

Test Java Code using SonarQube on Linux

This document provides a complete step-by-step guide to test Java code using SonarQube on a Linux server without installing SonarScanner or PostgreSQL. It utilizes Maven's built-in SonarQube plugin and SonarQube's embedded H2 database.

1. Pre-requisites

- A linux server with 2 CPU's and 4gb RAM
- Java (JDK 17 or later)
- Maven (3.6
- unzip, wget

Launch an instance with 2 CPU's and 4gb RAM

The screenshot displays the AWS Management Console interface for launching a new EC2 instance. The 'Name and tags' section has 'Sonarqube' entered as the name. The 'Application and OS Images (Amazon Machine Image)' section shows a search for 'Ubuntu Server 22.04 LTS (HVM), 550 Volume Type' (AMI ID: ami-0b8dd8c17ed981ef9). The 'Instance type' section shows 't2.medium' selected. The 'Summary' section on the right shows 'Number of instances: 1', 'Software Image (AMI): Canonical, Ubuntu, 22.04, amd64', 'Virtual server type (instance type): t2.medium', 'Firewall (security group): New security group', and 'Storage (volumes): 1 volume(s) - 8 GiB'. A 'Free tier' notification is visible, stating that the first year of opening an AWS account includes 750 hours of t2.micro instance usage. The 'Launch Instance' button is highlighted in orange.

Name and tags Info

Name: Sonarqube [Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Q Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), 550 Volume Type Free tier eligible

ami-0b8dd8c17ed981ef9 (64-bit (x86)) / ami-0102110wfs25172b (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 22.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 22.04, amd64 jammy image

Architecture: 64-bit (x86) AMI ID: ami-0b8dd8c17ed981ef9 Publish Date: 2025-08-22 Username: ubuntu [Verified provider](#)

▼ Instance type Info | Get advice

Instance type: t2.medium

Family: t2 2 vCPU 4 GB Memory Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour On-Demand Linux base pricing: 0.0464 USD per Hour

On-Demand RHEL base pricing: 0.0752 USD per Hour On-Demand Windows base pricing: 0.0644 USD per Hour

☐ All generations [Compare instance types](#)

▼ Summary

Number of instances: 1

Software Image (AMI)

Canonical, Ubuntu, 22.04, amd64 [read more](#)

ami-0b8dd8c17ed981ef9

Virtual server type (instance type)

t2.medium

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

[Cancel](#) [Launch Instance](#) [Preview code](#)

Create a key pair with a suitable name

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

Sonardube

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA

RSA encrypted private and public key pair

☐ ED25519

ED25519 encrypted private and public key pair

Private key file format

☒ .pem

For use with OpenSSH

☐ .ppk

For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn](#)

Cancel

Create key pair

After launching the instance, connect to it with the public IP address of the server

Session settings

SSH

Telnet

Rsh

Xdmcp

RDP

VNC

FTP

SFTP

Serial

File

Shell

Browser

Mosh

Aws S3

WSL

Basic SSH settings

Remote host * 54.219.240.228

☒ Specify username ubuntu

Port 22

Advanced SSH settings

Terminal settings

Network settings

Bookmark settings

☒ X11-Forwarding

☒ Compression

Remote environment: Interactive shell

Execute command:

☐ Do not exit after command ends

SSH-browser type: SFTP protocol

☐ Follow SSH path (experimental)

☒ Use private key C:\Users\ADMIN\Downloads\Def

Expert SSH settings

Execute macro at session start: <none>

OK

Cancel

After connecting to the server update the server to the latest package installations

```
ubuntu@ip-172-31-30-110:~$ sudo apt update -y
```

Now, install java version

```
ubuntu@ip-172-31-30-110:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-11-jre-headless # version 11.0.28+6-1ubuntu1~22.04.1, or
sudo apt install default-jre # version 2:1.11-72build2
sudo apt install openjdk-17-jre-headless # version 17.0.16+8~us1-0ubuntu1~22.04.1
sudo apt install openjdk-18-jre-headless # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-21-jre-headless # version 21.0.8+9~us1-0ubuntu1~22.04.1
sudo apt install openjdk-25-jre-headless # version 25+36-1~22.04.2
sudo apt install openjdk-8-jre-headless # version 8u462-ga~us1-0ubuntu2~22.04.2
ubuntu@ip-172-31-30-110:~$ sudo apt install openjdk-17-jre-headless
```

After installing java, we need to install maven

```
ubuntu@ip-172-31-30-110:~$ sudo apt install maven -y
```

By this step, we have installed all the necessary pre-requisites on the server.

