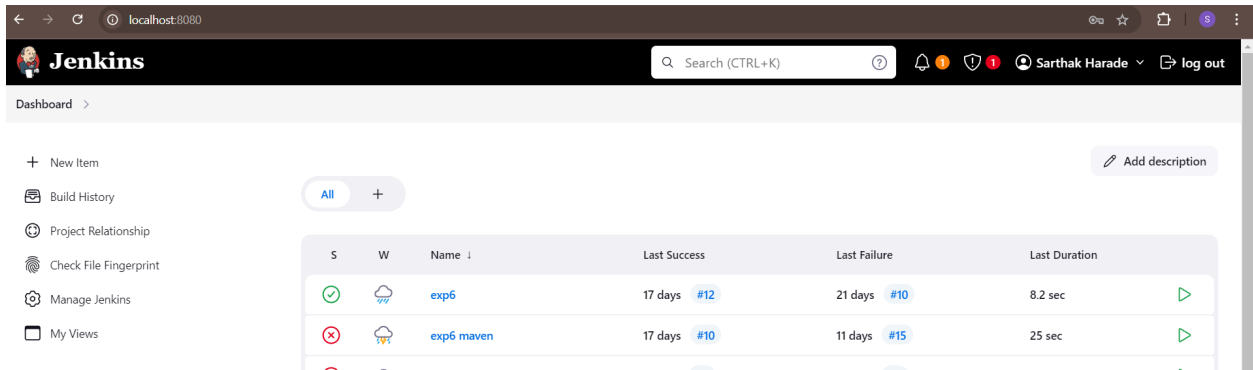
## **Integrating Jenkins with SonarQube:**

### **Prerequisites:**

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

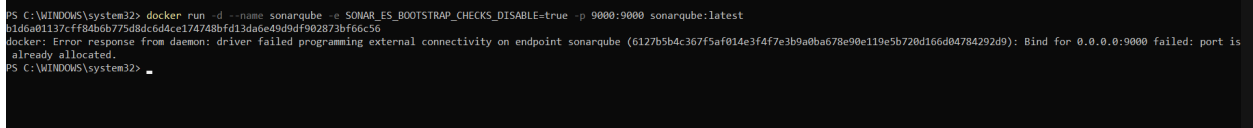
# Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.

