

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 Lambda console interface. At the top, there's a search bar and a 'Create New Function' button. Below that, a 'Function name' input field contains 'jenkins_lambda'. Under the 'Runtime' dropdown, 'Node.js 14.x' is selected. The 'Handler' dropdown shows 'index.handler'. In the 'Memory size' dropdown, '128 MB' is chosen. The 'Timeout' dropdown shows '3 minutes'. The 'Role' dropdown is set to 'Lambda execution role - lambda-execution-role'. The 'Code' section shows a 'Create New Layer' button and a 'Upload ZIP file' button. The 'Environment' section has a 'Configure' button. The 'Logs' section shows a 'View logs' button. The 'Monitoring' section has a 'CloudWatch Metrics' button. The 'Metrics' section shows a 'CloudWatch Metrics' button. The 'Logs' section shows a 'View logs' button.

2) instance type as t2.medium.

The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and a 'Create New Function' button. Below that, a 'Function name' input field contains 'jenkins_lambda'. Under the 'Runtime' dropdown, 'Node.js 14.x' is selected. The 'Handler' dropdown shows 'index.handler'. In the 'Memory size' dropdown, '128 MB' is chosen. The 'Timeout' dropdown shows '3 minutes'. The 'Role' dropdown is set to 'Lambda execution role - lambda-execution-role'. The 'Code' section shows a 'Create New Layer' button and a 'Upload ZIP file' button. The 'Environment' section has a 'Configure' button. The 'Logs' section shows a 'View logs' button. The 'Monitoring' section has a 'CloudWatch Metrics' button. The 'Metrics' section shows a 'CloudWatch Metrics' button. The 'Logs' section shows a 'View logs' button.

3) Create a key pair for our instance

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

iamkhetal

Create new key pair

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

Inbound rules	Type	Protocol	Port range	Source	Description - optional
sg-0666a74e77e1ec5c8	SSH	TCP	22	Anywhere- 0.0.0.0/0	
-	Custom TCP	TCP	0	Anywhere- 0.0.0.0/0	
-	All TCP	TCP	0 - 65535	Anywhere- 0.0.0.0/0	
-	HTTP	TCP	80	Anywhere- 0.0.0.0/0	
-	HTTPS	TCP	443	Anywhere- 0.0.0.0/0	

Add rule

⚠ Rules with source of 0.0.0.0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

5) SSH your connection

```
PS C:\Users\ADMIN\Downloads\advdevops> ssh -i "iamkhetal.pem" ec2-user@ec2-35-174-104-174.compute-1.amazonaws.com
#_###_
~~\_\####\_
~~ \###|_
~~ \#/ --- https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '-->
~~~ .-.
~~~ /-/
~~~ /m'|
Last login: Sat Oct 19 07:10:49 2024 from 49.33.251.45
[ec2-user@ip-172-31-82-90 ~]$ |
```

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.noarch 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

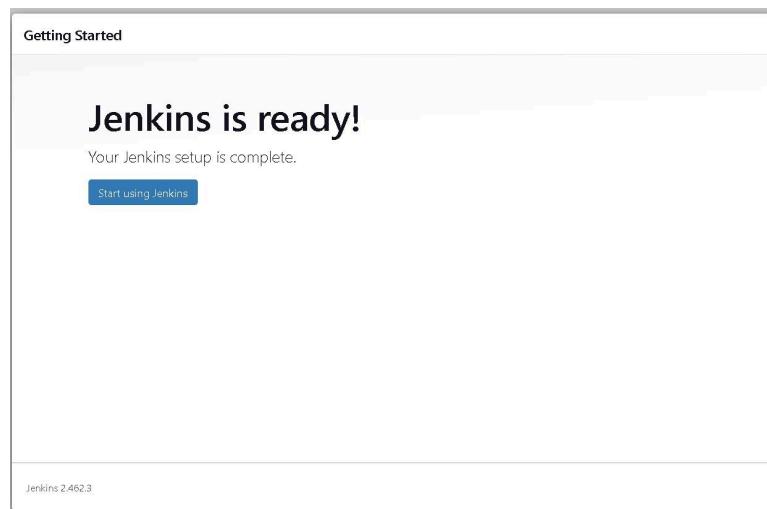
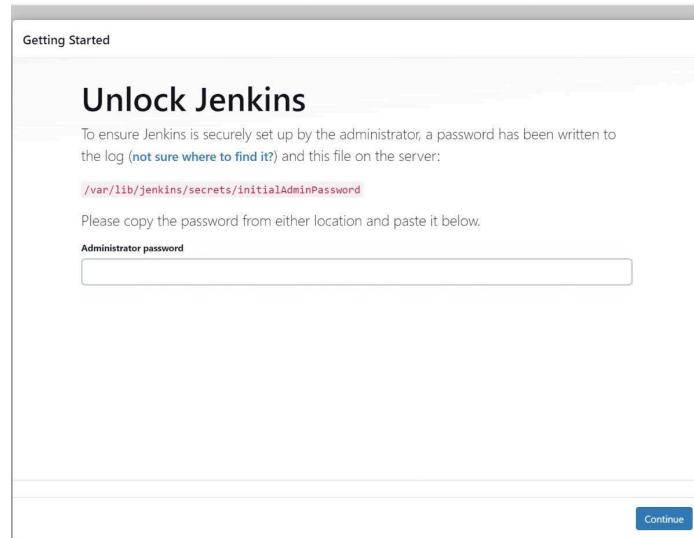
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing
  Running scriptlet: jenkins-2.462.3-1.1.noarch
  Installing : jenkins-2.462.3-1.1.noarch
  Running scriptlet: jenkins-2.462.3-1.1.noarch
  Verifying   : jenkins-2.462.3-1.1.noarch
                                         1/1
                                         1/1
                                         1/1
                                         1/1
                                         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; 1s ago
    Main PID: 25878 (java)
       Tasks: 46 (limit: 1112)
      Memory: 14.574M
         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$1#onAttained: Completed initialization
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.366+0000 [id:24]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal system[1]: Started jenkins.service - Jenkins Continuous Integration Server
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.436+0000 [id:47]      INFO      h.m.DownloadService$Downloadable#load: Obtained the updated configuration
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.Retrive#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: More than 10% of disk space is used by Jenkins
Lines 1-20 (END)
```



