

SonarQube 9.x

By

Anand Kulkarni

anand.kulkarni1@zensar.com

Table of Content

Module	Topic
Module 1:	Static Code Analysis
Module 2:	Introduction to SonarQube
Module 3:	SonarQube Architecture
Module 4:	SonarQube setup in SpringBoot app
Module 5:	SonarLint plugin
Module 6:	Code coverage with JaCoCo

Static Code Analysis

1. Static code analysis is a method of debugging by examining source code before a program actually runs.
2. It's done by analyzing a set of code against a set of coding rules.
3. Static code analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual code reviews. But using automated tools is much more effective.
4. Static code analysis is performed early in development, before software testing begins. Developers will know early on if there are any problems in their code. And it will be easier to fix those problems.

Benefits of static code analysis

- 1) ***Speed:*** It takes time for developers to do manual code reviews. Automated tools are much faster.
- 2) ***Depth:*** Testing can't cover every possible code execution path. But a static code analyzer can.
- 3) ***Accuracy:*** Manual code reviews are prone to human error. Automated tools are not.

Introduction to SonarQube

- 1) SonarQube is an open source platform developed by SonarSource for continuous inspection of code quality.
- 2) SonarQube performs automatic reviews.
- 3) It performs static code analysis to detect bugs, code smells and security vulnerabilities on 20+ programming languages.
- 4) It gives reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs & security vulnerabilities.
- 5) Find more information at <https://www.sonarqube.org/>

Installation & Running

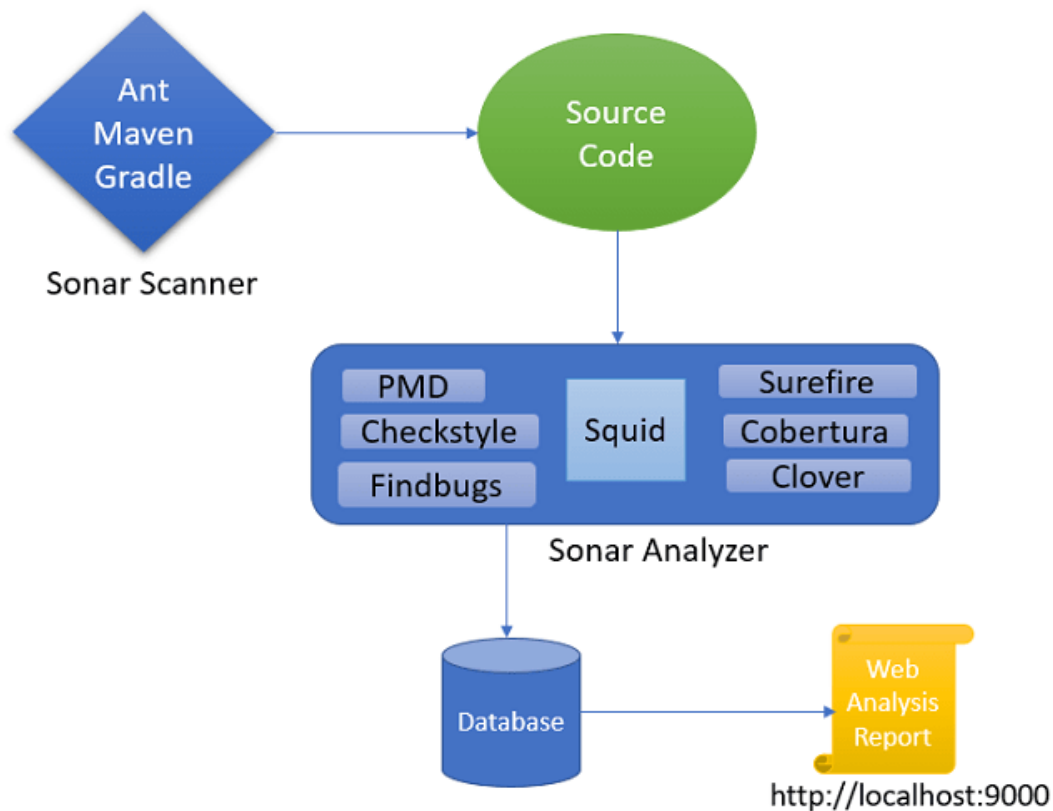
- 1) Please make sure you have JDK 11 or above. If you have lower version then download SonarQube <=7.8 otherwise download the latest version.
- 2) Download & extract the SonarQube zip file at root location say c:\ using <https://www.sonarqube.org/downloads/>
- 3) `cd C:\sonarqube-9.2.4.50792\sonarqube-9.2.4.50792\bin\windows-x86-64`
- 4) Run StartSonar.bat & visit SonarQube admin console <http://localhost:9000>
- 5) Login as admin/admin. You may need to change the password after first login.

