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:

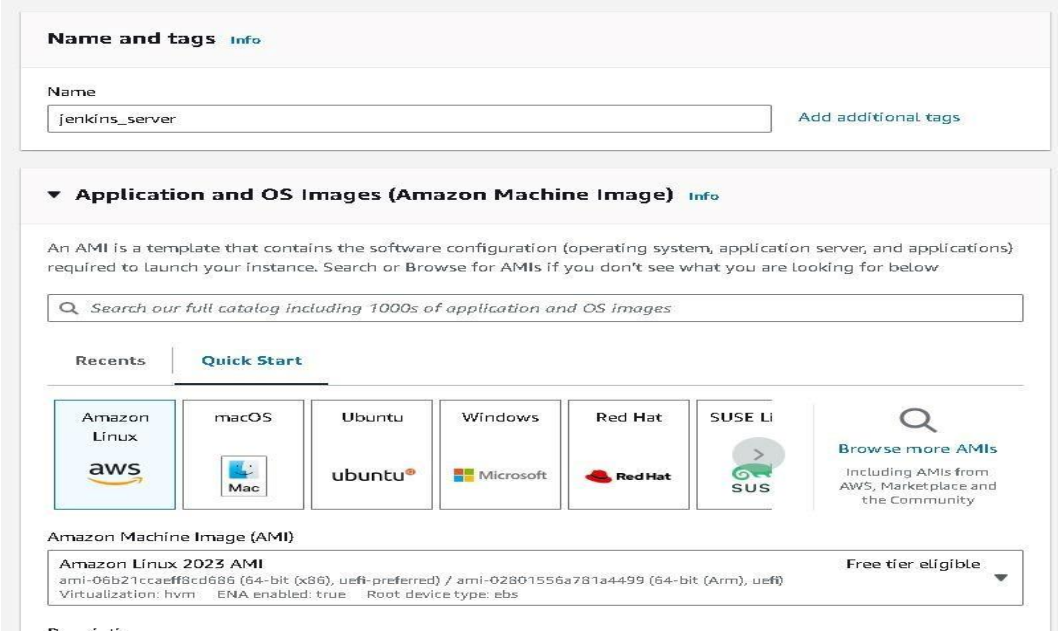
- **Automated Testing:** Every new feature, bug fix, or update will trigger automated tests, ensuring that the platform remains stable and bug-free after each code commit.
- **Code Quality:** SonarQube will continuously monitor the quality of the codebase, making sure the project grows with clean, maintainable 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

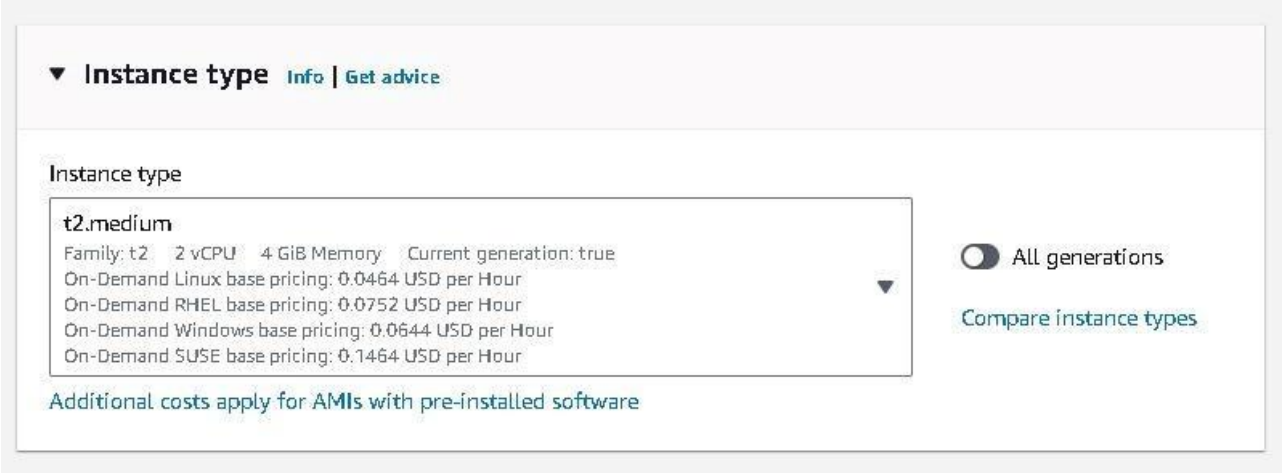
Problem Statement

1] Jenkins on an EC2 instance:Resource:https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

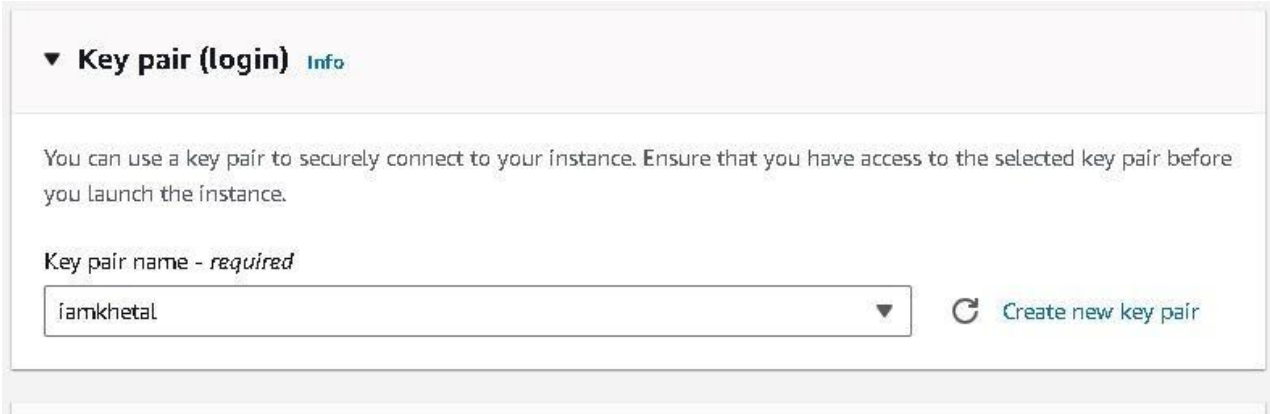
1) Launch a AWS EC2 instance with a Linux OS .



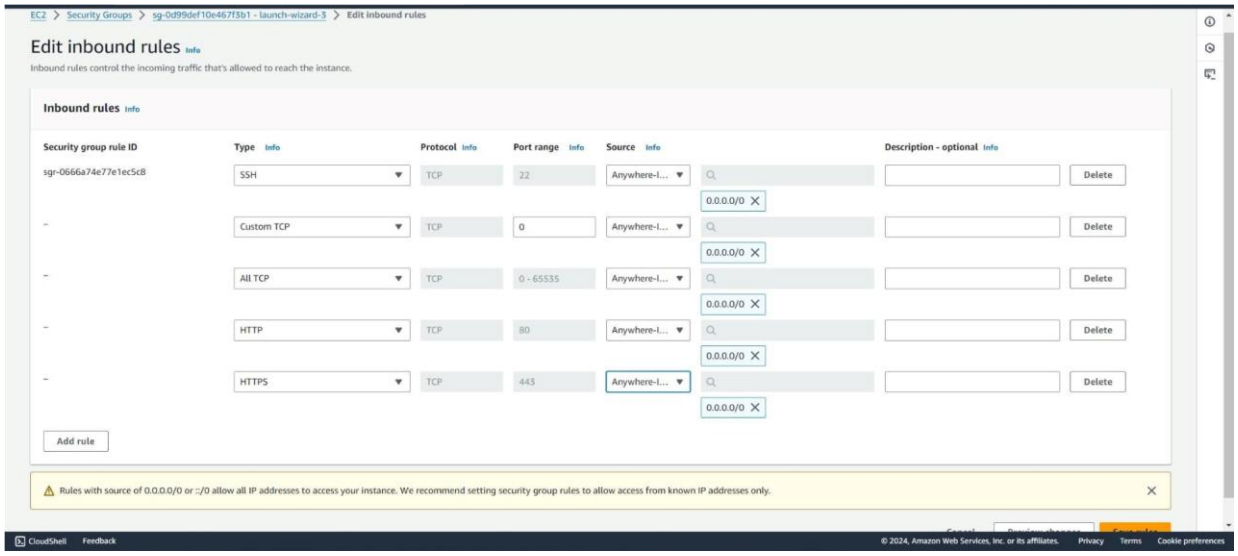
2) instance type as t2.medium.