The screenshot shows the Jenkins dashboard at `localhost:8080`. The left sidebar includes links for 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'My Views', and 'Open Blue Ocean'. The main area displays a table of build history with columns: Status (S), Workstation (W), Name, Last Success, Last Failure, Last Duration, and a star icon. The table lists five projects: 'casestudy_24', 'casestudy_24.java', 'casestudy_maven_24', 'javacodesmell', and 'my-new-project'. Below the table, the 'Build Executor Status' section shows 'Built-In Node' with 1 idle and 2 idle nodes, and a node named 'nishesh' which is offline. A footer bar at the bottom includes links for 'REST API' and 'Jenkins 2.462.1'.

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_24**.

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](#)

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 v10.7 \(96327\) ACTIVE](#) [LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#) [Web API](#)

2) Sonarqube configurations in jenkins:-

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name: sonarqube

Server URL: Default is http://localhost:9000
http://localhost:9000

Server authentication token: SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -

+ Add ▾

Advanced ▾

Save Apply

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name: sonarqube

Install automatically ?

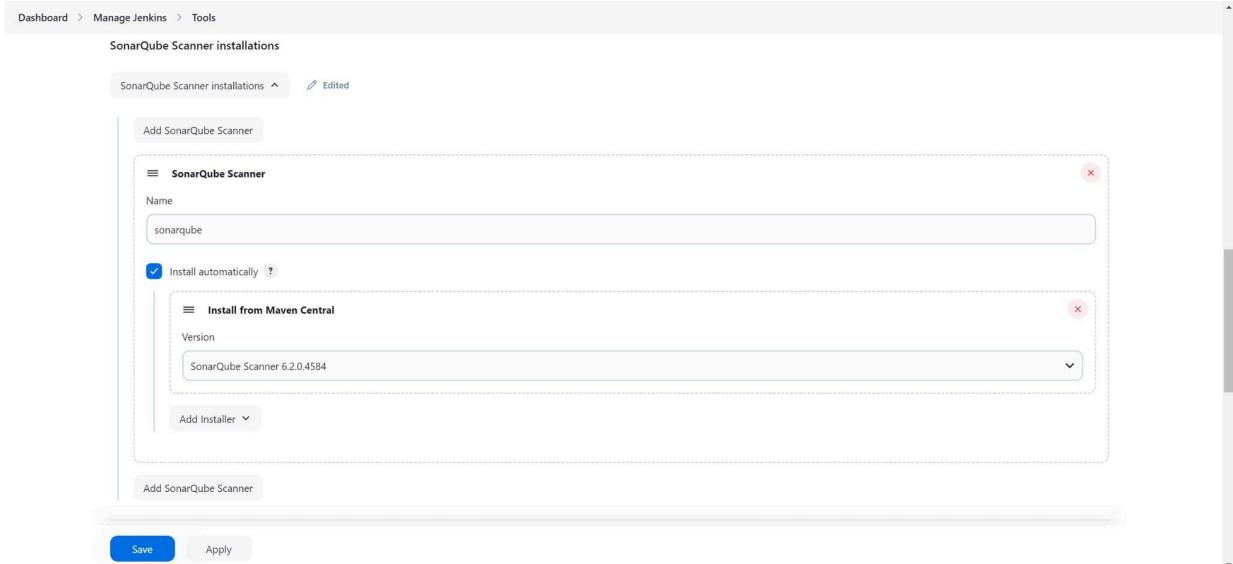
Install from Maven Central

Version: SonarQube Scanner 6.2.0.4584

Add Installer ▾

Add SonarQube Scanner

Save **Apply**



Dashboard > Manage Jenkins > Tools

Maven installations

Add Maven

Maven

Name: Maven

Install automatically ?

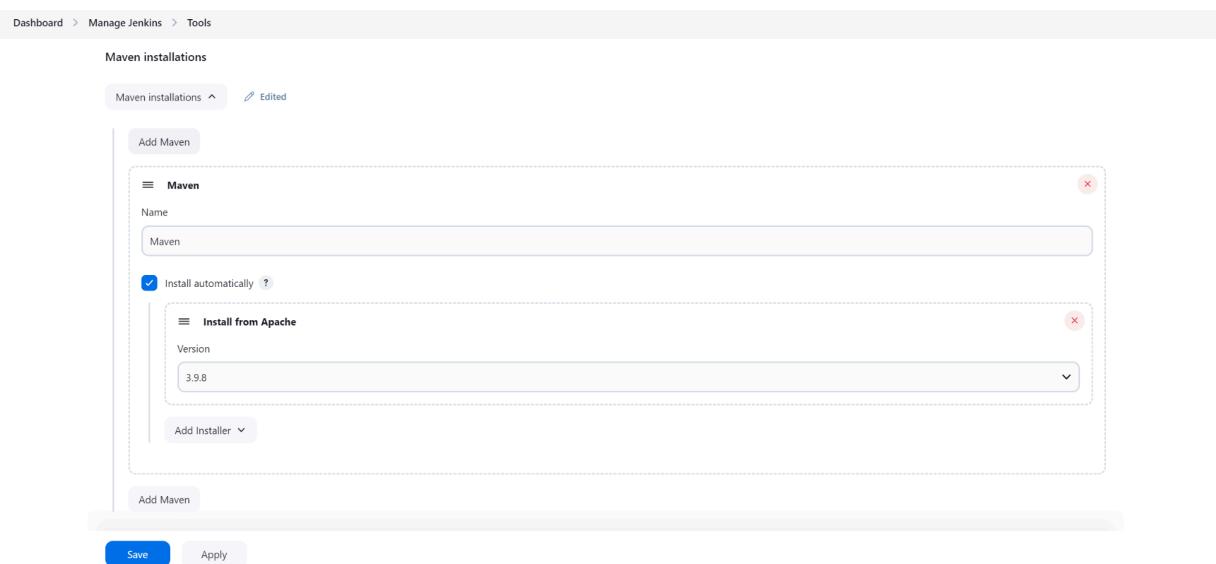
Install from Apache

Version: 3.9.8

Add Installer ▾

Add Maven

Save **Apply**



3) Jenkins Pipeline:-

The screenshot shows the Jenkins Pipeline configuration page for a project named 'casestudy_24'. The 'Pipeline' tab is selected under 'Advanced Project Options'. The 'Definition' dropdown is set to 'Pipeline script'. The main area contains a code editor with the following Groovy script:

```

1+ node {
2+   stage('Cloning the GitHub Repo') {
3+     git 'https://github.com/SonarSource/sonar-python.git'
4+   }
5+   stage('SonarQube Analysis') {
6+     withSonarQubeEnv('sonarqube') {
7+       bat """
8+         cd C:\Users\nish\Downloads\sonar-scanner-cli-6.1.0.4477-windows-x64\sonar-scanner-6.1.0.4
9+         sonar-scanner -Dsonar.projectKey=casestudy_24 ^
10+          -Dsonar.exclusions=vendor/**,resources/**,**/*.java ^
11+          -Dsonar.host.url=http://localhost:9000/
12+          """
13+     }
14+   }
15+ }

```

Below the code editor, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the page, there are 'Save' and 'Apply' buttons.