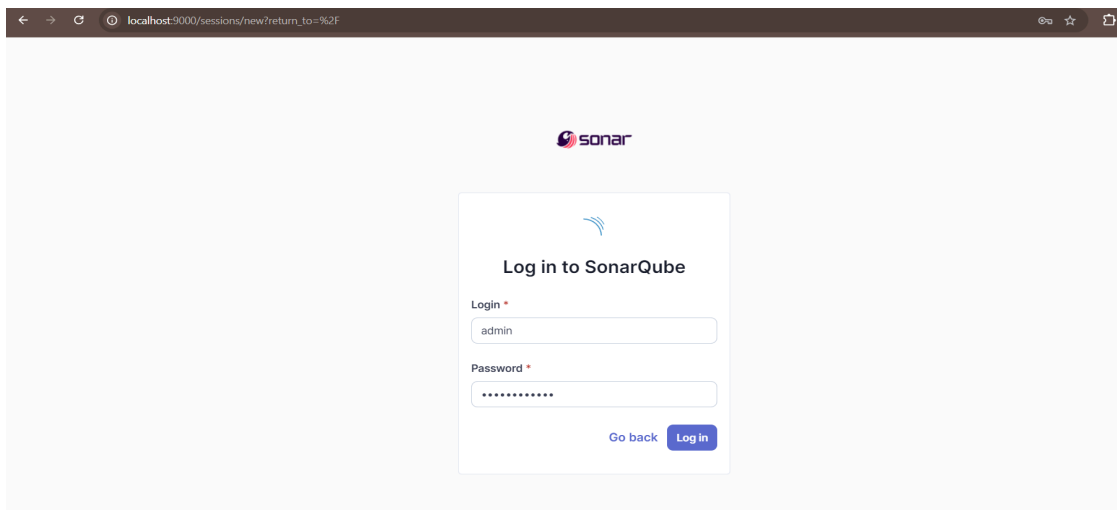


2. Run SonarQube in a Docker container using this command -

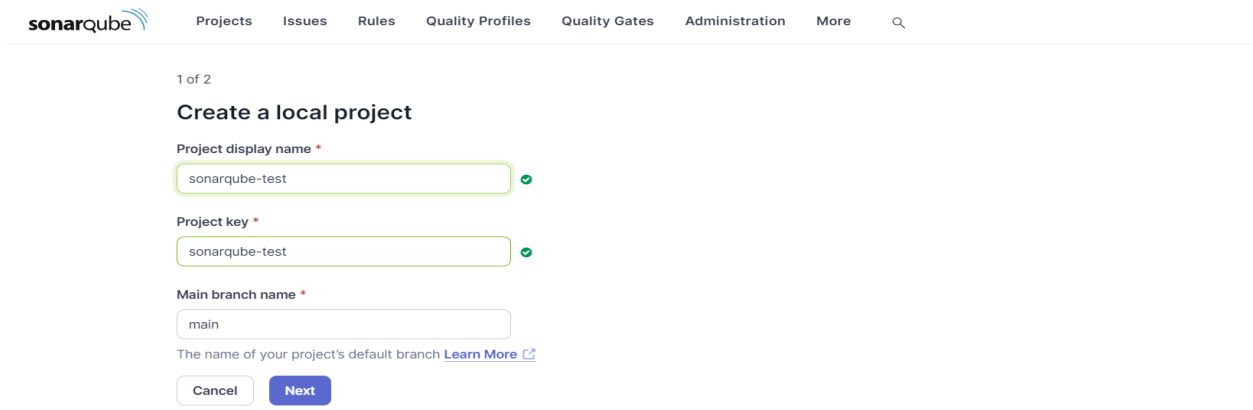
```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```



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**



The screenshot shows the 'Create a local project' form in SonarQube. The form has three input fields: 'Project display name' with the value 'sonarqube-test', 'Project key' with the value 'sonarqube-test', and 'Main branch name' with the value 'main'. Each field has a green checkmark icon to its right. Below the 'Main branch name' field, there is a link 'Learn More' and a note: 'The name of your project's default branch'. At the bottom of the form are two buttons: 'Cancel' and 'Next'.

1 of 2

### Create a local project

Project display name \*

sonarqube-test ✓

Project key \*

sonarqube-test ✓

Main branch name \*

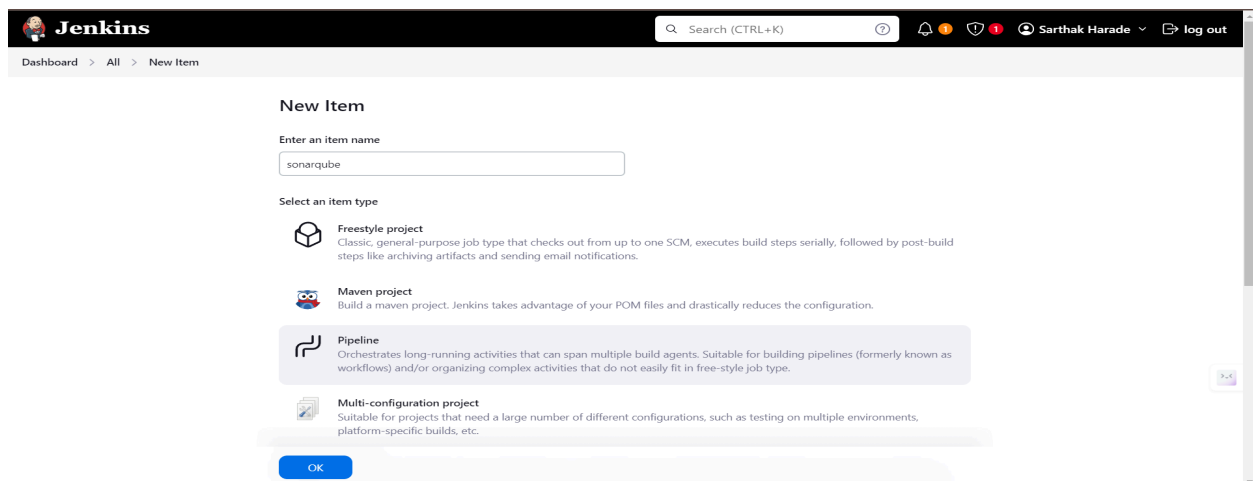
main

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

Cancel Next

The second screenshot shows the SonarQube project page for 'sonarqube-test'. A green notification box at the top right says 'Your project has been created.' The breadcrumb trail is 'sonarqube-test / main'. The navigation tabs include Overview, Issues, Security Hotspots, Measures, Code, and Activity. The 'Analysis Method' section is active, showing options for how to analyze the repository: With Jenkins, With GitHub Actions, With Bitbucket Pipelines, With GitLab CI, With Azure Pipelines, and Other CI. The 'Other CI' option is selected, with a description: 'SonarQube integrates with your workflow no matter which CI tool you're using.'