Now, clone the code from github where your Java code is present

```
ubuntu@ip-172-31-30-110:~$ git clone https://github.com/akracad/JavaWebCal.git
Cloning into 'JavaWebCal'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 29 (delta 3), reused 29 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (29/29), 5.78 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (3/3), done.
ubuntu@ip-172-31-30-110:~$
```

Now install sonarqube on the server

```
ubuntu@ip-172-31-27-244:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.6.0.92116.zip
--2025-10-21 11:24:07-- http://wget/
Resolving wget (wget)... failed: Temporary failure in name resolution.
wget: unable to resolve host address 'wget'
--2025-10-21 11:24:07-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.6.0.92116.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.188.45, 99.84.188.35, 99.84.188.106, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.188.45|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 731034761 (697M) [binary/octet-stream]
Saving to: 'sonarqube-10.6.0.92116.zip'

sonarqube-10.6.0.92116.zip 100%[=====] 697.17M 401MB/s in 1.7s

2025-10-21 11:24:09 (401 MB/s) - 'sonarqube-10.6.0.92116.zip' saved [731034761/731034761]

FINISHED --2025-10-21 11:24:09--
Total wall clock time: 1.8s
Downloaded: 1 files, 697M in 1.7s (401 MB/s)
ubuntu@ip-172-31-27-244:~$ ls
sonarqube-10.6.0.92116.zip
ubuntu@ip-172-31-27-244:~$
```

Now untar the sonarqube zip file

```
ubuntu@ip-172-31-27-244:~$ unzip sonarqube-10.6.0.92116.zip
Archive:  sonarqube-10.6.0.92116.zip
  creating:  sonarqube-10.6.0.92116/
  creating:  sonarqube-10.6.0.92116/jres/
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_x64_linux_hotspot_17.0.11_9.tar.gz
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_aarch64_linux_hotspot_17.0.11_9.tar.gz
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_x64_alpine-linux_hotspot_17.0.11_9.tar.gz
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_x64_windows_hotspot_17.0.11_9.zip
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_x64_mac_hotspot_17.0.11_9.tar.gz
  inflating: sonarqube-10.6.0.92116/jres/OpenJDK17U-jre_aarch64_mac_hotspot_17.0.11_9.tar.gz
  inflating: sonarqube-10.6.0.92116/dependency-license.json
  inflating: sonarqube-10.6.0.92116/COPYING
  creating:  sonarqube-10.6.0.92116/bin/
  creating:  sonarqube-10.6.0.92116/bin/windows-x86-64/
  inflating: sonarqube-10.6.0.92116/bin/windows-x86-64/SonarService.bat
  creating:  sonarqube-10.6.0.92116/bin/windows-x86-64/lib/
  inflating: sonarqube-10.6.0.92116/bin/windows-x86-64/lib/SonarServiceWrapper.exe
  inflating: sonarqube-10.6.0.92116/bin/windows-x86-64/lib/find_java.bat
  creating:  sonarqube-10.6.0.92116/bin/winsw-license/
  inflating: sonarqube-10.6.0.92116/bin/winsw-license/LICENSE.txt
  creating:  sonarqube-10.6.0.92116/data/
  inflating: sonarqube-10.6.0.92116/data/README.txt
  creating:  sonarqube-10.6.0.92116/extensions/
  creating:  sonarqube-10.6.0.92116/extensions/jdbc-driver/
```

Now start the sonarqube server by navigating into Sonar/bin/linux

```
ubuntu@ip-172-31-27-244:~/sonar$ ls
COPYING  bin  conf  data  dependency-license.json  elasticsearch  extensions  jres  lib  logs  temp  web
ubuntu@ip-172-31-27-244:~/sonar$ cd bin/
ubuntu@ip-172-31-27-244:~/sonar/bin$ ls
elasticsearch  linux-x86-64  macosx-universal-64  windows-x86-64  winsw-license
ubuntu@ip-172-31-27-244:~/sonar/bin$ cd linux-x86-64/
ubuntu@ip-172-31-27-244:~/sonar/bin/linux-x86-64$ ls
sonar.sh
ubuntu@ip-172-31-27-244:~/sonar/bin/linux-x86-64$ ./sonar.sh
/usr/bin/java
Usage: ./sonar.sh { console | start | stop | force-stop | restart | status | dump }
ubuntu@ip-172-31-27-244:~/sonar/bin/linux-x86-64$ ./sonar.sh start
/usr/bin/java
Starting SonarQube...
Started SonarQube.
```