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-6.1.0.4
      477-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

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 -- 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 -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

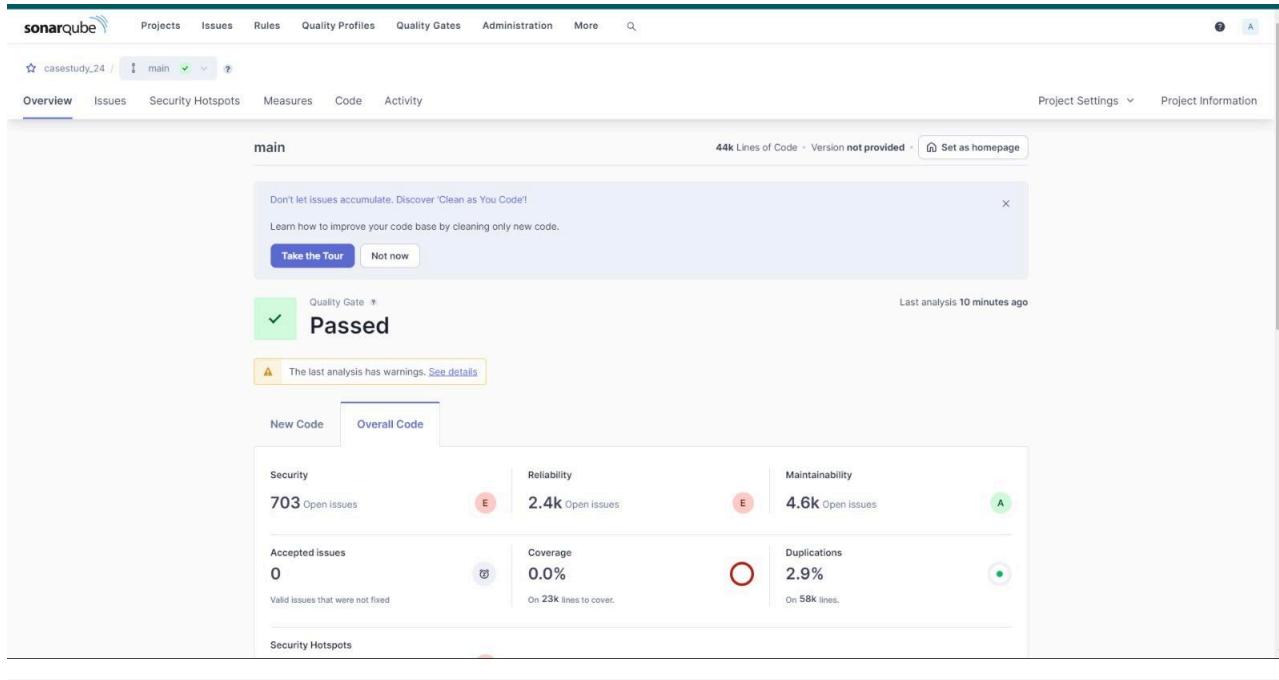
```

```

* 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=15676ms
11:17:30.996 INFO -----
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'
11:18:34.836 INFO Analysis report generated in 2670ms, dir size=6.2 MB
11:18:43.772 INFO Analysis report compressed in 8919ms, 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-cf89-4010-9aeb-f216744290b5
11:18:56.543 INFO Analysis total time: 7:20.256 s
11:18:56.559 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:-**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
casestudy_maven_24

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:

OK

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 *

casestudy24maven



Project key *

casestudy24maven



Main branch name *

main

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

Cancel

Next

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-6.1.0.4
477-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"
}

}
}
}

```

The screenshot shows the Jenkins Pipeline configuration page. Under the 'Pipeline' tab, the 'Definition' dropdown is set to 'Pipeline script'. The script content is as follows:

