Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

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.

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

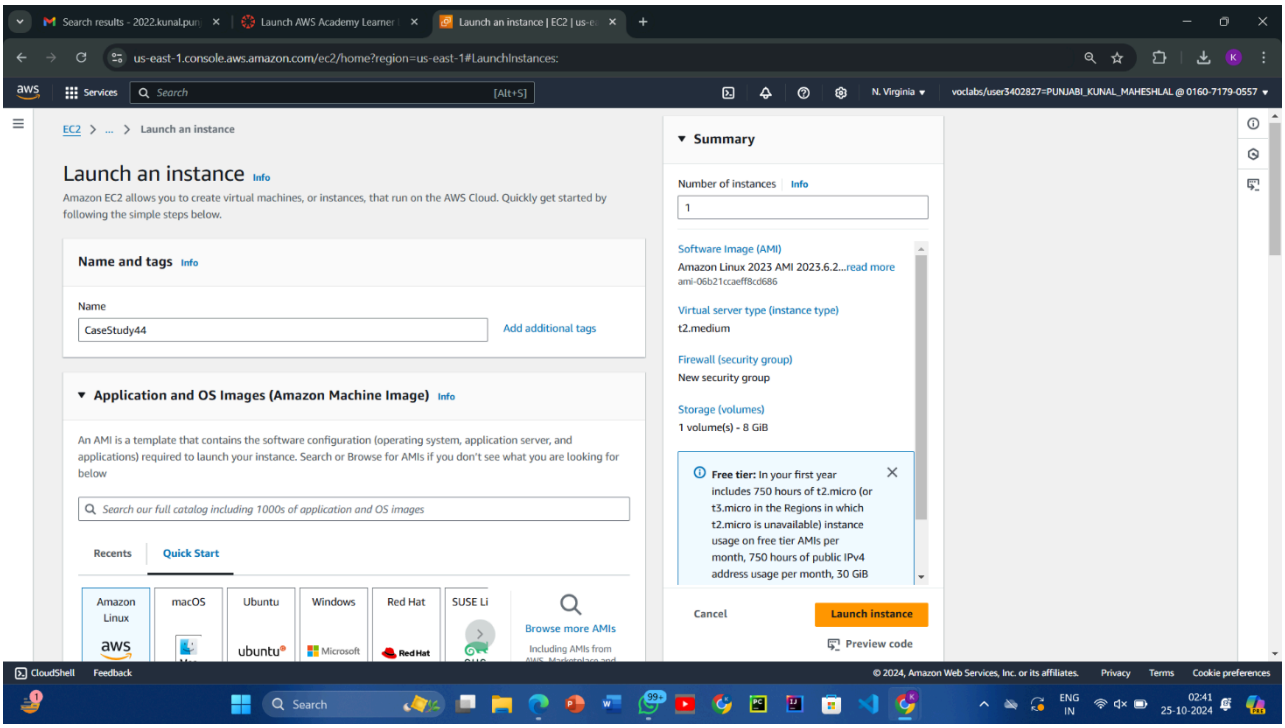
- **Security Scanning:** SonarQube's security checks will help detect vulnerabilities early, ensuring that sensitive data like event details and user information is protected before the code is deployed to production.

2. Demonstration

Problem Statement

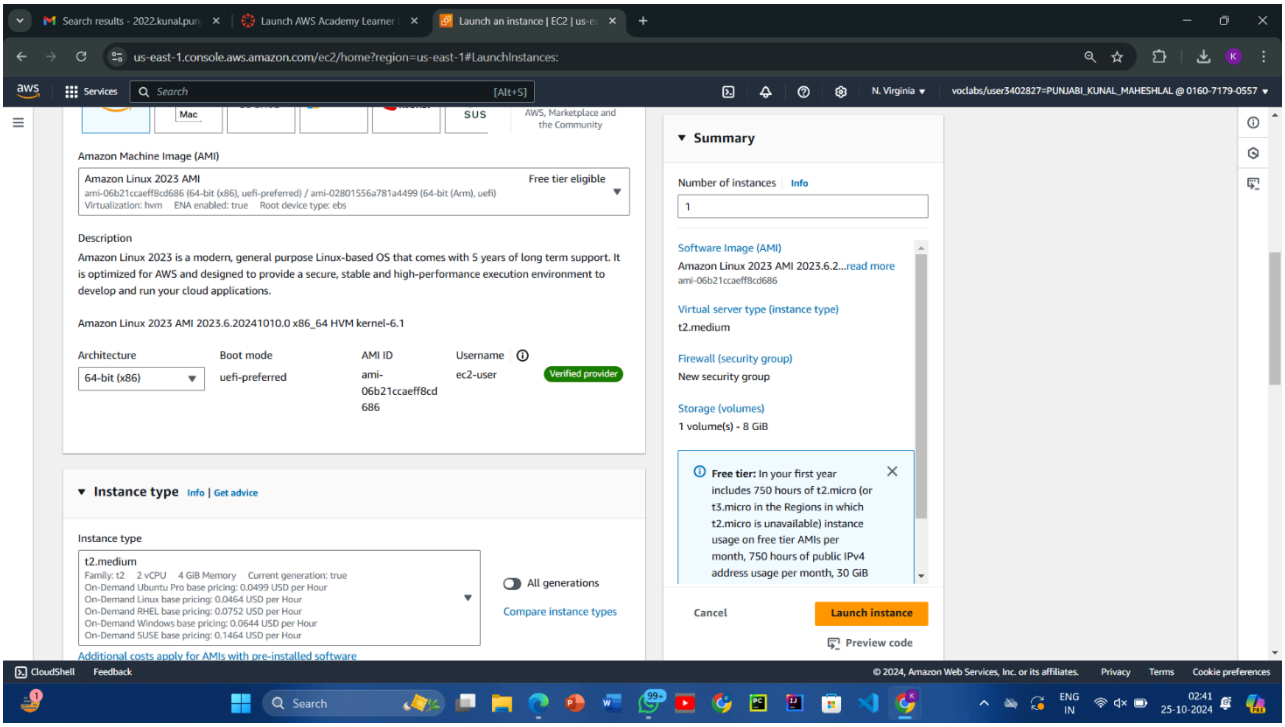
1] Jenkins on an EC2 instance:Resource:https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

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

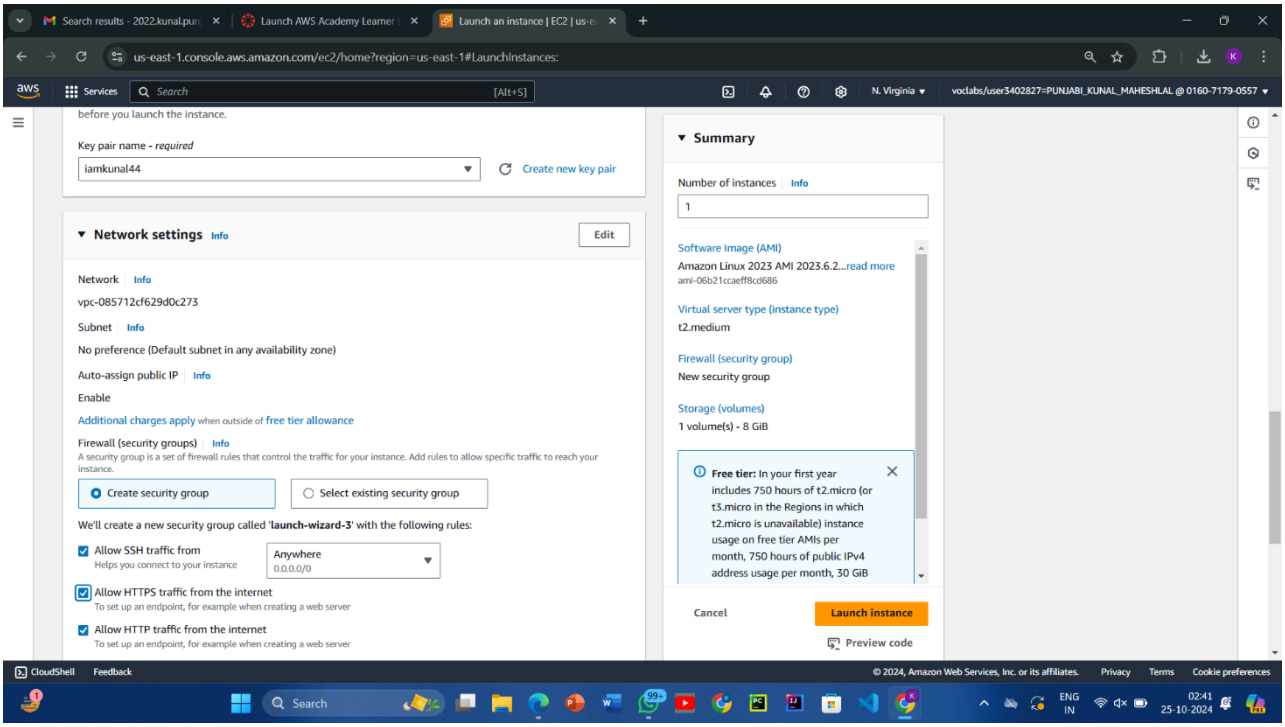


2) instance type as t2.medium.

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

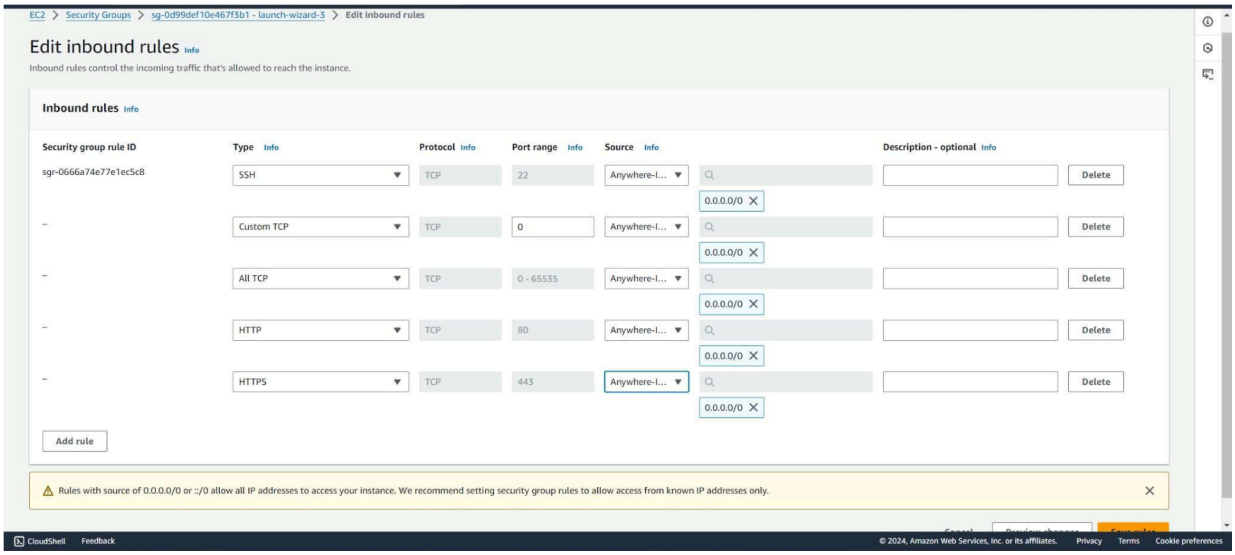


3) Create a key pair for our instance

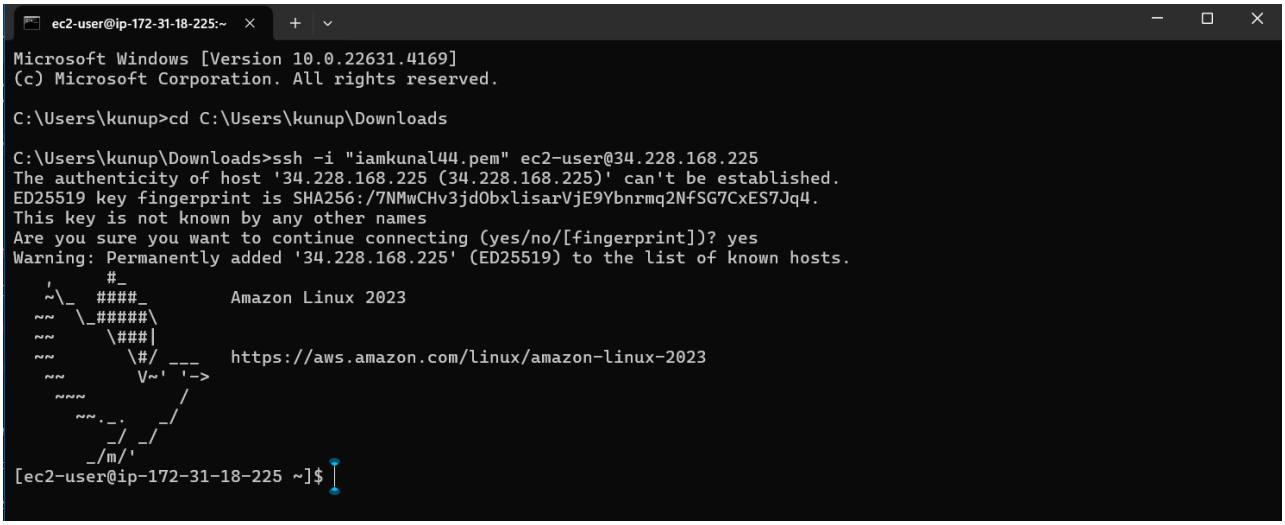


4) Allow the TCP, HTTP and HTTPS network access for all connections over the network.

