

1. Introduction

Case Study Overview:

This case study revolves around a DevOps-centric approach to software development and deployment. We will utilize Jenkins and SonarQube to implement a Continuous Integration (CI) pipeline with static code analysis. The focus is on setting up a Jenkins pipeline on AWS Cloud9 to automate the static code analysis of Python and Java applications via SonarQube. The objective is to ensure the reliability and maintainability of code through automatic checks for potential errors and security vulnerabilities during each integration.

Key Feature and Application:

The core of this study lies in integrating Jenkins for CI and SonarQube for static code analysis. This integration automates the process of analyzing code quality, helping developers detect bugs and vulnerabilities early in the development process. With this automation, coding standards can be enforced, and potential risks like code smells or security weaknesses can be identified before deployment.

Practical Application:

- **Jenkins:** Automates the CI pipeline, triggering builds and static code analysis whenever code changes occur.
- **SonarQube:** Performs static analysis to provide feedback on the maintainability, reliability, and security of the code.
- **AWS Cloud9:** Serves as the development environment, offering a cloud-based IDE that integrates smoothly with AWS services.

Third-Year Project Integration (EventEase Project):

The EventEase project, which I developed during my third year, can benefit from the integration of this CI/CD pipeline approach to optimize the development process. EventEase is a comprehensive event management platform where users can organize events, manage attendees, and schedule activities seamlessly. The platform also provides features such as real-time notifications, ticketing, and user history tracking for past events.

By incorporating Jenkins and SonarQube into EventEase, the following improvements can be achieved:

- **Automated Testing:** Every new feature, bug fix, or update will trigger automated tests, ensuring that the platform remains stable and bug-free after each code commit.
- **Code Quality:** SonarQube will continuously monitor the quality of the codebase, making sure the project grows with clean, maintainable code, which in turn reduces technical debt.
- **Security Scanning:** SonarQube's security checks will help detect vulnerabilities early, ensuring that sensitive data like event details and user information is protected before the code is deployed to production.

2. Demonstration

Problem Statement

1] Jenkins on an EC2

instance:Resource:<https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/>

1) Launch a AWS EC2 instance with a Linux OS .

The screenshot shows the AWS Management Console interface for launching an EC2 instance. The 'Name and tags' section has a text input field with 'jenkins_server' and a link to 'Add additional tags'. The 'Application and OS Images (Amazon Machine Image)' section includes a search bar, tabs for 'Recents' and 'Quick Start', and a grid of OS options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. Below the grid, the 'Amazon Linux 2023 AMI' is selected, showing its ID (ami-06b21c0aeff8cd686), architecture (64-bit x86), and other details. A 'Free tier eligible' badge is visible on the right.

2) instance type as t2.medium.

The screenshot shows the 'Instance type' section of the AWS Management Console. It features a dropdown menu with 't2.medium' selected. Below the dropdown, the instance's specifications are listed: Family: t2, 2 vCPU, 4 GiB Memory, and Current generation: true. Pricing information for On-Demand Linux, RHEL, Windows, and SUSE is provided. To the right, there is a toggle for 'All generations' and a link to 'Compare instance types'. A note at the bottom states: 'Additional costs apply for AMIs with pre-installed software'.

3) Create a key pair for our instance

4) Allow the TCP, HTTP and HTTPS network access for all connections over the network.

5) SSH your connection

6) Execute the following commands :-

- **sudo yum update -y**: Updates all installed packages on the system to the latest available versions.

- **sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo**: Downloads the Jenkins repository file and saves it in the system's repository folder.
- **sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key**: Imports the Jenkins GPG key to authenticate the Jenkins packages.
- **sudo yum upgrade**: Upgrades the system packages to their latest versions based on available repositories.
- **sudo yum install jenkins -y**: Installs Jenkins from the configured repository.
- **sudo systemctl enable jenkins**: Enables Jenkins to automatically start at system boot.
- **sudo systemctl start jenkins**: Starts the Jenkins service immediately.
- **sudo systemctl status jenkins**: Displays the current status of the Jenkins service.

```

[ec2-user@ip-172-31-47-120 ~]$ sudo yum update -y
Last metadata expiration check: 0:01:19 ago on Fri Oct 18 18:07:32 2024.
No match for argument: -y
Error: No packages marked for upgrade.
[ec2-user@ip-172-31-47-120 ~]$ |

```

```

[ec2-user@ip-172-31-47-120 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2024-10-18 18:09:04-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.34.133, 2a04:4e42:79::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.34.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenk 100%[=====>]          85  --.-KB/s   in 0s
2024-10-18 18:09:04 (3.28 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-47-120 ~]$ |

```

```

[ec2-user@ip-172-31-47-120 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/j
enkins.io-2023.key
[ec2-user@ip-172-31-47-120 ~]$ sudo yum upgrade
Jenkins-stable                               325 kB/s | 29 kB      00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-47-120 ~]$ |

```

```
[ec2-user@ip-172-31-47-120 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:02:03 ago on Fri Oct 18 18:09:29 2024.
Dependencies resolved.

=====
Package                Architecture      Version           Repository        Size
=====
Installing:
jenkins                noarch            2.462.3-1.1      jenkins           89 M

Transaction Summary
=====
Install 1 Package

Total download size: 89 M
Installed size: 89 M
Downloading Packages:
jenkins-2.462.3-1.1.noar 14% [===                ] 2.2 MB/s | 13 MB      00:34 ETA
```

```
[ec2-user@ip-172-31-47-120 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:02:03 ago on Fri Oct 18 18:09:29 2024.
Dependencies resolved.

=====
Package                Architecture      Version           Repository        Size
=====
Installing:
jenkins                noarch            2.462.3-1.1      jenkins           89 M

Transaction Summary
=====
Install 1 Package

Total download size: 89 M
Installed size: 89 M
Downloading Packages:
jenkins-2.462.3-1.1.noarch.rpm                3.7 MB/s | 89 MB      00:23
-----
Total                                          3.7 MB/s | 89 MB      00:23
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                :                               1/1
  Running scriptlet: jenkins-2.462.3-1.1.noarch          1/1
  Installing             : jenkins-2.462.3-1.1.noarch    1/1
  Running scriptlet: jenkins-2.462.3-1.1.noarch          1/1
  Verifying              : jenkins-2.462.3-1.1.noarch    1/1

Installed:
jenkins-2.462.3-1.1.noarch

Complete!
[ec2-user@ip-172-31-47-120 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-47-120 ~]$ sudo systemctl start jenkins
```

```
[ec2-user@ip-172-31-47-120 ~]$ sudo systemctl status jenkins
* jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
   Active: active (running) since Fri 2024-10-18 18:12:27 UTC; 13s ago
     Main PID: 25878 (java)
       Tasks: 46 (limit: 1112)
      Memory: 329.6M
         CPU: 14.574s
    CGroup: /system.slice/jenkins.service
            └─25878 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 18 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 18 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.331+0000 [id=32] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.366+0000 [id=24] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.436+0000 [id=47] INFO h.n.DownloadService$Downloadable#load: Obtained the updated
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.437+0000 [id=47] INFO hudson.util.Retrier#start: Performed the action check update
Oct 18 18:12:32 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:32.494+0000 [id=62] WARNING h.n.DiskSpaceMonitorDescriptor#markNodeOfflineOrOnline: M
Lines 1-20/20 (END)
```

