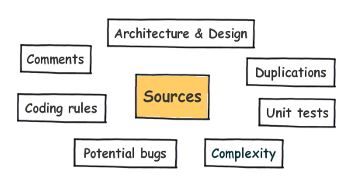
**SonarQube** (formerly **Sonar**) is an open platform to manage code quality.

As such, it covers the 7 axes of code quality:



**SonarQube Setup:**

1. Download Sonar server from:

<http://www.sonarqube.org/downloads/>

( here we are using **sonarqube-4.1.1.zip**)

1. Extract the content

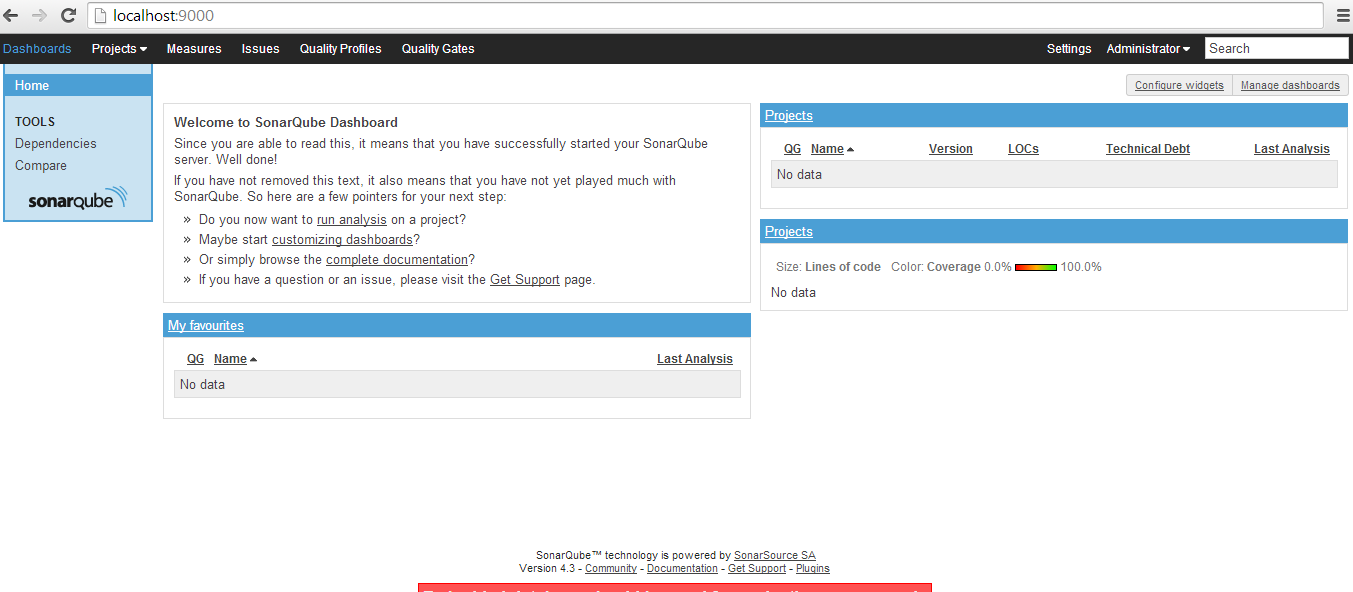
( like d:\ sonarqube-4.1.1)

1. Go to d:\sonarqube-4.1.1\bin\windows-x86-32

Double click on **StartSonar.bat**

it will start Sonar at **localhost:9000** .By default it starts at port 9000 we can change it by editing sonar.web.port=9000 property in d:\sonarqube-4.1.1\conf\sonar.properties.

( here we are using 32bit version of StartSonar.bat , you can also use 64-bit version which is located in d:\sonarqube-4.1.1\bin\windows-x86-64)



**Running Analysis**

First, you should install the plugin(s) for the language(s) of the project to be analyzed, either by [a direct download](http://docs.codehaus.org/display/SONAR/Plugin+Library) or through the [update center](http://docs.codehaus.org/display/SONAR/Update+Center).

