# SonarQube

-- > SonarQube use for code quality check.

-- > Code quality is an important part of the build process.

-- > SonarQube analyzes your application code and displays the results on sonar dashboard.

-- > Provide reports of duplicated code, coding standards, unit tests, integration tests,complex code, potential bugs, comments and design and architecture.

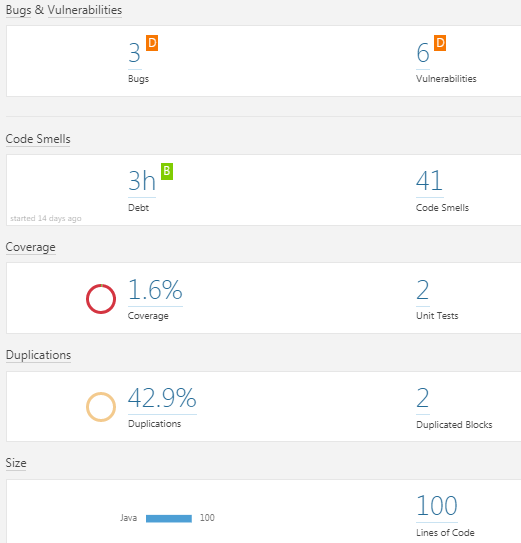
-- > SonarQube stores the results into a relational database.

-- > SonarQube by default embedded with H2 database.This is useful for testing purposes but not very scalable.

For a production environment, you will typically run Sonar on a real database such as MySQL or Postgres.

-- > SonarQube a great fit for working on Maven projects

-- > Sonar dashboard looks like below.



**Code Analysis:**

**---------------------**

In part of code analysis below things are captured

Coding standards

Duplication of code

Size of the code i.e Lines of Code

number of lines of code (NLOC),

number of lines per class

Dead code

Complex Expressions

Potential bugs

performance problems

Coding standards nothing but rules those defines the coding styles and conventions.

* code layout
* Formatting
* Naming conventions
* missing curly brackets after a condition

-- remember that coding standards are guidelines and recommendations not absolute rules.

-- Use failing code quality builds and code quality reports to improve the code.

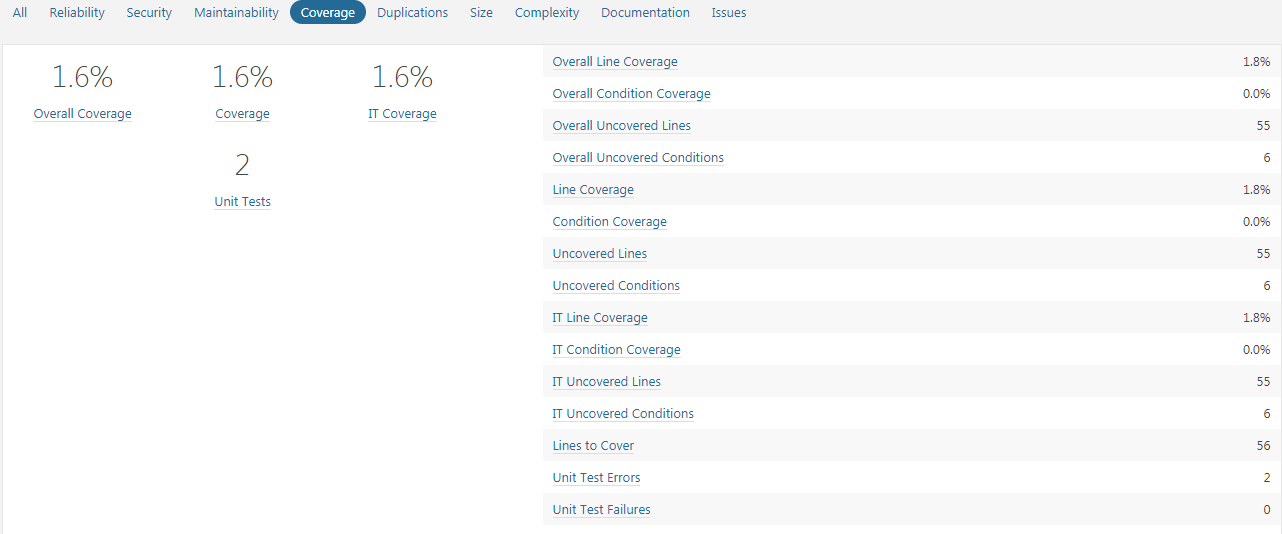
**Code Coverage**:

**----------------------**

**-->**  Code coverage is an indication of how much of your application code is actually executed during your tests

**-->**  It can be a useful tool in particular for finding areas of code that have not been tested by your test suites.

**-- >** It can also give some indication as to how well a team is applying good testing practices such as Test-Driven Development or Behavior-Driven Development.



**Benefits:**

--> High quality code contains less bugs, and is easier to understand and easier to maintain.

--> The code will be more readable for new developers on the team.