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

Getting Started


Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Jenkins 2.462.3

← → ↻ 🌐 localhost:8080 🔍 ⭐ 🌱 ⚙️ Error

 **Jenkins** 🔍 Search (CTRL+K) 🔔 🔒 🛡️ 👤 Nishant Sachin Khetal 🌐 log out

Dashboard >

+ New Item

📋 Build History

🔗 Project Relationship

🔍 Check File Fingerprint

⚙️ Manage Jenkins

📁 My Views

🌐 Open Blue Ocean

Build Queue

No builds in the queue.

Build Executor Status

📁 Built-In Node

1 Idle

2 Idle

📁 spash (offline)

ALL +

S	W	Name	Last Success	Last Failure	Last Duration	F
🟢	☁	casestudy_24	33 min #8	1 hr 59 min #4	5 min 1 sec	▶ ⭐
🟢	☁	casestudy_24_java	5 min 43 sec #2	7 min 50 sec #1	20 sec	▶ ⭐
🟢	☁	casestudy_maven_24	18 min #9	30 min #8	3 min 12 sec	▶ ⭐
🟢	☀	javacodessell	1 min 18 sec #1	N/A	19 sec	▶ ⭐
🟢	☁	my-new-project	9 days 20 hr #2	9 days 20 hr #1	0.19 sec	▶ ⭐

Icons: S M L

REST API Jenkins 2.462.3

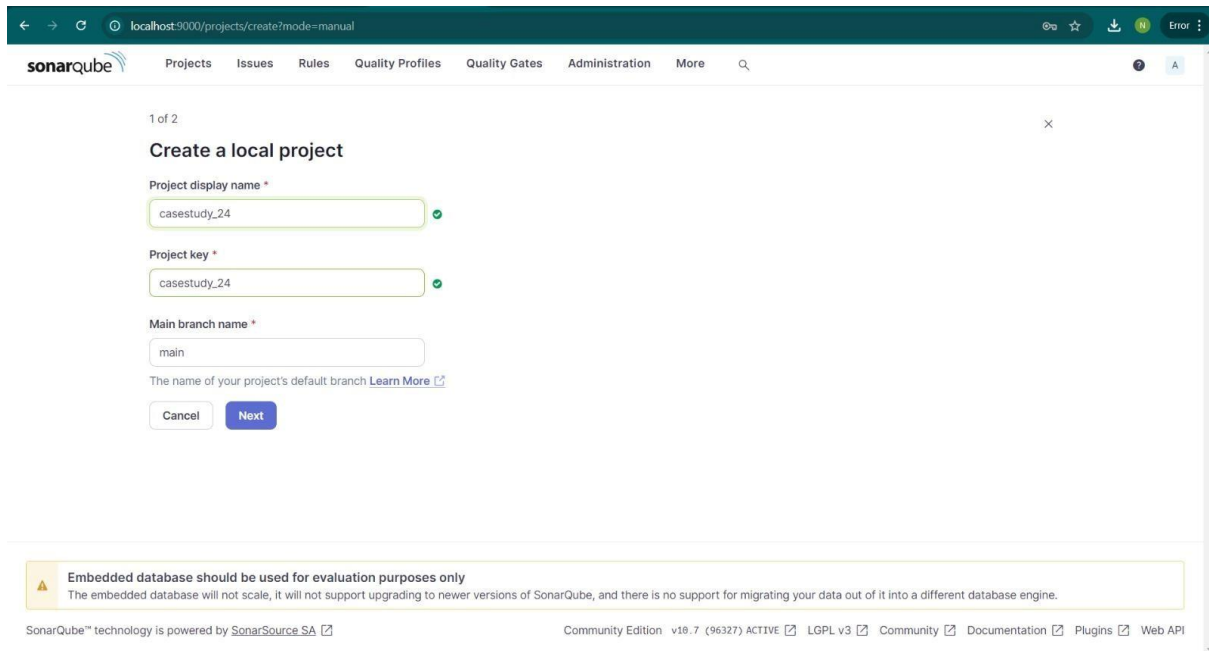
Thus, Jenkins is successfully configured on the EC2 Linux instance.