Now allow the port number into the inbound rules of security group of ec2-instance

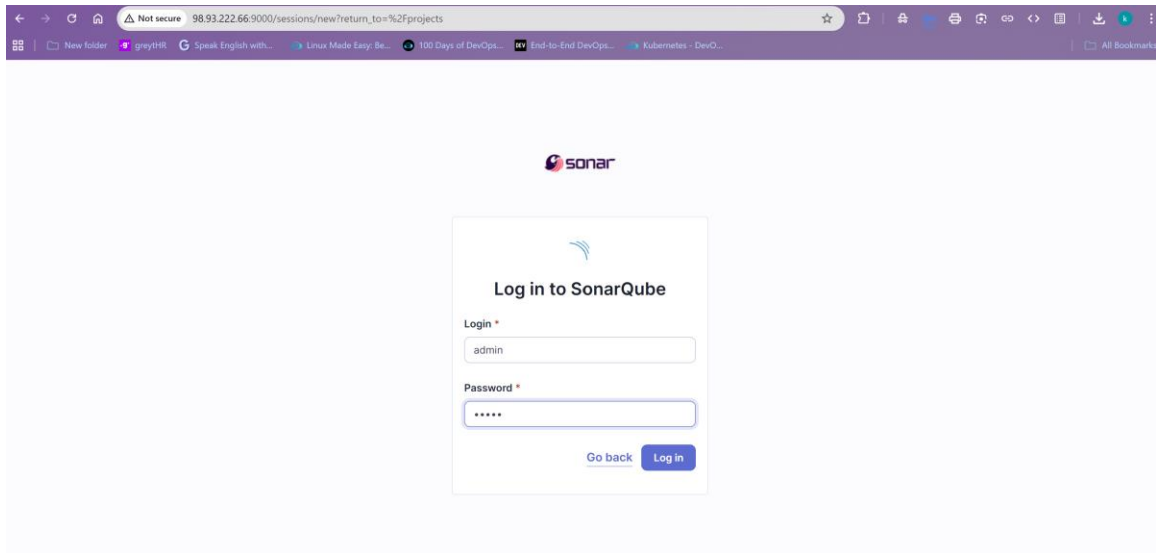
The screenshot shows the AWS Management Console interface for editing inbound rules of a security group. The page title is 'Edit inbound rules' with a sub-header 'Inbound rules control the incoming traffic that's allowed to reach the instance.' Below this, there's a table of inbound rules. The first rule is for SSH on port 22. The second rule is for Custom TCP on port 8081, which is highlighted with a blue border. The 'Source' field for the Custom TCP rule is set to '0.0.0.0/0'. At the bottom, there's a warning message: 'Rules with source of 0.0.0.0/0 or ::0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' The 'Add rule' button is visible at the bottom left.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
sgr-0d01686845410298d	SSH	TCP	22	Custom		Delete
-	Custom TCP	TCP	8081	Anywh...	For Nexus	Delete

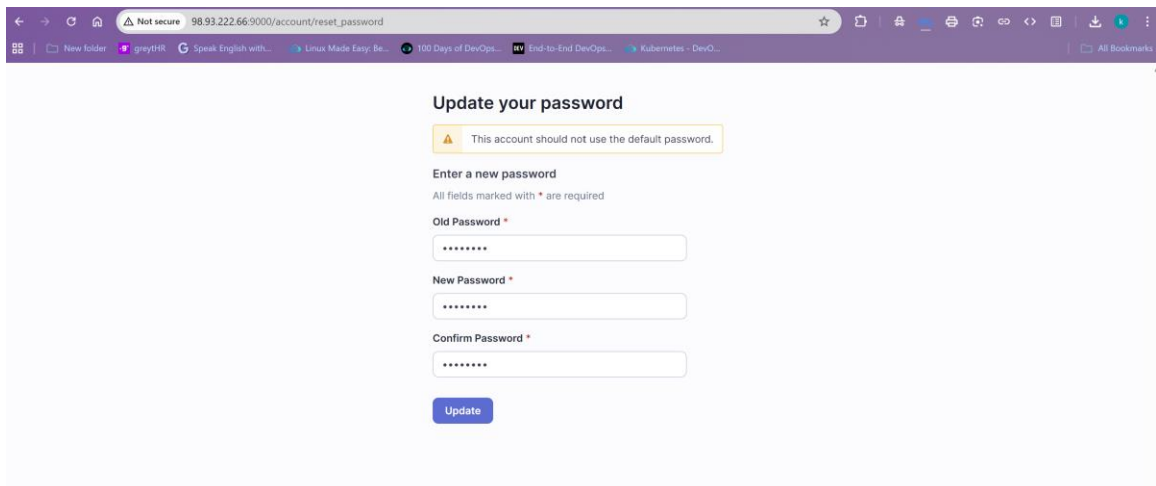
Rules with source of 0.0.0.0/0 or ::0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Buttons: Add rule, Cancel, Preview changes, Save rules

