Code Management in DevOps – Introduction to SonarQube

This worksheet covers the following topics to help you understand how to perform static code analysis using SonarQube.

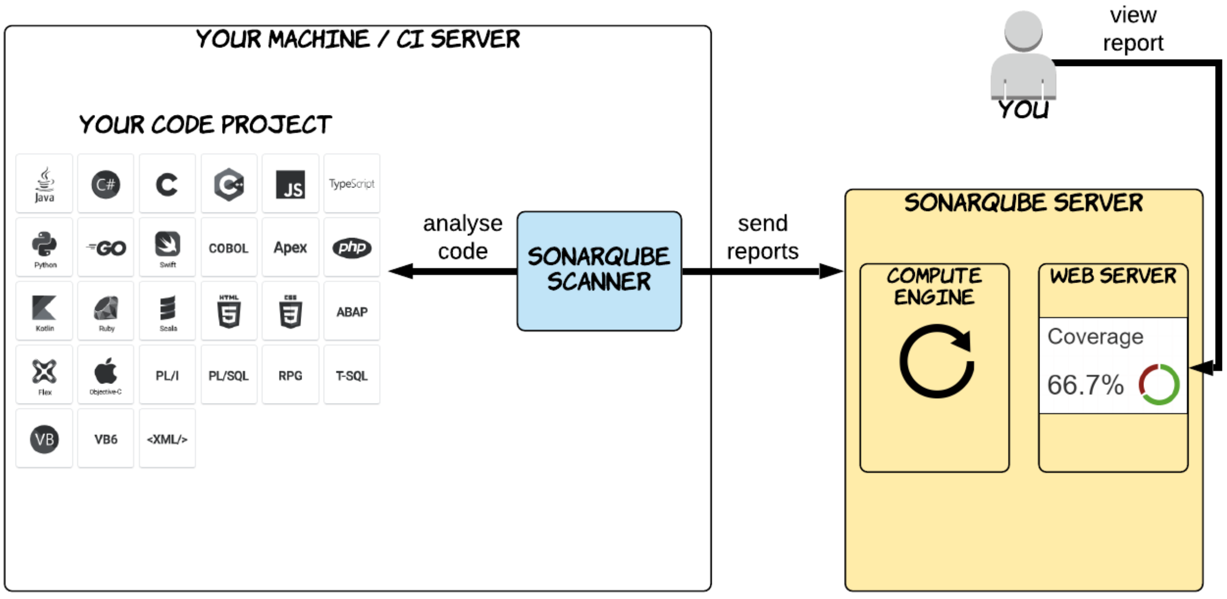
- Setup and **configure locally hosted SonarQube server**
- Identify how to use the **SonarLint** plugin and **practice code as you clean** using Eclipse IDE
- Examine **SonarQube dashboard** and identify the different components

Part 1: Step by Step guide to set up SonarQube server

A. What is SonarQube?

As a refresher, SonarQube server is a standalone service which allows you to browse reports from all the different projects which have been scanned. To scan a specific codebase you run the SonarQube scanner. This is a local process that analyses your code then sends reports to the SonarQube server.

SonarQube is an open-source code quality assurance tool that performs static code analysis and produces reports used for **increasing code reliability**, helps to **reduce technical debts,** and helps to **re-enforce application security** for a project. For this exercise, we will be experiencing how to set up and configure a locally hosted SonarQube server and use **various plugins** to help us apply the **clean as you code approach**.



Reference: https://docs.sonarqube.org/latest/

To introduce SonarQube into a project, we will first have to set up a SonarQube server locally. For this exercise, we will use the build automation tool Maven to manage the dependencies required automatically.

For this project, we will also introduce JaCoCo, an open-source toolkit for **measuring code coverage** in a code base and reporting it through visual reports. JaCoCo will be in charge of computing the code coverage metric, analyses the code and generates an XML report, which is later ingested by SonarQube. SonarQube will perform the consolidation and produce the **SonarQube reporting dashboard**.