```

node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/SonarSource/sonar-scanner-maven.git'
    }
    stage('SonarQube Analysis') {
        bat """
            cd C:\Users\nish\Downloads\sonar-scanner-6.1.0.4477-windows-x64\sonar-scanner-6.1.0.4477-windows-x64\bin
            sonar-scanner -D sonar.login=nish_bdec17a0b5920ba0dcb05538051b015e076c04
            -D sonar.password=casestudy108
            -D sonar.projectKey=casestudy24maven
            -D sonar.exclusions=vendor/**,resources/**,**/*.java
            -D sonar.host.url=http://localhost:9000
        """
    }
}

```

The 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons.

3) Build and check its console output:-

✓ 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 --force --progress --tags 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 7e0b6392e64cdcfc2498beed78c79309d000dacb5 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 7e0b6392e64cdcfc2498beed78c79309d000dacb5 # timeout=10
 > git.exe branch -D master # timeout=10
 > git.exe checkout -b master 7e0b6392e64cdcfc2498beed78c79309d000dacb5 # timeout=10
 Commit message: "sonarscanner with maven - maven multimodule project"
 > git.exe rev-list --no-walk 7e0b6392e64cdcfc2498beed78c79309d000dacb5 # timeout=10
 [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 have been analyzed
12:32:42.151 INFO 23/23 source files have been analyzed
12:32:42.154 INFO Sensor TextAndSecretsSensor [test] (done) | time=1272ms
12:32:42.169 INFO ----- Run sensors on project
12:32:42.162 INFO Sensor Zero Coverage Sensor
12:32:42.165 INFO Sensor Zero Coverage Sensor [done] | time=4ms
12:32:42.169 INFO SCM Publisher SCM provider for this project is: git
12:32:42.161 INFO SCM Publisher 18 source files to be analyzed
12:32:42.111 INFO SCM Publisher 18/18 source files have been analyzed (done) | time=540ms
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: '7e0b392d44cdcf24980eed78c79300d000dc5b'
12:32:43.583 INFO Analysis report generated in 153ms, dir size=338.3 kB
12:32:43.732 INFO Analysis report compressed in 151ms, zip size=68.9 kB
12:32:43.981 INFO Analysis report uploaded in 108ms
12:32:43.982 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=casestudy_24_Java
12:32:43.983 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
12:32:43.984 INFO More about the report processing at http://localhost:9000/api/ce/task?id=1637031a-60d5-4f75-ba51-8065c1fd436b
12:32:43.921 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

The dashboard displays the following key metrics:

- Status:** Passed
- Lines of Code:** 1.3k Lines of Code
- Version:** Version not provided
- Analysis Time:** Last analysis 7 minutes ago
- New Code:** 0 Open issues (A)
- Overall Code:** 0 Open issues (A)
- Maintainability:** 8 Open issues (A)
- Accepted issues:** 0
- Coverage:** On 0 lines to cover.
- Duplications:** 0.0% (On 1.5k lines)
- Security Hotspots:** 0 (A)

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.

The screenshot shows the SonarQube web interface. At the top, there are tabs for Overview, Issues (which is selected), Security Hotspots, Measures, Code, and Activity. On the right, there are Project Settings and Project Information. The main area displays a list of issues. One issue is highlighted with a red border: "ts/.../javascript-module/src/Person.js" with the message "Unexpected var, use let or const instead." It is categorized under "Intentionality" and has a status of "Open". Below this, there are two more issues listed: "its/.../python-module/src/badfortune.py" with the message "Remove the unused local variable "version"" and "its/.../python-module/src/badfortune.py" with the message "Remove the unused local variable "longlen"".

} 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:-

The screenshot shows the SonarQube interface for a Java Maven project. The 'Security Hotspots' tab is selected in the sidebar. The main view lists two code files: 'Person.js' and 'badfortune.py'. For 'Person.js', there is one high-severity issue: 'Unexpected var, use let or const instead.' (Intentionality: es2015 - bad-practice). For 'badfortune.py', there are two issues: 'Remove the unused local variable "version"' (Intentionality: unused) and 'Remove the unused local variable "longlen"' (Intentionality: unused). A warning message 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.'

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:

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](#)

[Cancel](#) [Next](#)

2) Jenkins Pipeline:

⚠️ 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.

Dashboard > All > New Item

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 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/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-6.1.0.4477-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+   stage('SonarQube Analysis') {
7+     withSonarQubeEnv('sonarqube') {
8+       C:\Users\janes\Downloads\sonar-scanner-clt-6.2.1.4610-windows-x64\sonar-scanner-6.2.1.4610-windows-x64\bin\sonar-scanner.bat ^
9+       -D sonar.login=admin ^
10+      -D sonar.password=HareKrishna#108 ^
11+      -D sonar.projectKey=casestudy_24_Java ^
12+      -D sonar.exclusions=vendor/**,resources/**,*/*.java ^
13+      -D sonar.host.url=http://localhost:9000/
14+      ***
15+    }
16+  }
17+ }

```

 Use Groovy Sandbox ?

