AdvanceDevOps Practical Case_Study:

NITISH BHOSLE

D15A – 04

4. Continuous Integration with Static Code Analysis

Concepts Used: Jenkins, SonarQube, and AWS Cloud9(EC2 Instance).
Problem Statement: "Set up a Jenkins pipeline using AWS Cloud9 IDE to perform a static analysis of a Java/Python application. Integrate SonarQube for code quality checks."
Tasks:

- ○ Install Jenkins and set up a basic pipeline.
- ○ Configure SonarQube as part of the pipeline for static code analysis.
- ○ Run the pipeline and generate a report for code quality issues.

1. Introduction

Case Study Overview

This case study demonstrates the setup of a Continuous Integration (CI) pipeline using Jenkins, integrated with SonarQube for static code analysis on AWS EC2 instance. The objective is to automate the quality checks of a Java/Python application and ensure that code quality issues are identified early in the development process. Jenkins, a popular CI/CD tool, automates code builds, testing, and deployment, while SonarQube provides comprehensive code quality and security analysis.

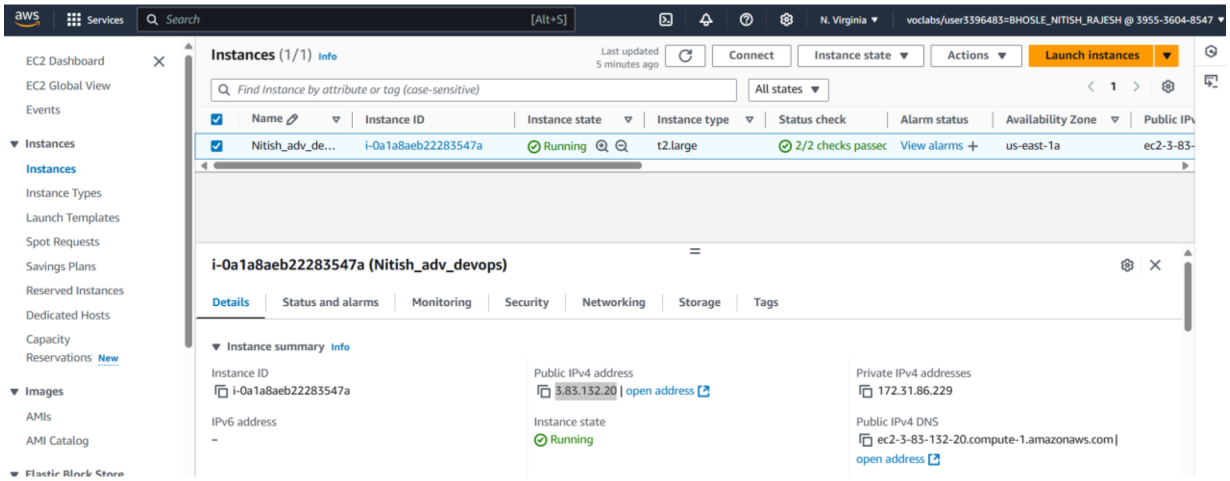
Key Feature and Application

The key feature of this project is the integration of static code analysis into the CI pipeline. This allows developers to ensure code quality and adherence to best practices without manual intervention. It ensures early detection of bugs, code smells, and potential security vulnerabilities, making the development process smoother and more reliable.

2. Step-by-Step Explanation

Step 1: Create EC2 Instances for Jenkins and SonarQube

1. Login to AWS Console:
- Navigate to the AWS Management Console and log in.
2. Launch EC2 Instances:
- Go to the EC2 Dashboard and click Launch Instance.
- Choose Ubuntu Server 22.04 LTS as your AMI.
- Select an instance type (t2.large for Jenkins and SonarQube).
- Configure instance details (ensure Auto-assign Public IP).
- Add storage (default 8 GB, adjust for larger projects).
- Configure security group (allow ports 22, 8080 for Jenkins, 9000 for SonarQube).
- Launch instances and download the key pair for SSH access.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
Nitish_adv_de...	i-0a1a8aeb22283547a	Running	t2.large	2/2 checks passec	View alarms +	us-east-1a	ec2-3-83-

i-0a1a8aeb22283547a (Nitish_adv_devops)

Instance summary Info

Instance ID: i-0a1a8aeb22283547a
Public IPv4 address: 3.83.132.20 | open address
Private IPv4 addresses: 172.31.86.229
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-3-83-132-20.compute-1.amazonaws.com | open address