6. Create a New Item in Jenkins, choose **Pipeline**.



The screenshot shows the 'New Item' form in Jenkins. The 'Enter an item name' field contains the value 'sonarqube'. Under the 'Select an item type' section, four options are listed: 'Freestyle project', 'Maven project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is highlighted with a blue background. At the bottom of the form is an 'OK' button.

### New Item

Enter an item name

sonarqube

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 environments, platform-specific builds, etc.

OK

7. Under Pipeline Script, enter the following -

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.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=Sarthak@1234 \
                -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 ?

8. 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

## Pipeline SonarQube

Lab 8

### Stage View

Average stage times: (Average full run time: ~36s)	
Cloning the GitHub Repo	SonarQube analysis
2s	33s

#8

Sep 20 12:36

No Changes

### 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

### Build History

trend

Filter builds...

Sep 20, 2023, 12:36 PM

Atom feed for all

Atom feed for failures

## 9.Console output:

Status

Changes

Console Output

Edit Build Information

Delete build '#5'

Timings

Git Build Data

Previous Build

## Console Output

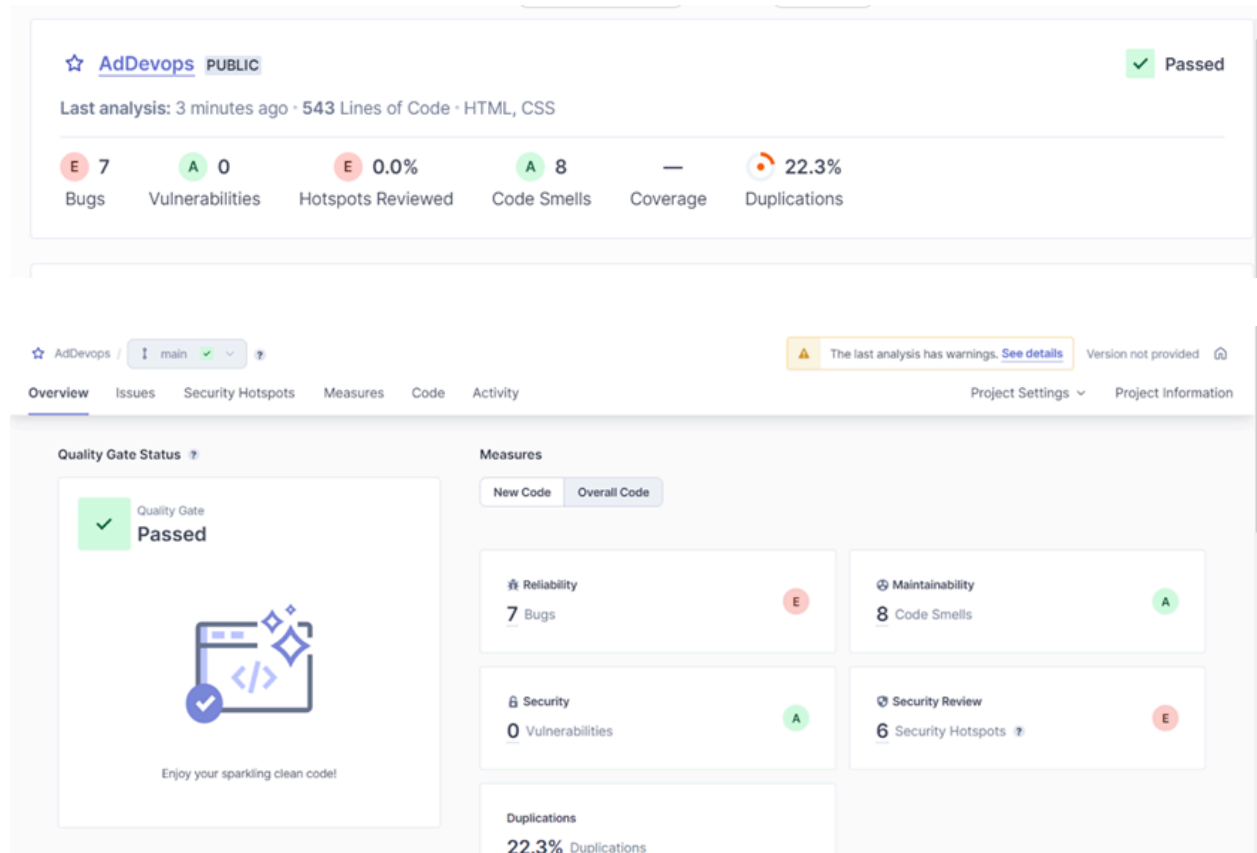
Download Copy View as plain text