**Url to access sonar:**

<http://localhost:9000> sonar default port is 9000

Credentials : admin/admin

**Commands:**

mvn sonar:sonar // do only code analysis

mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install sonar:sonar // code analysis + code coverage.

clean checkstyle:checkstyle pmd:pmd findbugs:findbugs cobertura:cobertura javadoc:javadoc

**Required Configuration:**

--> Required jdk-1.8 and maven 3.3 and later

--> Database. H2,Mysql,Oracle.

--> Sonar host details as to provide.

In setting.xml or pom.xml or maven target

Adding sonar host details in setting.xml

<profiles>

<profile>

<id>sonar</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<!-- Optional URL to server. Default value is http://localhost:9000 -->

**<sonar.host.url>**

**http://localhost:9000**

**</sonar.host.url>**

</properties>

</profile>

</profiles>

Adding sonar host details in pom.xml

<properties>

**<!-- SONAR SERVER ON A REMOTE HOST -->**

**<sonar.host.url>http://ipaddr:9000</sonar.host.url>**

</properties>

Adding sonar host details in maven targets

**mvn sonar:sonar -Dsonar.host.url=http://ipaddr:9000/**

**Installation :**

Run below command to run sonarqube image.

**docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube**

From second time, to start sonar use below command

**docker start sonarqube**

Before to run sonarqube should be install docker.

**Steps to Install docker**

1 sudo su

2 yum update -y

3 yum install docker

4 docker info

5 service docker start

6 chkconfig docker on

7 docker --version

**Code Quality:**

-- > High quality code contains less bugs, and is easier to understand and easier to maintain.

-- > Only an experienced developer can really judge code quality in all its aspects.

-- > The code will also be more readable for new developers arriving on the team.

-- > sonar is the role of code reviews.

-- > Sonar tools analyze your application source code

-- > In particular, only a human eye can decide if a piece of code is truly well written.

-- > Sonar mainly focus on more sophisticated static analysis.

-- > The aim of this process is to produce project-wide code quality metrics

-- > poor practices in your code.

-- > looking for potential bugs.

-- > Java static analysis tools

Checkstyle,

PMD,

FindBugs,

Cobertura, and JDepend.

fxcop and NCover, are focused on .NET applications.

The notable exception to this rule is Sonar. Sonar can extract code quality metrics from any Maven project, with no additional configuration required in your Maven project.

-- provide feedback for developers on code quality issues.

-- In addition to your normal unit and integration test build jobs, set up a dedicated code quality build, which runs after the normal build and test.

-- It is important to run this job separately, because code coverage analysis and many static analysis tools can be quite slow to run.

-- set up a dedicated code quality build, which runs after the normal build and test, and configure the build to fail if code quality metrics are not at an acceptable level.

In the Java world, three static analysis tools have

-- Checkstyle excels at checking coding standards and conventions, coding

practices, as well as other metrics such code complexity.

-- PMD is a static analysis tool similar to Checkstyle, more focused on coding and design practices.

-- FindBugs is an innovative tool focuses on identifying potentially dangerous and buggy code.

**Checkstyle**:

Checkstyle1 is a static analysis tool for Java. Originally designed to enforce a set of highly-configurable

coding standards, Checkstyle now also lets you check for poor coding practices, as well as overly

complex and duplicated code.

Checkstyle supports a very large number of rules, including ones relating to naming conventions,

annotations, javadoc comments, class and method size, code complexity metrics, poor coding practices,

and many others.

Checkstyle provides some support for the detection of duplicated code, but

more specialized tools such as CPD do a better job in this area

PMD3 is another popular static analysis tool. It focuses on potential coding problems such as unused

or suboptimal code, code size and complexity, and good coding practices. Some typical rules include

“Empty If Statement,” “Broken Null Check,” “Avoid Deeply Nested If Statements,” “Switch Statements

Should Have Default,” and “Logger Is Not Static Final.” There is a fair amount of overlap with some

of the Checkstyle rules, though PMD does have some more technical rules, and more specialized ones

such as rules related to JSF and Android.

**PMD** also comes with CPD, a robust open source detector of duplicated and near-duplicated code.

CPD- copy paste detector.

PMD scans Java source code and looks for potential problems

Possible bugs - empty try/catch/finally/switch statements

Dead code - unused local variables, parameters and private methods

Suboptimal code - wasteful String/StringBuffer usage

complicated expressions - unnecessary if statements, for loops that could be while loops

Duplicate code - copied/pasted code means copied/pasted bugs

**FindBugs** can detect some fairly significant issues such as null pointer exceptions, infinite loops, and unintentionally accessing the internal state of an object

Reporting on Code Complexity:

complex code typically has large numbers of nested conditional statements and loops, which

make the code harder to understand and to debug.

set up code quality reporting in your Jenkins builds

Code Quality in Your Build Process

Popular Java and Groovy Code Quality Analysis Tools

It uses several Maven plugins (Checkstyle, PMD, FindBugs,

Cobertura or Clover, and others) to analyse Maven projects and generate a comprehensive set of code

quality metrics reports

Sonar reports on code coverage, rule compliance, and documentation, but also

on more high-level metrics such as complexity, maintainability and even technical debt.

In one of the most common usages of Sonar, Sonar automatically runs a set of Maven code quality related plugins against your Maven project, and stores the results into a relational database