Task: SonarQube analysis of a Java/Python Project on Jenkins Pipeline:-

A] Sonarqube project:-

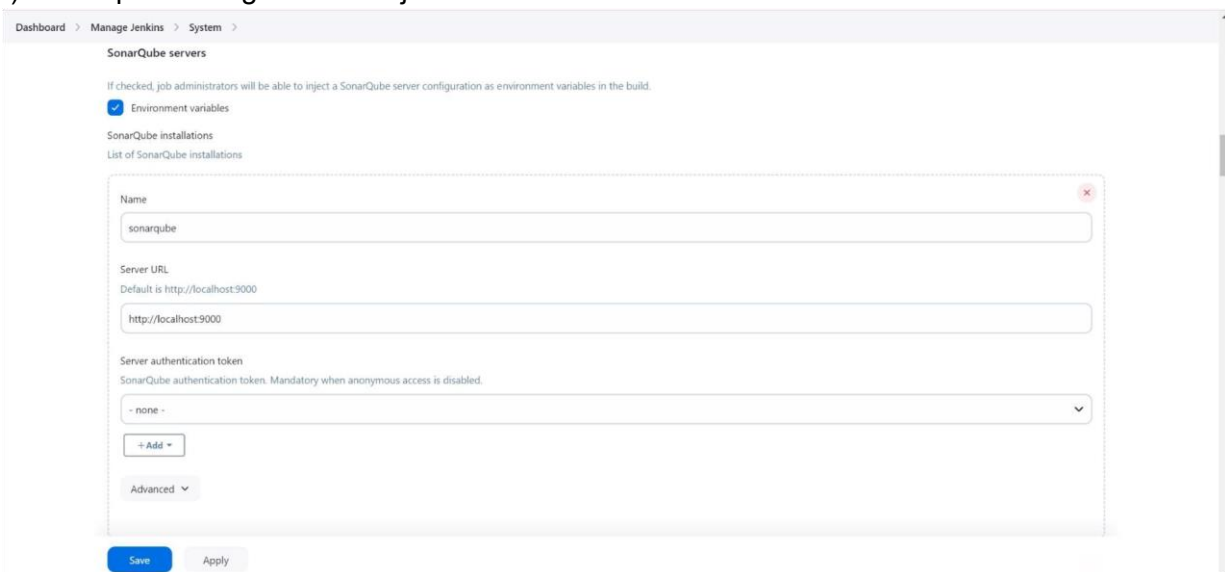
Python Project : <https://github.com/piomin/sample-java-sonar>

1) Create a sonarqube project named **casestudy_64**.



The screenshot shows the SonarQube web interface at the URL `localhost:9000/projects/create?mode=manual`. The page title is "Create a local project". It contains three input fields: "Project display name *" with the value "casestudy_24", "Project key *" with the value "casestudy_24", and "Main branch name *" with the value "main". Below these fields is a link "Learn More" and two buttons: "Cancel" and "Next". A yellow warning box at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The footer includes "SonarQube™ technology is powered by SonarSource SA" and "Community Edition v10.7 (96327) ACTIVE" along with links for LGPL v3, Community, Documentation, Plugins, and Web API.

2) Sonarqube configurations in jenkins:-



The screenshot shows the Jenkins "System" configuration page for "SonarQube servers". It includes a checkbox for "Environment variables" which is checked. Below this is a section for "SonarQube installations" with a "List of SonarQube installations" table. The table has one entry with "Name" "sonarqube", "Server URL" "http://localhost:9000", and "Server authentication token" "none". There are buttons for "+ Add" and "Advanced" (expanded). At the bottom are "Save" and "Apply" buttons.

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

SonarQube Scanner installations ^ Edited

Add SonarQube Scanner

≡ SonarQube Scanner

Name

sonarqube

☒ Install automatically ?

≡ Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer ▾

Add SonarQube Scanner

Save Apply

Dashboard > Manage Jenkins > Tools

Maven installations

Maven installations ^ Edited

Add Maven

≡ Maven

Name

Maven

☒ Install automatically ?

≡ Install from Apache

Version

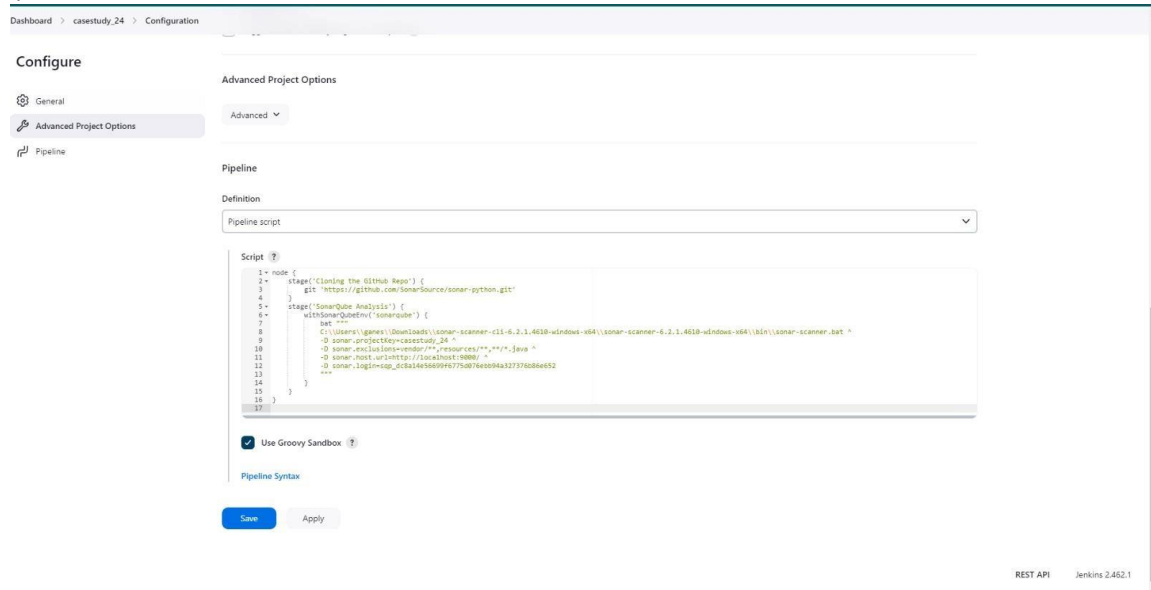
3.9.8

Add Installer ▾

Add Maven

Save Apply

3) Jenkins Pipeline:-



Pipeline Script:node

```
{
  stage('Cloning the GitHub Repo') { git
    'https://github.com/SonarSource/sonar-python.git'
  }
}
```

```
stage('SonarQube Analysis') {
  withSonarQubeEnv('sonarqube') {
    bat
```

```
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat " +
```

```
"-D sonar.login=admin " +
```

```
"-D sonar.password=HareKrishna#108 " +
```

```
"-D sonar.projectKey=casestudy_24 " +
```

```
"-D sonar.exclusions=vendor/**,resources/**,*/*.java " +
```

```
"-D sonar.host.url=http://localhost:9000/"
```

```
}
```

```
}
```

```
}
```

4) Jenkins Output :

Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Nishant Sachin Khetal
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\jenkins\jenkins\workspace\casestudy_24
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the Github Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\jenkins\jenkins\workspace\casestudy_24\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/SonarSource/sonar-python.git # timeout=10
Fetching upstream changes from https://github.com/SonarSource/sonar-python.git
> git.exe --version # timeout=10
> git --version # 'git version 2.43.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/SonarSource/sonar-python.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse 'refs/remotes/origin/master':[commit]' # timeout=10
Checking out Revision 45fd33ef84936b07fb0f9796b4d33918ac4ef900 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 45fd33ef84936b07fb0f9796b4d33918ac4ef900 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master 45fd33ef84936b07fb0f9796b4d33918ac4ef900 # timeout=10
Commit message: "Update all non-major dependencies (#2067)"
> git.exe rev-list --no-walk 45fd33ef84936b07fb0f9796b4d33918ac4ef900 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SonarQube Analysis)
[Pipeline] withSonarQubeEnv
Injecting SonarQube environment variables using the configuration: sonarqube
[Pipeline] {
[Pipeline] bat
```

```
* The filename starts with "test"
* The filename contains "test." or "tests."
* Any directory in the file path is named: "doc", "docs", "test" or "tests"
* Any directory in the file path has a name ending in "test" or "tests"

11:17:16.212 INFO Using git CLI to retrieve untracked files
11:17:16.274 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
11:17:17.643 INFO 1731 source files to be analyzed
11:17:27.652 INFO 1730/1731 files analyzed, current file: python-checks/src/main/resources/org/sonar/python/checks/hardcoded_credentials_call_check_meta.json
11:17:30.976 INFO 1731/1731 source files have been analyzed
11:17:30.980 INFO Sensor TextAndSecretsSensor [text] (done) | time=15678ms
11:17:30.996 INFO ----- Run sensors on project
11:17:31.320 INFO Sensor Zero Coverage Sensor
11:17:32.544 INFO Sensor Zero Coverage Sensor (done) | time=1224ms
11:17:33.720 INFO CPD Executor 259 files had no CPD blocks
11:17:33.720 INFO CPD Executor Calculating CPD for 723 files
11:17:34.561 INFO CPD Executor CPD calculation finished (done) | time=840ms
11:17:34.593 INFO SCM revision ID '45fd33ef84936b07fb0f9796b4d33918ac4ef900'
11:18:34.636 INFO Analysis report generated in 2670ms, dir size=6.2 MB
11:18:43.772 INFO Analysis report compressed in 8910ms, zip size=3.8 MB
11:18:43.899 INFO Analysis report uploaded in 126ms
11:18:43.901 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=casestudy_24
11:18:43.901 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
11:18:43.901 INFO More about the report processing at http://localhost:9000/api/ce/task?id=8127bf41-cf89-4010-9eeb-f216744290b5
11:18:56.543 INFO Analysis total time: 7:20.256 s
11:18:56.550 INFO SonarScanner Engine completed successfully
11:18:57.150 INFO EXECUTION SUCCESS
11:18:57.151 INFO Total time: 7:23.179s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Sonarqube Analysis:-

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

casestudy_24 / main

Overview Issues Security Hotspots Measures Code Activity

main 44k Lines of Code - Version not provided Set as homepage

Don't let issues accumulate. Discover 'Clean as You Code'!

Learn how to improve your code base by cleaning only new code.

Take the Tour Not now

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

The last analysis has warnings. [See details](#)

New Code Overall Code

Security	Reliability	Maintainability
703 Open Issues E	2.4k Open Issues E	4.6k Open Issues A

Accepted issues	Coverage	Duplications
0 U	0.0% E On 23k lines to cover;	2.9% D On 58k lines;

Security Hotspots

Build casestudy_24



Thus, the Python project was successfully analyzed with SonarQube.







B]For Maven Project:-

- 1) Make a Jenkins Pipeline.

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.
-  **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.
-  **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Java code to be analyzed :- <https://github.com/SonarSource/sonar-scanner-maven.git>

2) Create a sonarqube project named **casestudy24maven**.

1 of 2

Create a local project

Project display name *

 ✓

Project key *

 ✓

Main branch name *

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

Pipeline Script:node

```
{
  stage('Cloning the GitHub Repo') {
    git 'https://github.com/SonarSource/sonar-scanner-maven.git'
  }
  stage('SonarQube Analysis') {
    withSonarQubeEnv('sonarqube') {
```

bat

"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat " +

"-D sonar.login=admin " +

"-D sonar.password=HareKrishna#108 " +

"-D sonar.projectKey=casestudy24maven " +

"-D sonar.exclusions=vendor/**,resources/**,**/*.java " +

"-D sonar.host.url=http://localhost:9000/"

```
}  
}  
}
```

Dashboard > casestudy_maven_24 > Configuration

☐ Trigger builds remotely (e.g., from scripts) ?

Configure

- General
- Advanced Project Options
- Pipeline

Advanced Project Options: Advanced

Pipeline Definition: Pipeline script

```
1 node {  
2   stage('Cloning the Github Repo') {  
3     git 'https://github.com/SonarSource/sonar-scanner-maven.git'  
4   }  
5  
6   stage('SonarQube Analysis') {  
7     withSonarQubeEnv('sonarqube') {  
8       bat '''  
9         C:\Users\ganes\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 -D sonar.password=HareKrishna#108 -D sonar.projectKey=casestudy_maven_24  
11        -D sonar.exclusions=vendor/**,resources/**,**/*.java  
12        -D sonar.host.url=http://localhost:9000/  
13      '''  
14    }  
15  }  
16 }  
17 }
```

☒ Use Groovy Sandbox ?