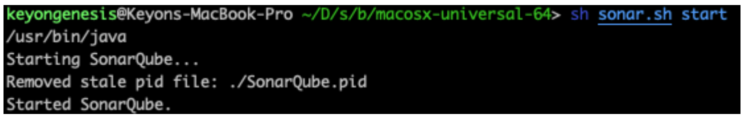
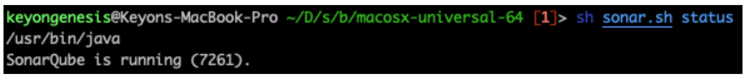
Different ways of Integrating SonarQube

- You can install a plugin — Sonarlint in Eclipse or Intellij Idea, and get your code analyzed even before pushing it to the central repository.
- You can also create a Jenkins Job, and trigger it on new code push. It will take code from your code base and will analyze them.
- You can have a local instance of SonarQube running on your/ remote machine and using the Maven plugin you can analyze the code.
- There is one more way, using Docker. Nowadays setting up any tool is easy with Docker. We will see this in the subsequent module.

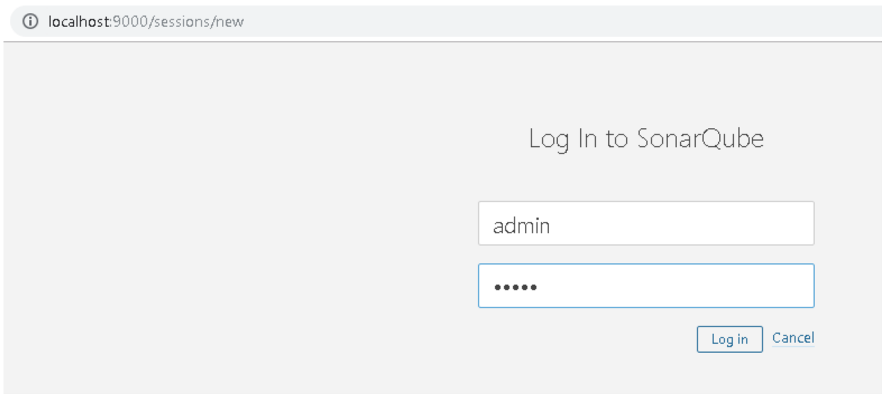
B. Installation of and setting up of locally hosted SonarQube Server

1. Firstly, let us download SonarQube Community Edition.
    - For this exercise, we will be using SonarQube 7.8 version due to its compatibility with **Java 8**. We will **not** use the latest version for this module and will not deploy the server locally.
    - After you have downloaded the file, copy the .zip file into a working folder (for this example, we will be using the Documents folder).
    - Unzip the SonarQube folder and navigate to **bin >** (Depending on your OS version)

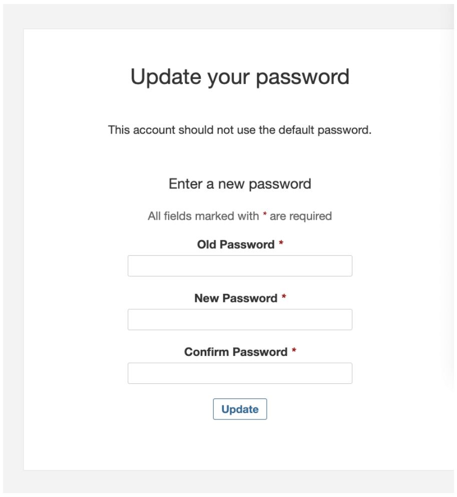
Windows	1. There will be a file, with the name StartSonar.bat. 2. Use this to launch SonarQube. (Windows firewall might attempt to block the process's execution, select allow all to start up the SonarQube application server. 3. In the command prompt there should be a success message, stating that "SonarQube is up". Note: You can access **conf > sonar.properties** for configuration purposes such as database, LDAP, webserver, SSO authentication, logging, port number, and many more. For this exercise, we will not be amending any configuration. By default, SonarQube runs on "http://localhost:9000". If you want to change the port, open sonar.properties file from config folder and update.

macosx	In the command prompt
	1. sh sonar.sh start
	
	`keyongenesis@Keyons-MacBook-Pro ~/D/s/b/macosx-universal-64> sh sonar.sh start`
	`/usr/bin/java`
	`Starting SonarQube...`
	`Removed stale pid file: ./SonarQube.pid`
	`Started SonarQube.`
	2. sh sonar.sh status
	
	`keyongenesis@Keyons-MacBook-Pro ~/D/s/b/macosx-universal-64 [1]> sh sonar.sh status`
	`/usr/bin/java`
	`SonarQube is running (7261).`

- Once the server started, log in to the SonarQube server on **http://localhost:9000** using the default username and password credentials:
    1. login: **admin**
    2. password: **admin**



You should see the screen below if you have successfully started and login to the SonarQube server. Where it will force the account to update the password.