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

```
PS C:\Users\ADMIN\Downloads\advdevops> ssh -i "iamkhetal.pem" ec2-user@ec2-35-174-104-174.compute-1.amazonaws.com
       #_
       ~\_  ####_        Amazon Linux 2023
      ~~  \_#####\
      ~~     \###|
      ~~       \#/ ___   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 ~]$
```

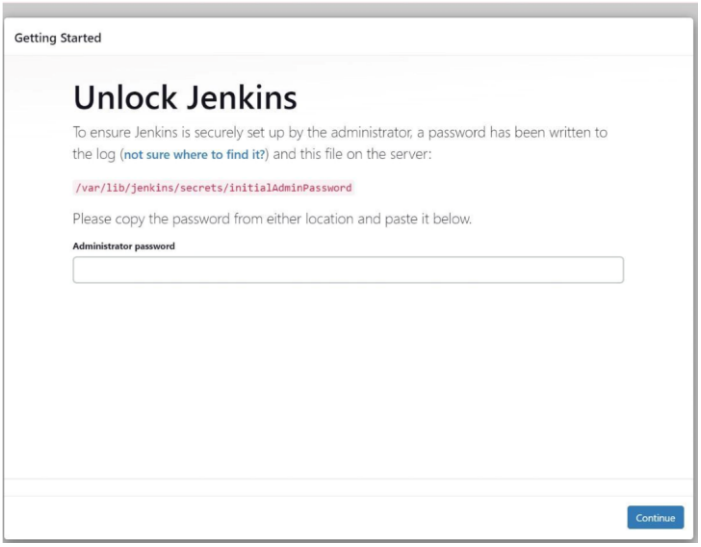
```
[ec2-user@ip-172-31-47-120 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2024-10-18 18:09:04--  https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.34.133, 2a04:4e42:79::645
Connecting to 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 ~]$
```

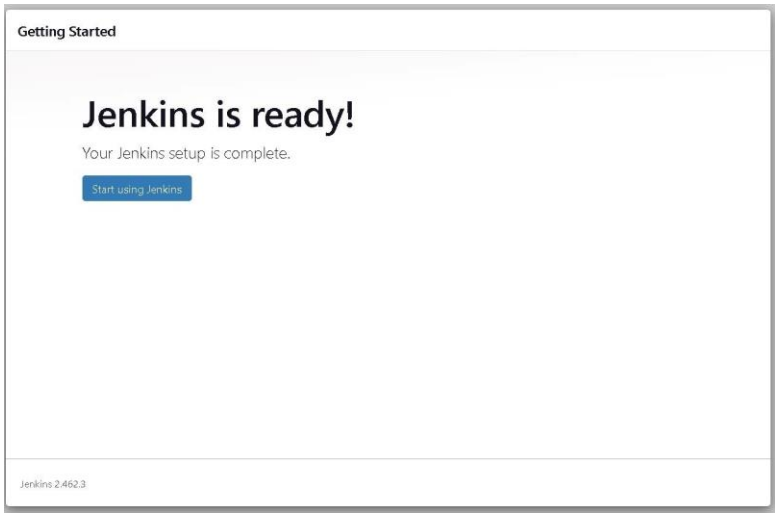
```
[ec2-user@ip-172-31-47-120 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/j
enkins.io-2023.key
[ec2-user@ip-172-31-47-120 ~]$ sudo yum upgrade
Jenkins-stable                                           325 kB/s |  29 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-47-120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

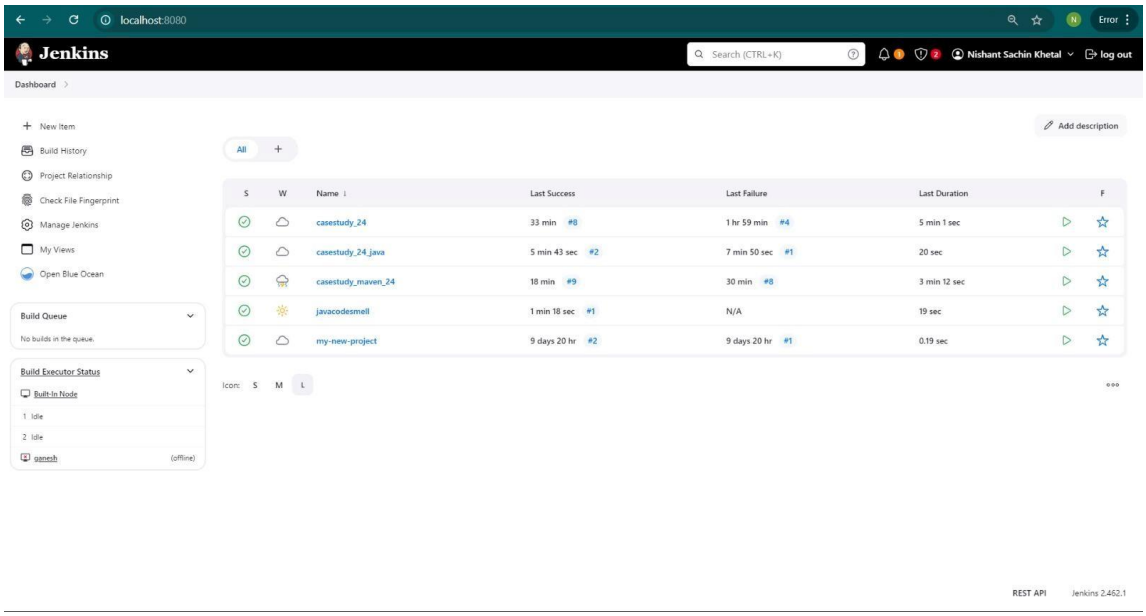
```
[ec2-user@ip-172-31-47-120 ~]$ sudo yum install jenkins -y
Last metadata expiration check: 0:02:03 ago on Fri Oct 18 18:09:29 2024.
Dependencies resolved.
================================================================================================
 Package             Architecture      Version                  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.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$1#onAttained: Completed initializa
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.366+0000 [id=24]	INFO	hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 18 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.436+0000 [id=47]	INFO	h.m.DownloadService$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
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
ganesh (offline)
Add description
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
javacodesmell 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
Icon: S M L
REST API Jenkins 2.462.1

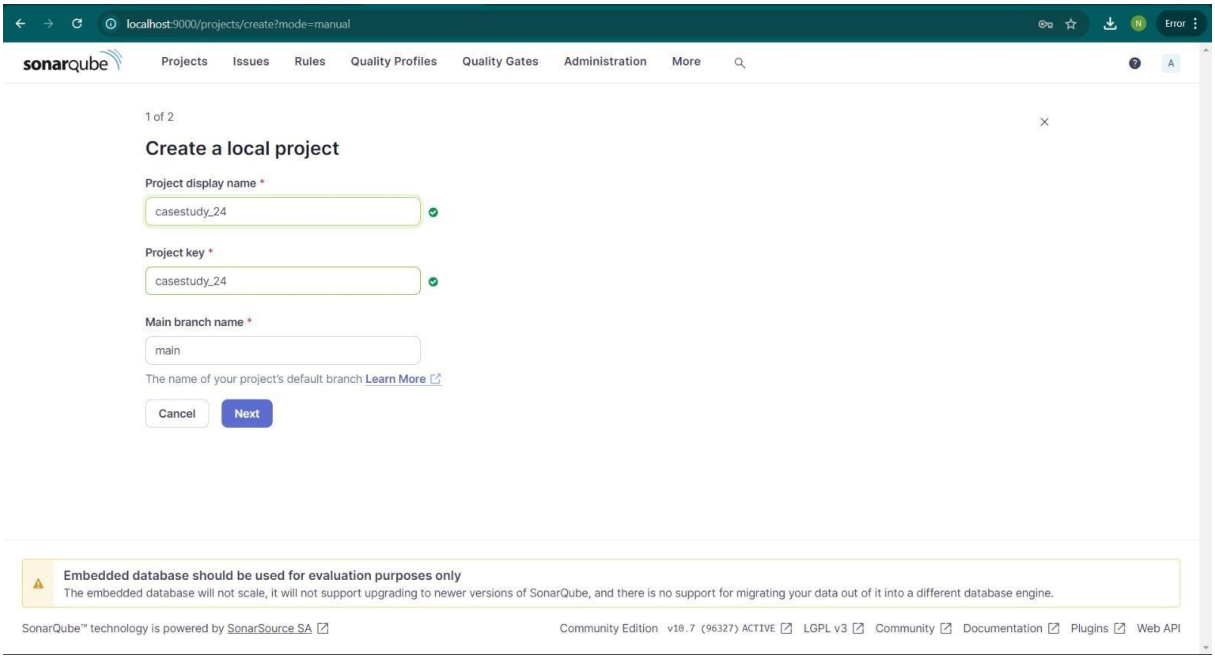
Thus, Jenkins is successfully configured on the EC2 Linux instance.