Pipeline Syntax

REST API Jenkins 2.462.1

3) Build and check its console output:-

Started by user Nishant Sachin Khetal

[Pipeline] Start of Pipeline

[Pipeline] node

Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\casestudy_24_java

[Pipeline] [

[Pipeline] stage

[Pipeline] { (Cloning the Github Repo)

[Pipeline] git

The recommended git tool is: NONE

No credentials specified

> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\casestudy_24_java\.git # timeout=10

Fetching changes from the remote Git repository

> git.exe config remote.origin.url https://github.com/mohsiqba/sonar-maven.git # timeout=10

Fetching upstream changes from https://github.com/mohsiqba/sonar-maven.git

> git.exe --version # timeout=10

> git --version # 'git version 2.43.0.windows.1'

> git.exe fetch --tags --force --progress -- https://github.com/mohsiqba/sonar-maven.git +refs/heads/*:refs/remotes/origin/* # timeout=10

> git.exe rev-parse --refs/remotes/origin/master^{commit} # timeout=10

Checking out Revision 7e0b6392e64cdf2498beed78c79309d000dacb5 (refs/remotes/origin/master)

> git.exe config core.sparsecheckout # timeout=10

> git.exe checkout -f 7e0b6392e64cdf2498beed78c79309d000dacb5 # timeout=10

> git.exe branch -a -v --no-abbrev # timeout=10

> git.exe branch -D master # timeout=10

> git.exe checkout -b master 7e0b6392e64cdf2498beed78c79309d000dacb5 # timeout=10

Commit message: "sonarscanner with maven - maven multimodule project"

> git.exe rev-list --no-walk 7e0b6392e64cdf2498beed78c79309d000dacb5 # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (SonarQube Analysis)