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



5) SSH your connection

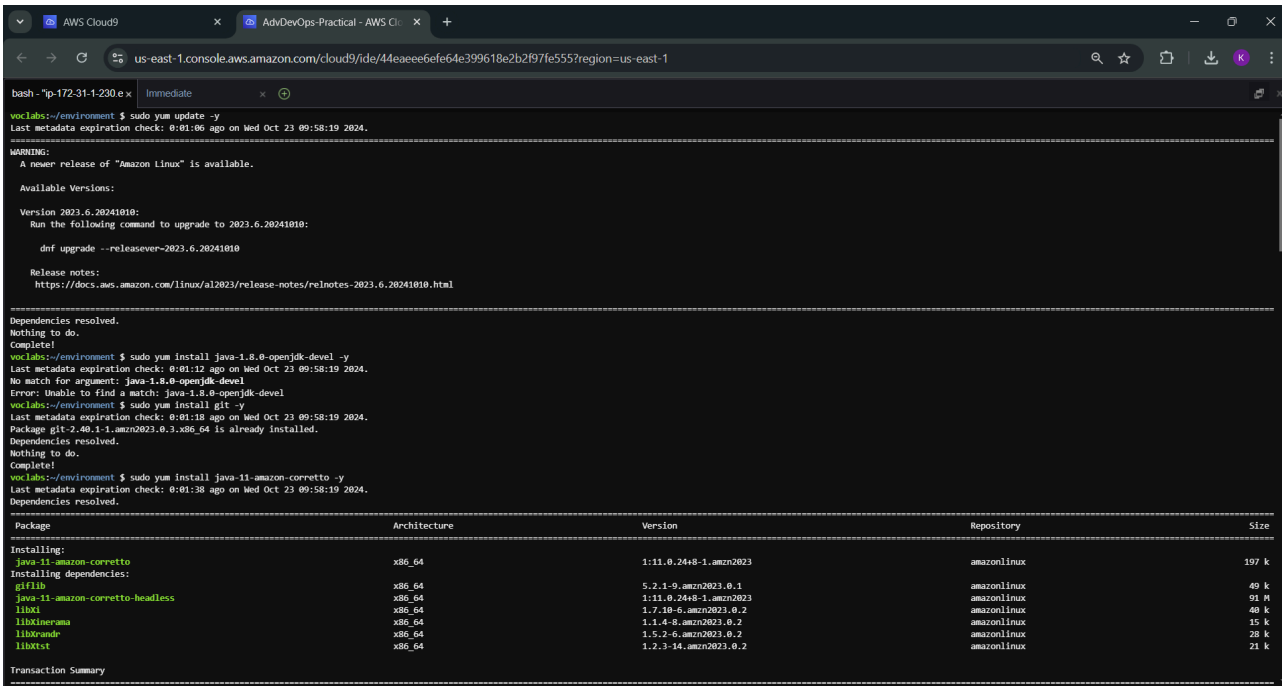


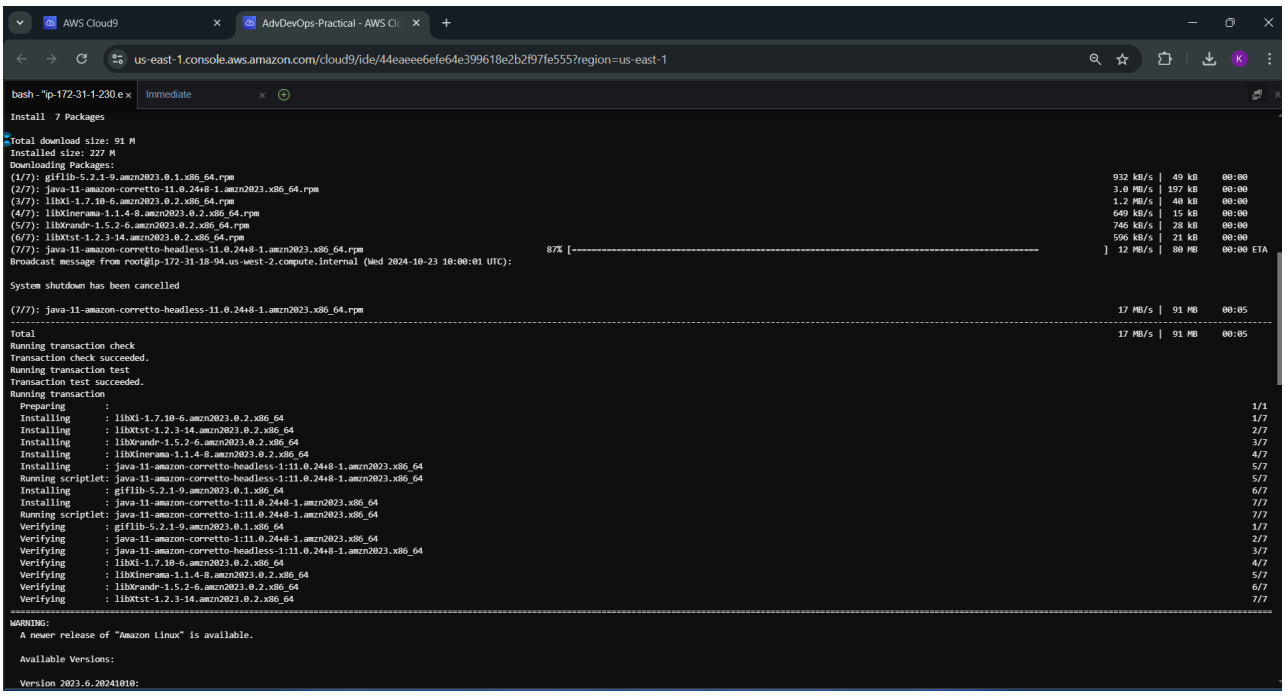
6) Execute the following commands :-

- **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.

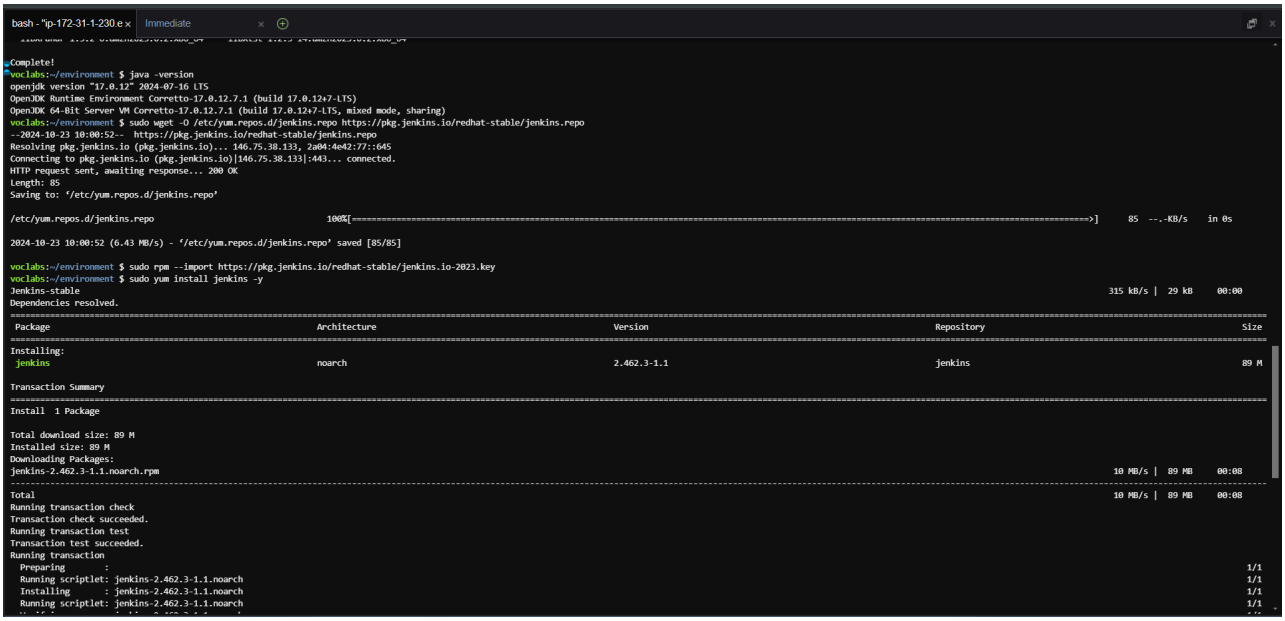
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

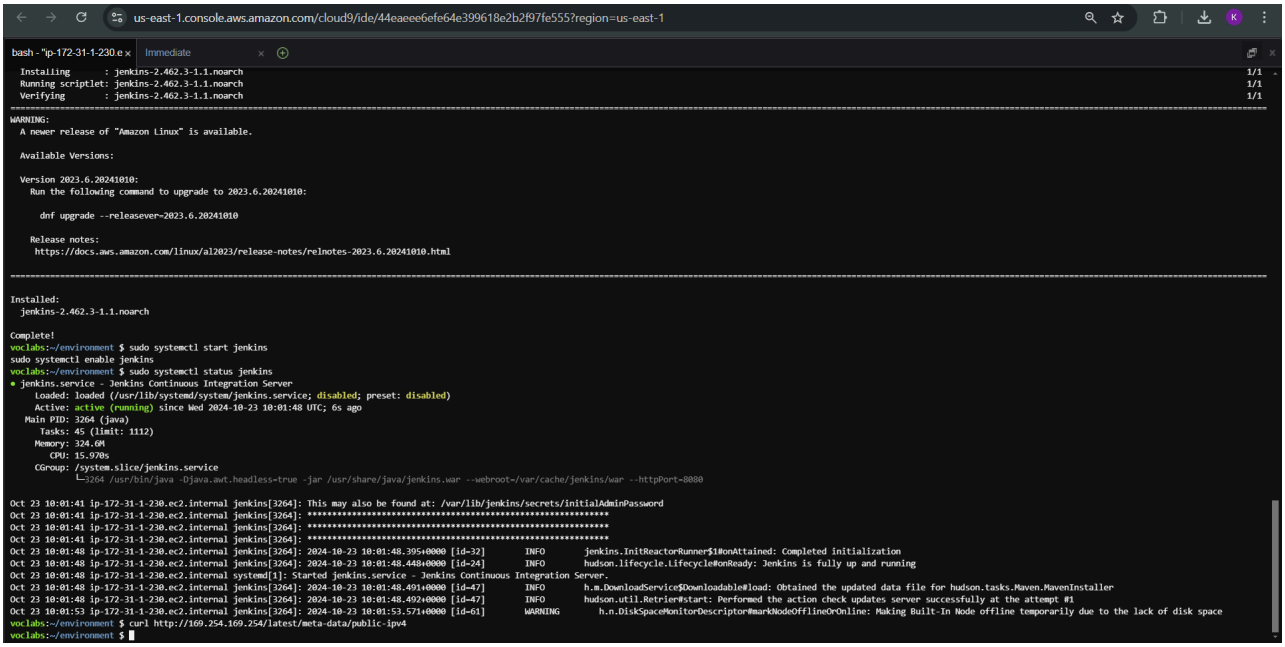
- **sudo yum upgrade**: Upgrades the system packages to their latest versions based on available repositories.
- **sudo yum install jenkins -y:** Installs Jenkins from the configured repository.
- **sudo systemctl enable jenkins**: Enables Jenkins to automatically start at system boot.
- **sudo systemctl start jenkins**: Starts the Jenkins service immediately.
- **sudo systemctl status jenkins**: Displays the current status of the Jenkins service.