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

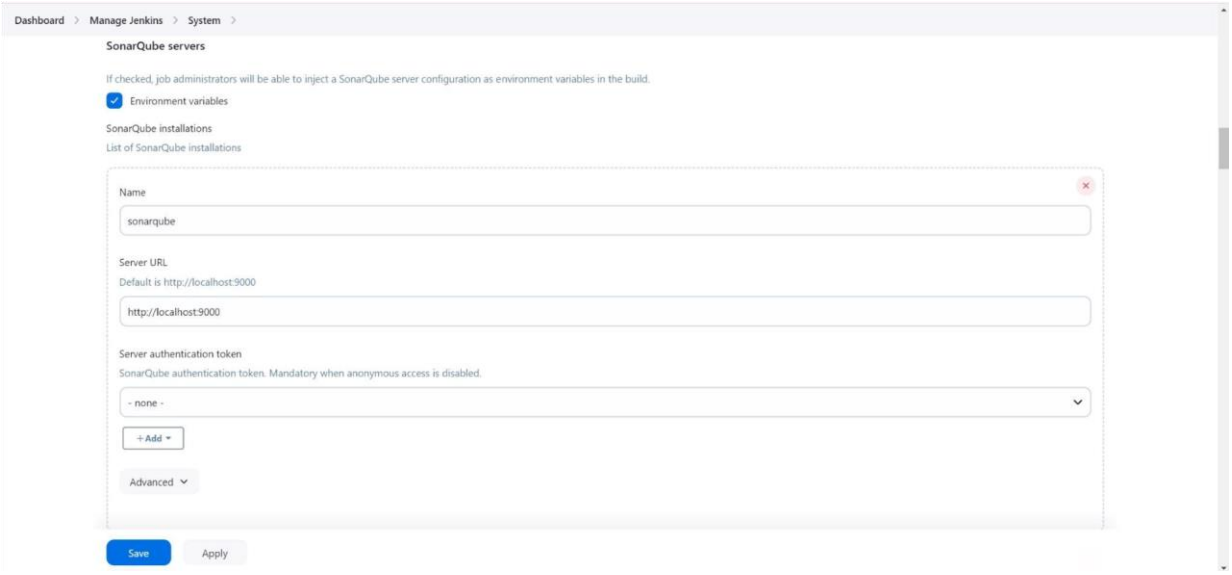
A] Sonarqube project:-

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

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



2) Sonarqube configurations in jenkins:-



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

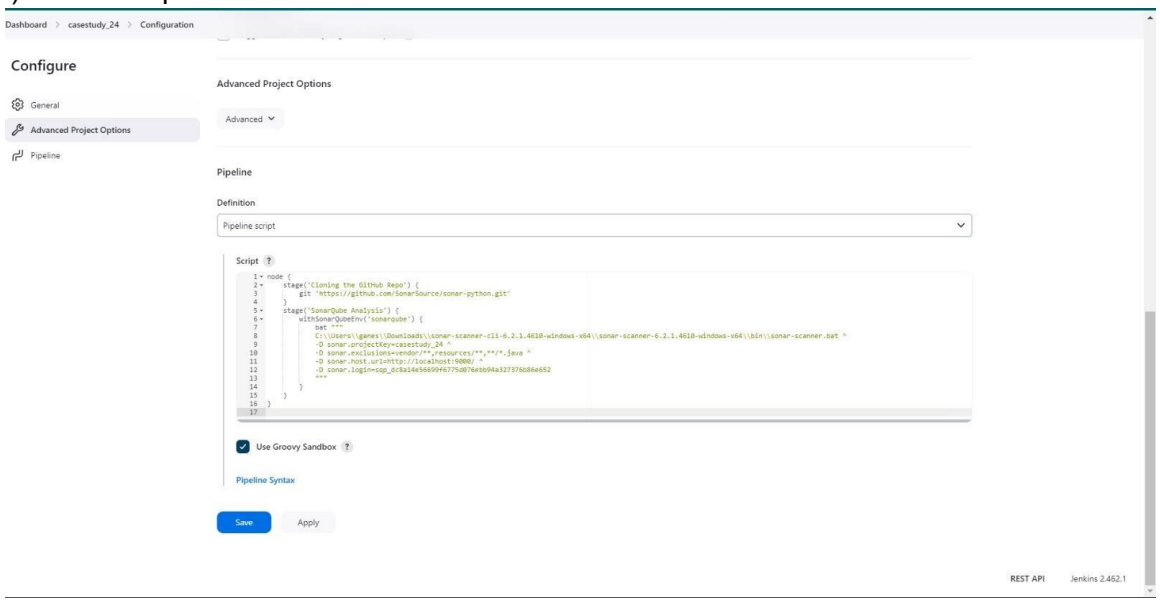
3.9.8

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-6.1.0.4
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 :