[Pipeline] withSonarQubeEnv

Injecting SonarQube environment variables using the configuration: sonarqube

[Pipeline] [

[Pipeline] bat

```
Dashboard > casestudy_24_java > #2

* The filename contains "test." or "tests."
* Any directory in the file path is named: "doc", "docs", "test" or "tests"
* Any directory in the file path has a name ending in "test" or "tests"

12:32:42.052 INFO Using git CLI to retrieve untracked files
12:32:42.100 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
12:32:42.159 INFO 23 source files to be analyzed
12:32:42.351 INFO 23/23 source files have been analyzed
12:32:42.354 INFO Sensor TextAndSecretsSensor [text] (done) | time=1272ms
12:32:42.369 INFO ----- Run sensors on project
12:32:42.652 INFO Sensor Zero Coverage Sensor
12:32:42.656 INFO Sensor Zero Coverage Sensor (done) | time=4ms
12:32:42.659 INFO SCM Publisher SCM provider for this project is: git
12:32:42.661 INFO SCM Publisher 18 source files to be analyzed
12:32:43.211 INFO SCM Publisher 18/18 source files have been analyzed (done) | time=549ms
12:32:43.214 INFO CPD Executor Calculating CPD for 0 files
12:32:43.215 INFO CPD Executor CPD calculation finished (done) | time=0ms
12:32:43.234 INFO SCM revision ID '7e0b6392e64cdf240b0eed78c79309d000dadb5'
12:32:43.583 INFO Analysis report generated in 153ms, dir size=338.3 kB
12:32:43.752 INFO Analysis report compressed in 131ms, zip size=68.9 kB
12:32:43.901 INFO Analysis report uploaded in 160ms
12:32:43.902 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=casestudy_24_java
12:32:43.903 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
12:32:43.904 INFO More about the report processing at http://localhost:9000/api/ce/task?id=1637031a-60d5-4f75-ba51-8005c1f4d5b0
12:32:43.911 INFO Analysis total time: 15.476 s
12:32:43.922 INFO SonarScanner Engine completed successfully
12:32:44.020 INFO EXECUTION SUCCESS
12:32:44.022 INFO Total time: 17.991s

[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.462.1

SonarQube analysis

localhost:9000/dashboard?id=casestudy_24_java&codeScope=overall

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More

casestudy_24_java / main

Overview Issues Security Hotspots Measures Code Activity

main 1.3k Lines of Code · Version not provided · Set as homepage

Quality Gate **Passed** Last analysis 7 minutes ago

The last analysis has warnings. [See details](#)

New Code Overall Code

Metric	Value	Grade
Security	0 Open issues	A
Reliability	0 Open issues	A
Maintainability	8 Open issues	A
Accepted issues	0	
Coverage	0.0%	
Duplications	0.0%	
Security Hotspots	0	A

Activity

Attributes:-

Consistency: Ensures the code follows uniform patterns and styles throughout, making it easier to read and understand.

Intentionality: Reflects that the code is written with clear purpose and reasoning, avoiding unnecessary complexity.

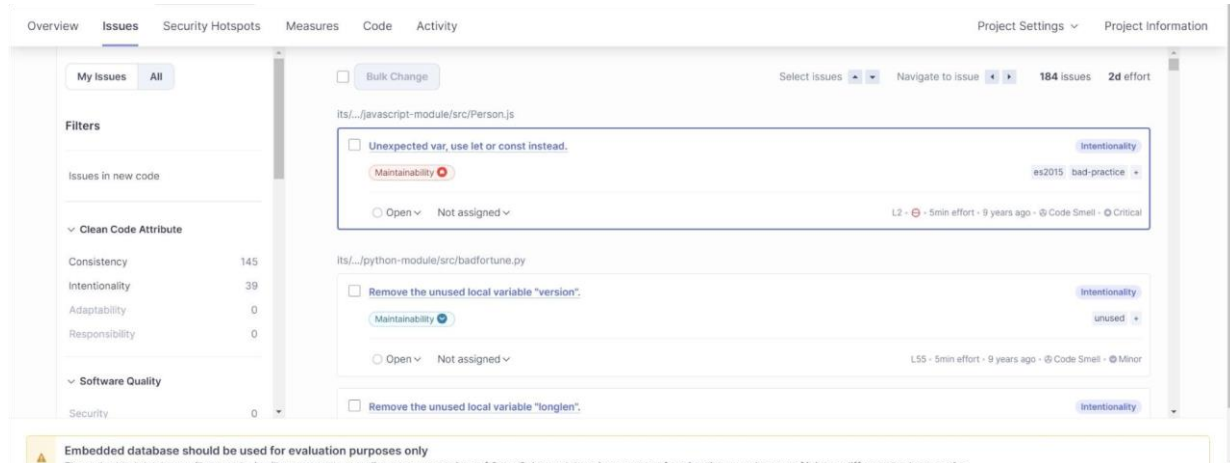
Adaptability: Indicates how flexible the code is to accommodate changes or new features without requiring major rewrites.

Responsibility: Reflects that the code is well-structured with clear ownership of functions or classes, adhering to the Single Responsibility Principle.

Security: Measures how well the code protects against vulnerabilities and malicious attacks.

Reliability: Assesses the code's ability to function correctly under different conditions and its resistance to failures.

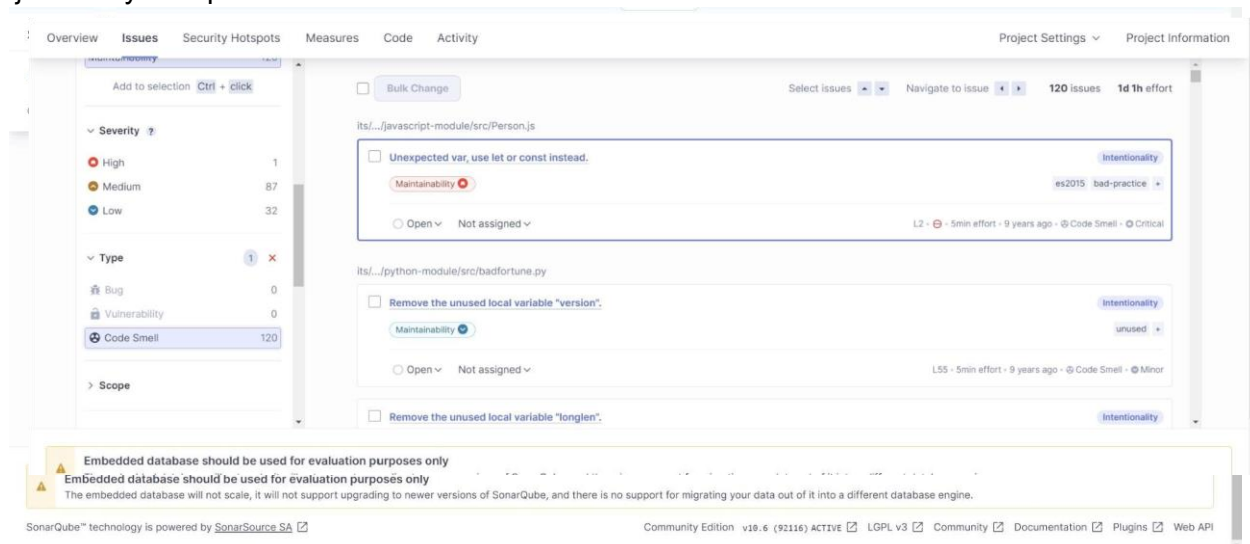
Maintainability: Evaluates how easily the code can be modified, debugged, or enhanced for long-term use.



} Code smells:-

A code smell is an indicator of potential design issues in code that, while not causing immediate errors, can lead to problems over time, such as maintainability and scalability issues. Common examples include large classes, long methods, and duplicated code, which increase complexity and technical debt. Identifying and addressing code smells through refactoring improves code quality and readability, ensuring long-term reliability and maintainability

ii} Security Hotspots:-



A **security hotspot** is a piece of security-sensitive code that requires review to determine if it poses a threat. Unlike vulnerabilities, which need immediate fixes, hotspots must be assessed by the developer to decide if action is required. Fixing hotspots enhances an application's

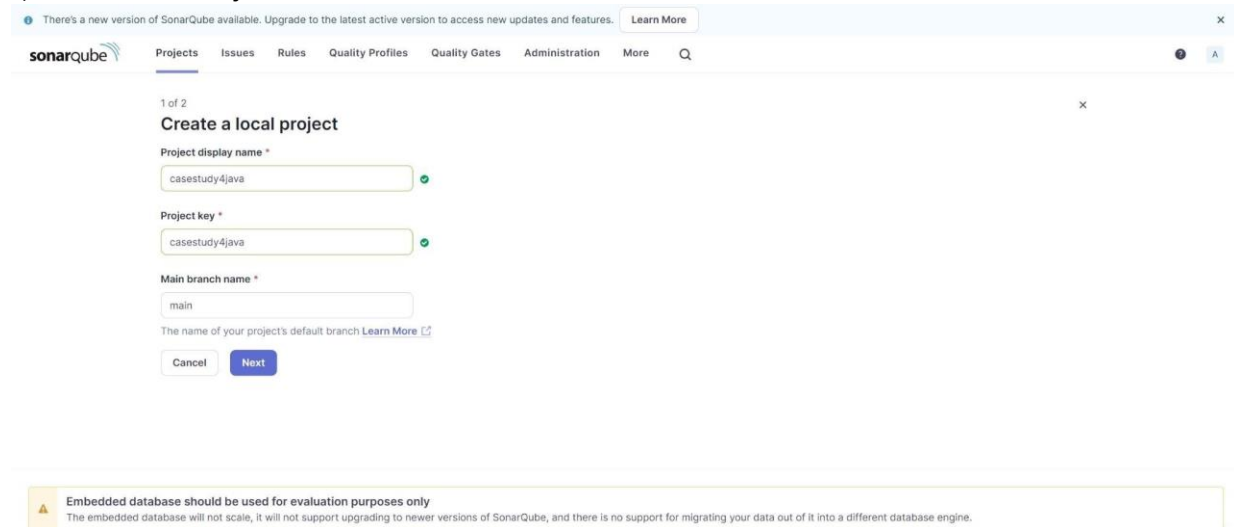
resilience against attacks. SonarQube assigns review priority based on standards like OWASP Top 10 and CWE Top 25, with hotspots categorized as high, medium, or low priority. Developers review the code, assess risks, and apply necessary fixes or mark it as safe. Addressing security hotspots strengthens code security while allowing for informed, context-based decisions.

Successfully analyzed the Java Maven project using SonarQube.

3] Java Project:-

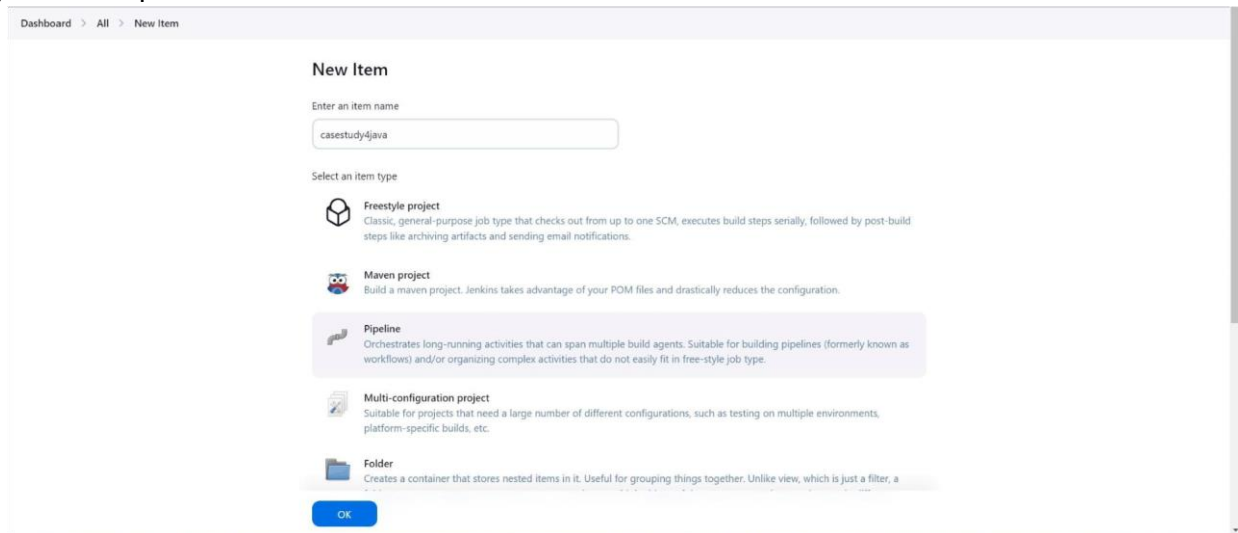
Project Link: <https://github.com/mohsiqba/sonar-maven.git>

1)SonarQube Project:



The screenshot shows the SonarQube web interface. At the top, there is a notification bar stating: "There's a new version of SonarQube available. Upgrade to the latest active version to access new updates and features." Below this is the navigation menu with options: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main content area is titled "1 of 2 Create a local project". It contains three input fields: "Project display name *" with the value "casestudy4java", "Project key *" with the value "casestudy4java", and "Main branch name *" with the value "main". Below these fields is a link "The name of your project's default branch Learn More". At the bottom of the form are "Cancel" and "Next" buttons. A yellow warning box at the bottom of the page states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine."

2)Jenkins Pipeline:



The screenshot shows the Jenkins "New Item" form. At the top, there is a breadcrumb trail: "Dashboard > All > New Item". The form has a title "New Item". Below the title is a text input field "Enter an item name" with the value "casestudy4java". Below this is a section "Select an item type" with five options: "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.), and "Folder" (Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder is a real container). At the bottom of the form is an "OK" button.

Pipeline Script:-