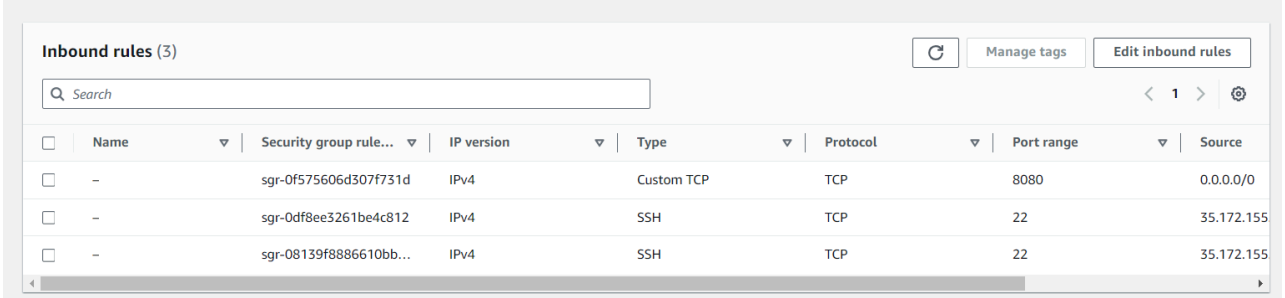




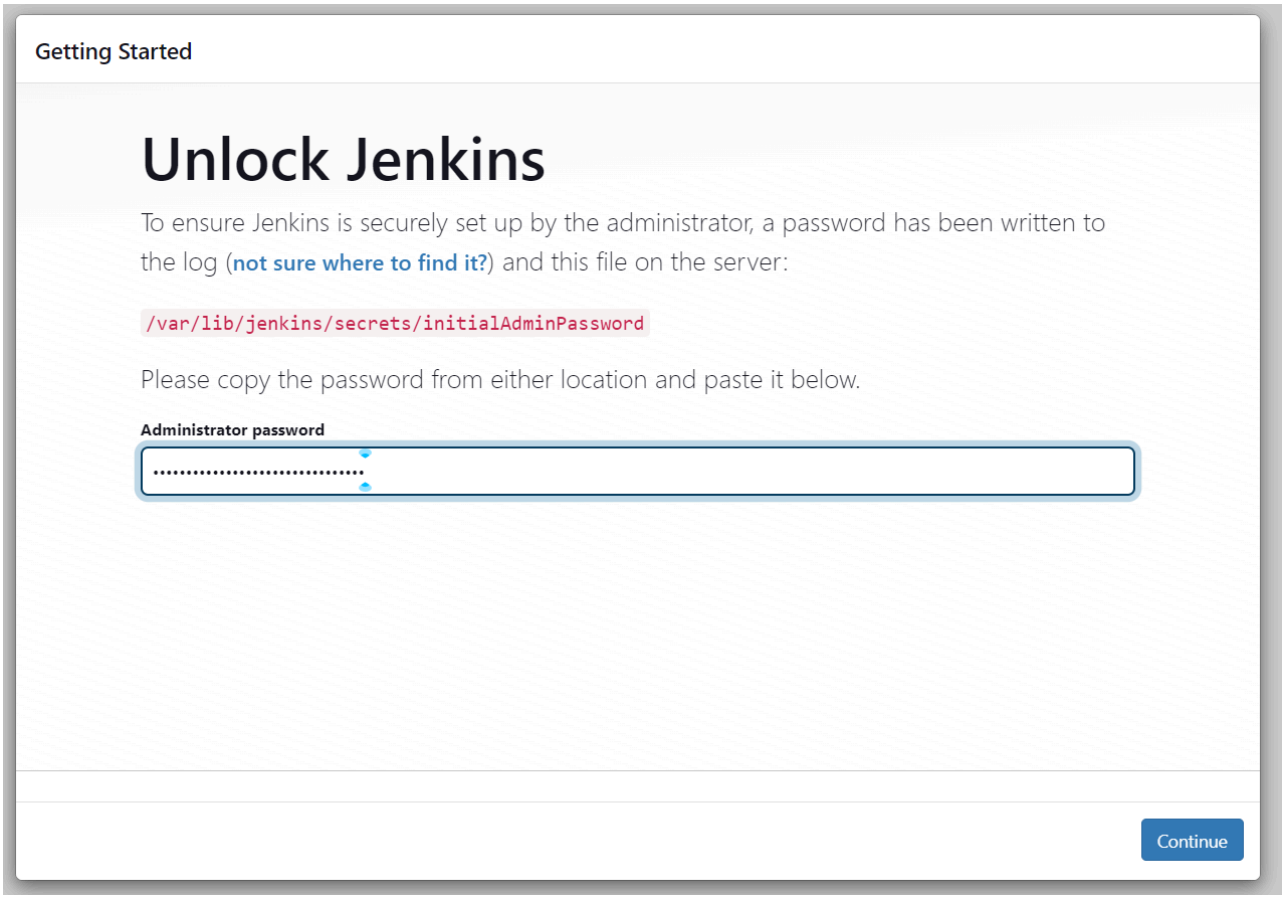
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



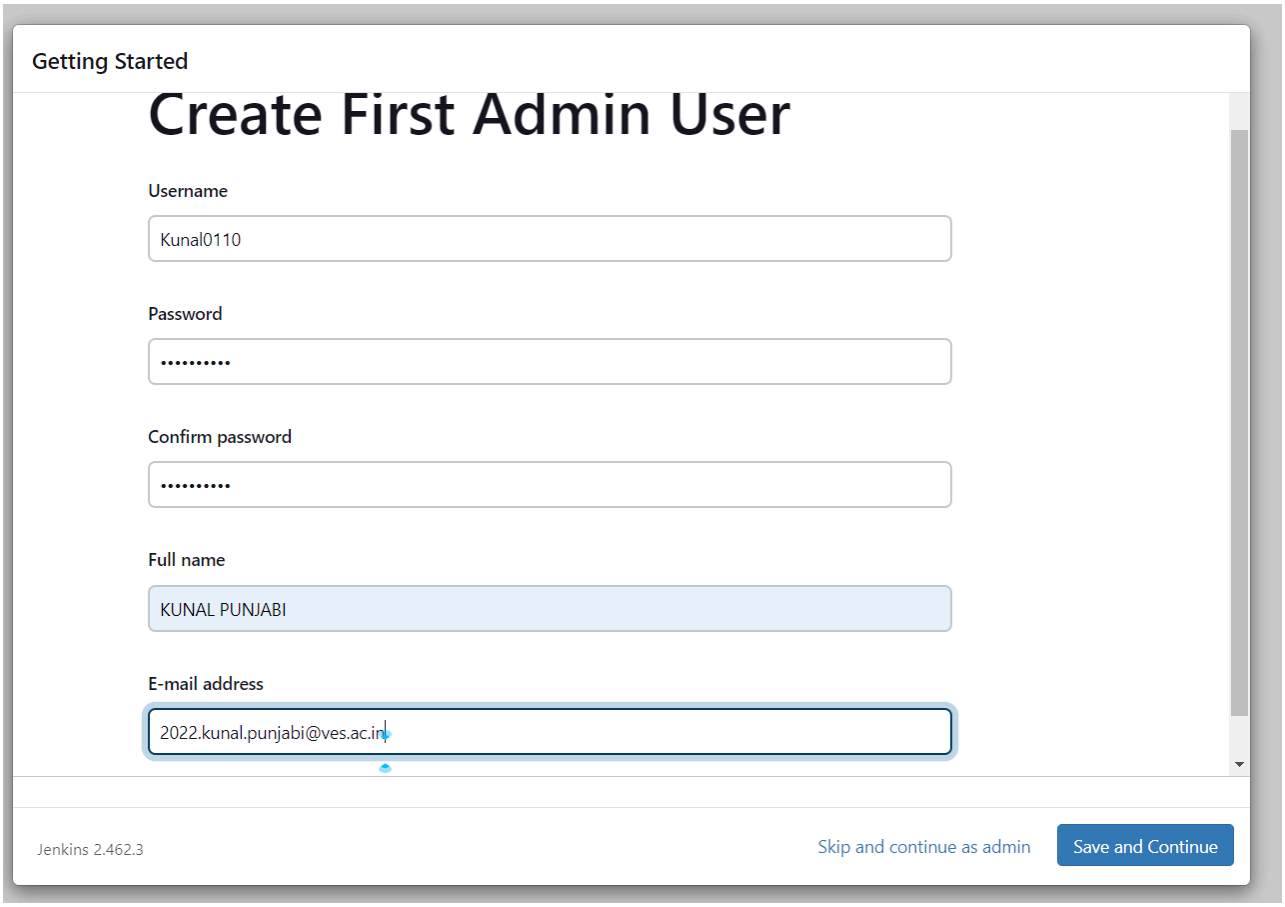




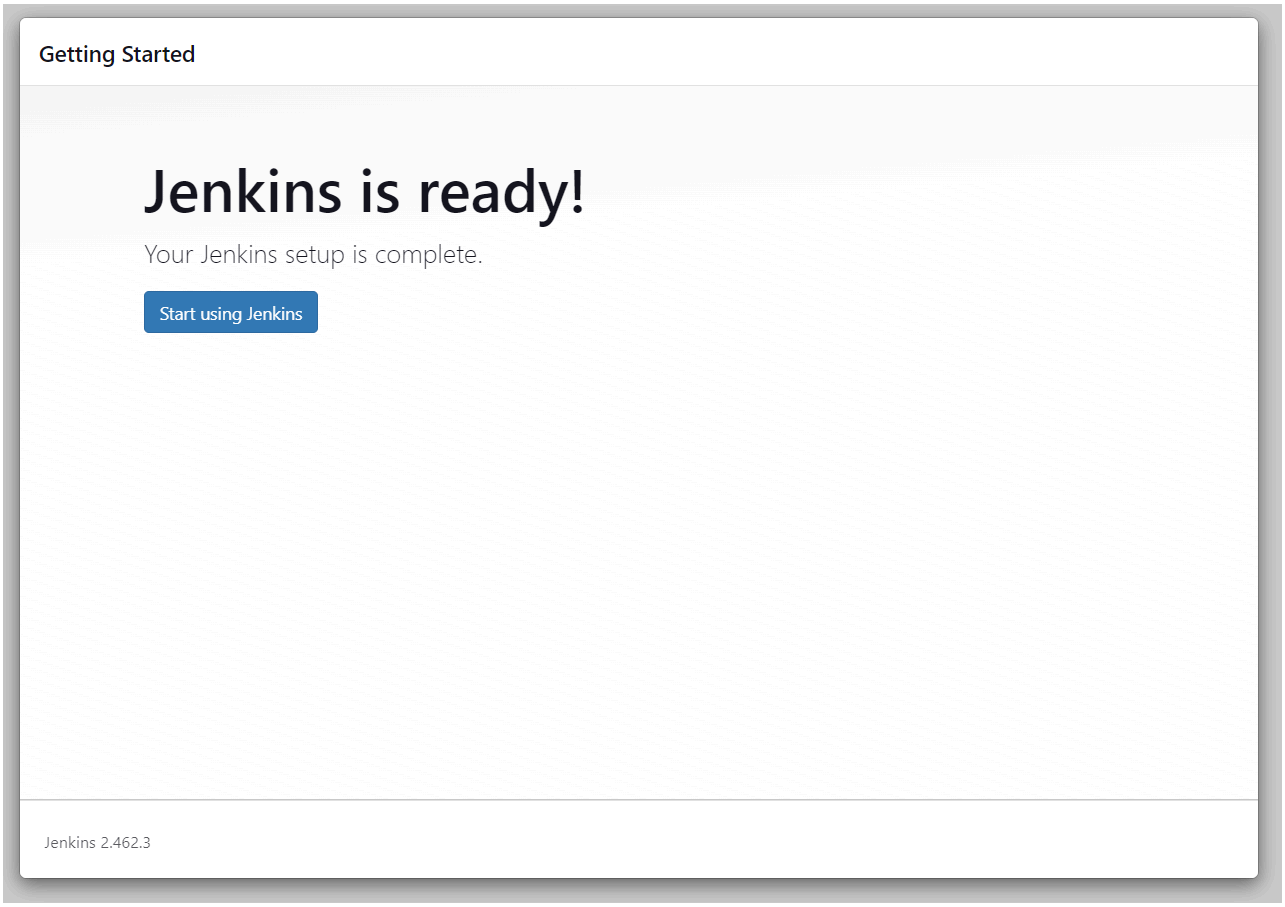
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



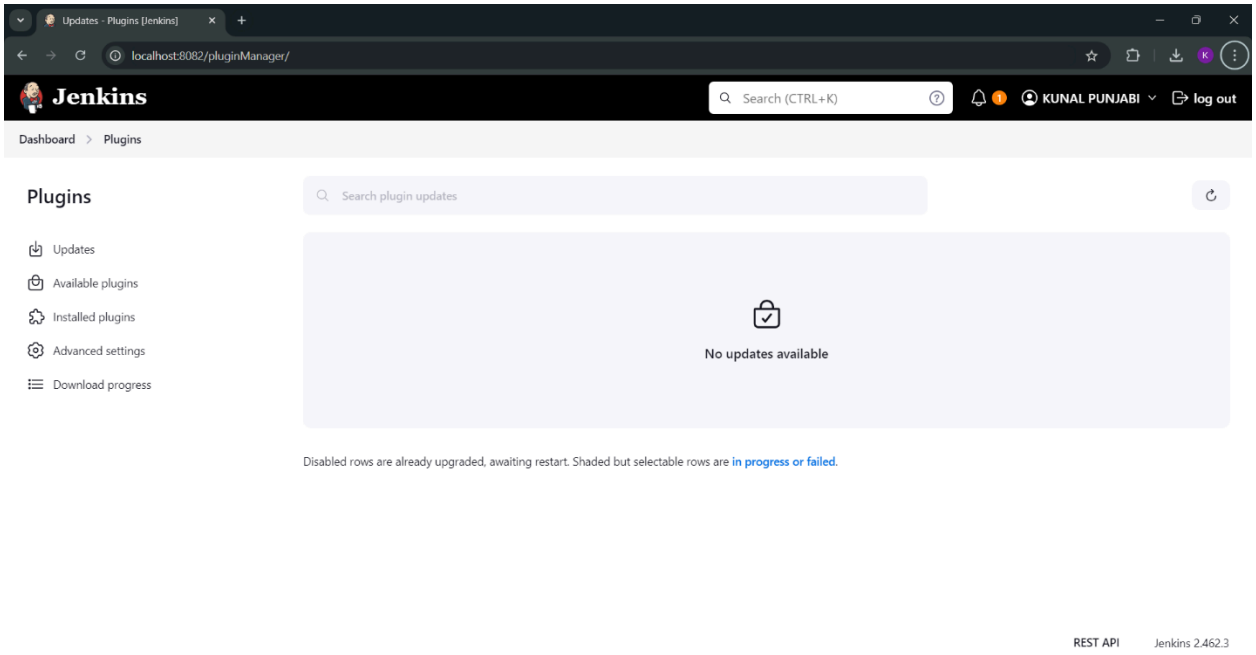
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

