

## 1. Introduction :

**Case Study Overview:** This case study explores a DevOps-driven approach to software development and deployment. We focus on integrating Jenkins and SonarQube to implement a Continuous Integration (CI) pipeline with static code analysis. The primary objective is to ensure that code remains reliable and maintainable through continuous quality checks during integration. This study specifically applies to my third-year project, EventEase.

### Key Features:

- **Jenkins:** Automates CI by triggering builds and static code analysis whenever code changes are made.
- **SonarQube:** Provides continuous feedback on code quality, reliability, and security.
- **AWS Cloud9:** Serves as the development environment for seamless integration.

## 2. Third-Year Project Integration: EventEase

EventEase is a comprehensive event management system that helps users plan and manage events, handle attendees, and schedule activities. The platform includes real-time notifications, ticketing, and a history of past events. By integrating Jenkins and SonarQube into EventEase, the following benefits are achieved:

- **Automated Testing:** Jenkins ensures that every code commit triggers tests, keeping the platform stable and bug-free.
- **Code Quality Monitoring:** SonarQube monitors code quality, enforcing clean, maintainable code.
- **Security Scanning:** SonarQube catches vulnerabilities before deployment, protecting sensitive event data.

## 3. Demonstration :

**Problem Statement:** Jenkins on an EC2 instance

- Set up an AWS EC2 instance with Linux (t2.micro) and configure Jenkins to automate the CI pipeline.
- Execute commands to install and configure Jenkins.

## 4. Jenkins Configuration:

[illegible]

- **sudo yum update -y**
- **sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhatstable/jenkins.repo>**
- **sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>**
- **sudo yum install jenkins -y**
- **sudo systemctl enable jenkins**
- **sudo systemctl start jenkins**
- **sudo systemctl status 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 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 18:12:21 ip-172-31-47-120.ec2.internal jenkins[25878]: *****
Oct 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$1:nonAttained: Completed initialization
Oct 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.366+0000 [id=24] INFO hudson.lifecycle.Lifecycle:Lifecycle#onReady: Jenkins is fully up and
Oct 18:12:27 ip-172-31-47-120.ec2.internal systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 18:12:27 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:27.436+0000 [id=47] INFO h.n.DownloadService$Downloadable#load: Obtained the updated
Oct 18:12:27 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 18:12:32 ip-172-31-47-120.ec2.internal jenkins[25878]: 2024-10-18 18:12:32.494+0000 [id=62] WARNING h.n.DiskSpaceMonitorDescriptor#markNodeOfflineOrOnline: Me

```

## Getting Started

# Unlock Jenkins

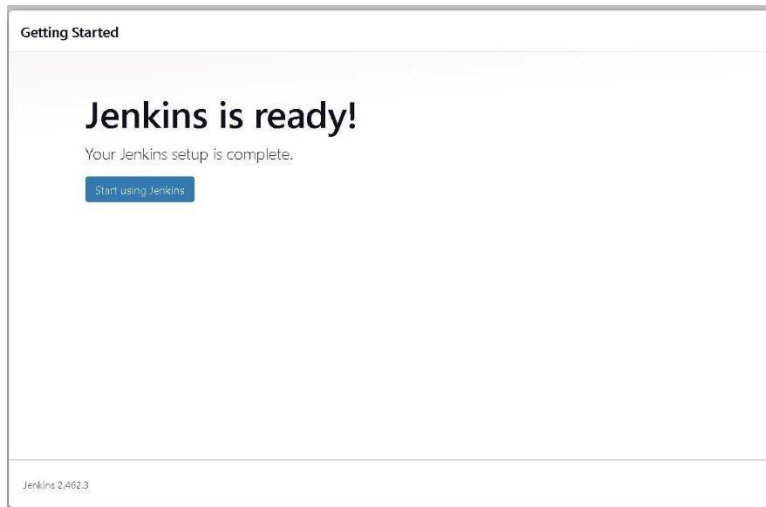
To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

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

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

**Administrator password**

Continue

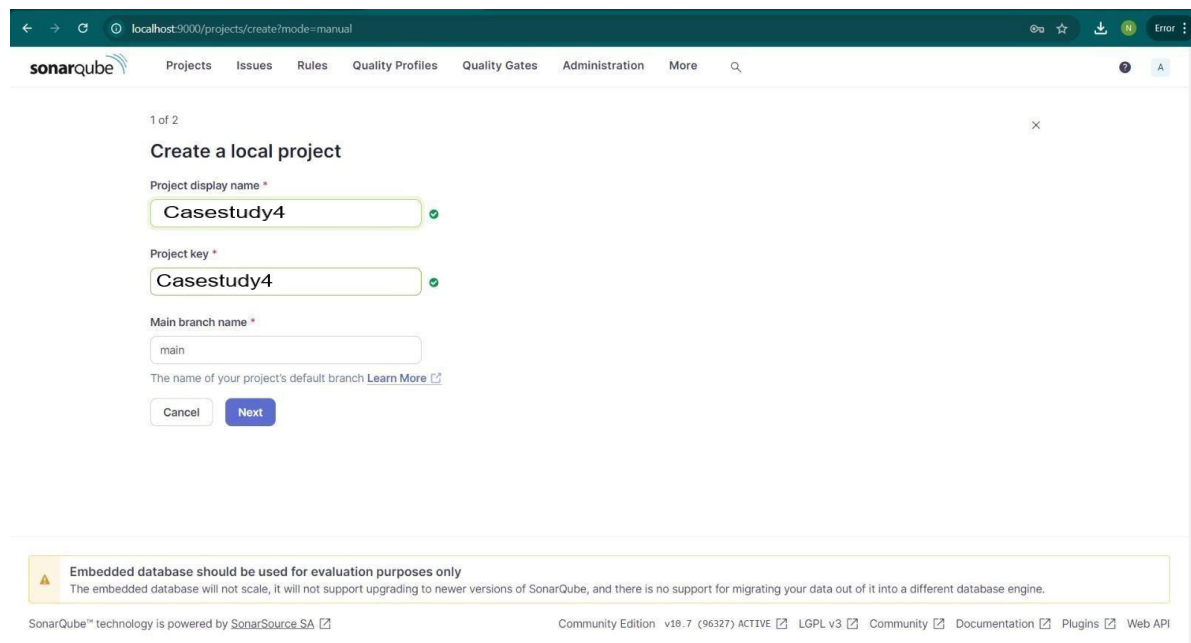


## 5. Task: SonarQube analysis of a Java/Python Project on Jenkins Pipeline:-

A] Sonarqube project:-

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

### 1) Create a sonarqube project named casestudy\_24.



### 2) Configuration of SonarQube 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

### 3) Deploy Code on Jenkins Pipeline :

Dashboard > casestudy\_24 > Configuration

### Configure

General

Advanced Project Options

Pipeline

Advanced Project Options

Advanced

Pipeline

Definition

Pipeline script

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       C:\Users\lgines\Downloads\sonar-scanner-cli-6.2.1.4010-windows-x64\sonar-scanner-6.2.1.4010-windows-x64\bin\sonar-scanner.bat
9       -D sonar.projectkey=casestudy_24
10      -D sonar.exclusions=src/**/*.java
11      -D sonar.host.url=http://localhost:9000
12      -D sonar.login=admin
13      -D sonar.password=Admin@123456
14    }
15  }
16 }
17