Step 2: Install Jenkins on EC2

1. Connect to EC2 via SSH:
- Use the key pair to SSH into the Jenkins EC2 instance.
2. Update the system:

```
sudo apt update
sudo apt upgrade
```

3. Install Java 11:

```
sudo apt install openjdk-11-jdk
```

4. Add Jenkins Repository:
- Add Jenkins GPG key and repository.

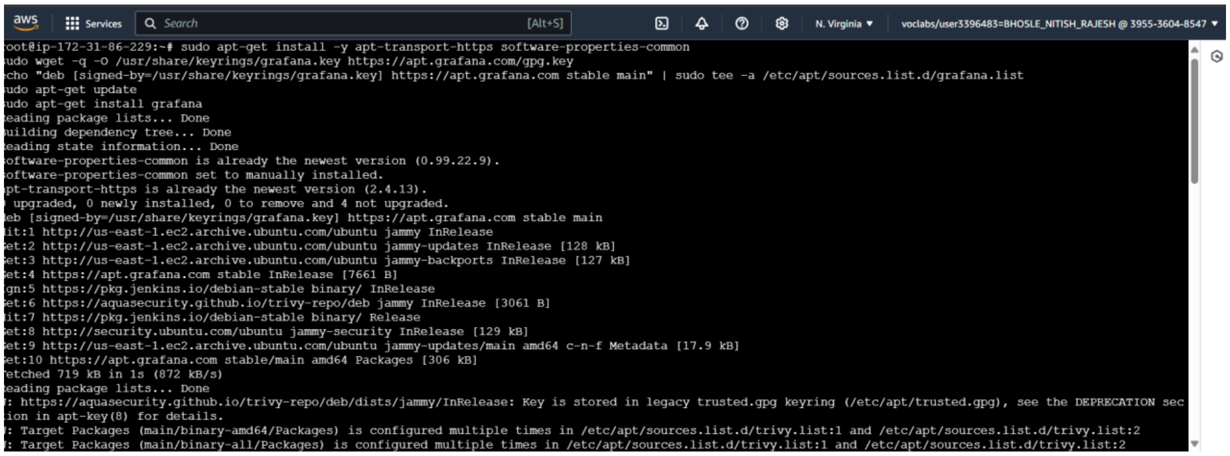
```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources
```

5. Install Jenkins:

```
sudo apt update
sudo apt install jenkins
```

6. Start Jenkins:

```
sudo systemctl start jenkins
sudo systemctl status jenkins
```



```
:oot@ip-172-31-86-229:~# sudo apt-get install -y apt-transport-https software-properties-common
:udo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key
:cho "deb [signed-by=/usr/share/keyrings/grafana.key] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
:udo apt-get update
:udo apt-get install grafana
:eading package lists... Done
:uilding dependency tree... Done
:eading state information... Done
:oftware-properties-common is already the newest version (0.99.22.9).
:oftware-properties-common set to manually installed.
:pt-transport-https is already the newest version (2.4.13).
 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
:eb [signed-by=/usr/share/keyrings/grafana.key] https://apt.grafana.com stable main
:it:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
:et:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
:et:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
:et:4 https://apt.grafana.com stable InRelease [7661 B]
:gn:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
:et:6 https://aquasecurity.github.io/trivy-repo/deb jammy InRelease [3061 B]
:it:7 https://pkg.jenkins.io/debian-stable binary/ Release
:et:8 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
:et:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.9 kB]
:et:10 https://apt.grafana.com stable/main amd64 Packages [306 kB]
:etched 719 kB in 1s (872 kB/s)
:eading package lists... Done
: https://aquasecurity.github.io/trivy-repo/deb/dists/jammy/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION sec
:ion in apt-key(8) for details.
: Target Packages (main/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
: Target Packages (main/binary-all/Packages) is configured multiple times in /etc/apt/sources.list.d/trivy.list:1 and /etc/apt/sources.list.d/trivy.list:2
```