Pipeline Syntax

SaveApply

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_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 --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 7e0b6392e64cdcf2498beed78c79309d000dacb5 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 7e0b6392e64cdcf2498beed78c79309d000dacb5 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master 7e0b6392e64cdcf2498beed78c79309d000dacb5 # timeout=10
Commit message: "sonarscanner with maven - maven multimodule project"
> git.exe rev-list --no-walk 7e0b6392e64cdcf2498beed78c79309d000dacb5 # 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
12:32:42.104 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
12:32:42.158 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 -----
12:32:42.369 INFO ----- Run sensors on project
12:32:42.369 INFO Sensor Zero Coverage Sensor
12:32:42.456 INFO Sensor Zero Coverage Sensor (done) | time=4ms
12:32:42.459 INFO SCM Publisher SCM provider for this project is: git
12:32:42.661 INFO SCM Publisher 18 source files to be analyzed
12:32:43.214 INFO SCM Publisher 18/18 source files have been analyzed (done) | time=549ms
12:32:43.214 INFO CPU Executor Calculating CPU for 0 files
12:32:43.215 INFO CPU Executor CPU calculation finished (done) | time=0ms
12:32:43.234 INFO SCM revision ID '7e0b0392e64cdcf2490beed78c793090000dacb5'
12:32:43.583 INFO Analysis report generated in 15ms, dir size=38.3 kB
12:32:43.732 INFO Analysis report compressed in 13ms, zip size=68.9 kB
12:32:43.901 INFO Analysis report uploaded in 10ms
12:32:43.902 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=casestudy_24_Java
12:32:43.906 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
12:32:43.906 INFO More about the report processing at http://localhost:9000/api/ce/task?id=1e37031a-60d5-4f75-ba51-8005c1fd436b
12:32:43.911 INFO Analysis total time: 15.476 s
12:32:43.912 INFO SonarScanner Engine completed successfully
12:32:44.024 INFO EXECUTION SUCCESS
12:32:44.022 INFO Total time: 17.991s
[Pipeline]
[Pipeline] // withdrawsonarQubeEnv
[Pipeline] // stage
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.462.1

Successfully analyzed the Java project using SonarQube.

4] Java Project:-

[Project Link: https://github.com/nerdschoolbergen/code-smells.git](https://github.com/nerdschoolbergen/code-smells.git)

1) SonarQube Project:

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](#)

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.

2) Jenkins Pipeline:

Dashboard > All > New Item

New Item

Enter an item name

Select an item type

Pipeline 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.