Then, you need to choose an analysis method. The following are available:

* [Analyzing with SonarQube Runner](http://docs.codehaus.org/display/SONAR/Analyzing+with+SonarQube+Runner) (recommended analyzer)
* [Analyzing with SonarQube Ant Task](http://docs.codehaus.org/display/SONAR/Analyzing+with+SonarQube+Ant+Task)
* [Analyzing with Maven](http://docs.codehaus.org/display/SONAR/Analyzing+with+Maven)
* [Analyzing with Gradle](http://docs.codehaus.org/display/SONAR/Analyzing+with+Gradle)
* [CI Engines](http://docs.codehaus.org/display/SONAR/Continuous+Integration)

**Analyzing with Maven:**

You need to add the following plugin in pom.xml

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>sonar-maven-plugin</artifactId>

<version>2.1</version>

</plugin>

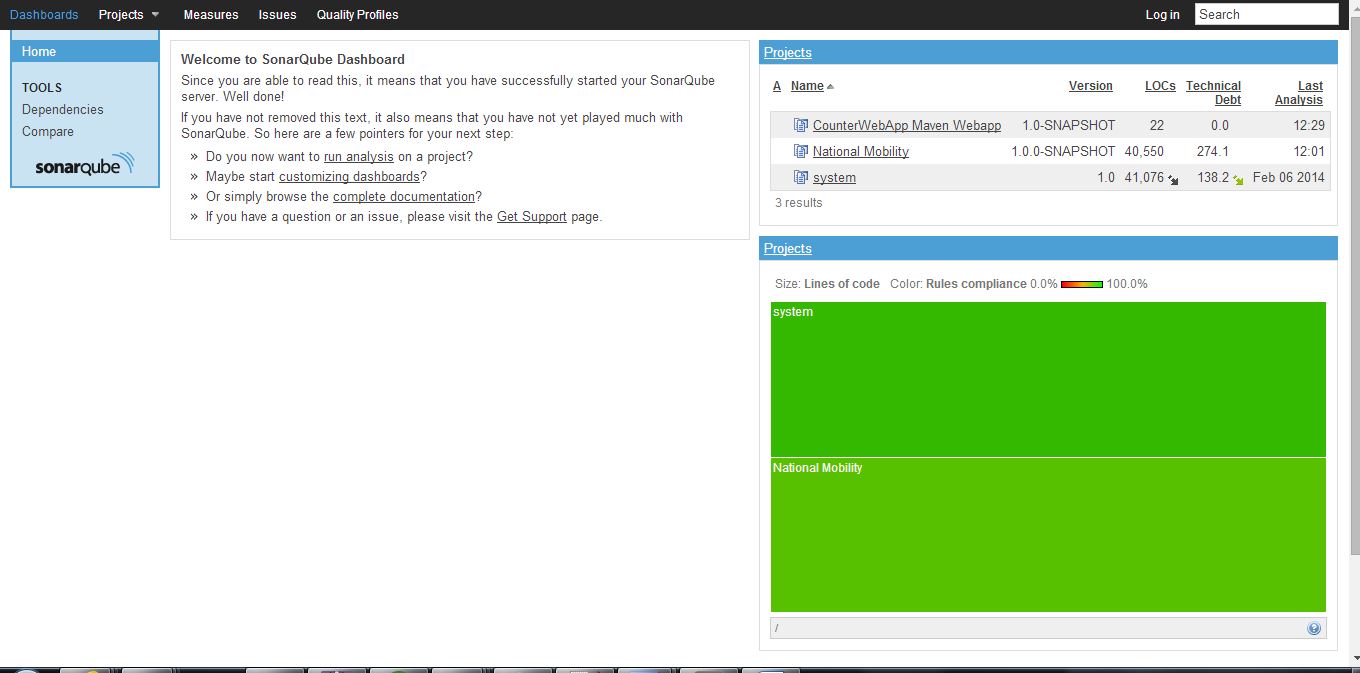
Analyzing a Maven project consists of running a Maven goal: sonar:sonar in the directory where the pom.xml file sits.