Update your password

This account should not use the default password.

Enter a new password

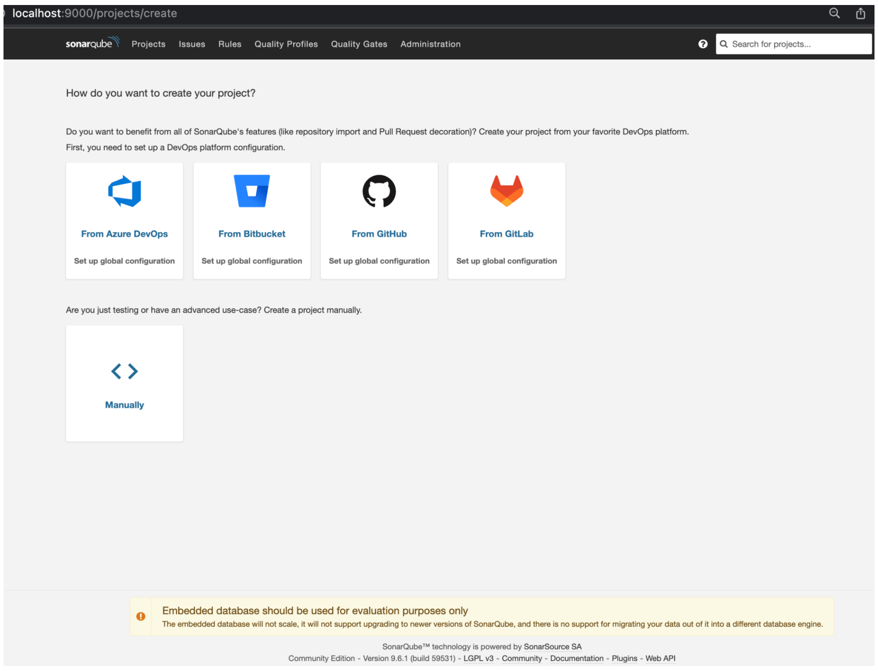
All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Currently the Home page after login will not have any information, as we are yet to Integrate our project with SonarQube.



How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.

From Azure DevOps
Set up global configuration
From Bitbucket
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration

Are you just testing or have an advanced use-case? Create a project manually.

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

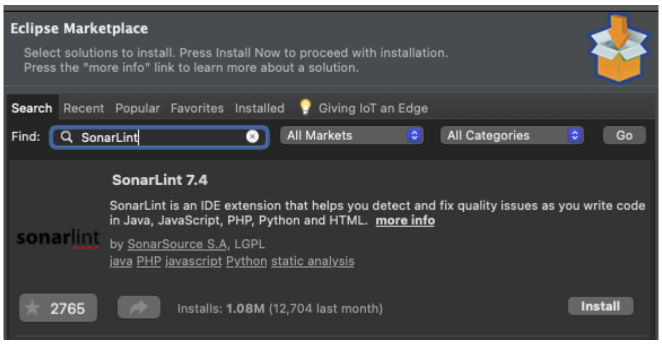
SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.6.1 (build 59531) - LGPL v3 - Community - Documentation - Plugins - Web API

- We will come back to the server later when we want to analyze our project. Now, let's install our SonarLint plugin and create our maven project using Eclipse IDE.

C. Identify how to use the SonarLint plugin and practice code as you clean using Eclipse IDE

SonarLint is an IDE extension that enables us to fix coding issues before they exist, similar to the concept of a spell checker. In Eclipse, we can make use of the Eclipse Marketplace to install the plugin for free.