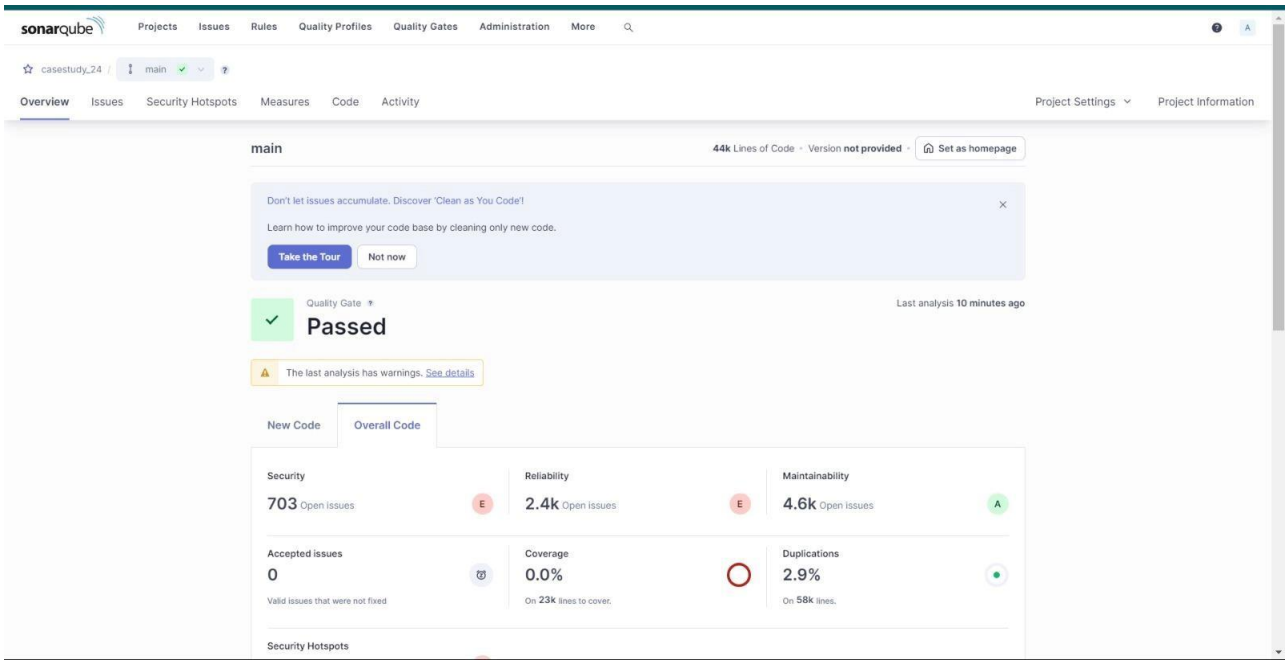
Console Output

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=15676ms
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'
11:18:34.836 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
11:18:56.550 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
```

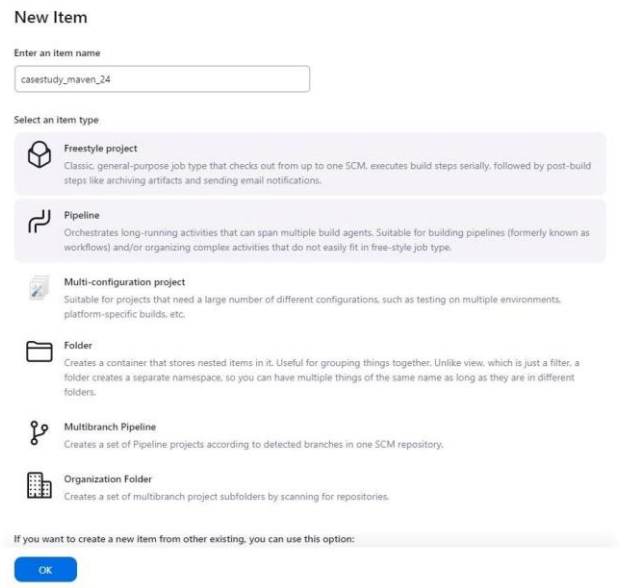
Sonarqube Analysis:-



Thus, the Python project was successfully analyzed with SonarQube.

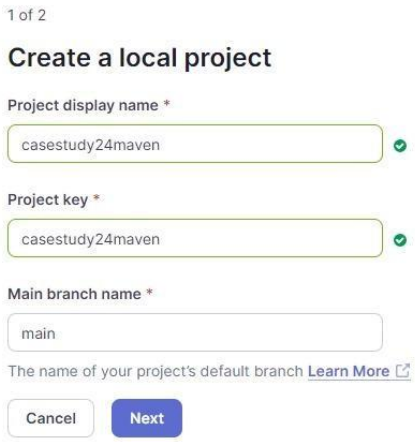
B]For Maven Project:-

1) Make a Jenkins Pipeline.



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

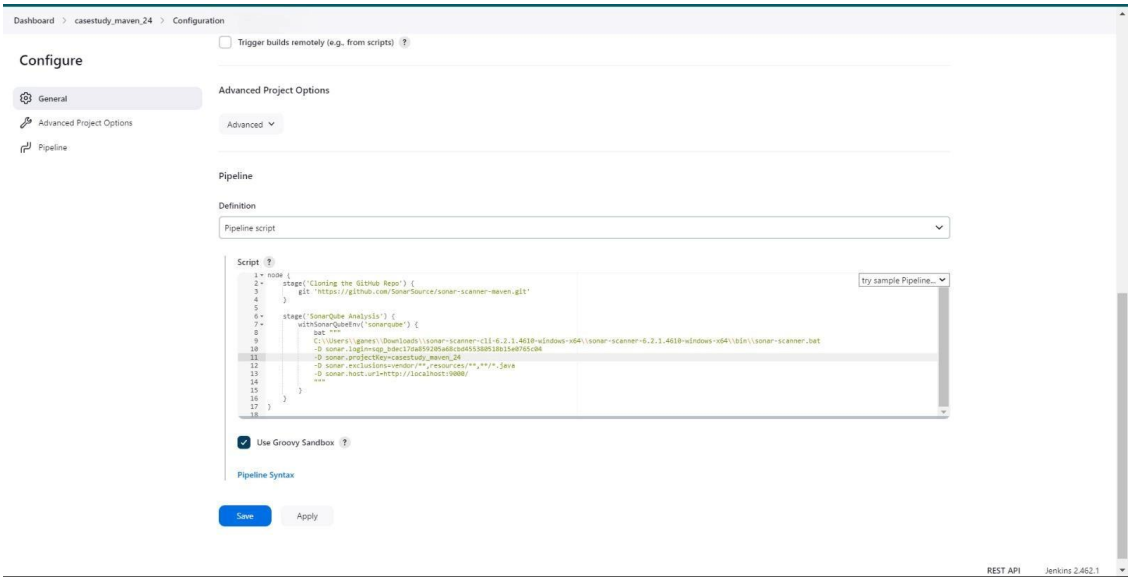
2) Create a sonarqube project named **casestudy24maven.**



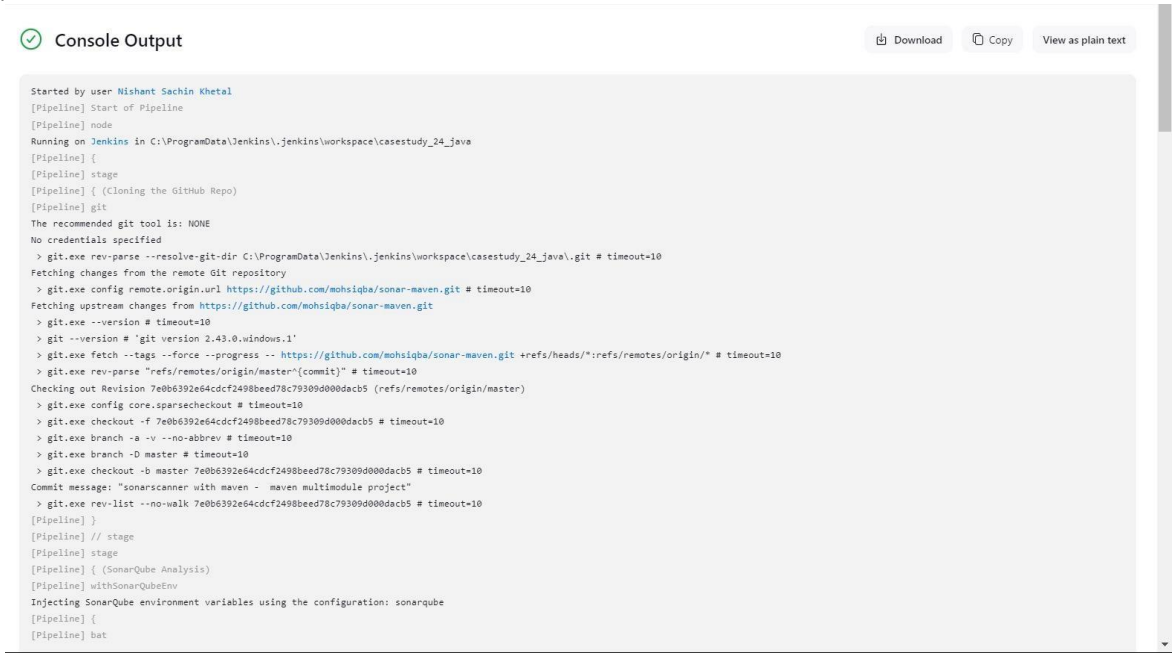
Pipeline Script:node

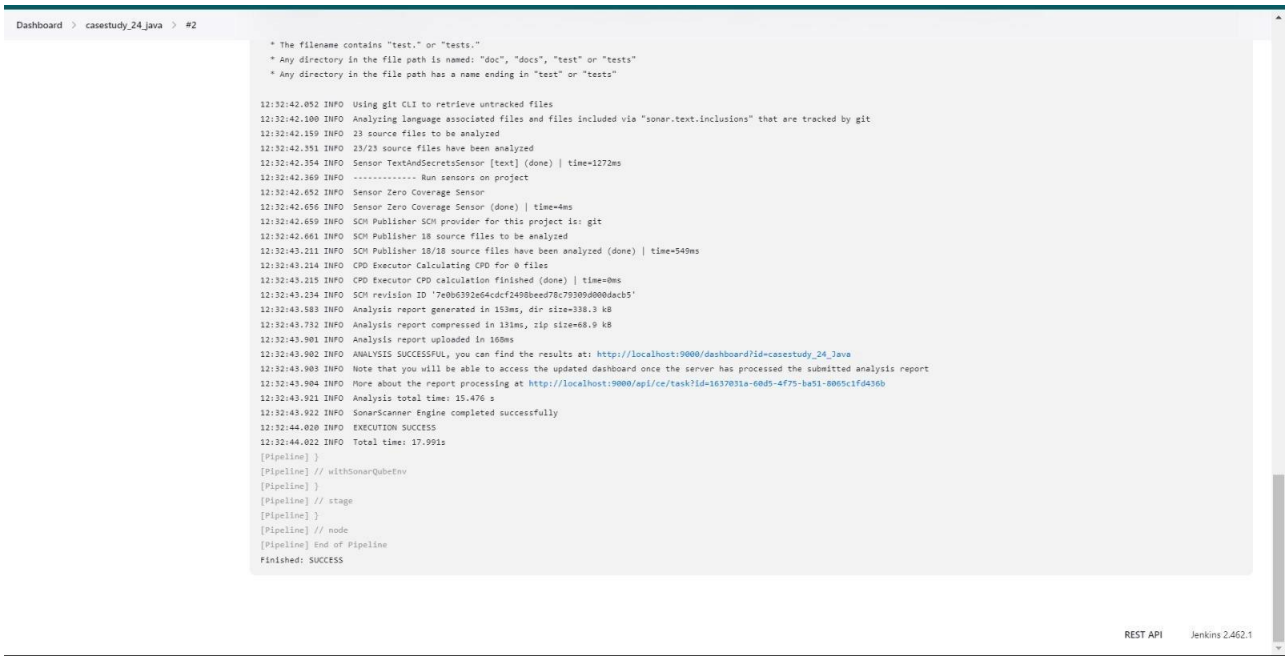
```
{
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/SonarSource/sonar-scanner-maven.git'
   }
   stage('SonarQube Analysis') {
      withSonarQubeEnv('sonarqube') {
```