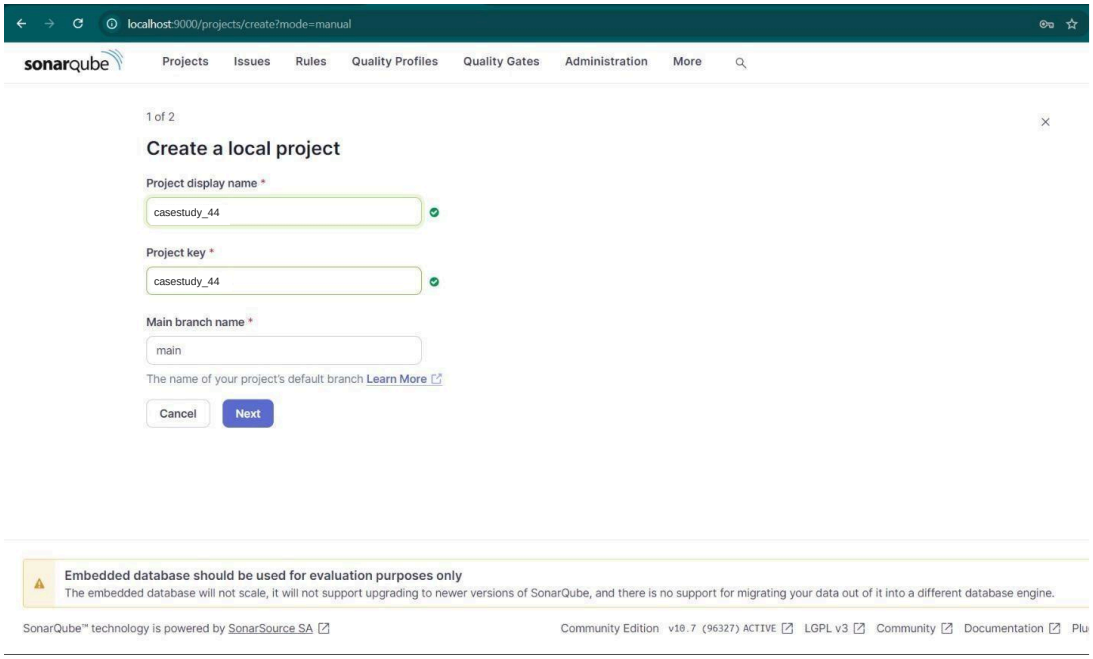
Thus, Jenkins is successfully configured on the EC2 Linux instance.

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

A] Sonarqube project:-

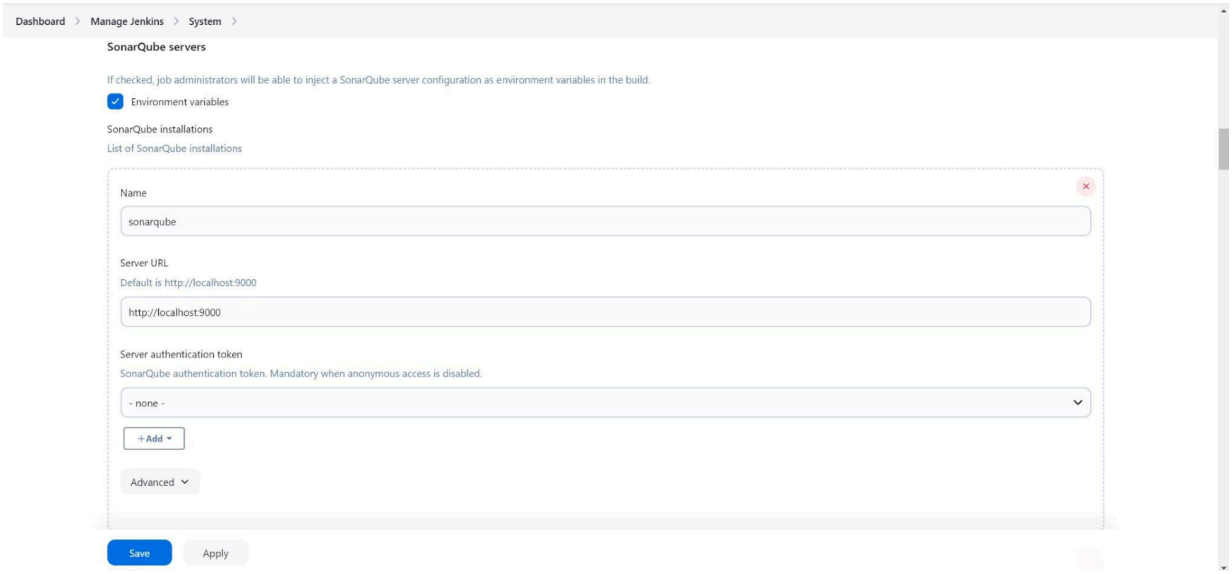
Python Project : https://github.com/piomin/sample-java-sonar

1) Create a sonarqube project named **casestudy_24.**

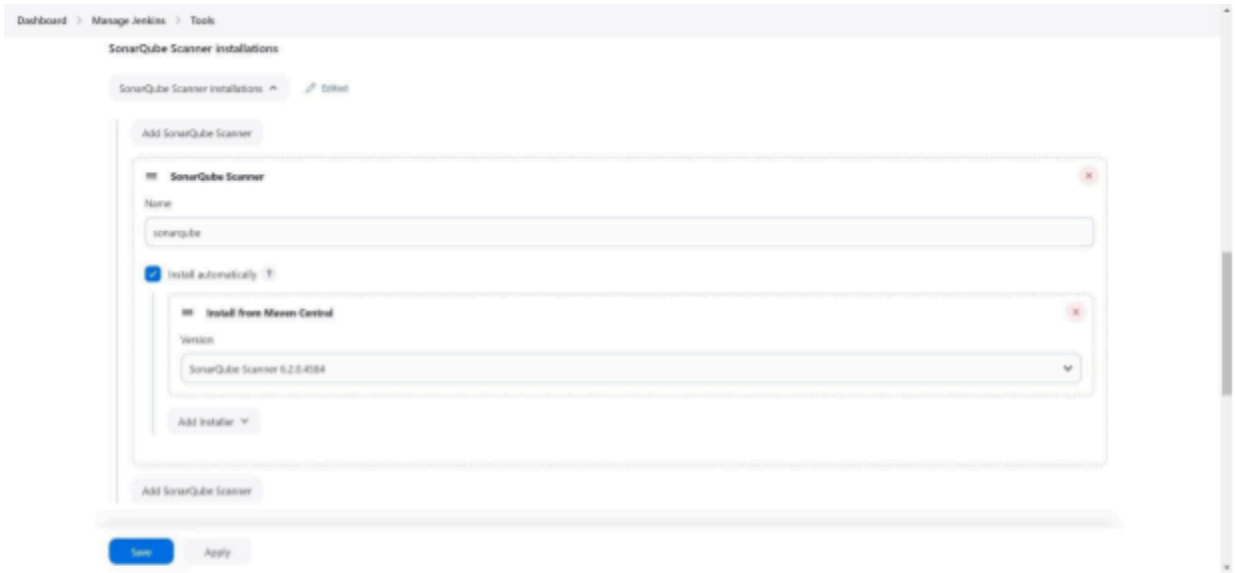


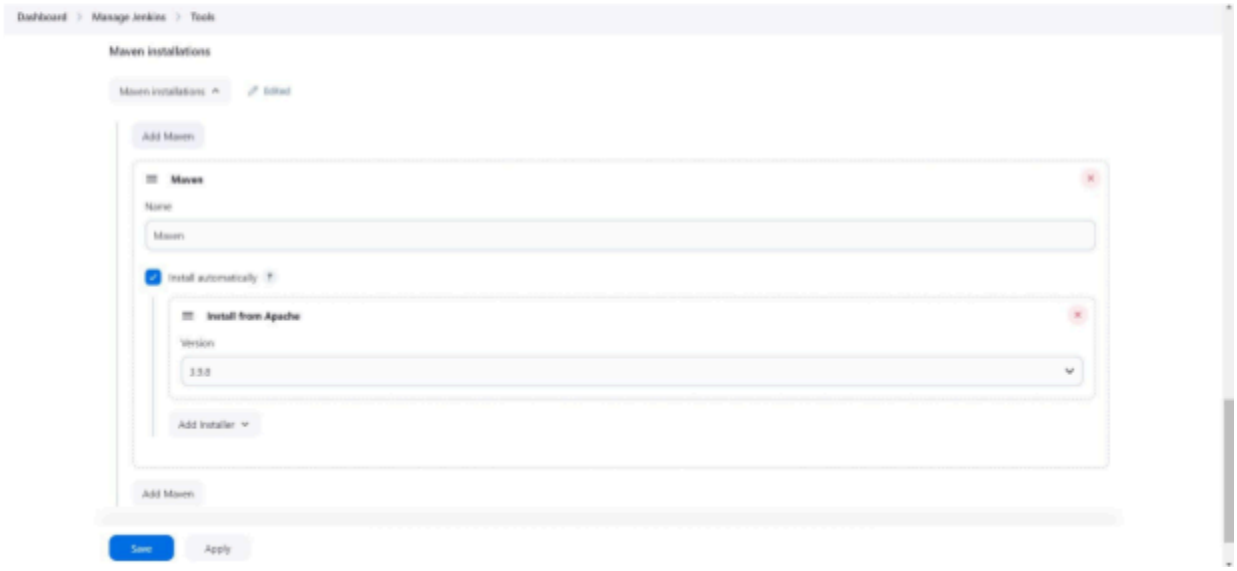
2) Sonarqube configurations in jenkins:-

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



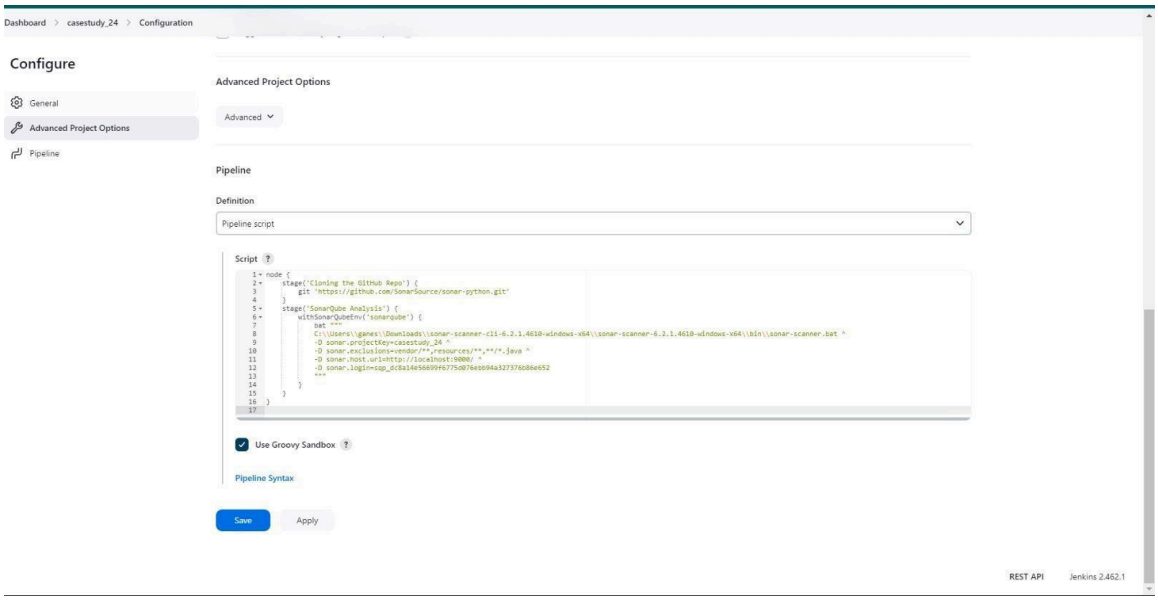
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44





Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

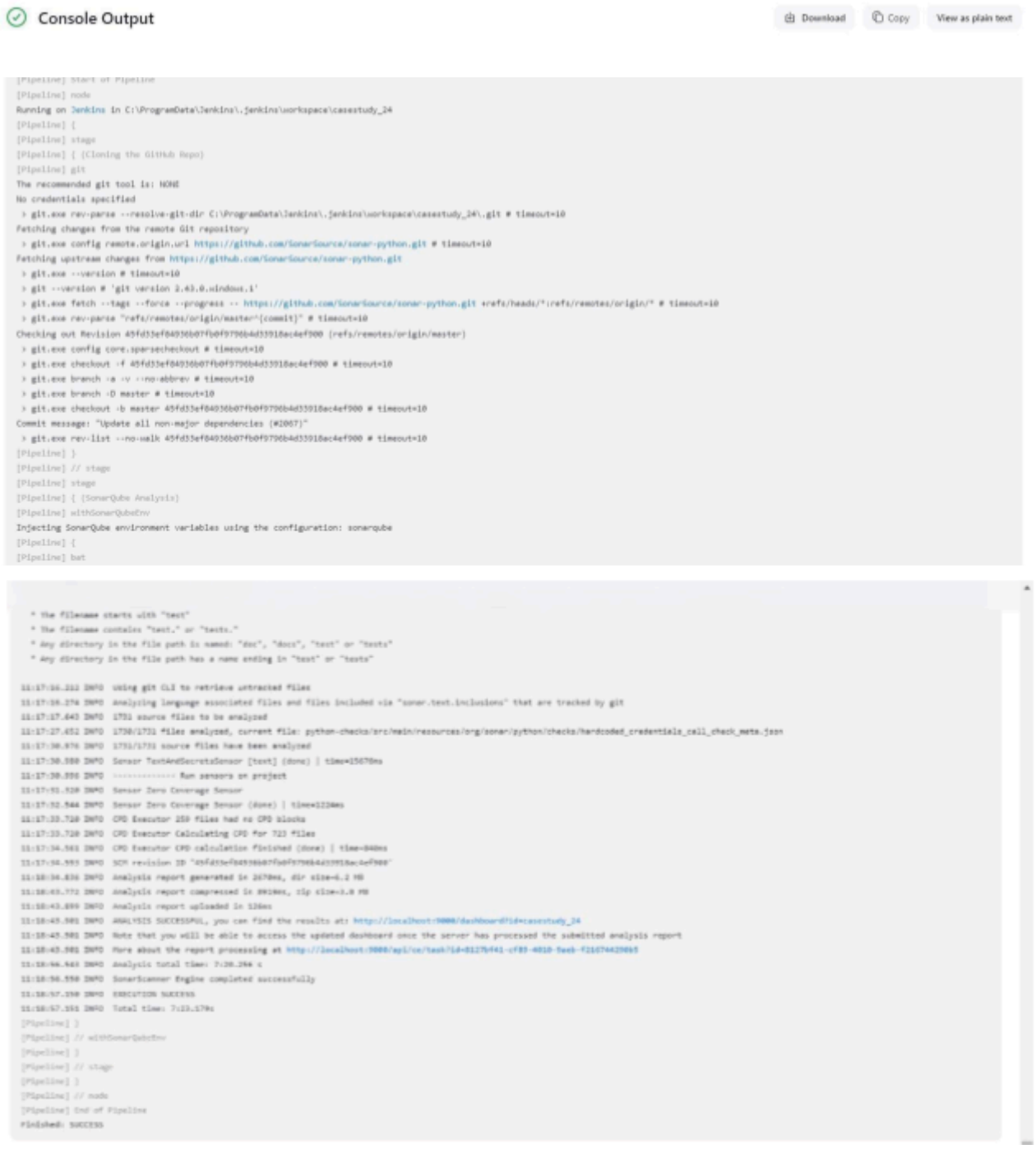
3) Jenkins Pipeline:-



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/"
}
}
}

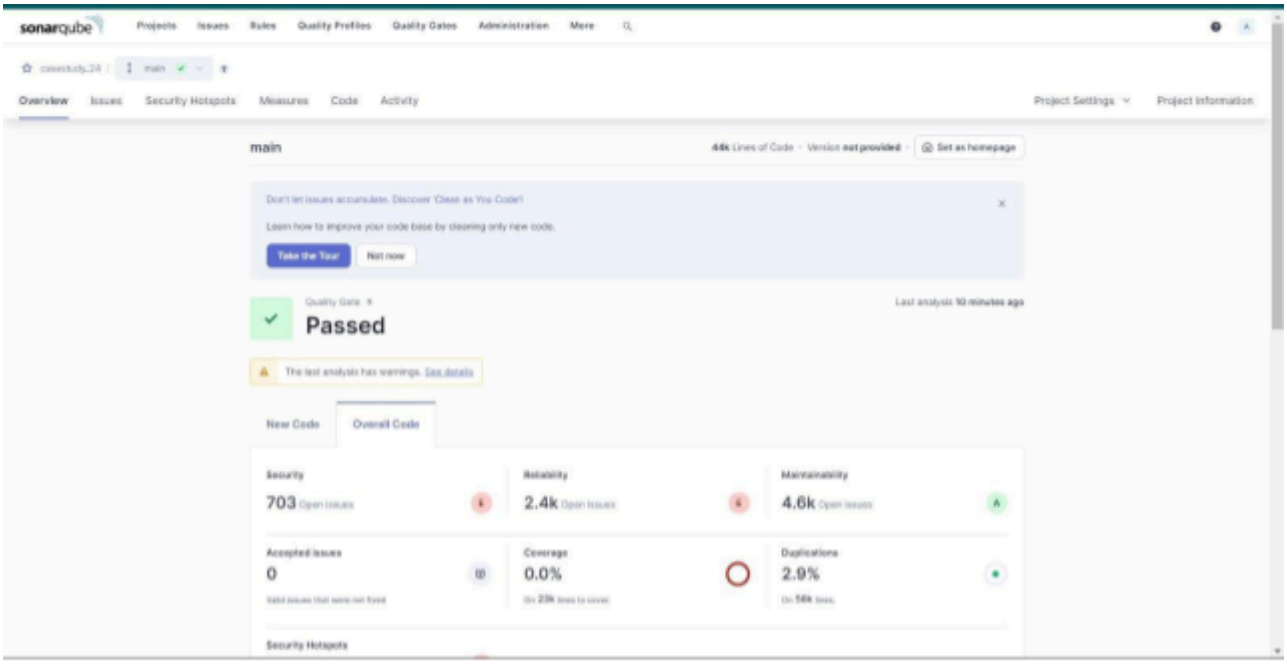
4) Jenkins Output :

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



Sonarqube Analysis:-

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

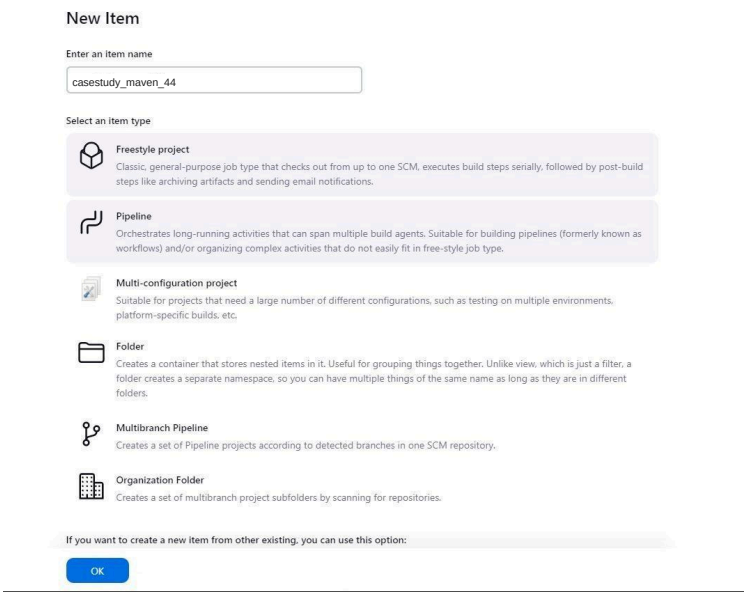


Thus, the Python project was successfully analyzed with SonarQube.

B]For Maven Project:-

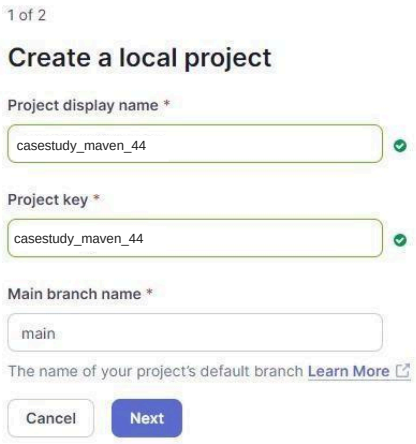
1) Make a Jenkins Pipeline.

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



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

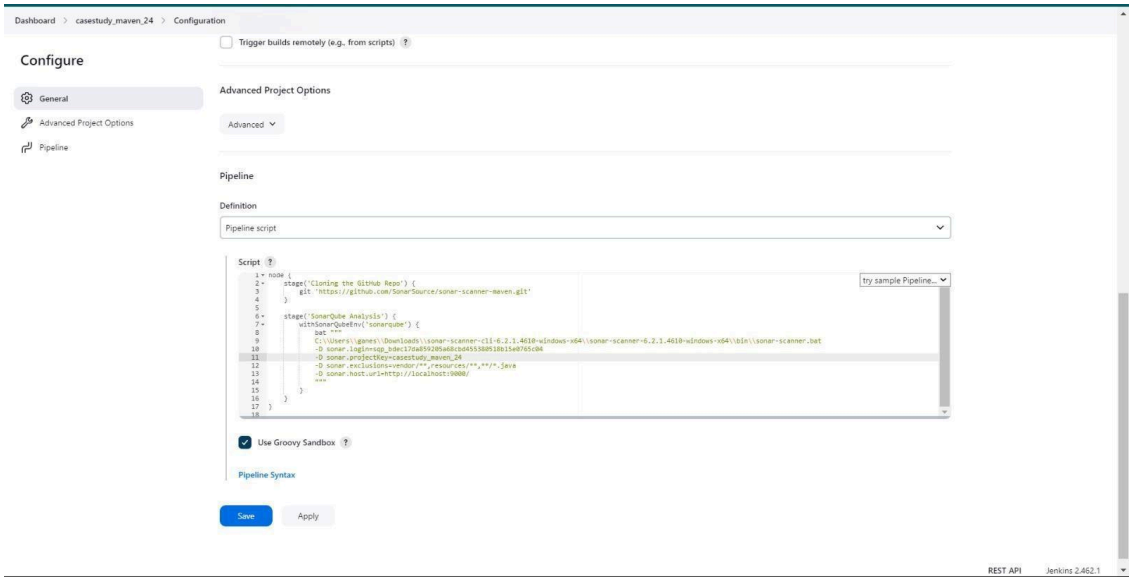
2) Create a sonarqube project named **casestudy24maven.**



Pipeline
Script:node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/SonarSource/sonar-scanner-maven.git'
    }

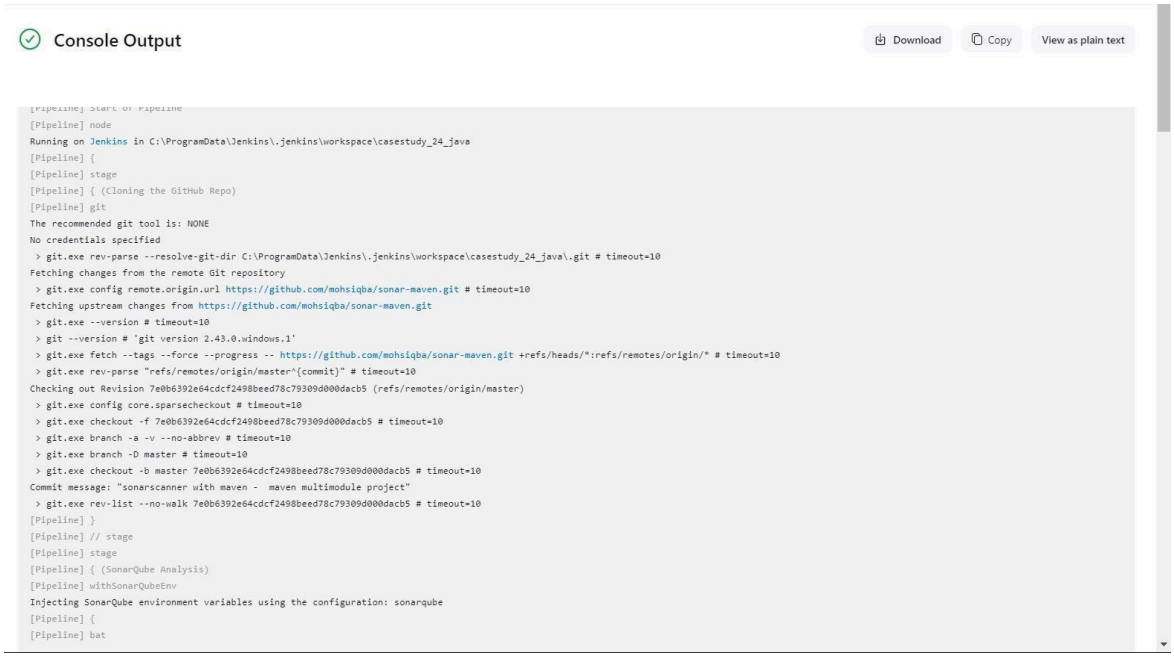
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