SonarQube supported databases

SonarQube by default uses internal database called H2 database. However, it can be configured with below supported databases:

- MS SQL Server
- PostgreSQL
- Oracle

SonarQube Architecture



SonarQube Architecture

SonarQube Architecture mainly have four components:

Sonar Scanner: SonarScanner is a separate client type application that in connection with the SonarQube server will run project analysis and then send the results to the SonarQube server to process it.

Source Code: Source code which need to be analyzed.

Sonar Analyzer: Sonar receives the request and starts analyzing the source code of the project.

SonarQube Database: When the analysis is finished, the results will be stored in the database for future reference and history tracking.

SonarQube setup in SpringBoot app

1. Add `sonar-project.properties` inside project root directory with below properties:

sonar.projectKey=product_app

sonar.projectName=Product App

sonar.projectVersion=1.0

sonar.sources=.

sonar.language=java

sonar.sourceEncoding=UTF-8

SonarQube setup in SpringBoot app continue..

2. Run following command on maven project where you have kept sonar-project.properties file:

```
mvn clean install sonar:sonar -Dsonar.login=admin -Dsonar.password=admin123
```

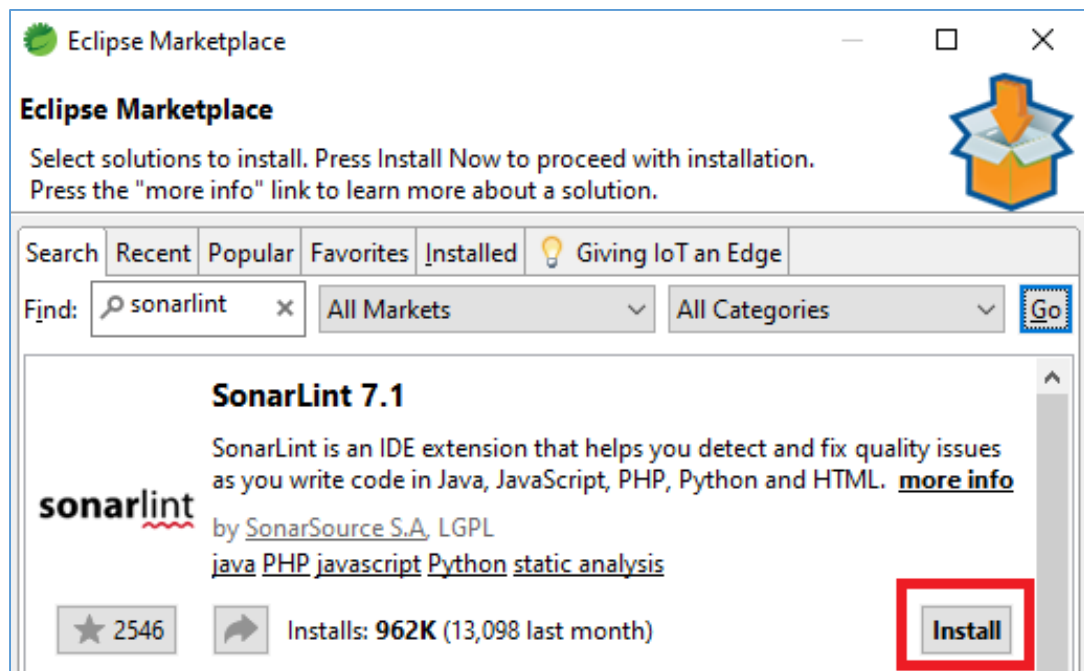
3. Now open the dashboard at <http://localhost:9000>. You will find the project 'Product App' is registered with detailed report on Bugs, Code smells, vulnerabilities etc.