```
        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/"
    }
  }
}
```

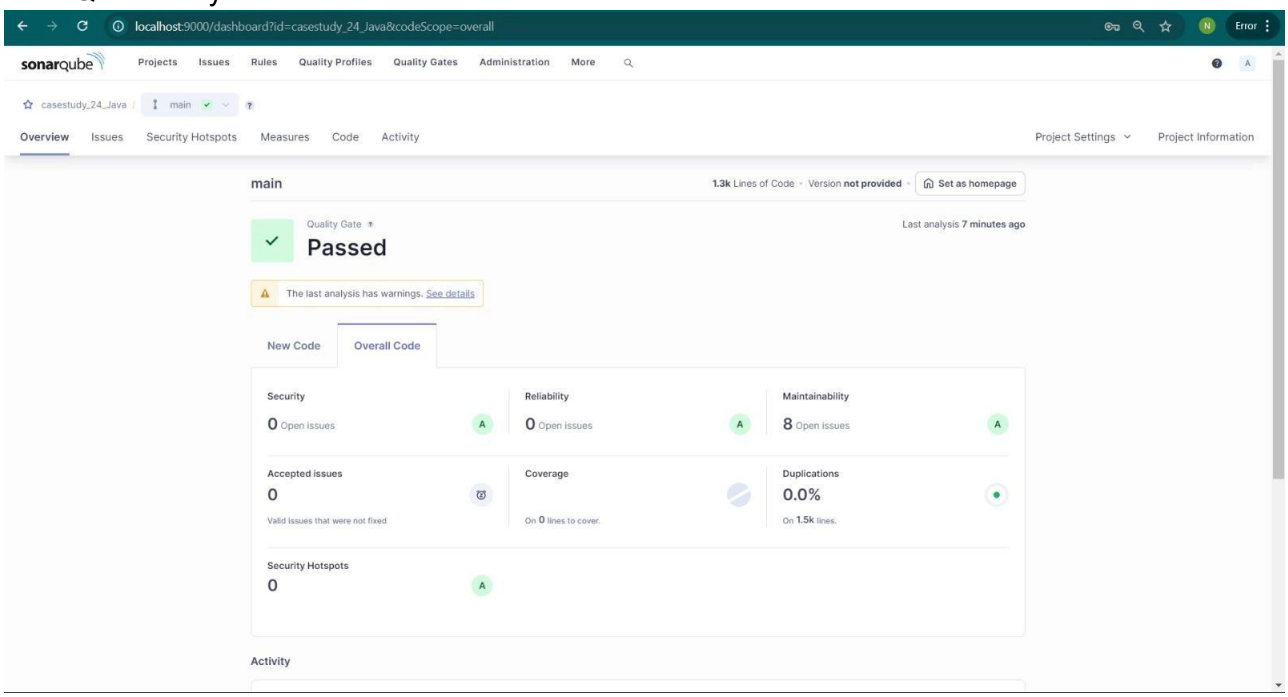


3) Build and check its console output:-





SonarQube analysis



Attributes:-

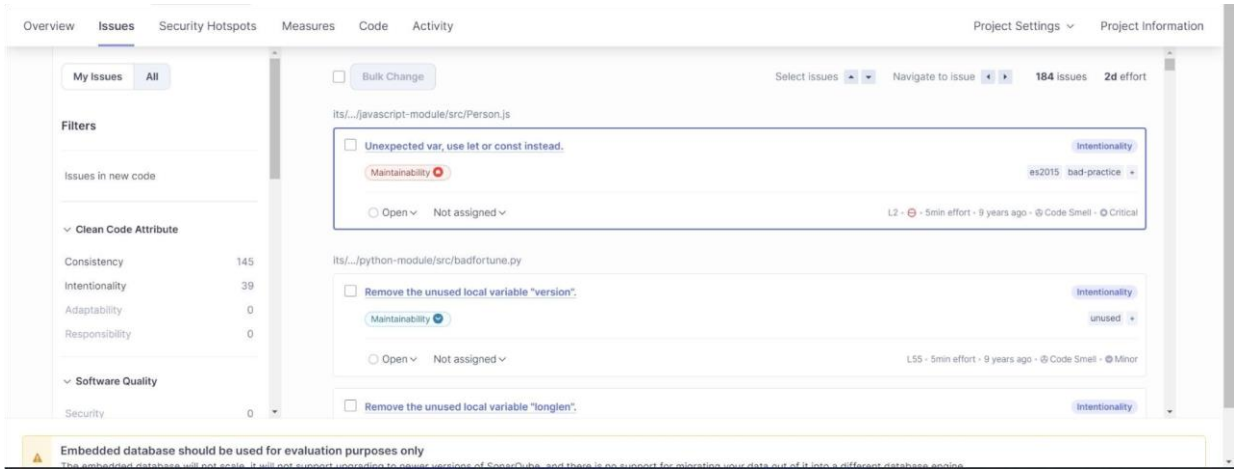
Consistency: Ensures the code follows uniform patterns and styles throughout, making it easier to read and understand.

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

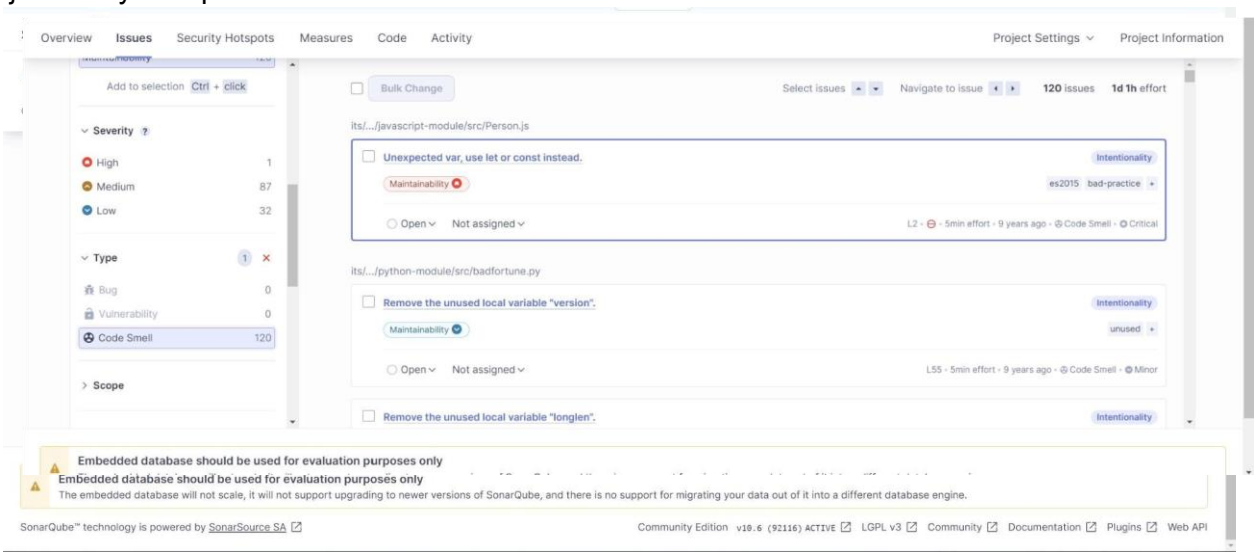
Adaptability: Indicates how flexible the code is to accommodate changes or new features without requiring major rewrites.

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

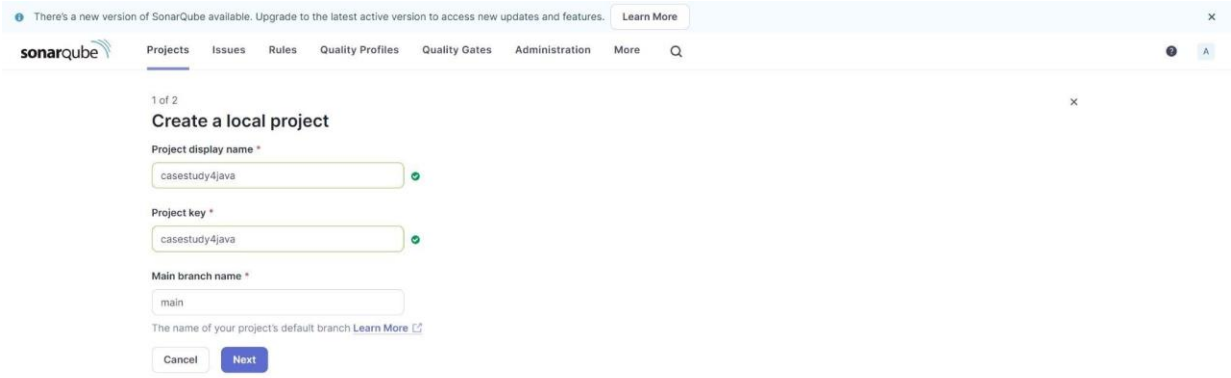
resilience against attacks. SonarQube assigns review priority based on standards like OWASP Top 10 and CWE Top 25, with hotspots categorized as high, medium, or low priority. Developers review the code, assess risks, and apply necessary fixes or mark it as safe. Addressing security hotspots strengthens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