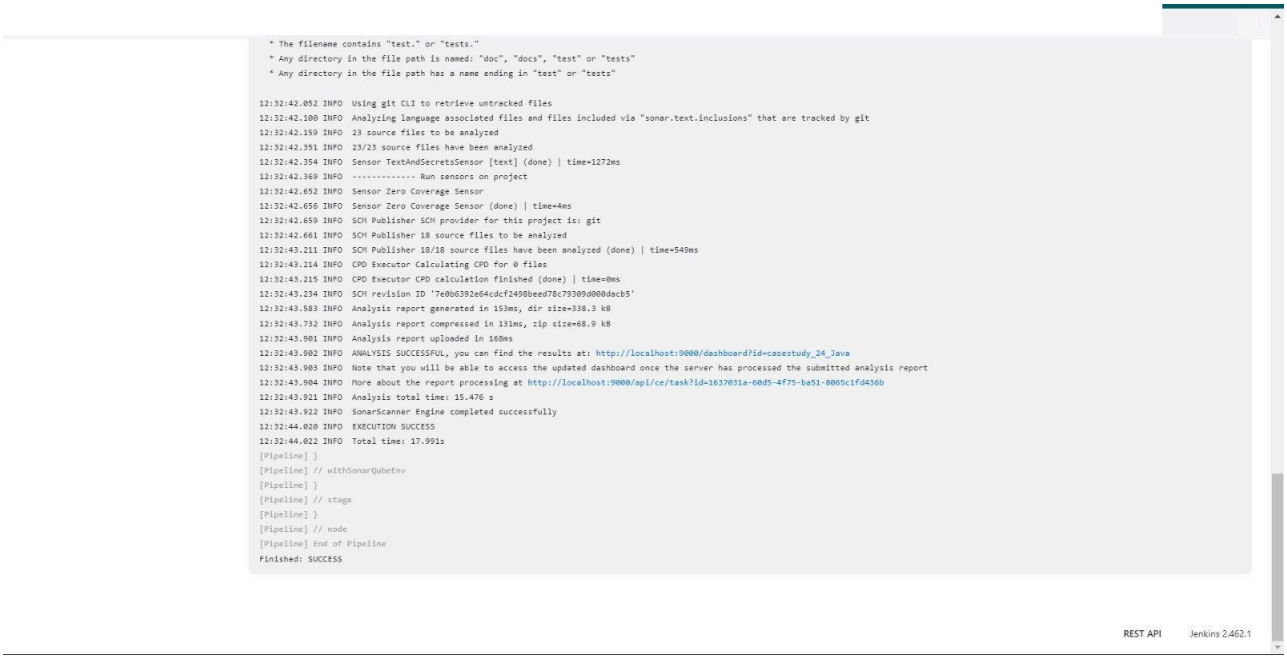
```
        stage('SonarQube Analysis') {
            withSonarQubeEnv('sonarqube') {
                bat
    "C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4
    477-windows-x64\\bin\\sonar-scanner.bat " +
                    "-D sonar.login=admin " +
                    "-D sonar.password=HareKrishna#108 " +
                    "-D sonar.projectKey=casestudy24maven " +
                    "-D sonar.exclusions=vendor/**,resources/**,**/*.java " +
                    "-D sonar.host.url=http://localhost:9000/"
            }
        }
    }
```



3) Build and check its console output:-

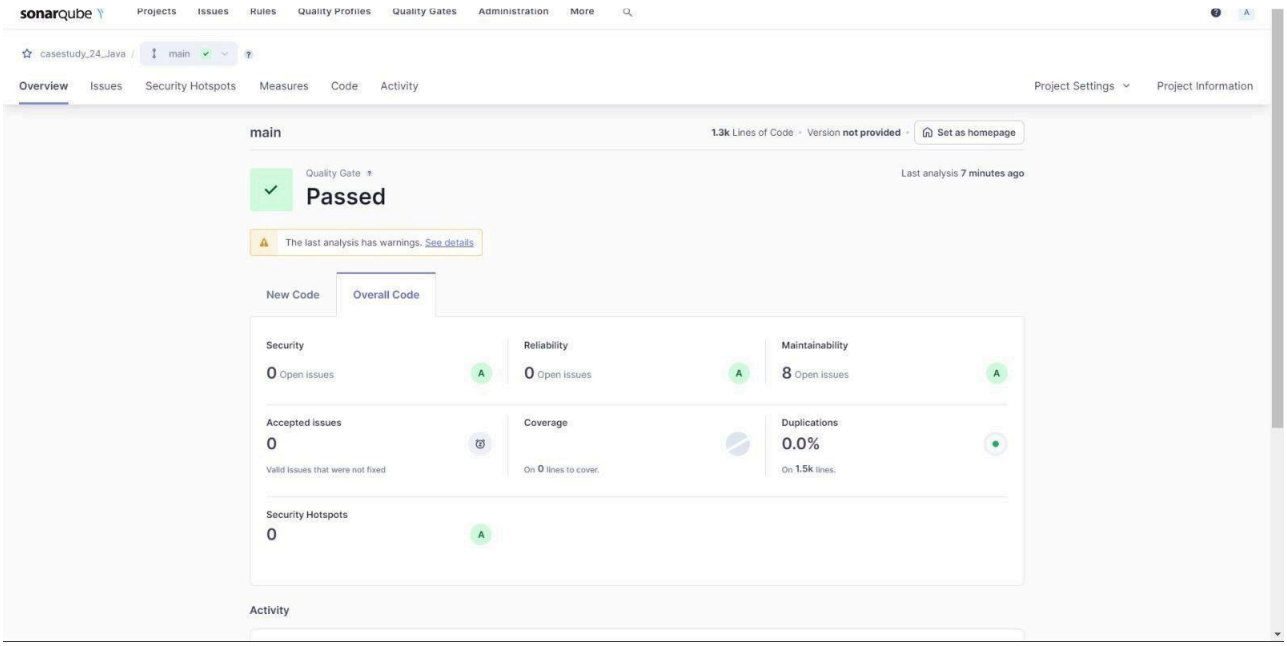
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44





SonarQube analysis

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



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.

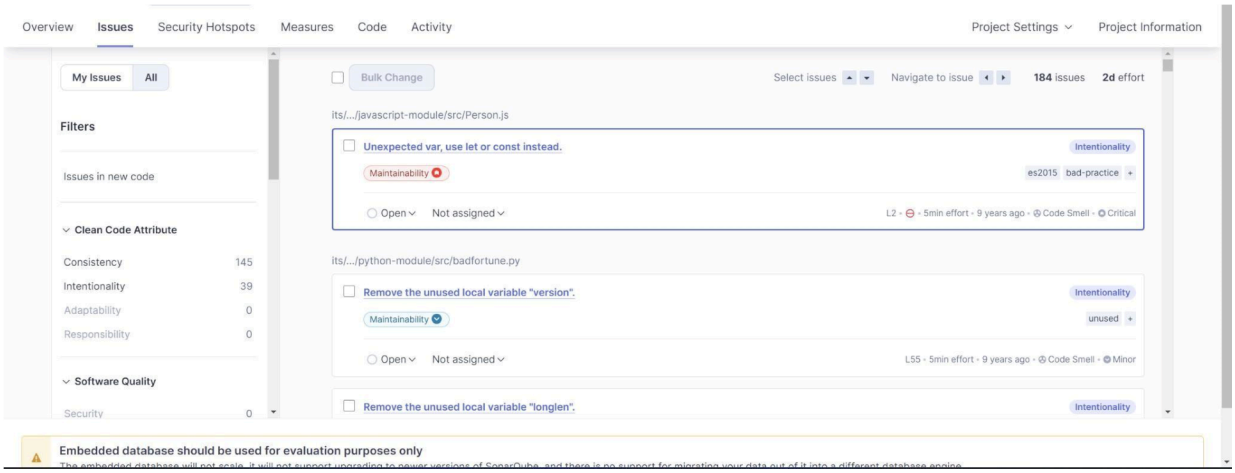
Adaptability: Indicates how flexible the code is to accommodate changes or new features without requiring major rewrites.

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.

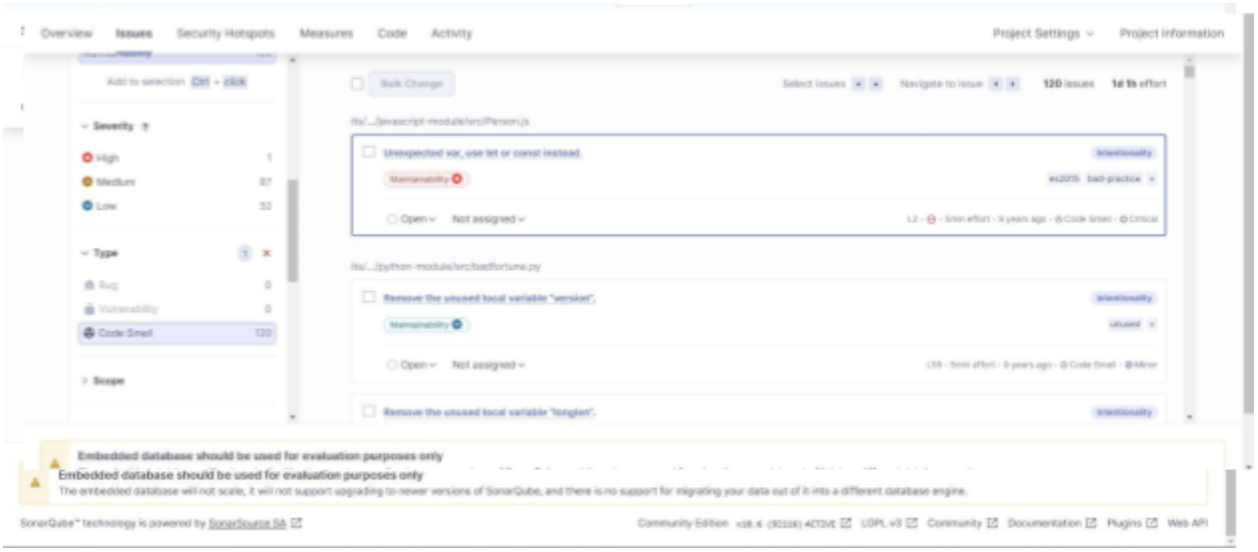
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



} 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.

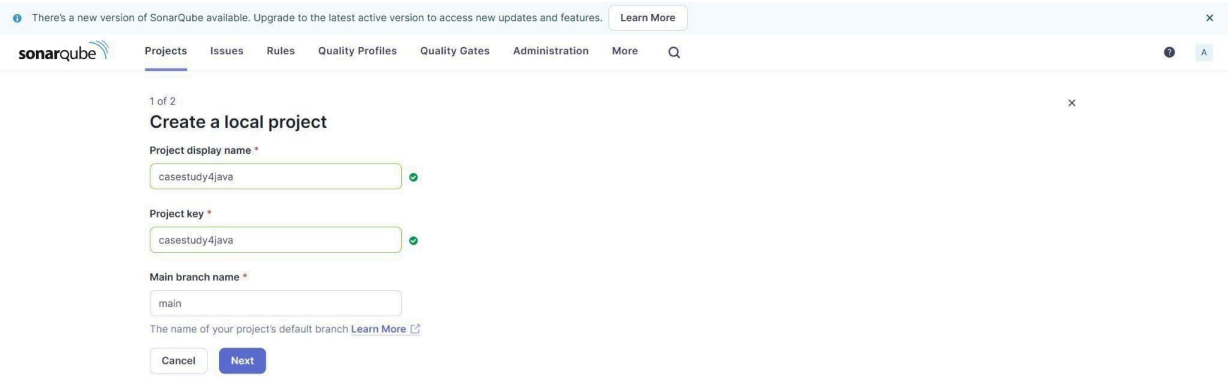
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

Successfully analyzed the Java Maven project using SonarQube.