Freestyle project 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:-

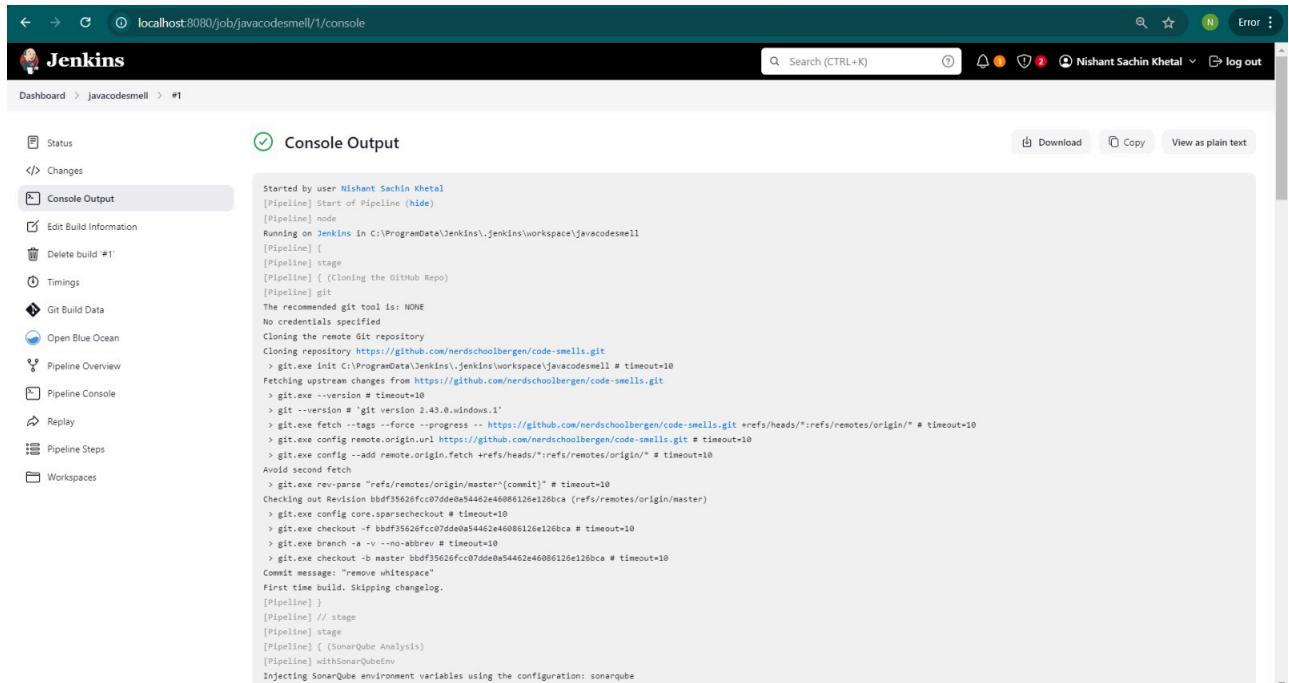
```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/nerdschoolbergen/code-smells.git'
    }
    stage('SonarQube Analysis') {
        withSonarQubeEnv('sonarqube') {
            bat
            "C:\\\\Users\\\\nish\\\\Downloads\\\\sonar-scanner-cli-6.1.0.4477-windows-x64\\\\sonar-scanner-6.1.0.4
477-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 Pipeline configuration page for a job named 'javacodesmell'. The 'Advanced Project Options' tab is selected. In the 'Definition' section, the 'Pipeline script' tab is chosen. The pipeline script is displayed in a code editor:

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

Below the script editor, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the page, there are 'Save' and 'Apply' buttons, and a footer indicating 'REST API' and 'Jenkins 2.462.1'.

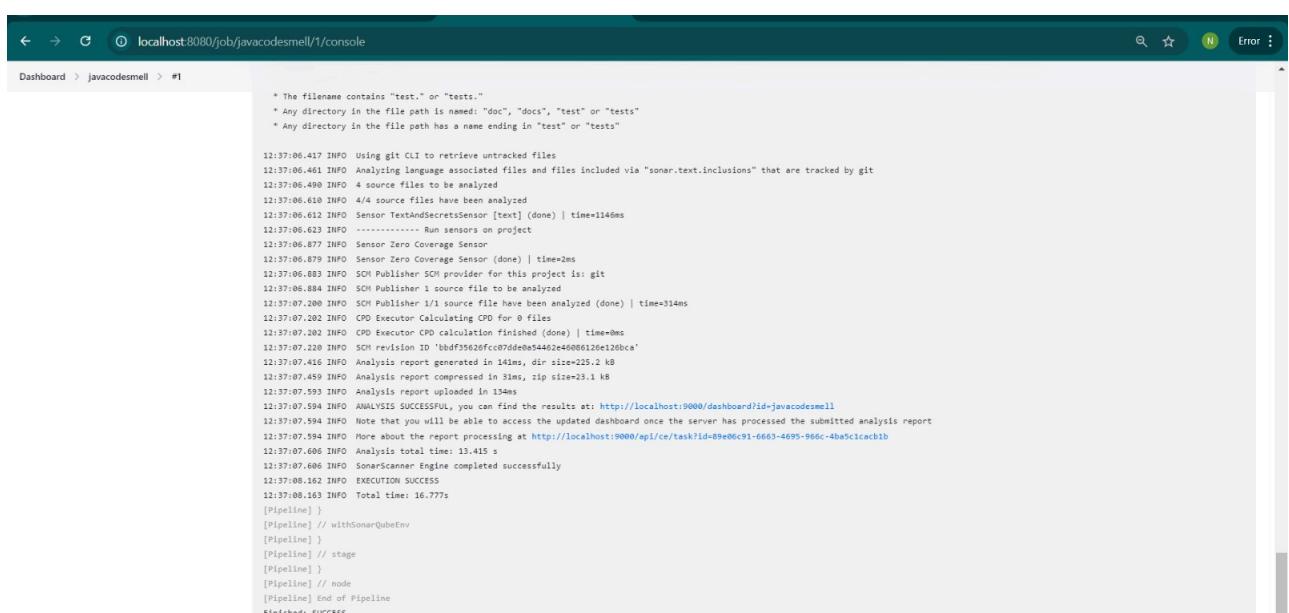
3) Console Output:

 Jenkins Console Output for job javacodesmell #1. The output shows the Jenkins pipeline starting, cloning the GitHub repository, and running SonarQube analysis. It includes logs for git fetch, SonarQube analysis, and the final success message.