SonarLint plugin

SonarLint plugin is an IDE extension that helps you to detect and fix quality issues as you write code. Like a spell checker, SonarLint highlights Bugs and Security Vulnerabilities as you write the code, with clear remediation guidance so you can fix them before the code is even committed.

SonarLint plugin installation

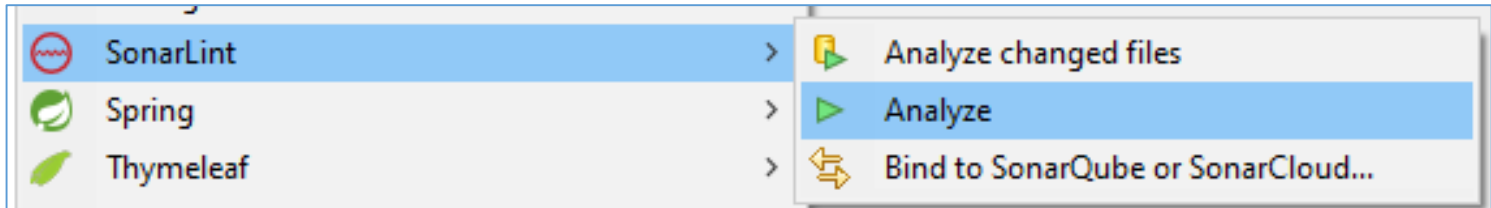
In order to install SonarLint plugin, please visit Help > Eclipse Marketplaces... option & install SonarLint plugin.



SonarLint plugin installation

After adding SonarLint plugin, you need to restart the IDE.

Now when you right-click on your project, you find SonarLint option. You can click on Analyze to analyze your current project code.



SonarLint supported IDEs

- 1) Eclipse
- 2) Spring Tool Suite (STS)
- 3) IntelliJ IDEA
- 4) Visual Studio
- 5) VS Code

Code coverage

Code coverage is measured by determining the number of lines that are run through unit tests. This is also called line coverage. Line coverage simply reports the percentage of lines of code executed by taking the number of executed lines over the total number of lines of code. For example, if a particular method is 10 lines long and the unit test ran 7 of the lines, then we would say the method has a line coverage of 70%.

JaCoCo

1. JaCoCo stands for Java Code Coverage.
2. JaCoCo is an open source toolkit for measuring code coverage in a code base and reporting it through visual reports.
3. JaCoCo can generate reports in HTML, XML & CSV formats.
4. The reports are published in the directory `/target/site/jacoco`.

Adding JaCoCo in Spring Boot app

Add JaCoCo plugin entry into pom.xml

```
<plugin>
<groupId>org.jacoco</groupId>
<artifactId>jacoco-maven-plugin</artifactId>
<version>0.8.7</version>
<executions>
<execution>
<goals>
<goal>prepare-agent</goal>
</goals>
</execution>
<execution>
<id>report</id>
<phase>test</phase>
<goals>
<goal>report</goal>
</goals>
</execution>
```

Adding JaCoCo in Spring Boot app









```
<execution>
<id>jacoco-check</id>
<goals>
<goal>check</goal>
</goals>
<configuration>
<rules>
<rule>
<element>PACKAGE</element>
<limits>
<limit>
<counter>LINE</counter>
<value>COVEREDRATIO</value>
<minimum>0.5</minimum>
</limit>
</limits>
</rule>
</rules>
</configuration>
</execution>
</executions>
</plugin>
```

Building JaCoCo Reports

1. Run the below maven command to generate JaCoCo report:

Mvn clean test

2. Now open the `target/site/jacoco/index.html` to see detailed report as shown below.

ProductApp												
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 com.product.controller		24%		0%	16	19	22	29	7	10	0	1
 com.product.dto		18%		0%	13	14	23	29	12	13	0	1
 com.product		37%		n/a	1	2	2	3	1	2	0	1
Total	204 of 264	22%	20 of 20	0%	30	35	47	61	20	25	0	3

Thank you!!