
🧾 SonarQube Installation & Java Application Code Analysis Guide

Document Purpose:

This document provides a complete step-by-step guide to install and configure **SonarQube** (using the built-in H2 database), connect it with a **Java-based application**, and perform **static code analysis** to identify bugs, vulnerabilities, and code smells.

🧱 1. Prerequisites

Component	Requirement
OS	Ubuntu 22.04 LTS / Amazon Linux 2 / Windows Server
RAM	Minimum 4 GB (8 GB recommended)
Java	Java 17 (LTS version required by SonarQube 9.x and above)
Database	**Embedded H2 Database** (default, no external DB setup required)

User Privileges	sudo privileges required
Application Code	Java-based project (Maven or Gradle build)

⚙️ 2. SonarQube Installation Steps

🧩 Step 1: Install Java

sudo apt update

sudo apt install openjdk-17-jdk -y

java -version

🧩 Step 2: Create a SonarQube User

sudo useradd -m -d /opt/sonarqube -r -s /bin/bash sonar

🧩 Step 3: Download and Configure SonarQube

cd /opt

sudo apt install wget unzip -y

sudo wget
https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.6.0.92116.zip

sudo unzip sonarqube-10.6.0.92116.zip

sudo mv sonarqube-10.6.0.92116 sonarqube

sudo chown -R sonar:sonar /opt/sonarqube

Edit the SonarQube configuration file:

sudo nano /opt/sonarqube/conf/sonar.properties

Update configuration to use the **embedded H2 database** (default).
Ensure these lines are **commented out**:

sonar.jdbc.username=

sonar.jdbc.password=

sonar.jdbc.url=

Enable external access if required:

sonar.web.host=0.0.0.0

sonar.web.port=9000

Save and exit (`CTRL+O`, `CTRL+X`).

🧩 Step 4: Create a SonarQube System Service

sudo nano /etc/systemd/system/sonarqube.service

Paste the following configuration:

[Unit]

Description=SonarQube service

After=network.target

[Service]

Type=forking

User=sonar

Group=sonar

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

Restart=always

LimitNOFILE=65536

LimitNPROC=4096

[Install]

WantedBy=multi-user.target

Enable and start SonarQube:

sudo systemctl daemon-reload

sudo systemctl enable sonarqube

sudo systemctl start sonarqube

sudo systemctl status sonarqube

🧩 Step 5: Access the SonarQube Dashboard

Open your web browser and navigate to:

http://<server-ip>:9000

Default credentials:

Username: admin

Password: admin

You'll be prompted to change the password on first login.

💡 3. Configure Java Application for SonarQube Analysis

Assume you have a **Maven-based Java project** (e.g., *Currency Converter*).

🧩 Step 1: Install Sonar Scanner

sudo apt install unzip -y

cd /opt

sudo wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-6.2.0.61181-linux.zip

sudo unzip sonar-scanner-6.2.0.61181-linux.zip

sudo mv sonar-scanner-6.2.0.61181-linux sonar-scanner

Add the scanner to your PATH:

sudo nano /etc/profile.d/sonar-scanner.sh

Add:

export PATH=$PATH:/opt/sonar-scanner/bin

Apply and verify:

source /etc/profile.d/sonar-scanner.sh

sonar-scanner -v

🧩 Step 2: Generate a Sonar Token

Login to SonarQube → **My Account** → **Security** → **Generate Tokens**

- Name: `maven-analysis`
- Copy and store the token safely.

🧩 Step 3: Add Sonar Configuration to Maven Project

In your project root, create a file named:

sonar-project.properties

Add:

sonar.projectKey=CurrencyConverter

sonar.projectName=Currency Converter Java App

sonar.projectVersion=1.0

sonar.sources=src

sonar.language=java

sonar.java.binaries=target/classes

sonar.host.url=http://<sonarqube-server-ip>:9000

sonar.login=<your-generated-token>

🧩 Step 4: Build and Run Sonar Analysis

Option 1 — Using Sonar Scanner:

mvn clean install

sonar-scanner

Option 2 — Directly via Maven:

Example using your token and server IP (`http://18.218.37.139:9000`):

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=my-java-app \
  -Dsonar.host.url=http://18.218.37.139:9000 \
  -Dsonar.login=squ_c2e738e7a1dd54a92d75a005513b313fe21e656c
```

🔍 4. Verify SonarQube Analysis Results

Login to the SonarQube dashboard → **Projects** → **Your App**
You'll find metrics such as:

- ✅ **Code Smells**
- 🐞 **Bugs**
- 🔐 **Vulnerabilities**
- 📊 **Duplications**
- 🧪 **Test Coverage** (if tests are included)

⚙️ 5. Optional — Integrate SonarQube with Jenkins CI/CD

Jenkins Plugin Setup:

1. Go to **Manage Jenkins → Manage Plugins → Available**
2. Search for **SonarQube Scanner**
3. Install & restart Jenkins

Configure Jenkins:

1. **Manage Jenkins → Configure System**
2. Add **SonarQube Server URL**
3. Add **Authentication Token**

Add Stage in Jenkinsfile:

```
stage('SonarQube Analysis') {
   steps {
      script {
         withSonarQubeEnv('SonarQubeServer') {
            sh 'mvn clean verify sonar:sonar'
         }
      }
```

```
  }
}
```

📊 6. Common Troubleshooting

Issue	Possible Cause	Fix
SonarQube not starting	Insufficient memory	Increase VM RAM (≥ 4GB)
“Java not found”	Wrong JAVA_HOME	Export correct JAVA_HOME
Scanner fails	Invalid token or wrong URL	Recheck token and SonarQube URL
Port 9000 busy	Another service using it	Change `sonar.web.port` in `sonar.properties`

✅ 7. Validation Checklist

Step	Description	Status

Java Installed	`java -version`	✅
SonarQube Running	`systemctl status sonarqube`	✅
Web Access	`http://<ip>:9000`	✅
Sonar Scanner Installed	`sonar-scanner -v`	✅
Java Project Analyzed	Results visible in dashboard	✅

🧠 8. Summary

SonarQube provides deep **static code analysis** for continuous improvement in **code quality, security, and maintainability**.

Using its **embedded database** setup, you can quickly deploy SonarQube for **training, testing, and CI/CD integrations** — without requiring an external PostgreSQL instance.

It seamlessly integrates with:

- ☕ **Maven / Gradle**
- 🧩 **Jenkins / CI pipelines**

- 🧠 **Developers' workflows** via real-time feedback.