```

☒ Use Groovy Sandbox

Pipeline Syntax

Save Apply

REST API Jenkins 2.452.1

### 4) Pipeline Script :

Script:- node

```

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

```

```

"C:\bhagt\lnish\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=Admin@123456 " +

```

```
"-D sonar.projectKey=casestudy_24 " +  
"-D sonar.exclusions=vendor/**,resources/**,**/*.java  
" + "-D sonar.host.url=http://localhost:9000/"  
}  
}  
}
```

## 5)Open Console Output on Jenkins to check whether the Output is Success or not.

Console Output

```
started by user Roshan Bhagiani  
[Pipeline] Start of Pipeline  
[Pipeline] node  
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\Case study 4  
[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\Case study 4\.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 2.47.0.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" # timeout=10  
Checking out Revision 55836fb727b22c08f8f5f8e2577861cd97b75c9 (refs/remotes/origin/master)  
> git.exe config core.sparsecheckout # timeout=10  
> git.exe checkout -f 55836fb727b22c08f8f5f8e2577861cd97b75c9 # timeout=10  
> git.exe branch -a -v --no-abbrev # timeout=10  
> git.exe branch -D master # timeout=10  
> git.exe checkout -D master 55836fb727b22c08f8f5f8e2577861cd97b75c9 # timeout=10  
Commit message: "SQUARBY:2188 Introduce the concept of unresolvedReportType (#2077)"  
> git.exe rev-list --no-walk 45f633ef84936b07f0b9f796b4d3918ac4ef900 # timeout=10  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (SonarQube Analysis)  
[Pipeline] withSonarQubeEnv  
Injecting SonarQube environment variables using the configuration: sonarqube  
[Pipeline] {  
[Pipeline] bat  
  
C:\ProgramData\Jenkins\workspace\Case study 4>"C:\Users\bhagt\Downloads\sonar-scanner-c11-6.2.1.4610-windows-x64\sonar-scanner-6.2.1.4610-windows-x64\bin\sonar-scanner.bat" -D sonar.login=admin -D  
sonar.password=admin@123456 -D sonar.projectKey=casestudy4 -D sonar.exclusions=vendor/**,resources/**,**/*.java -D sonar.host.url=http://localhost:9000/  
21:13:52.615 INFO Scanner configuration file: C:\Users\bhagt\Downloads\sonar-scanner-c11-6.2.1.4610-windows-x64\sonar-scanner-6.2.1.4610-windows-x64\bin\..conf\sonar-scanner.properties
```

## 6)SonarQube assessment(Analysis):

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More

casestudy4 / main

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

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

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

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

Take the Tour Not now

Quality Gate

Passed

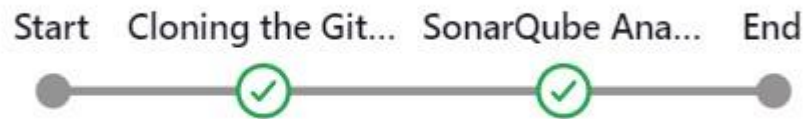
Last analysis 12 minutes ago

The last analysis has warnings. See details

New Code Overall Code

New Code: Since October 20, 2024 Started 1 day ago

#11



**Thus, the Python project was successfully analyzed with SonarQube.**

## **6. Conclusion :**

This case study highlights the successful integration of Jenkins and SonarQube to automate continuous integration and static code analysis for Python and Java projects. This CI pipeline enhances code quality and ensures compliance with security standards before deployment, fostering continuous improvement and accountability within the development team. Ultimately, this integration reduces vulnerabilities and streamlines the software delivery lifecycle.

enhances code quality and ensures compliance with security standards before deployment, fostering continuous improvement and accountability within the development team. Ultimately, this integration reduces vulnerabilities and streamlines the software delivery lifecycle.