Successfully analyzed the Java Maven project using SonarQube.

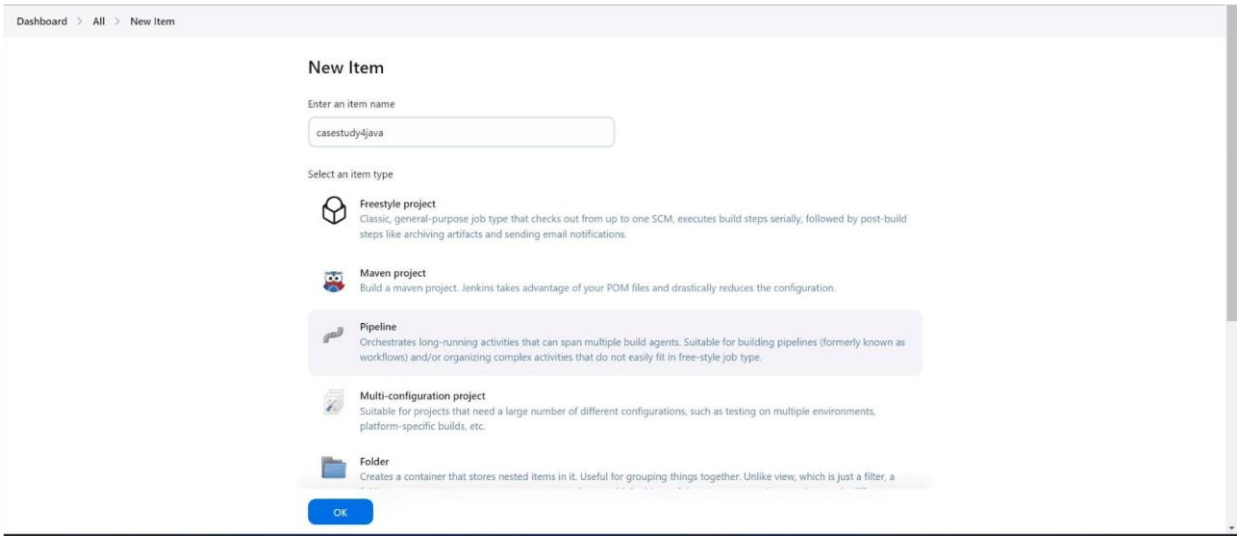
3] Java Project:-

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

1)SonarQube Project:



2)Jenkins Pipeline:



Pipeline Script:-

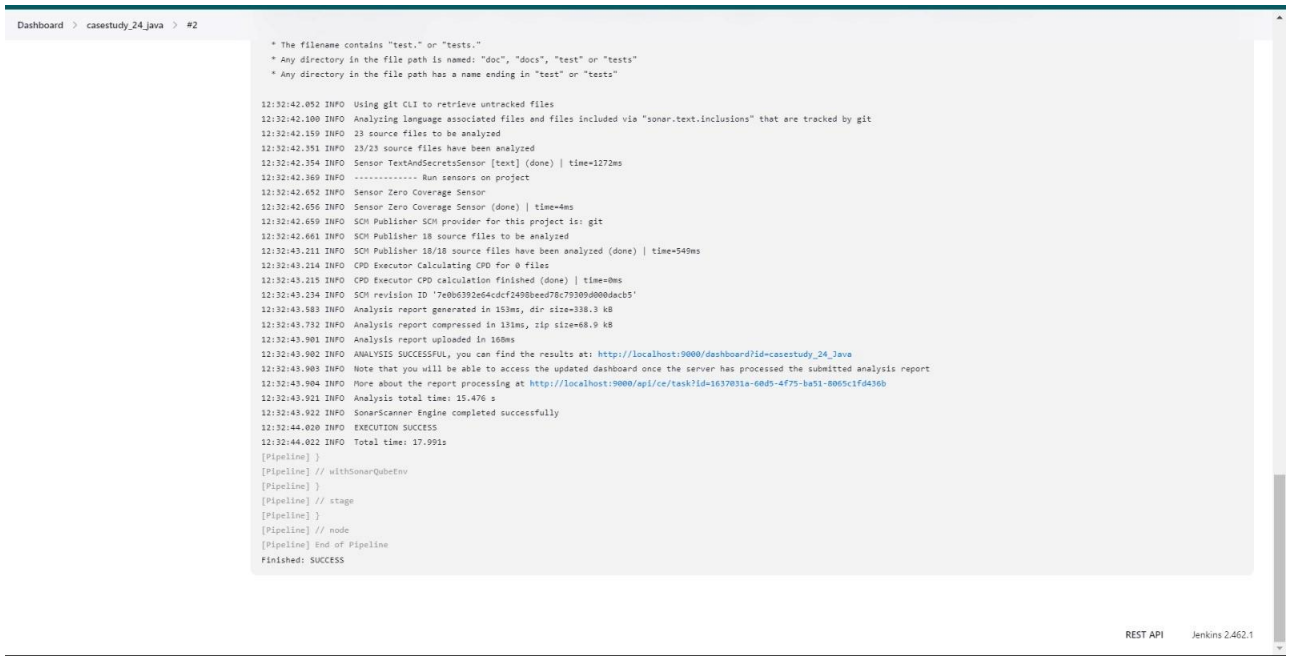
```
node {
   stage('Cloning the GitHub Repo') { git
      'https://github.com/mohsiqba/sonar-maven.git'
   }
```

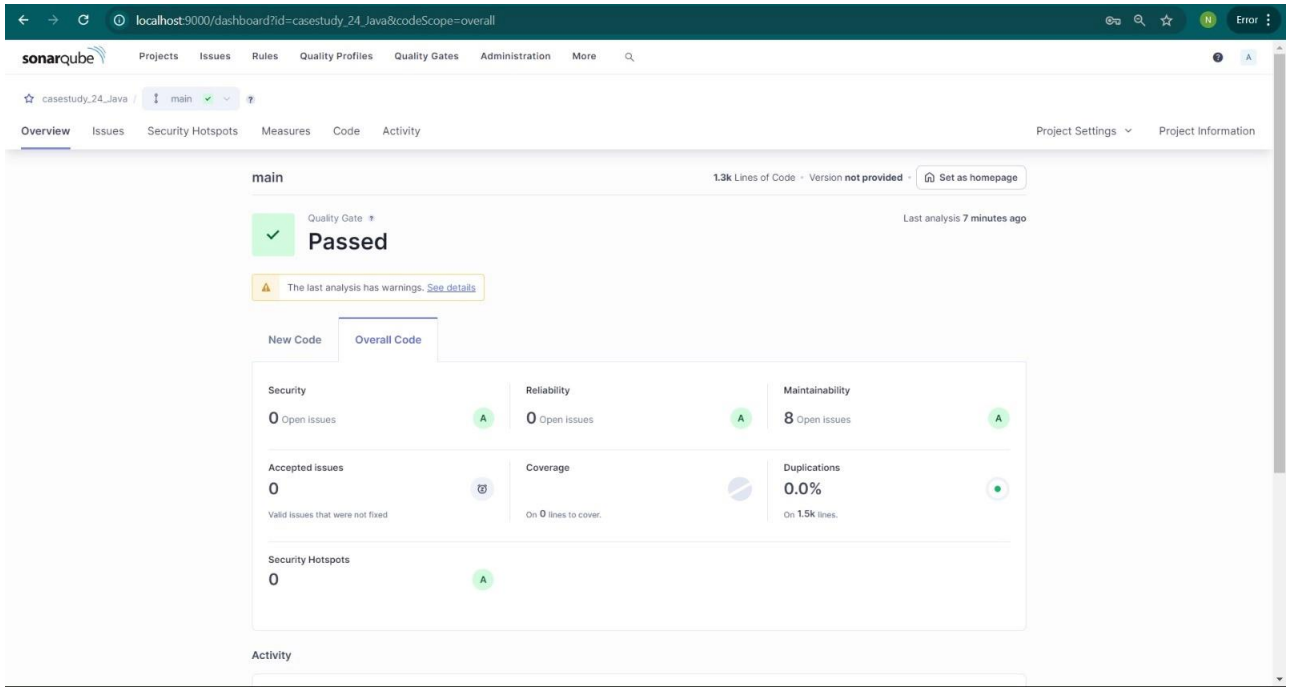
```
    stage('SonarQube Analysis') {
        withSonarQubeEnv('sonarqube') {
            bat
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4
477-windows-x64\\bin\\sonar-scanner.bat " +
                "-D sonar.login=admin " +
                "-D sonar.password=HareKrishna#108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