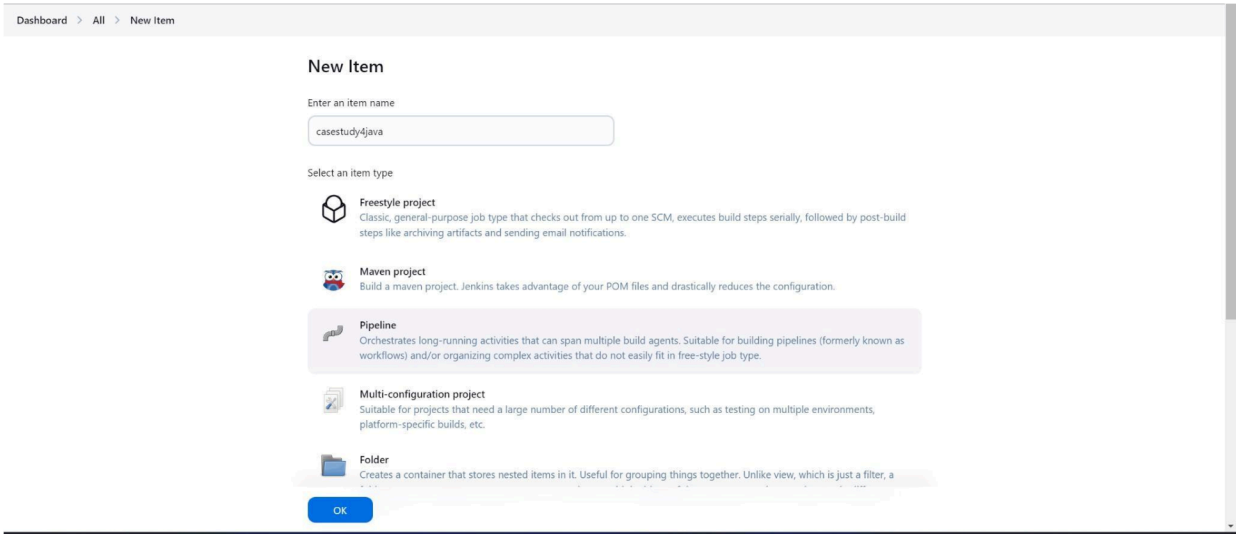
3] Java Project:-

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

1)SonarQube Project:



2)Jenkins Pipeline:



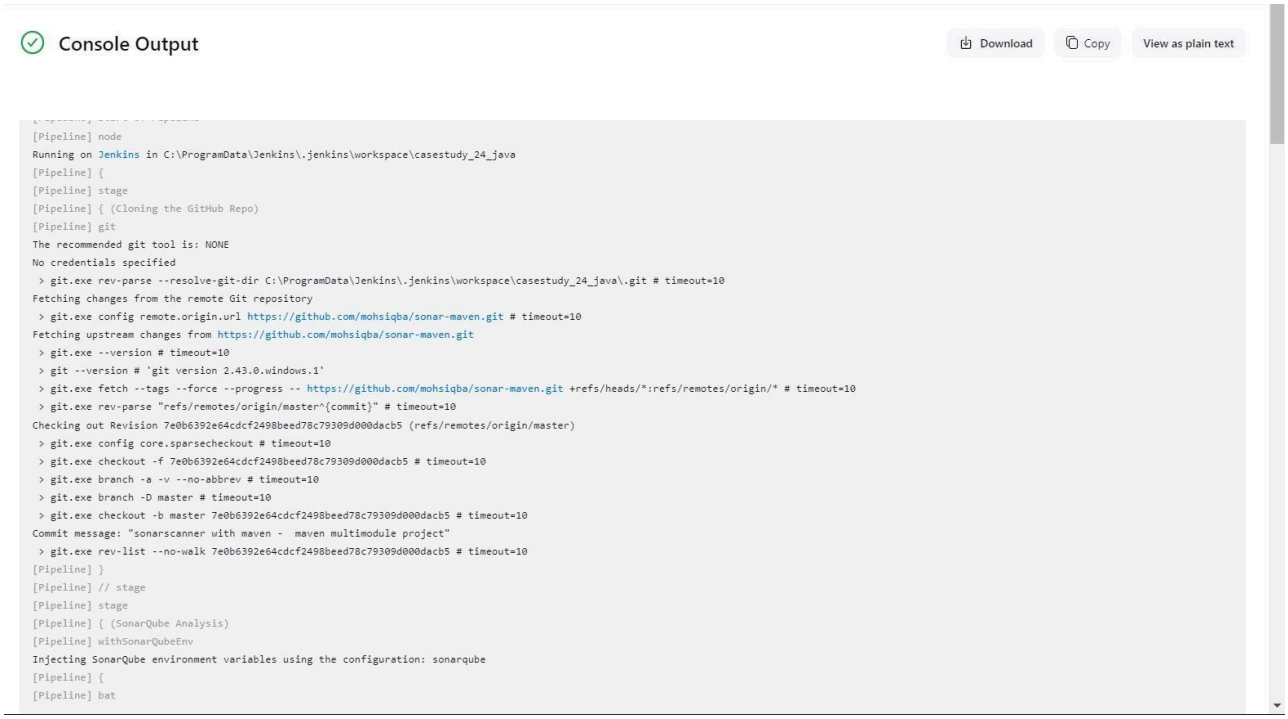
Pipeline Script:-

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

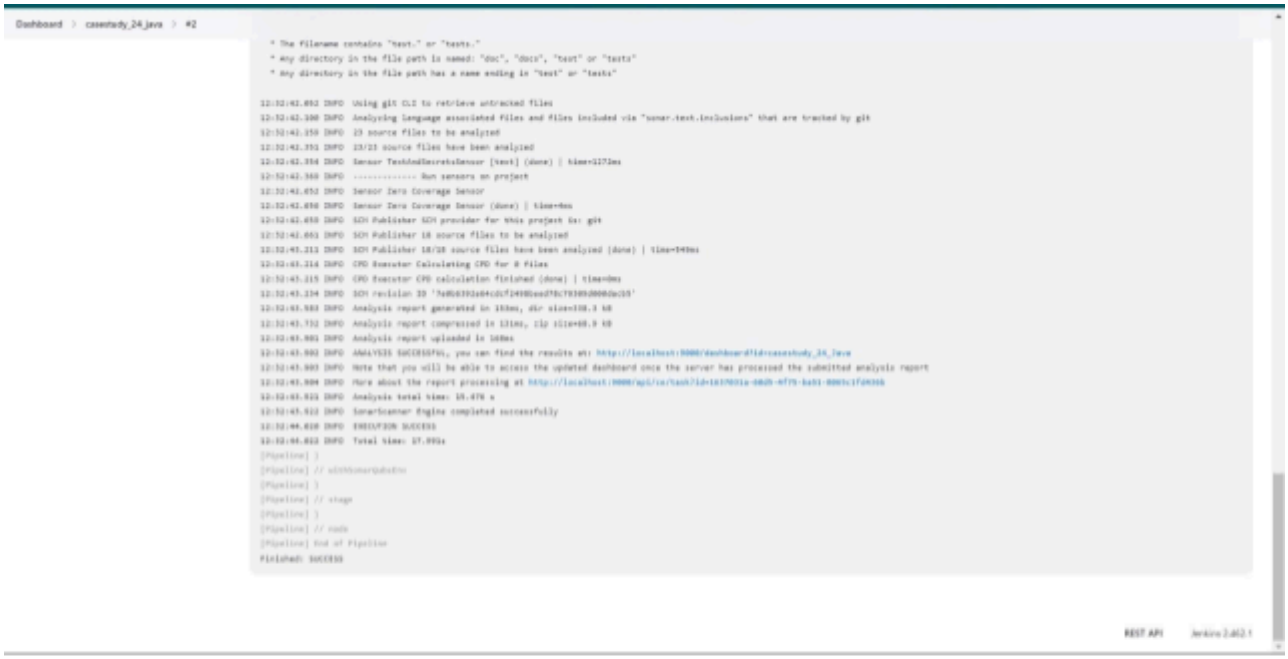
Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