```
node {  
  stage('Cloning the GitHub Repo') { git  
    'https://github.com/mohsiqba/sonar-maven.git'  
  }  
}
```



```

stage('SonarQube Analysis') {
    withSonarQubeEnv('sonarqube') {
        bat
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4
477-windows-x64\\bin\\sonar-scanner.bat " +
        "-D sonar.login=admin " +
        "-D sonar.password=HareKrishna#108 " +
        "-D sonar.projectKey=casestudy24java " +
        "-D sonar.exclusions=vendor/**,resources/**,**/*.java " +
        "-D sonar.host.url=http://localhost:9000/"
    }
}
}

```

Pipeline

Definition

Pipeline script

Script ?

```

1 node {
2     stage('Cloning the GitHub Repo') {
3         // Clone the GitHub repository
4         git 'https://github.com/mohsiqba/sonar-maven.git'
5     }
6
7     stage('SonarQube Analysis') {
8         withSonarQubeEnv('sonarqube') {
9             bat """
10             C:\\Users\\ganes\\Downloads\\sonar-scanner-cli-6.2.1.4610-windows-x64\\sonar-scanner-6.2.1.4610-windows-x64\\bin\\sonar-scanner.bat ^
11             -D sonar.login=admin ^
12             -D sonar.password=HareKrishna#108 ^
13             -D sonar.projectKey=casestudy_24_Java ^
14             -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
15             -D sonar.host.url=http://localhost:9000/
16             """
17         }
18     }
19 }

```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save

Apply

✓ Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Nishant Sachin Khetal
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\workspace\casestudy_24_java
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the Github Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\workspace\casestudy_24_java\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/mohsiqba/sonar-maven.git # timeout=10
Fetching upstream changes from https://github.com/mohsiqba/sonar-maven.git
> git.exe --version # timeout=10
> git --version # 'git version 2.43.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/mohsiqba/sonar-maven.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 7e0b6392e64cdf2498beed78c79309d000dabc5 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 7e0b6392e64cdf2498beed78c79309d000dabc5 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master 7e0b6392e64cdf2498beed78c79309d000dabc5 # timeout=10
Commit message: "sonarscanner with maven - maven multimodule project"
> git.exe rev-list --no-walk 7e0b6392e64cdf2498beed78c79309d000dabc5 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SonarQube Analysis)
[Pipeline] withSonarQubeEnv
Injecting SonarQube environment variables using the configuration: sonarqube
[Pipeline] {
[Pipeline] bat
```