**mvn clean install**

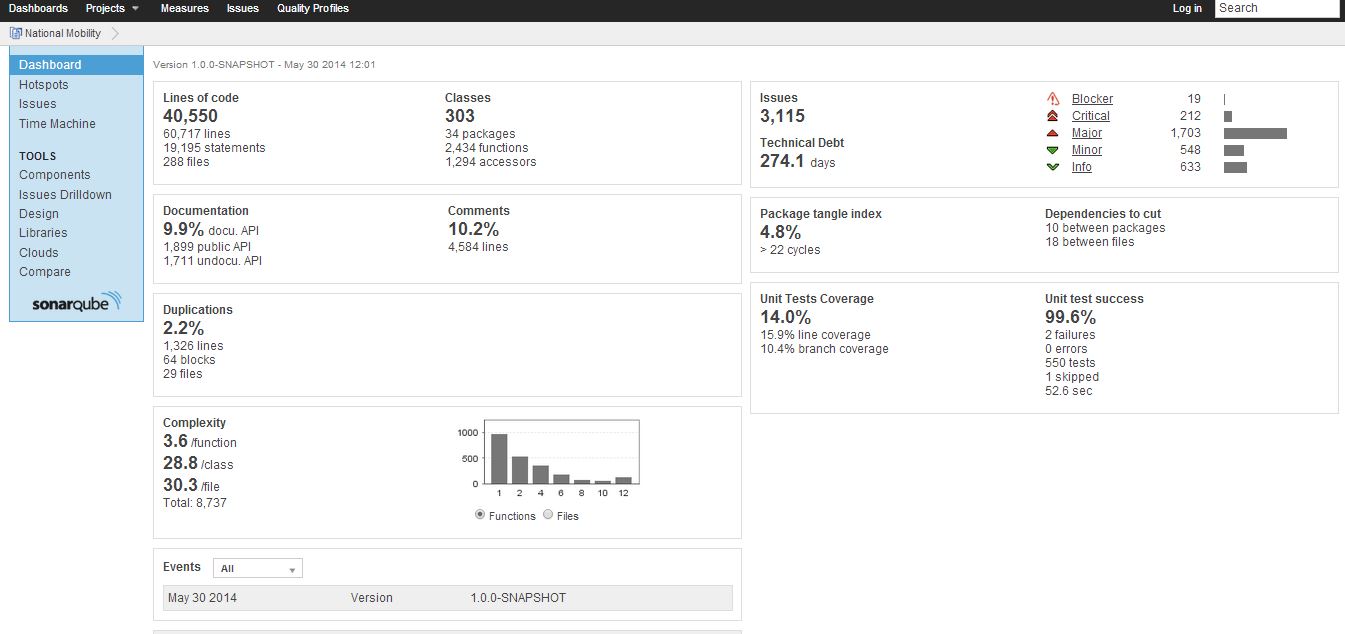
**mvn sonar:sonar**

Now if you hit <http://localhost:9000> , we will find all the analysis details of the current project are deployed inside SonarQube server.

**Snapshot of Dashboard :**



Now if we select our project from dashboard we will see the following screen which contains the analysis details of our project like Line of code, different issues, unit test coverage etc.



In above approach maven will look for Sonarqube in <http://localhost:9000> by default, but if sonar is hosted on some other port or ip then

mvn sonar:sonar will fail with the following message:

SonarQube server can not be reached at http://loost:9000. Please check the parameter 'sonar.host.url'. Connection refused.

To solve this problem you can add the following property in pom.xml.

<properties>

<sonar.host.url>

http://localhost:9000

</sonar.host.url>

</properties>

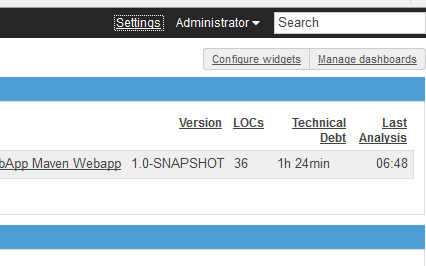
Here in **sonar.host.url** propertywe are setting url of **SonarQube** server **.**

**Configuring your favorite code-coverage tool**

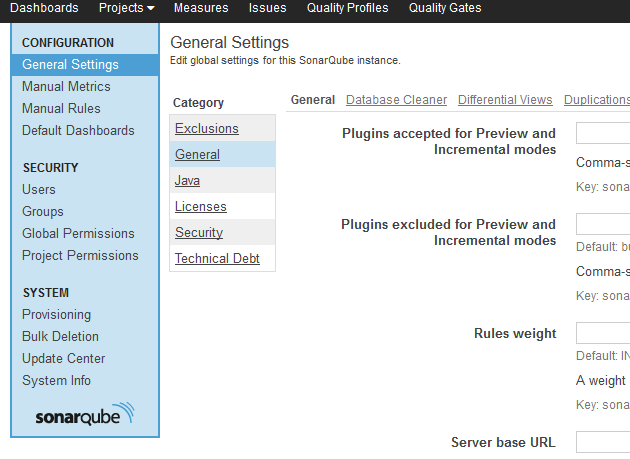
SonarQube’s default code-coverage tool for Java projects is JaCoCo and there are plugins to support other code coverage tools.

Lets install cobertura plugin

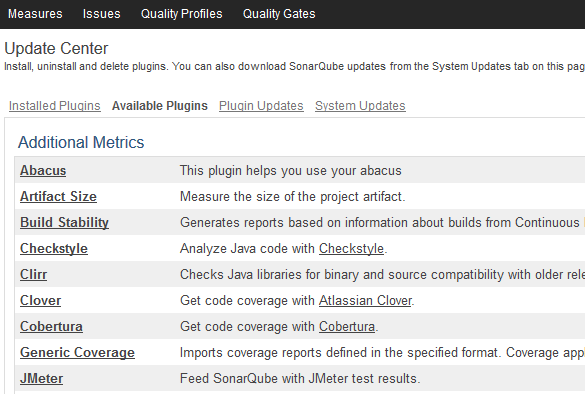
**Step 1** : click on settings at top right



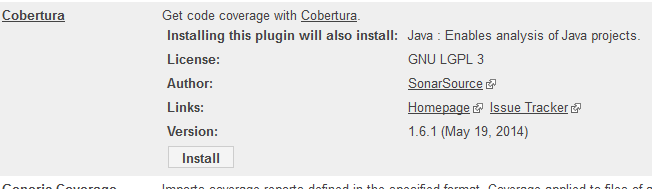
**Step 2**: Goto Update Center under SYSTEM menu



**Step 3**: Select Available Plugins tab, here you will see list of all available plugins. Select cobertura from this list.