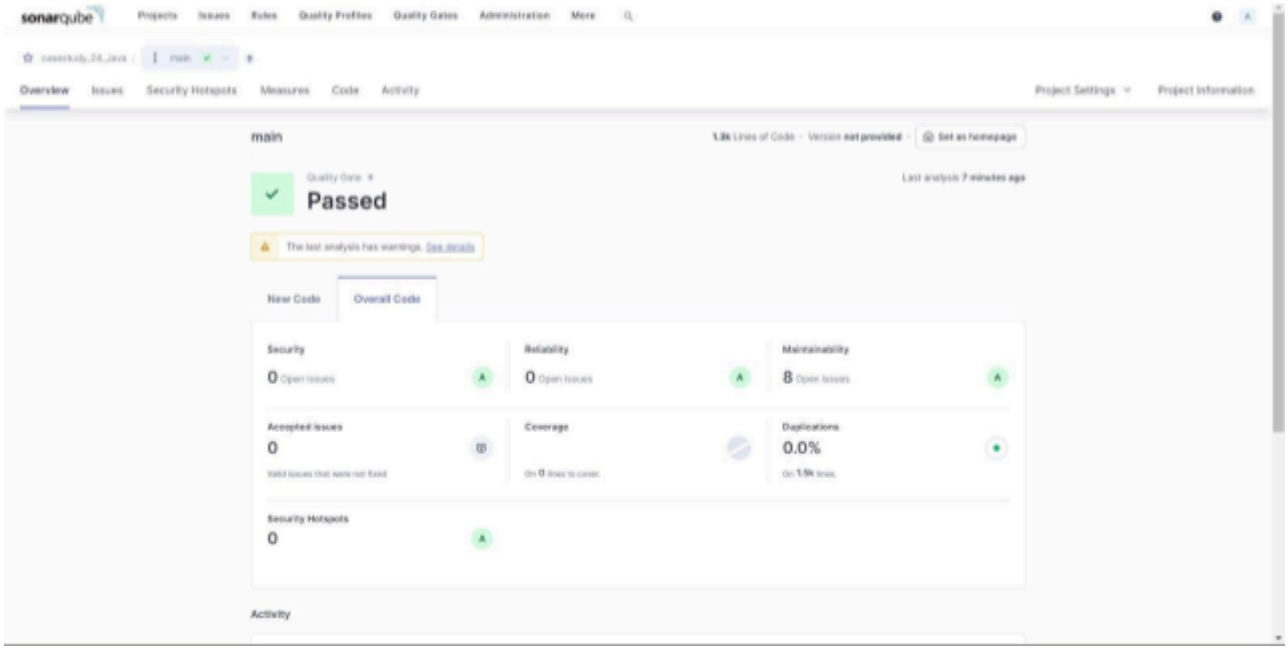
```
"C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4
477-windows-x64\\bin\\sonar-scanner.bat " +
            "-D sonar.login=admin " +
            "-D sonar.password=HareKrishna#108 " +
            "-D sonar.projectKey=casestudy24java " +
            "-D sonar.exclusions=vendor/**,resources/**,**/*.java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

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

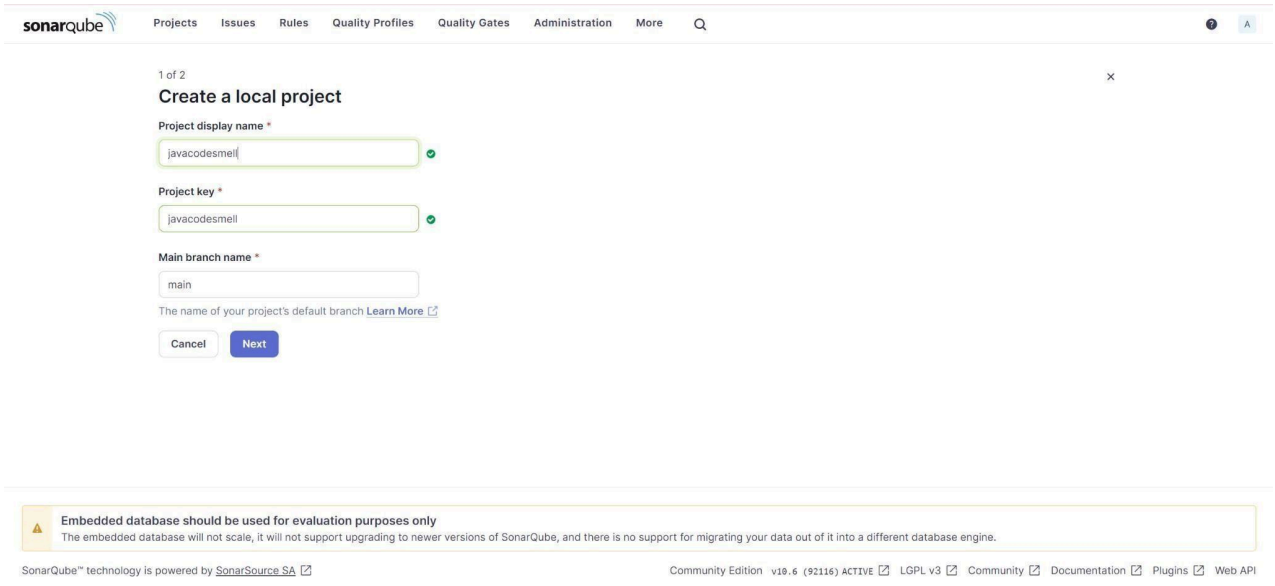




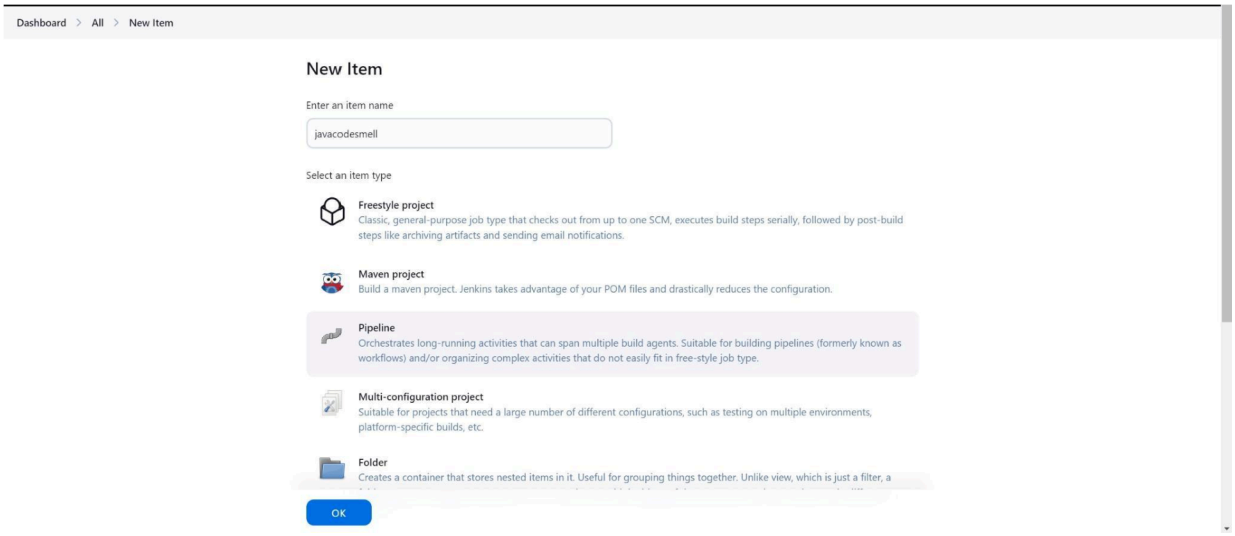
Successfully analyzed the Java project using SonarQube. 4] Java Project:-
Project Link: https://github.com/nerdschoolbergen/code-smells.git

1)SonarQube Project:

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



2)Jenkins Pipeline:

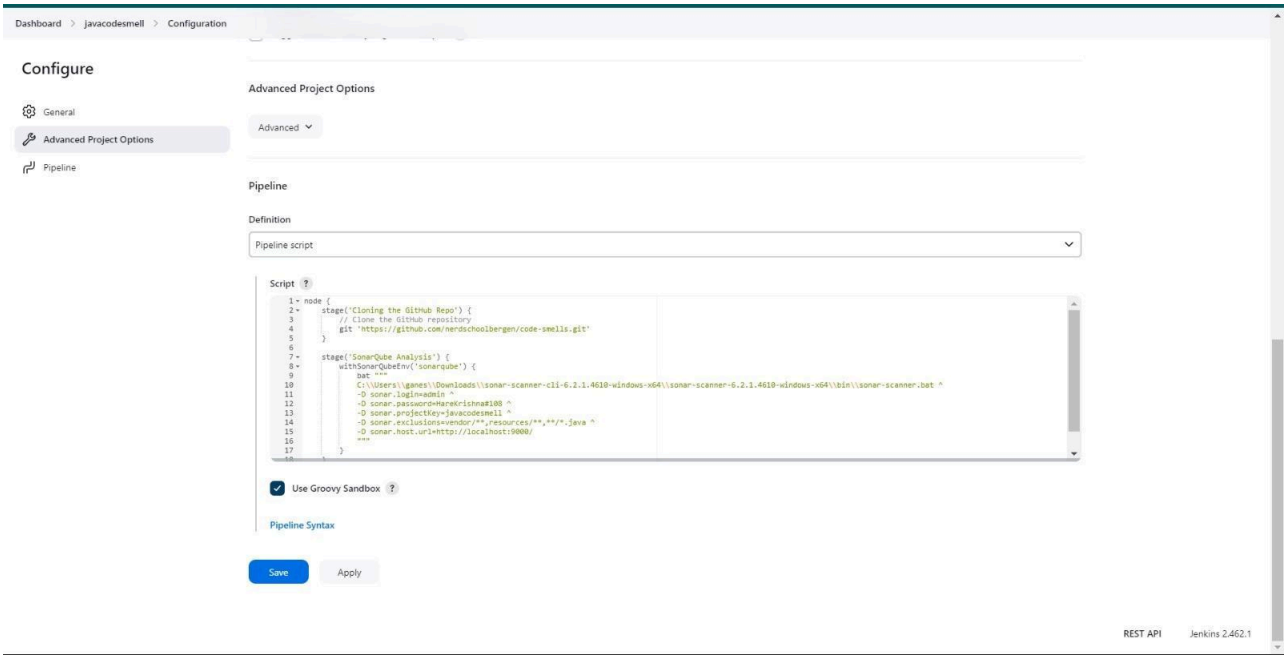


Pipeline Script:-

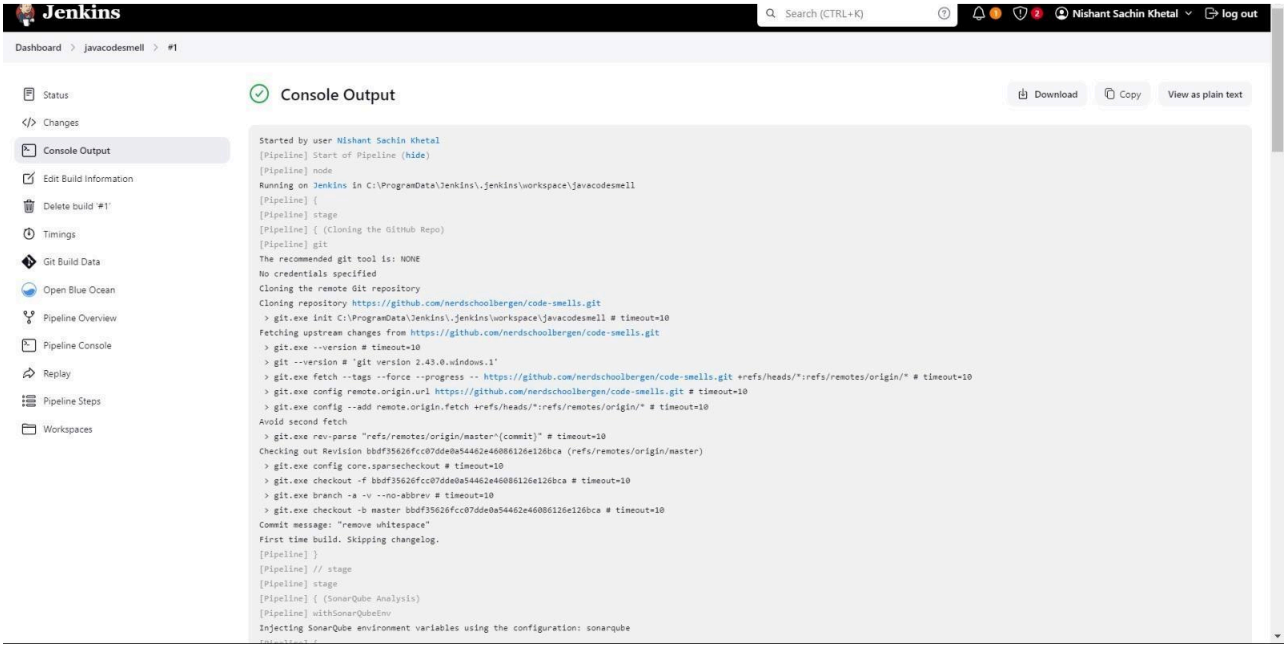
```
node {
  stage('Cloning the GitHub Repo') { git
     'https://github.com/nerdschoolbergen/code-smells.git'
    }
    stage('SonarQube Analysis') {
      withSonarQubeEnv('sonarqube') {
         bat
  "C:\\Users\\nish\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat " +
                "-D sonar.login=admin " + // Ensure there is a space before the next option
                "-D sonar.password=HareKrishna#108 " +
```

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

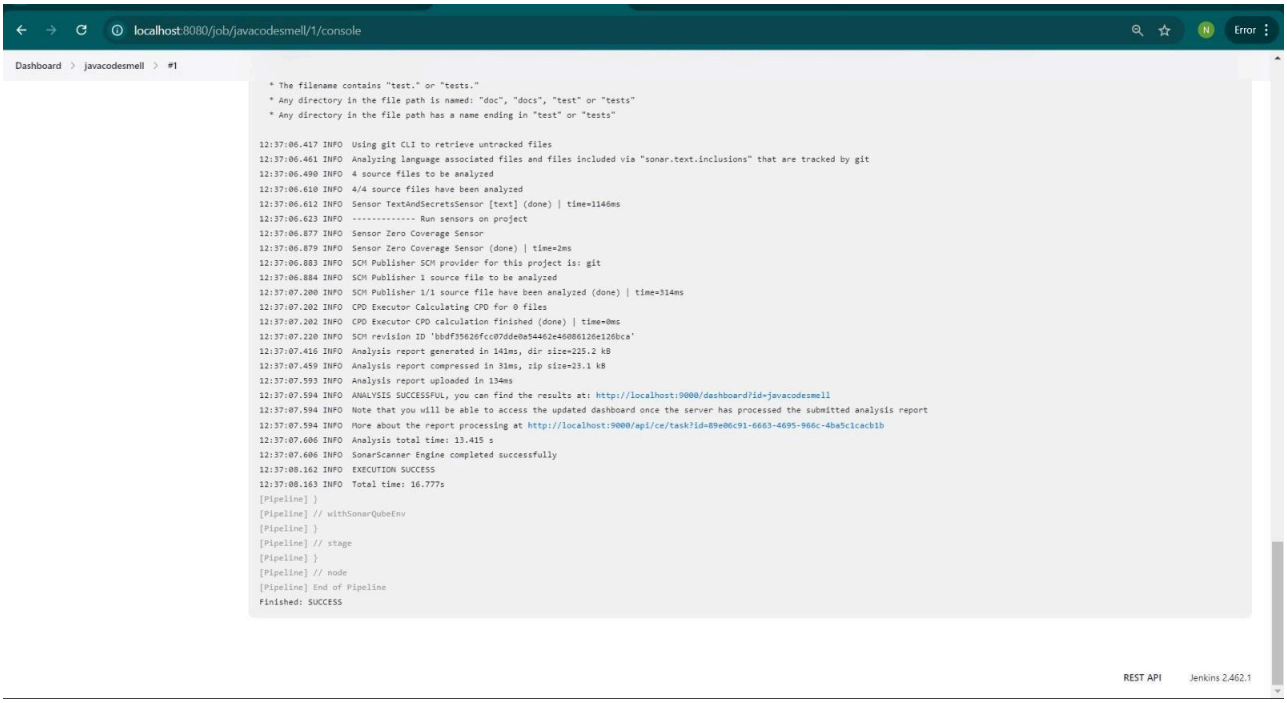
```
                "-D sonar.projectKey=javacodesmell " +
                "-D sonar.exclusions=vendor/**,resources/**,**/*.java " +
                "-D sonar.host.url=http://localhost:9000/"
        }
    }
}
```



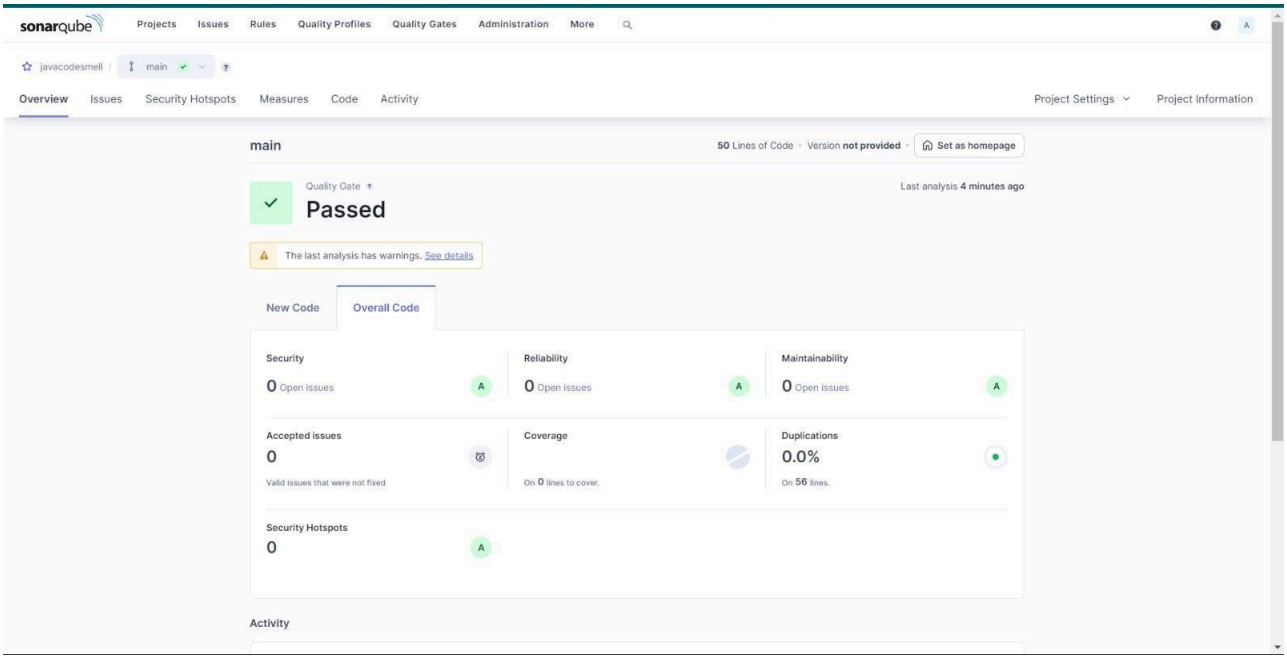
3) Console Output:



Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44



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

Name : Kunal Mahesh Punjabi
D15A
ROLL NO – 44

helps maintain high code quality and ensures that code adheres to security standards before being pushed to production.