```

Started by user Nishant Sachin Khetal
[Pipeline] Start of Pipeline [hide]
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\javacodesmell
[Pipeline] [
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/nerdschoolbergen/code-smells.git
> git.exe init C:\ProgramData\Jenkins\jenkins\workspace\javacodesmell # timeout=10
Fetching upstream changes from https://github.com/nerdschoolbergen/code-smells.git
> git.exe -v --version # timeout=10
> git.exe --version # git version 2.43.0.windows.1"
> git.exe fetch --tags --force --progress -- https://github.com/nerdschoolbergen/code-smells.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/nerdschoolbergen/code-smells.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision bbdff5962fccc07d0e854462e440886126e126bcfa (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f bbdff5962fccc07d0e854462e440886126e126bcfa # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe checkout -b master bbdff5962fccc07d0e854462e440886126e126bcfa # timeout=10
Commit message: "remove whitespace"
First time build. Skipping changelog.
[Pipeline] [
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SonarQube Analysis)
[Pipeline] withSonarQubeEnv
Injecting SonarQube environment variables using the configuration: sonarqube
sonarqubeEnv

```

 Jenkins Console Output for job javacodesmell #1. The output shows the Jenkins pipeline starting, cloning the GitHub repository, and running SonarQube analysis. It includes logs for git fetch, SonarQube analysis, and the final success message.

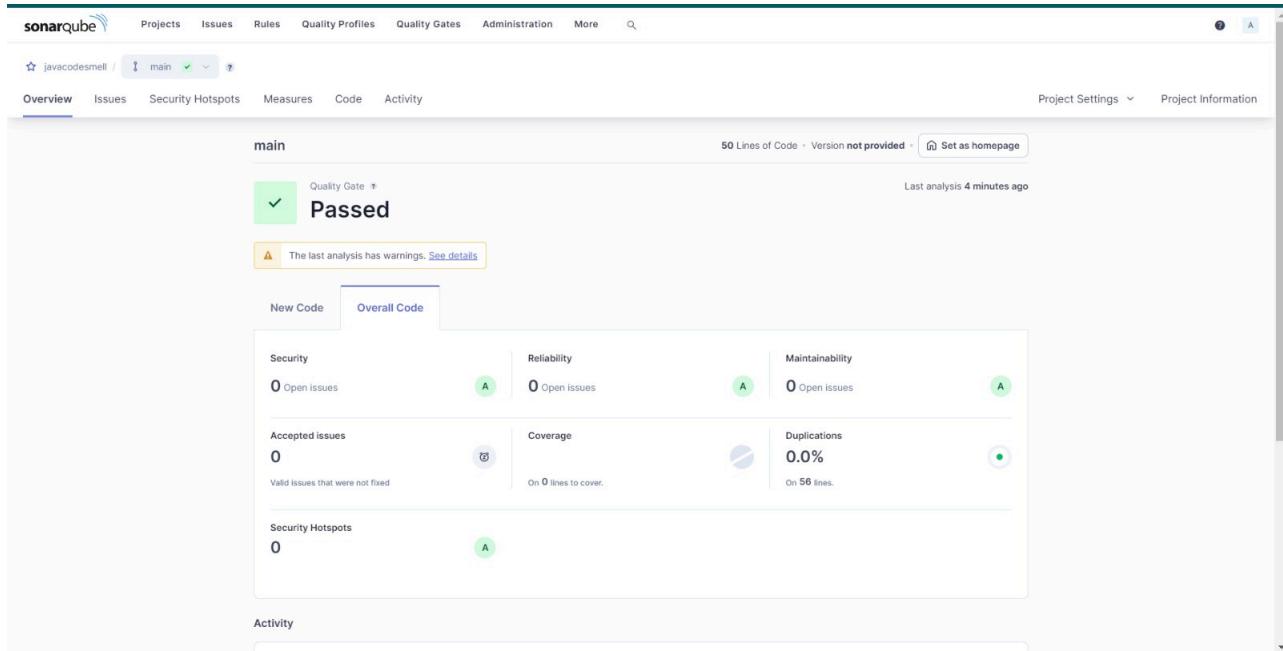
```

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
12:37:06.610 INFO 4/4 source files have been analyzed
12:37:06.612 INFO Sensor TextAndSecretsSensor [text] (done) | time=114ms
12:37:06.623 INFO -----
12:37:06.623 INFO ----- Run sensors on project
12:37:06.677 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 CPD Executor 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 'bbdff5962fccc07d0e854462e440886126e126bcfa'
12:37:07.416 INFO Analysis report generated in 14mins, dir size=215.2 kB
12:37:07.459 INFO Analysis report compressed in 3mins, zip size=23.1 kB
12:37:07.593 INFO Analysis report uploaded in 13mins
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
12:37:07.594 INFO More about the report processing at http://localhost:9000/api/ce/task?id=89e06c91-6663-4695-966c-4ba5c1cabc1b
12:37:07.606 INFO Analysis total time: 13.415 s
12:37:08.066 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:



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.