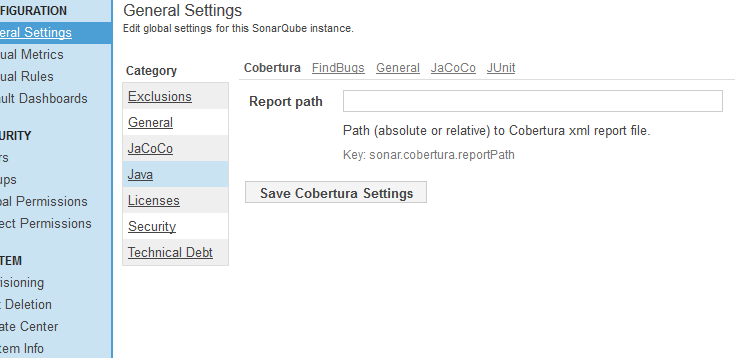
****

**Step 4**: Click install button

****

**Step 5**: Restart sonar server. Now you will find a new jar named sonar-cobertura-plugin-X.X.X.jar inside SONAR\_HOME\extensions\plugins. Plugin installation is complete now.

Now go to Settings at top right and select Java option under Category, here you will find a new tab for Cobertura which we just installed.

****

**Configure sonar with cobertura in maven project**

In pom.xml add the following

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<formats>

<format>html</format>

<format>xml</format>

</formats>

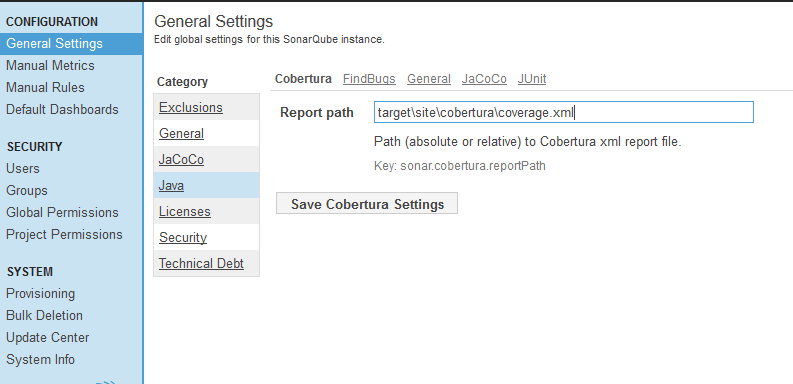
</configuration>

</plugin>

Now if you run : **mvn cobertura:cobertura**

It will generate type of code coverage report one in the form of xml and the other one in html. You can find those report inside **target\site\cobertura in our project.**

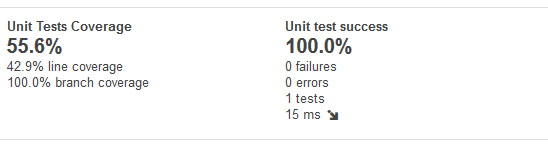
To integrate cobertura with sonarqube we have to give the path of generated xml report in sonarqube, so that sonarqube can display results generated by cobertura. As shown in the following figure.



Add **<sonar.java.coveragePlugin>cobertura</sonar.java.coveragePlugin>** property in pom.xml, this property tells which coverage tool sonar should pick. If we don’t give this property by default sonar will use jacoco.

Now we can run : **mvn sonar:sonar**

Now we can see the report generated by cobertura as shown below



By default cobertura will make coverage analysis on every java classes. We can configure cobertura to skip certain classes as follows

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<formats>

<format>html</format>

<format>xml</format>

</formats>

<instrumentation>

<excludes>

**<exclude>com/\*</exclude>**

</excludes>

</instrumentation>

</configuration>

</plugin>

Here  **<exclude>com/\*</exclude>** tells cobertura to skip all the clases inside com package and its sub packages

# [Quality Profiles](http://docs.codehaus.org/display/SONAR/Quality+Profiles)

The Quality Profiles service is the heart of SonarQube, since it is where you define your requirements by defining sets of **coding rules** (ex: Methods must not have a complexity greater than 10).

**TODO**