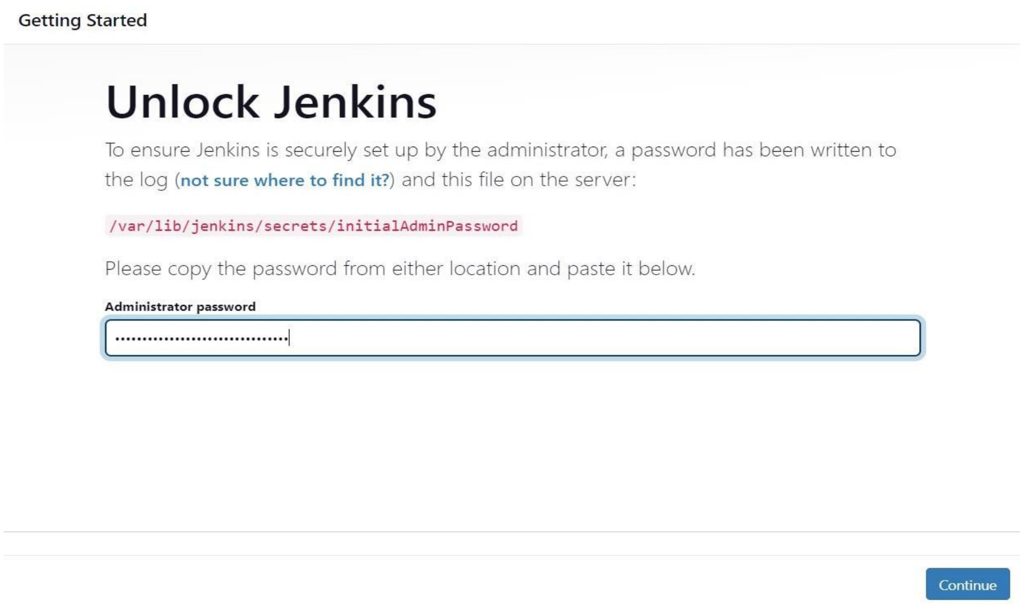
7. Access Jenkins:

- Navigate to in your browser :

```
http://<EC2-Jenkins-Public-IP>:8080
```

- Follow on-screen instructions and use the initialAdminPassword to unlock Jenkins.

```
root@ip-172-31-89-83:~# cat /var/lib/jenkins/secrets/initialAdminPassword
8a833890ecbc406382bac565582d5842
```



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

••••••••••••••••••••••••••••••••

Step 3: Install SonarQube on EC2

1. Connect to EC2 via SSH:

- Use the key pair to SSH into the SonarQube EC2 instance.

```
ssh -i /path/to/key.pem ubuntu@<EC2-SonarQube-Public-IP>
```

2. Update the system:

```
sudo apt update
sudo apt upgrade
```

3. Install prerequisites (unzip, wget):

```
sudo apt install unzip wget
```

4. Download and install SonarQube:

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.
unzip sonarqube-9.9.0.65466.zip
sudo mv sonarqube-9.9.0.65466 /opt/sonarqube
```

5. Start SonarQube:

```
cd /opt/sonarqube/bin/linux-x86-64
./sonar.sh start
```

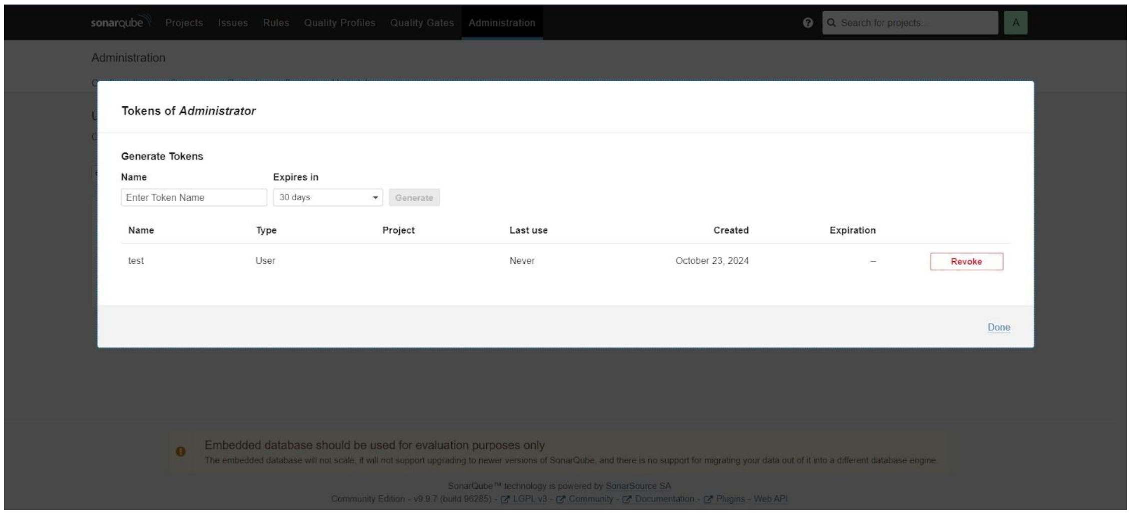
6. Access SonarQube:

- Navigate to in your browser :