After starting sonarqube, access it on the browser with the public ip and port number.

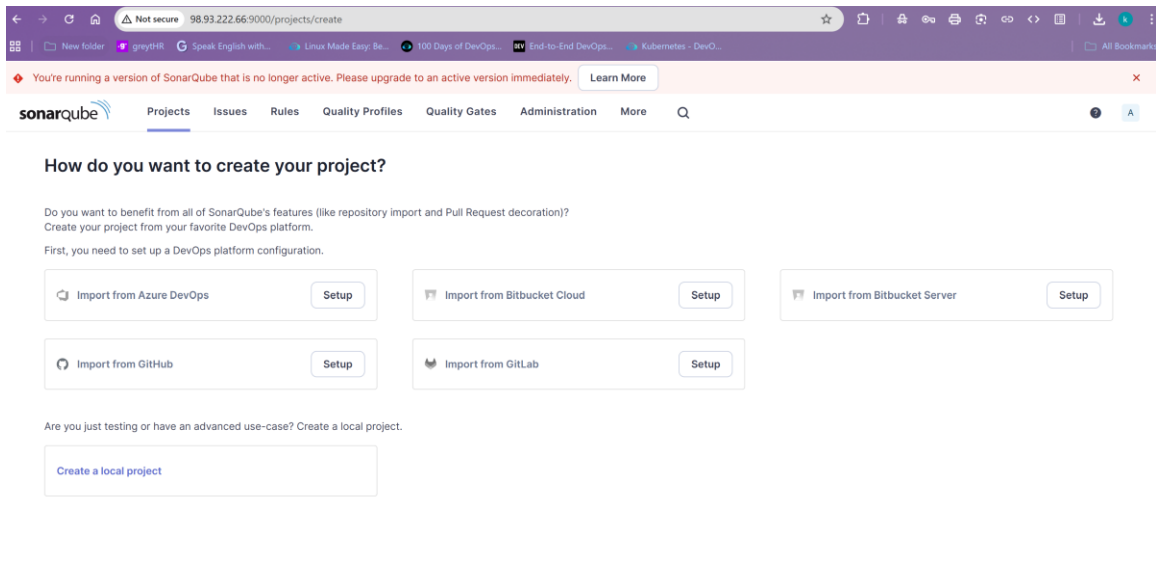


Update your password, once you login into sonarqube

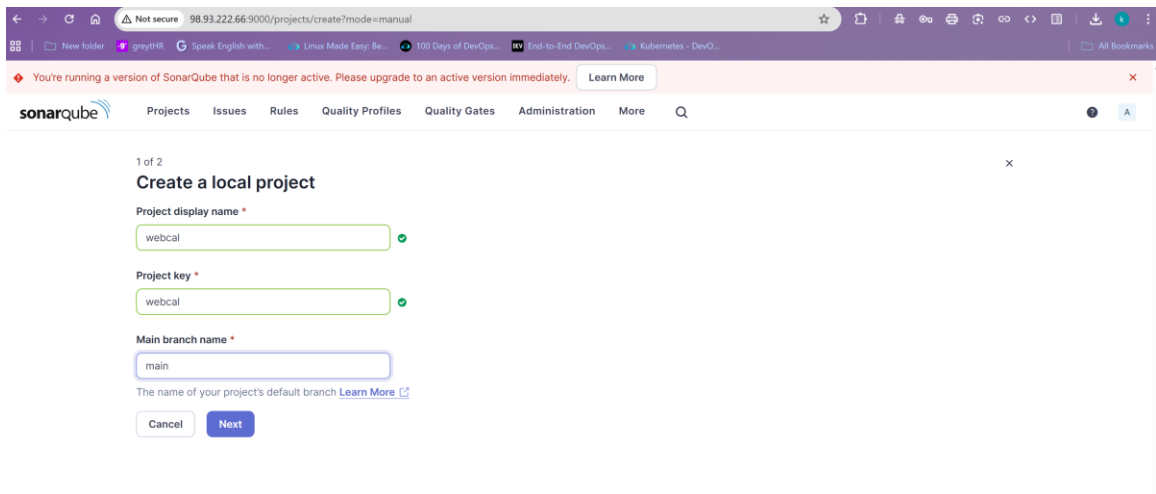


Create a Project in SonarQube UI

1. Log in to SonarQube.
2. Go to Projects → Create Project → Manually.

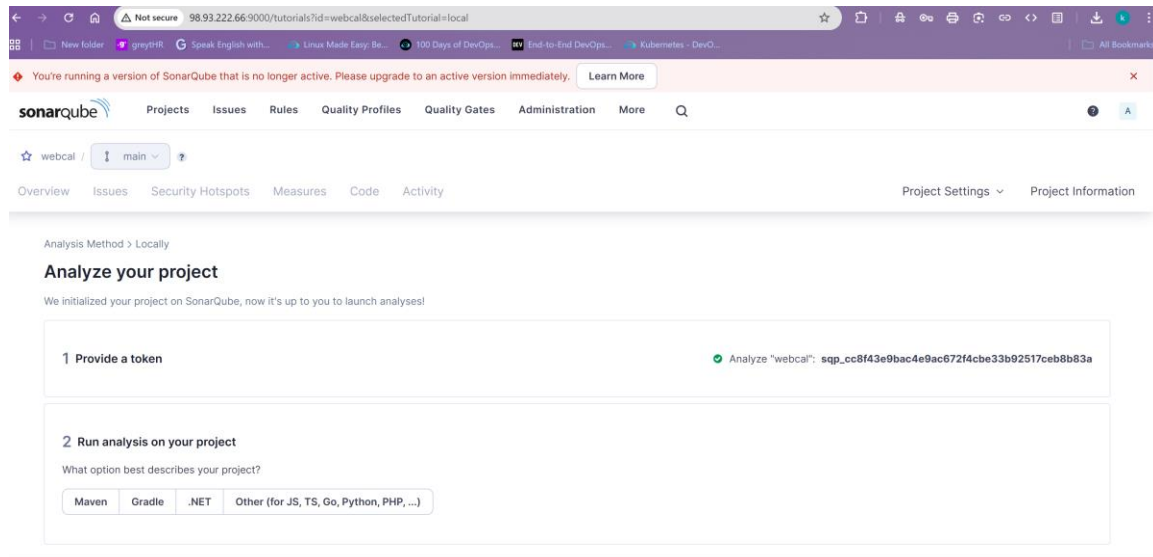


3. Enter Project Key: webcal.
4. Enter Project Name: webcal.



5. Select "Use Maven" method.

6. Generate an authentication token and save it.



Navigate to Java Project directory

Assume your project directory is `/home/user/my-java-app/` and contains a `pom.xml` file.

Run Analysis using Maven (No SonarScanner)

```
cd /home/user/my-java-app/  
mvn clean verify sonar:sonar \  
-Dsonar.projectKey=my-java-app \  
-Dsonar.host.url=http://<server-ip>:9000 \  
-Dsonar.login=<your-generated-token>
```

```
ubuntu@ip-172-31-26-216:~/JavaWebCal$ mvn clean verify sonar:sonar \  
-Dsonar.projectKey=webcal \  
-Dsonar.host.url=http://http://98.93.222.66:9000 \  
-Dsonar.login=snp_cc8f43e9bac4e9ac672f4cbe33b92517ceb8b83a  
[INFO] Scanning for projects...  
[WARNING] The artifact org.codehaus.mojo:sonar-maven-plugin:jar:4.0.0.4121 has been relocated to org.sonarsource.scanner.maven:sonar-maven-plugin:jar:4.0.0.4121: SonarQube plugin was moved to SonarSource organisation  
[INFO]  
[INFO] -----< com.web.cal:webapp >-----  
[INFO] Building WebAppCal Maven Webapp 0.2  
[INFO] -----[ war ]-----  
[INFO]  
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ webapp ---  
[INFO] Deleting /home/ubuntu/JavaWebCal/target
```

This command builds, tests, and analyzes your Java code automatically.