Dashboard > casestudy_24_java > #2

```
* The filename contains "test." or "tests."
* Any directory in the file path is named: "doc", "docs", "test" or "tests"
* Any directory in the file path has a name ending in "test" or "tests"

12:32:42.092 INFO Using git CLI to retrieve untracked files
12:32:42.100 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
12:32:42.159 INFO 23 source files to be analyzed
12:32:42.351 INFO 23/23 source files have been analyzed
12:32:42.354 INFO Sensor TextAndSecretsSensor [text] (done) | time=1272ms
12:32:42.369 INFO ..... Run sensors on project
12:32:42.652 INFO Sensor Zero Coverage Sensor
12:32:42.659 INFO Sensor Zero Coverage Sensor (done) | time=4ms
12:32:42.659 INFO SCM Publisher SCM provider for this project is: git
12:32:42.661 INFO SCM Publisher 18 source files to be analyzed
12:32:42.211 INFO SCM Publisher 18/18 source files have been analyzed (done) | time=549ms
12:32:42.214 INFO CPD Executor Calculating CPD for 0 files
12:32:42.215 INFO CPD Executor CPD calculation finished (done) | time=0ms
12:32:42.234 INFO SCM revision ID '7e0b6392e64cdf2490bed78c79309d000dadb5'
12:32:42.583 INFO Analysis report generated in 151ms, dir size=338.3 kB
12:32:42.732 INFO Analysis report compressed in 151ms, zip size=68.9 kB
12:32:42.901 INFO Analysis report uploaded in 169ms
12:32:42.902 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=casestudy_24_java
12:32:42.903 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
12:32:42.904 INFO More about the report processing at http://localhost:9000/api/ce/task?id=1637831a-68d5-4f75-ba51-8065c1f0436b
12:32:42.921 INFO Analysis total time: 15.476 s
12:32:42.922 INFO SonarScanner Engine completed successfully
12:32:44.020 INFO EXECUTION SUCCESS
12:32:44.022 INFO Total time: 17.091s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.462.1

localhost:9000/dashboard?id=casestudy_24_java&codeScope=overall

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More

casestudy_24_java / main

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

main 1.3K Lines of Code - Version not provided - Set as homepage

Quality Gate **Passed** Last analysis 7 minutes ago

The last analysis has warnings. [See details](#)

New Code Overall Code

Security	Reliability	Maintainability
0 Open issues	0 Open issues	8 Open issues
Accepted issues	Coverage	Duplications
0	On 0 lines to cover.	0.0%
Valid issues that were not fixed		On 1.5K lines.
Security Hotspots		
0		

Activity

Successfully analyzed the Java project using SonarQube. 4] Java Project:-
Project Link: <https://github.com/nerdschoolbergen/code-smells.git>

1)SonarQube Project:

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

1 of 2

Create a local project

Project display name *

javacodesmell

Project key *

javacodesmell

Main branch name *

main

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

Cancel Next

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.6 (92116) ACTIVE LGPL v3 Community Documentation Plugins Web API

2) Jenkins Pipeline:

Dashboard > All > New Item

New Item

Enter an item name

javacodesmell

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.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

OK

Pipeline Script:-

```
node {
  stage('Cloning the GitHub Repo') { git
    'https://github.com/nerdschoolbergen/code-smells.git'
  }
  stage('SonarQube Analysis') {
    withSonarQubeEnv('sonarqube') {
      bat
```

```
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat " +
```

```
"-D sonar.login=admin " + // Ensure there is a space before the next option
```

```
"-D sonar.password=HareKrishna#108 " +
```

```
"-D sonar.projectKey=javacodesmell " +
```

```
"-D sonar.exclusions=vendor/**,resources/**,**/*.java " +
"-D sonar.host.url=http://localhost:9000/"
```

```
}
}
}
```

The screenshot shows the Jenkins Configuration page for a pipeline named 'javacodesmell'. The 'Advanced Project Options' tab is selected. Under the 'Pipeline' section, the 'Definition' is set to 'Pipeline script'. A 'Script' editor contains a Groovy script for cloning a repository and running SonarQube analysis. The script includes environment variable injection for SonarQube. The 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons. The bottom right corner shows 'REST API' and 'Jenkins 2.462.1'.

3) Console Output:

The screenshot shows the Jenkins Console Output for the 'javacodesmell' pipeline. The output starts with 'Started by user Nishant Sachin Khetal' and '[Pipeline] Start of Pipeline (hide)'. It then shows the pipeline steps: 'node', 'git' (cloning the repository), and 'stage' (SonarQube Analysis). The 'git' step shows the repository being cloned from 'https://github.com/nerdschoolbergen/code-smells.git'. The 'stage' step shows the SonarQube analysis being run with the configuration: 'sonar.exclusions=vendor/**,resources/**,**/*.java' and 'sonar.host.url=http://localhost:9000/'. The output ends with 'Commit message: "remove whitespace"' and 'First time build. Skipping changelog.'

```
localhost:8080/job/javacodesmell/1/console

Dashboard > javacodesmell > #1

* The filename contains "test." or "tests."
* Any directory in the file path is named: "doc", "docs", "test" or "tests"
* Any directory in the file path has a name ending in "test" or "tests"

12:37:06.417 INFO Using git CLI to retrieve untracked files
12:37:06.461 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
12:37:06.490 INFO 4 source files to be analyzed
12:37:06.499 INFO 4/4 source files have been analyzed
12:37:06.612 INFO Sensor TextAndSecretsSensor [text] (done) | time=1146ms
12:37:06.623 INFO ----- Run sensors on project
12:37:06.877 INFO Sensor Zero Coverage Sensor
12:37:06.879 INFO Sensor Zero Coverage Sensor (done) | time=2ms
12:37:06.883 INFO SCM Publisher SCM provider for this project is: git
12:37:06.884 INFO SCM Publisher 1 source file to be analyzed
12:37:07.200 INFO SCM Publisher 1/1 source file have been analyzed (done) | time=314ms
12:37:07.202 INFO CPD Executor Calculating CPD for 0 files
12:37:07.202 INFO CPD Executor CPD calculation finished (done) | time=0ms
12:37:07.220 INFO SCM revision ID 'bbdf35626fcc07dde0e54462e46886126e126bce'
12:37:07.410 INFO Analysis report generated in 142ms, dir size=225.2 kB
12:37:07.459 INFO Analysis report compressed in 51ms, zip size=25.1 kB
12:37:07.593 INFO Analysis report uploaded in 134ms
12:37:07.594 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=javacodesmell
12:37:07.594 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
12:37:07.606 INFO More about the report processing at http://localhost:9000/api/ci/task?id=89e0ec91-6663-4695-966c-4b6c1cac1b
12:37:07.606 INFO Analysis total time: 13.415 s
12:37:07.606 INFO SonarScanner Engine completed successfully
12:37:08.162 INFO EXECUTION SUCCESS
12:37:08.163 INFO Total time: 16.777s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

SonarQube Analysis:

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More

javacodesmell main

Overview Issues Security Hotspots Measures Code Activity

main 50 Lines of Code - Version not provided - Set as homepage

Quality Gate **Passed** Last analysis 4 minutes ago

The last analysis has warnings. [See details](#)

New Code Overall Code

Security 0 Open issues A	Reliability 0 Open issues A	Maintainability 0 Open issues A
Accepted issues 0 Valid issues that were not fixed	Coverage 0 on 0 lines to cover	Duplications 0.0% On 56 lines
Security Hotspots 0 A		

Activity

Successfully analyzed the Java project using SonarQube.

Conclusion :

Through this case study, Jenkins and SonarQube were successfully integrated to automate the continuous integration and static code analysis of Python and Java projects. The CI pipeline helps maintain high code quality and ensures that code adheres to security standards before being pushed to production.