```
http://<EC2-SonarQube-Public-IP>:9000
```

- Login using default credentials (admin/admin) and complete the setup.

7. Generate a token for Jenkins:



Administration

Tokens of *Administrator*

Generate Tokens

Name: Enter Token Name

Expires in: 30 days

Generate

Name	Type	Project	Last use	Created	Expiration	
test	User		Never	October 23, 2024	–	Revoke

Done

Embedded database should be used for evaluation purposes only

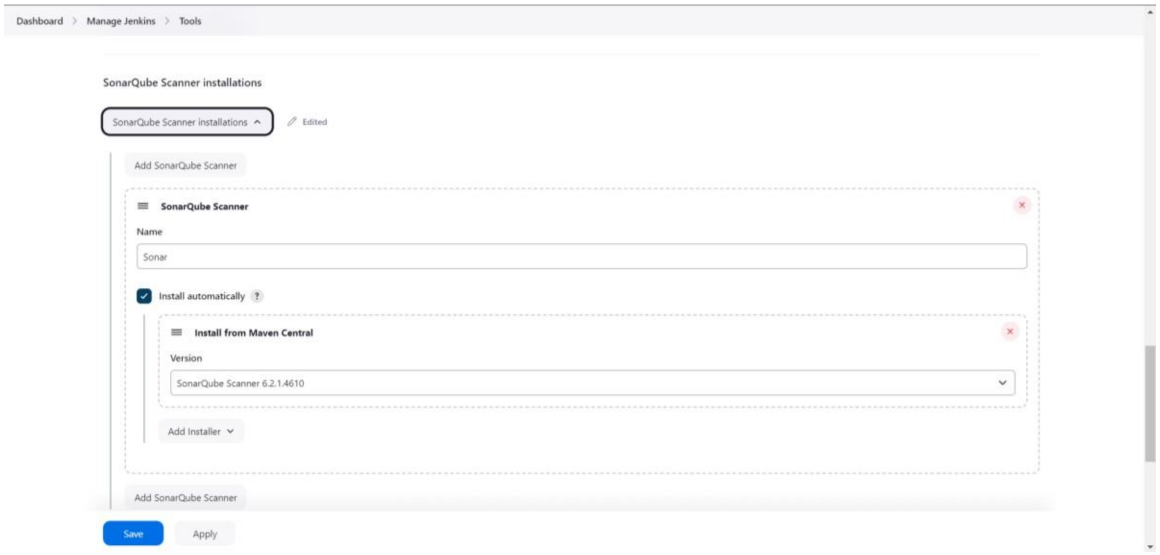
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

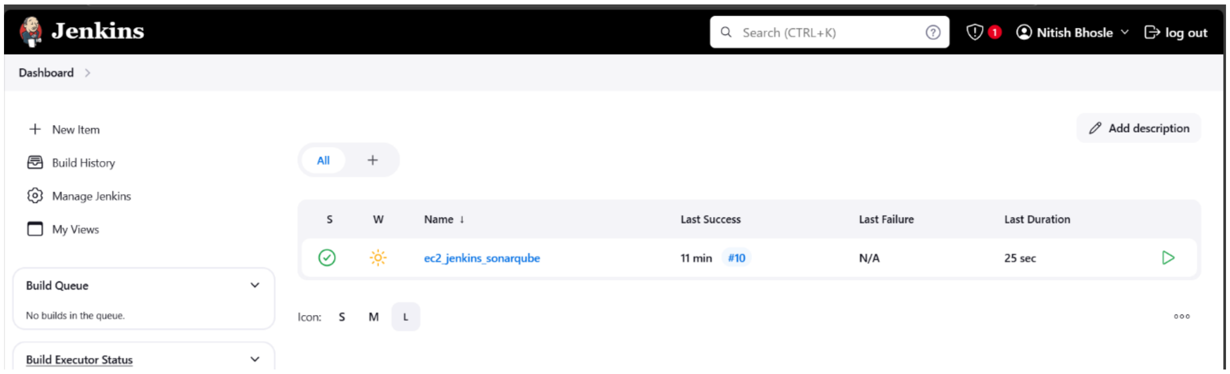
SonarQube™ technology is powered by SonarSource SA

Community Edition - v9.9.7 (build 96285) - LGPL v3 - Community - Documentation - Plugins - Web API

Step 4: Configure Jenkins to Integrate with SonarQube

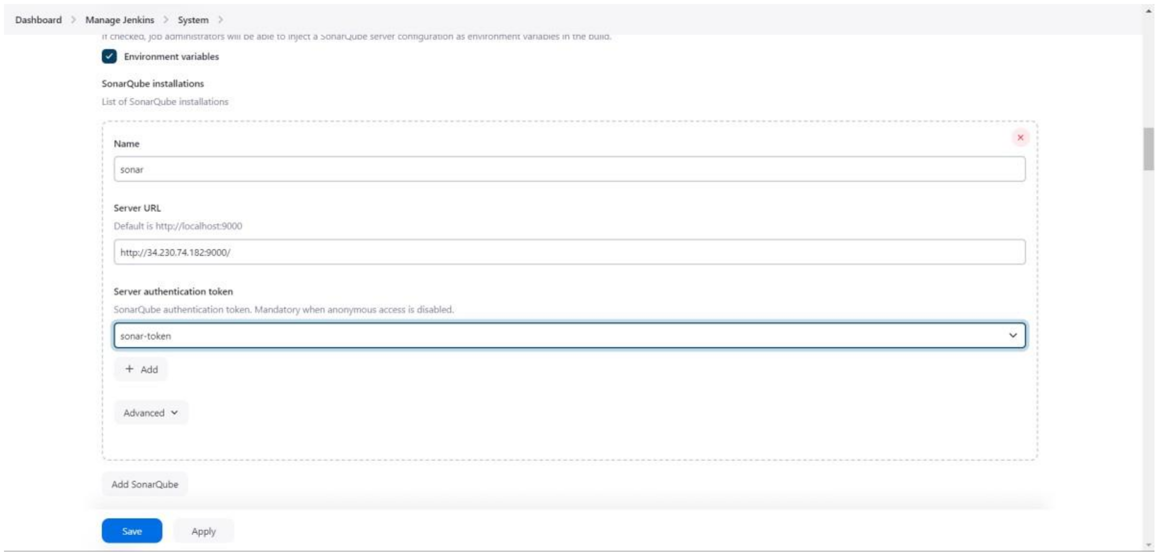
1. Install SonarQube Plugin in Jenkins:
- Go to Manage Jenkins → Manage Plugins.
- Install SonarQube Scanner plugin.


Dashboard > Manage Jenkins > Tools
SonarQube Scanner installations
SonarQube Scanner installations
Edited
Add SonarQube Scanner
SonarQube Scanner
Name
Sonar
Install automatically
Install from Maven Central
Version
SonarQube Scanner 6.2.1.4610
Add Installer