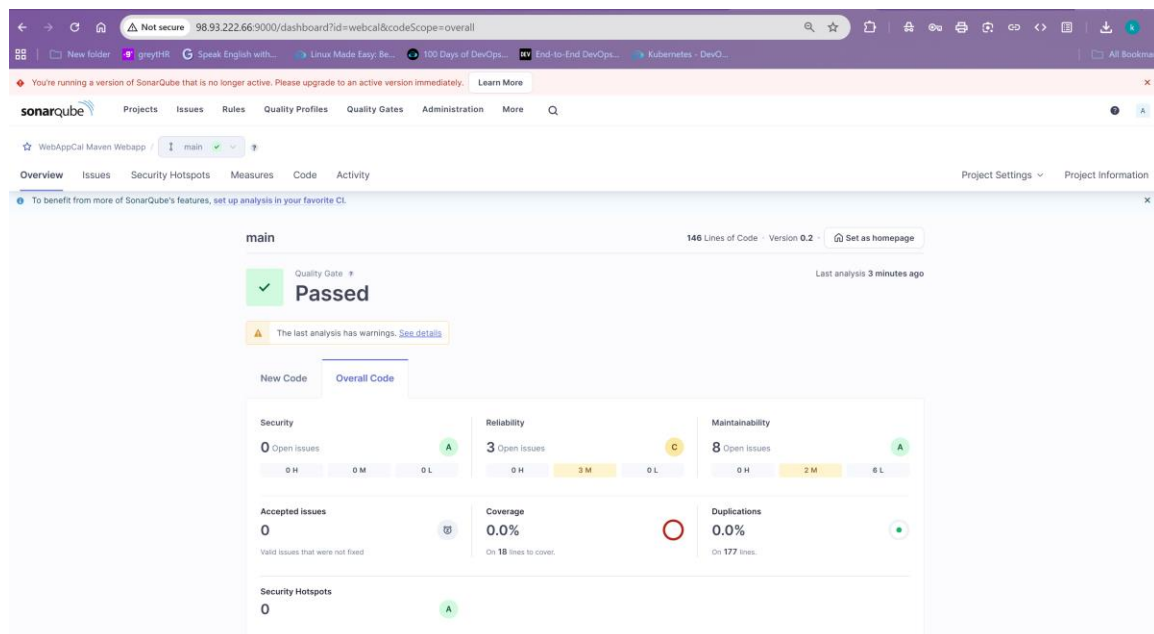
```

[INFO] 11:52:56.223 SCM Publisher 5 source files to be analyzed
[INFO] 11:52:56.424 SCM Publisher 4/5 source files have been analyzed (done) | time=200ms
[WARNING] 11:52:56.424 Missing blame information for the following files:
[WARNING] 11:52:56.425 * pom.xml
[WARNING] 11:52:56.425 This may lead to missing/broken features in SonarQube
[INFO] 11:52:56.429 CPD Executor Calculating CPD for 2 files
[INFO] 11:52:56.437 CPD Executor CPD calculation finished (done) | time=7ms
[INFO] 11:52:56.445 SCM revision ID '8d9c87629e894e160ee506685ad896e084661847'
[INFO] 11:52:56.583 Analysis report generated in 128ms, dir size=212.9 kB
[INFO] 11:52:56.629 Analysis report compressed in 44ms, zip size=29.9 kB
[INFO] 11:52:56.660 Analysis report uploaded in 29ms
[INFO] 11:52:56.662 ANALYSIS SUCCESSFUL, you can find the results at: http://98.93.222.66:9000/dashboard?id=webcal
[INFO] 11:52:56.663 Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] 11:52:56.664 More about the report processing at http://98.93.222.66:9000/api/ce/task?id=25db8891-014a-4e29-a837-ce04a94c67a8
[INFO] 11:52:56.689 Analysis total time: 14.118 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 19.716 s
[INFO] Finished at: 2025-10-21T11:52:56Z
[INFO]

```

8. View Analysis Results

After the analysis completes, open <http://<server-ip>:9000/projects> to view:



- Bugs
- Code Smells
- Vulnerabilities
- Coverage (if tests exist)

✓ You have successfully tested Java code using SonarQube without PostgreSQL or SonarScanner.