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 --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 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
```

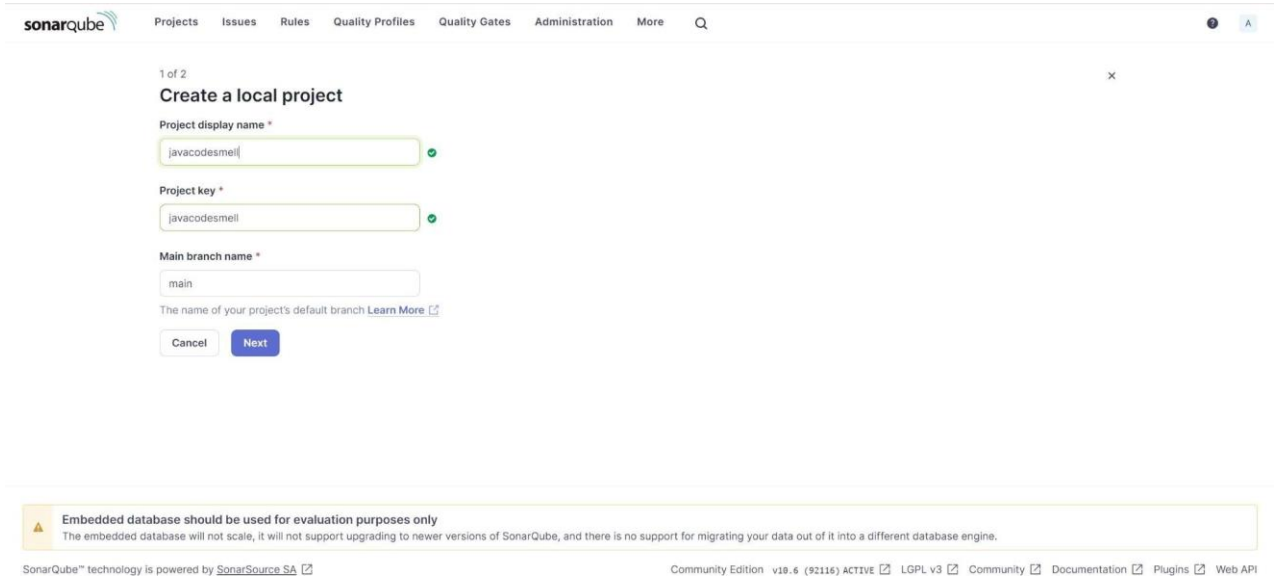




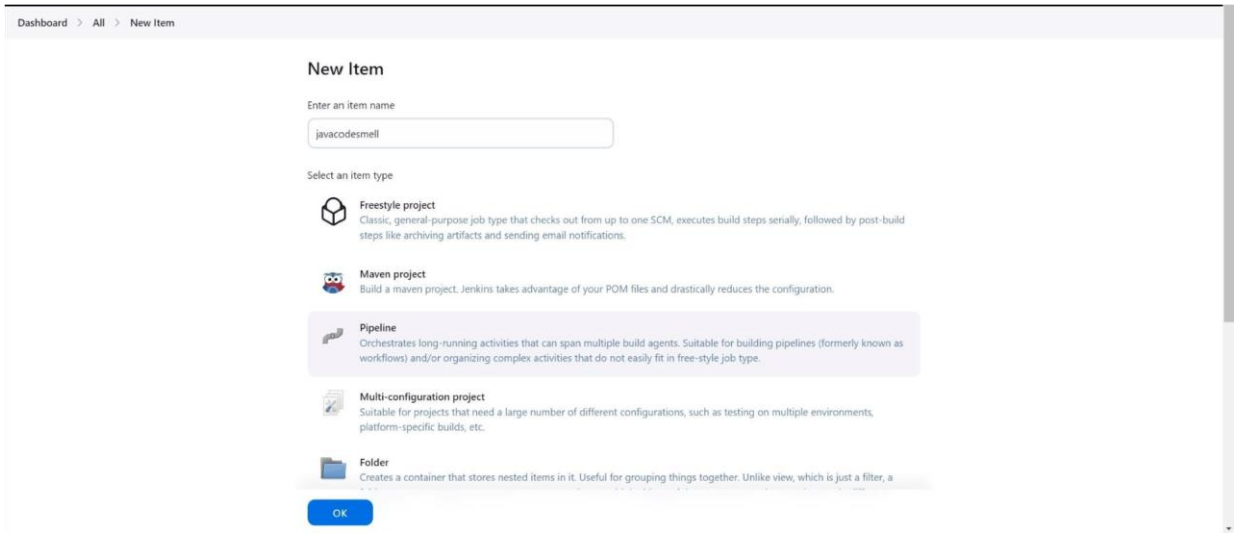
Successfully analyzed the Java project using SonarQube. 4] Java Project:-

Project Link: https://github.com/nerdschoolbergen/code-smells.git

1)SonarQube Project:



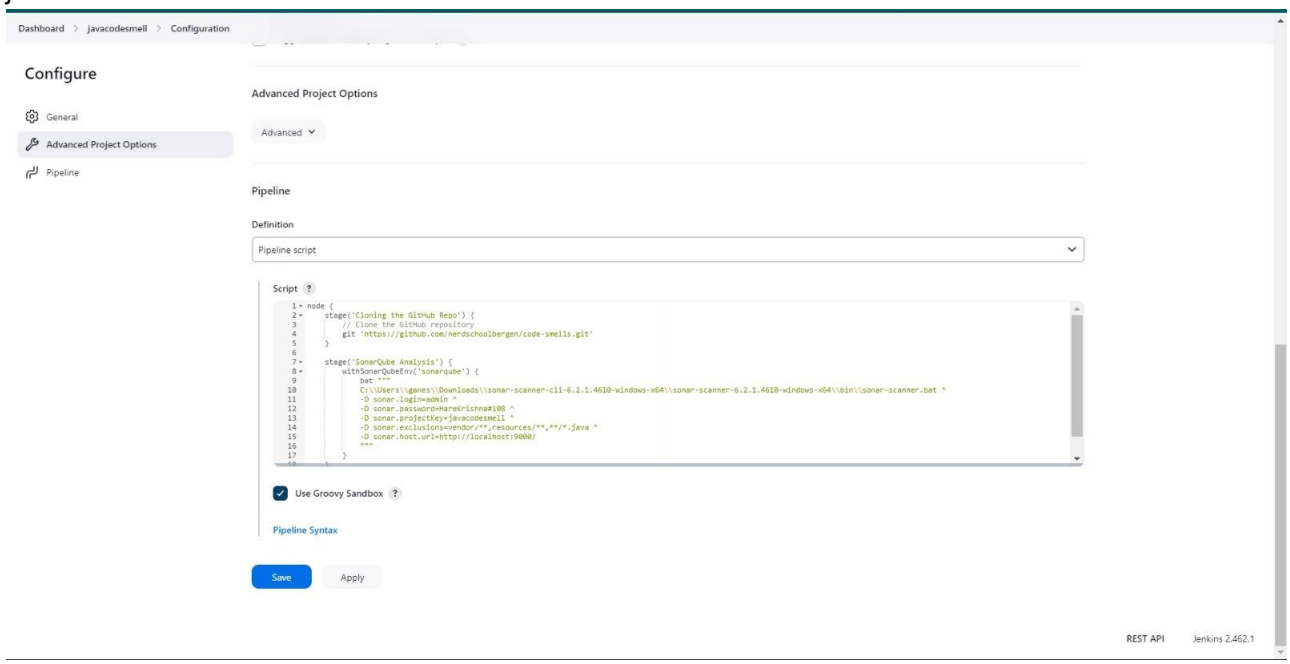
2)Jenkins Pipeline:



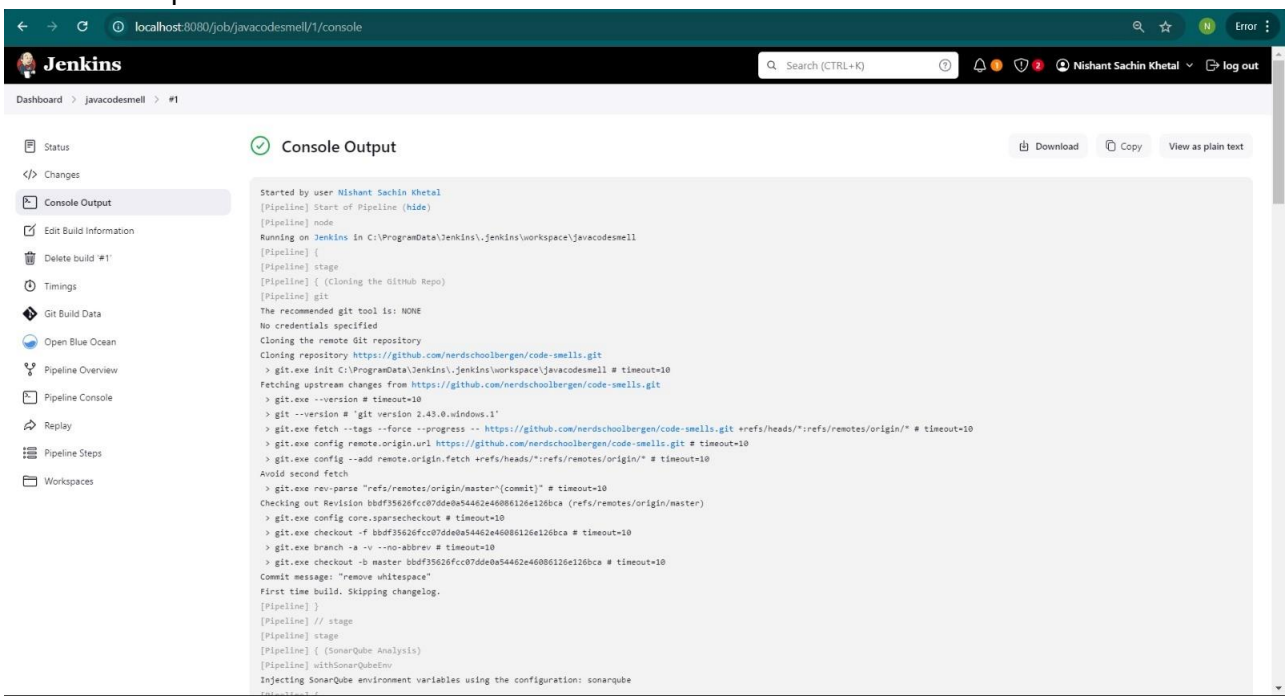