Add SonarQube Scanner
Save
Apply



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	ec2_jenkins_sonarqube	11 min #10	N/A	25 sec

2. Configure SonarQube in Jenkins:
- Go to Manage Jenkins → Configure System.
- Add SonarQube server details (IP, token) and configure SonarQube Scanner under Global Tool Configuration.



Dashboard > Manage Jenkins > System >

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

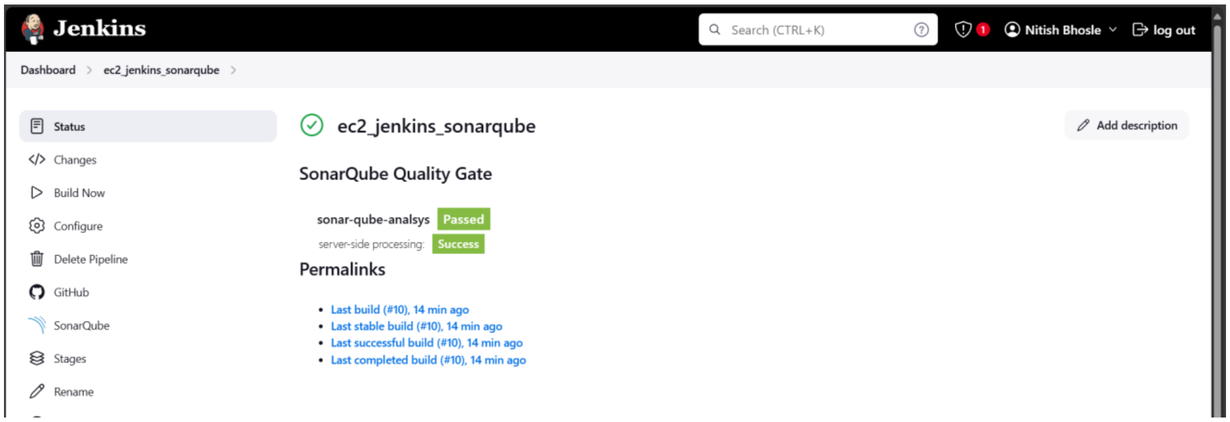
Name: sonar

Server URL (Default is http://localhost:9000): http://34.230.74.182:9000/

Server authentication token (SonarQube authentication token. Mandatory when anonymous access is disabled.): sonar-token

Step 5: Set Up a Jenkins Pipeline for Static Code Analysis

1. Create a new Jenkins Pipeline job.
2. In the pipeline script, add the following basic pipeline code:



Dashboard > ec2_jenkins_sonarqube >

ec2_jenkins_sonarqube

SonarQube Quality Gate

sonar-qube-analsys: Passed

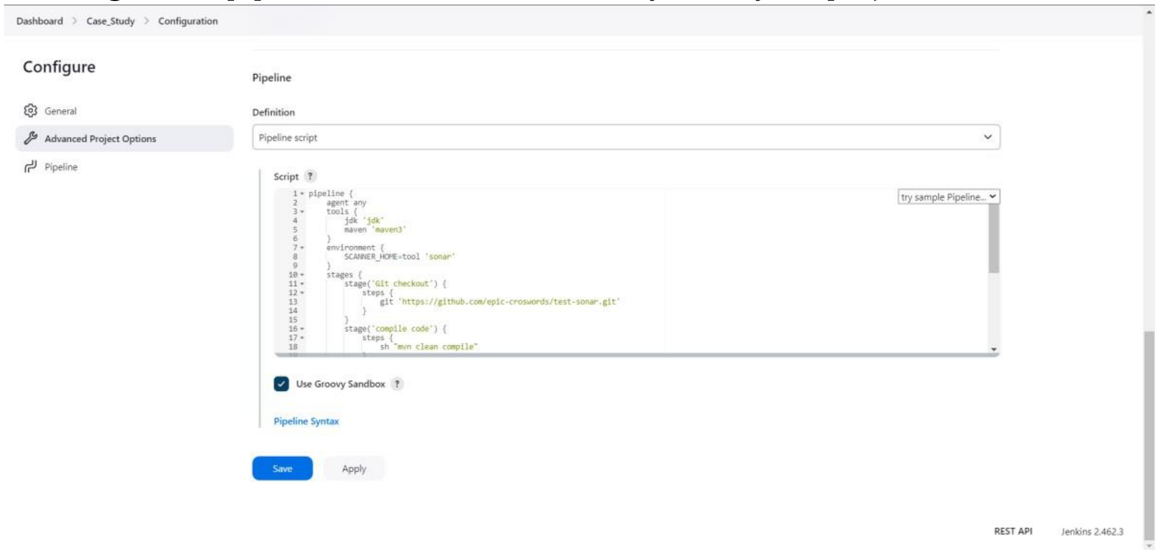
server-side processing: Success

Permalinks

- Last build (#10), 14 min ago
- Last stable build (#10), 14 min ago
- Last successful build (#10), 14 min ago
- Last completed build (#10), 14 min ago