```

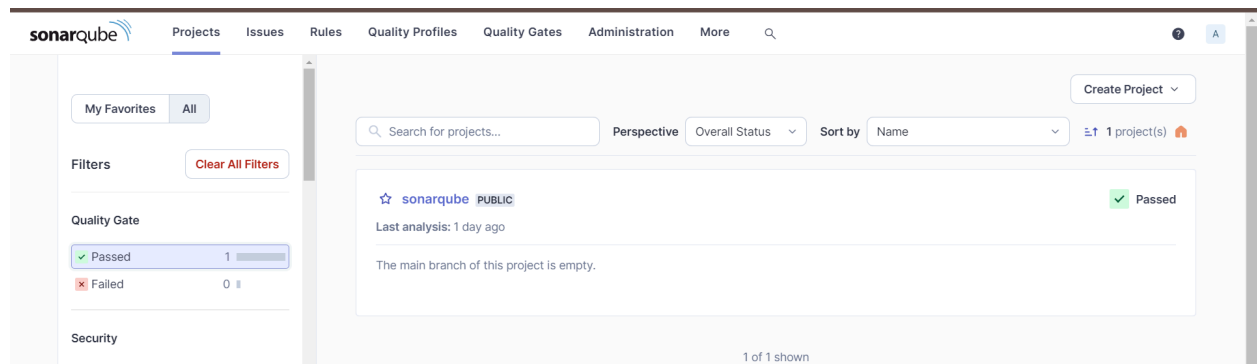
Started by user Sarthak Harade
Running as SYSTEM
Building on the built-in node in workspace C:\ProgramData\Jenkins\jenkins\workspace\sarthak-sonarqube
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\sarthak-sonarqube\.git # 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 2.46.0.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" # 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
[sarthak-sonarqube] $ C:\ProgramData\Jenkins\jenkins\tools\hudson.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.sources=. -
Dsonar.password=Sarthak@1234 -Dsonar.projectBaseDir=C:\ProgramData\Jenkins\jenkins\workspace\sarthak-sonarqube
08:44:17.047 WARN Property 'sonar.host.url' with value 'http://localhost:9000' is overridden with value 'http://localhost:9000'
08:44:17.060 INFO Scanner configuration file:
C:\ProgramData\Jenkins\jenkins\tools\hudson.plugins.sonar.SonarRunnerInstallation\sonarqube\bin\..\conf\sonar-scanner.properties
08:44:17.063 INFO Project root configuration file: NONE
08:44:17.087 INFO SonarScanner CLI 6.2.1.4610
08:44:17.088 INFO Java 21.0.3 Oracle Corporation (64-bit)

```

## 10.sonarqube:



## 11. Reliability:



## 12. Bugs



AdDevops / main [dropdown] [help]

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

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

Bugs 0

Rating

Remediation Effort 0

Overall Code

**Bugs 7**

Rating

Remediation Effort 28min

Security [help] >

Security Review [help] >

Bugs 7 [See history](#)

New Code: Since September 20, 2023

box.html	0
class.css	0
demo.html	1
element.css	0
float property.html	0
home.html	0
id.css	0
index.html	1
payment.html	1
scroll.css	1

## 13.Maintainability

AdDevops / main [dropdown] [help]

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

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

Project Overview

Reliability [help] >

Security [help] >

Security Review [help] >

**Maintainability [help] v**

Overview

New Code

Code Smells 0

AdDevops View as List Select files [dropdown] [icon] Navigate [icon] [icon] 15 files

Maintainability Rating [See history](#)

New Code: Since September 20, 2023

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

## 13.Security

AdDevops / main [check] [help]

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

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

Project Overview
Reliability [help] >
Security [help] >
Security Review [help] v
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](#)

## 14.Duplications

AdDevops / main [check] [help]

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

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

Project Overview
Reliability [help] >
Security [help] >
Security Review [help] >
Maintainability [help] >
Coverage >
Duplications v
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

- **Conclusion:**

In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc. Thus, we have successfully integrated Jenkins with SonarQube.