Pipeline Script:-

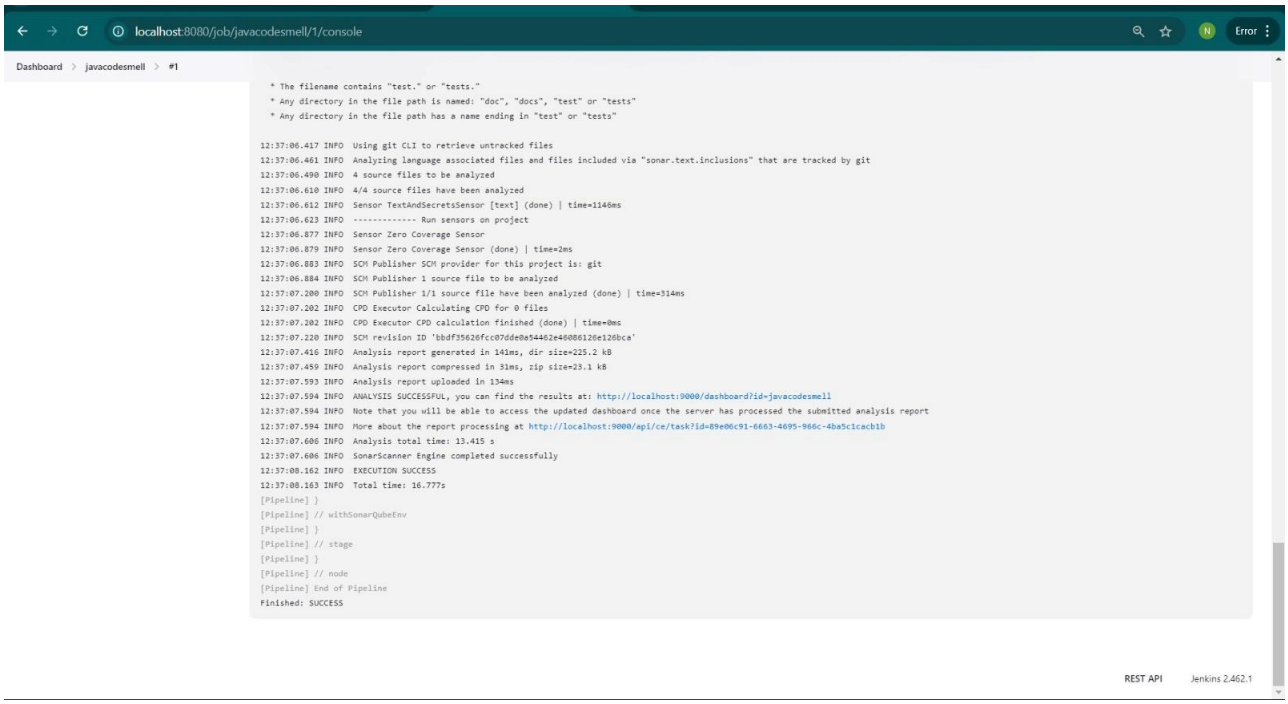
```
node {
   stage('Cloning the GitHub Repo') { git
      'https://github.com/nerdschoolbergen/code-smells.git'
   }
   stage('SonarQube Analysis') {
      withSonarQubeEnv('sonarqube') {
         bat
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat " +
            "-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/"
        }
    }
}
```

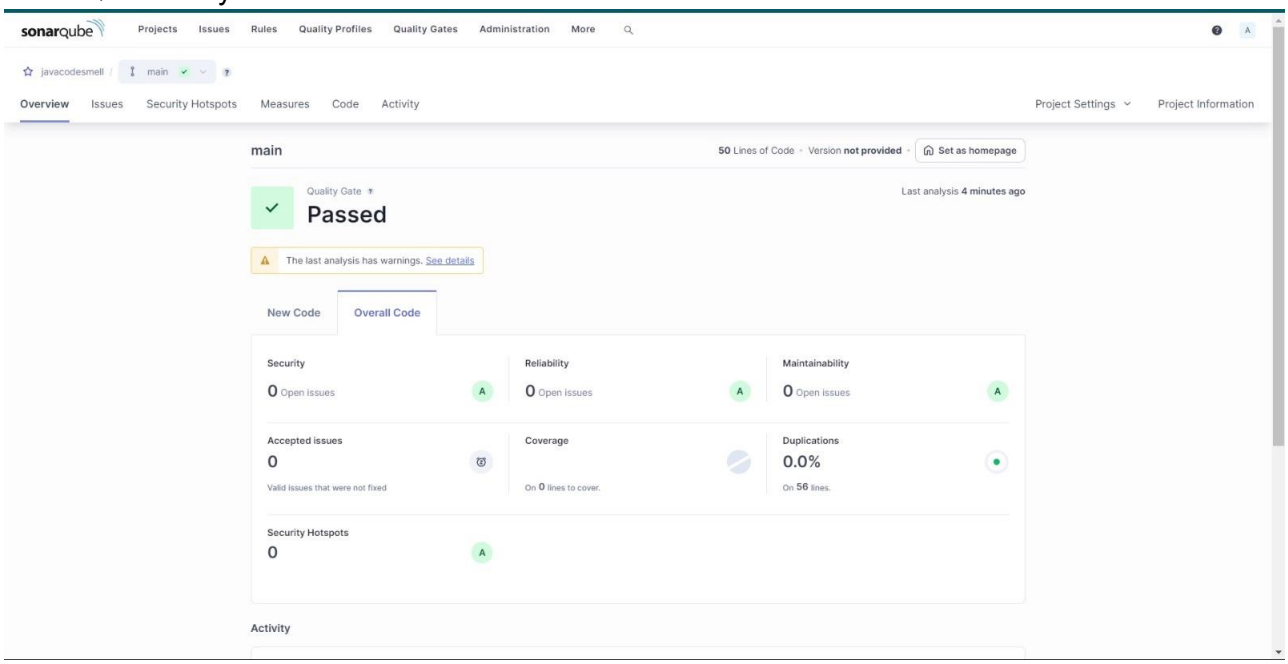


3) Console Output:





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.