```
pipeline {

  agent any

  stages {

    stage('SCM Checkout') {

      steps {

        git 'https://github.com/your-repo.git'

      }
    }

    stage('SonarQube Analysis') {

        steps {

            withSonarQubeEnv('SonarQube') {

                sh 'mvn clean verify sonar:sonar'

            }

        }

    }

}

}
```

3. Configure the pipeline to run SonarQube analysis for your project.



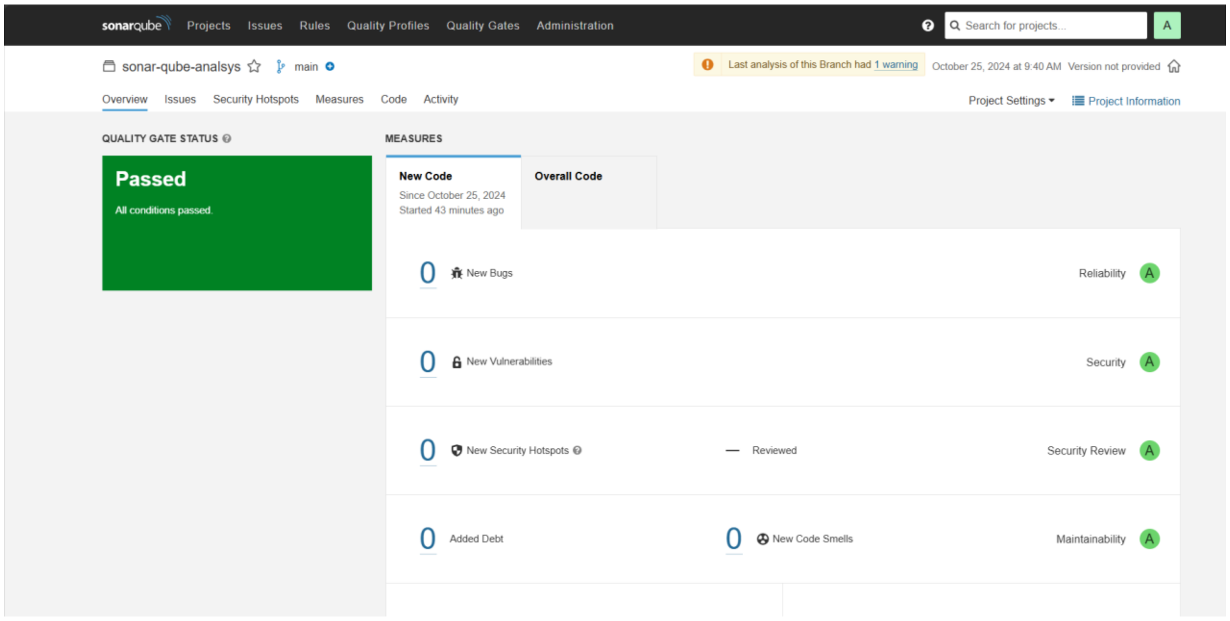
Pipeline script

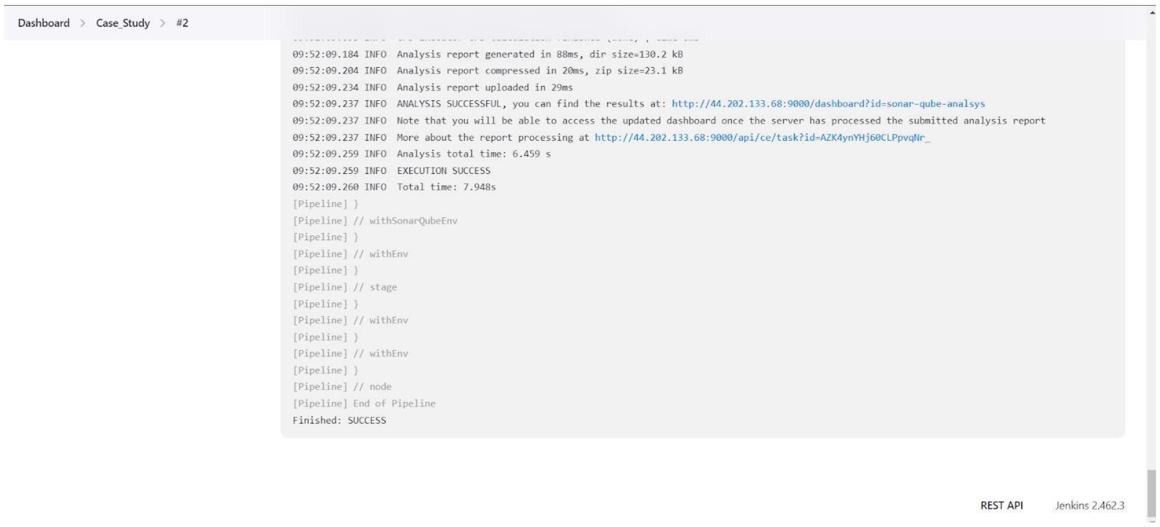
```
pipeline {
    agent any
    tools {
        jdk 'jdk'
        maven 'maven3'
    }
    environment {
        SCANNER_HOME=tool 'sonar'
    }
    stages {
        stage('Git checkout') {
            steps {
                git 'https://github.com/epic-croswords/test-sonar.git'
            }
        }
        stage('compile code') {
            steps {
                sh "mvn clean compile"
```

[✓] Use Groovy Sandbox

Step 6: Run the Jenkins Pipeline

1. Trigger the pipeline manually by clicking Build Now, or set it to run automatically after code push.
2. Monitor pipeline execution via Jenkins console output.

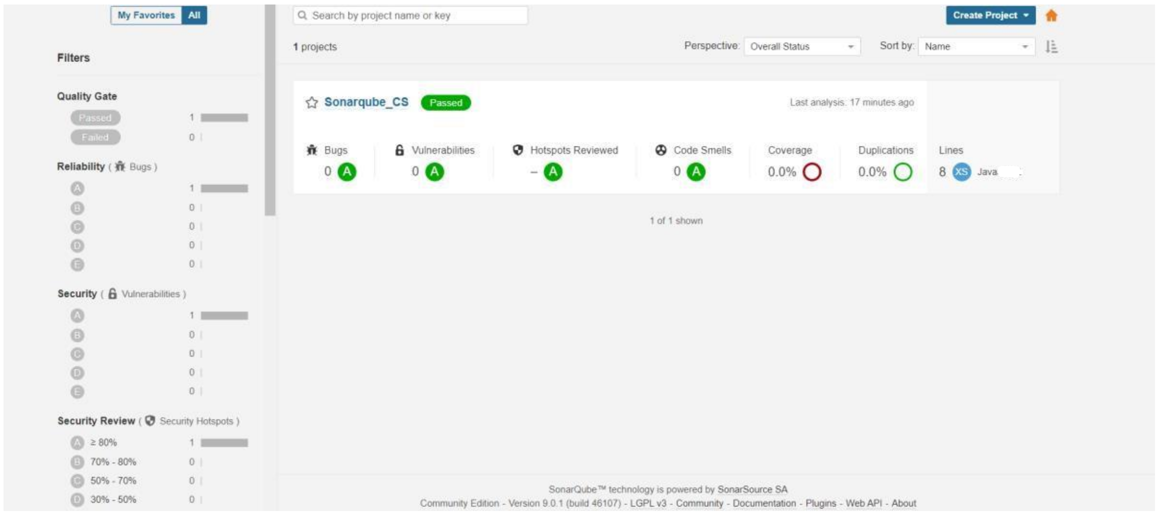




```
09:52:09.184 INFO  Analysis report generated in 88ms, dir size=130.2 kB
09:52:09.204 INFO  Analysis report compressed in 20ms, zip size=23.1 kB
09:52:09.234 INFO  Analysis report uploaded in 29ms
09:52:09.237 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://44.202.133.68:9000/dashboard?id=sonar-qube-analsys
09:52:09.237 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
09:52:09.237 INFO  More about the report processing at http://44.202.133.68:9000/api/ce/task?id=AZK4ynYHj60CLPpvqNr_
09:52:09.259 INFO  Analysis total time: 6.459 s
09:52:09.259 INFO  EXECUTION SUCCESS
09:52:09.260 INFO  Total time: 7.948s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Step 7: Analyze SonarQube Report

1. Access SonarQube dashboard at http://<EC2-SonarQube-Public-IP>:9000.
2. View code quality reports (bugs, code smells, vulnerabilities) and address issues accordingly.


My Favorites
All
Search by project name or key
Create Project
Filters
Quality Gate
Passed 1
Failed 0
Reliability (Bugs)
Security (Vulnerabilities)
Security Review (Security Hotspots)
≥ 80% 1
70% - 80% 0
50% - 70% 0
30% - 50% 0
1 projects
Perspective: Overall Status
Sort by: Name
Sonarqube_CS Passed
Last analysis: 17 minutes ago
Bugs 0 A
Vulnerabilities 0 A
Hotspots Reviewed – A
Code Smells 0 A
Coverage 0.0%
Duplications 0.0%
Lines 8 XS Java
1 of 1 shown
SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.0.1 (build 46107) - LGPL v3 - Community - Documentation - Plugins - Web API - About

UNIQUE FEATURES

1. **Automated Slack Notifications:**
    - Configure Jenkins to send notifications to a Slack channel whenever the pipeline completes or issues are detected by SonarQube.
    - Use the Slack Notification plugin in Jenkins.
2. **GitHub Webhook Integration:**
    - Set up a GitHub webhook so that any new code pushed to the repository automatically triggers the Jenkins pipeline.
3. **SonarQube Quality Gate Enforcement:**
    - Set up quality gates in SonarQube to automatically fail the Jenkins build if the code doesn't meet specific quality standards (e.g., a certain number of bugs, coverage percentage, or vulnerability issues).
4. **Real-time Code Metrics Dashboard:**
    - Configure a Grafana Dashboard with SonarQube metrics to visualize real-time code quality data. This could include trend graphs of code coverage, technical debt, or issue count.