1. Firstly, we will be creating a sample maven project using Eclipse for today's lab exercise.
2. Make use of the "maven-archetype-quickstart" archetype to create a maven project. You can refer to previous lab exercises on how to do so.
3. Enter **com.sddevops** as the Group Id and **sonarqube_maven.eclipse** as the Artifact Id. The package name should be **com.sddevops.sonarqube _maven.eclipse**
4. After you have created the project, we will walk through how to install the SonarLint plugin and retrieve the **required dependencies for SonarQube** from maven repositories to facilitate static code analysis for the project.
   - Firstly, we will install the plugin in Eclipse:
     i. Click on Help > Eclipse Marketplace
     ii. Search for "SonarLint"



Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

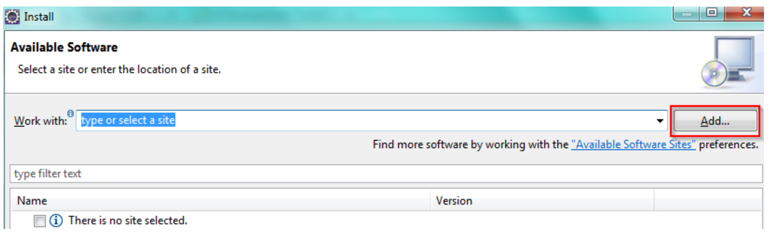
SonarLint 7.4

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code in Java, JavaScript, PHP, Python and HTML. more info

by SonarSource S.A, LGPL
java PHP javascript Python static analysis

     iii. Click on Install > Confirm (agree to the license agreement) and Finish to install the plugin.

     Note: If you are getting errors when trying to install sonarlint. Perhaps you are behind a corporate proxy and might have SSL issues. Also check java 11 as a runtime and the latest Eclipse version.
     Alternatively, you may download the latest version (zip). Followed by unzipping the file and Click on Help > Install New Software. Locate your file.



Install

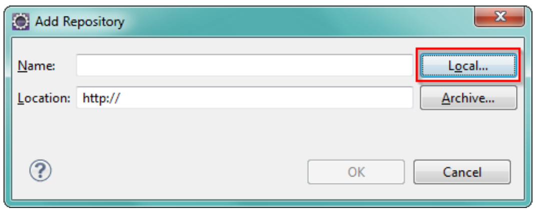
Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

There is no site selected.



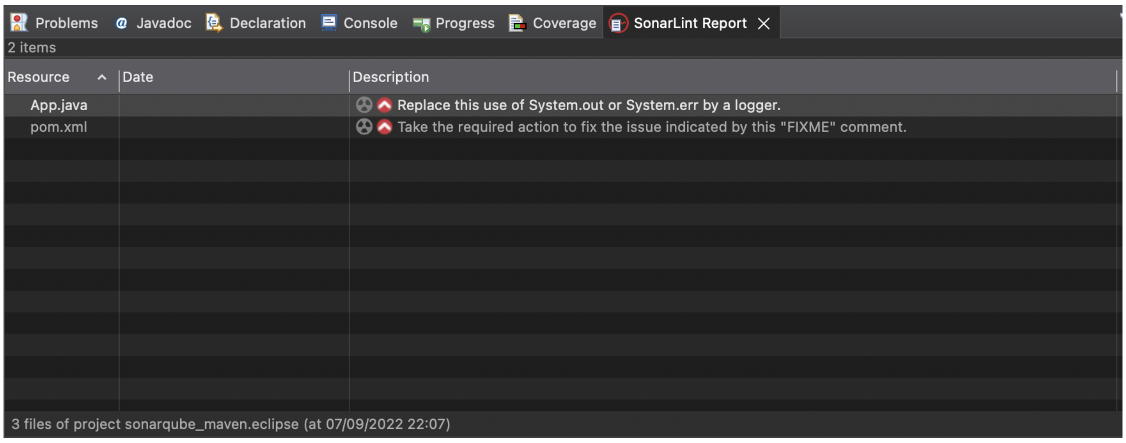
Add Repository

Name:

Location: http://

     iv. Once the installation is complete, a prompt should ask you to restart Eclipse IDE and **click on Restart.**

- After you have restarted Eclipse IDE, SonarLint should auto analyze your project; you can also right-click on your project or files you want to analyze and click **SonarLint > Analyze.**



2 items

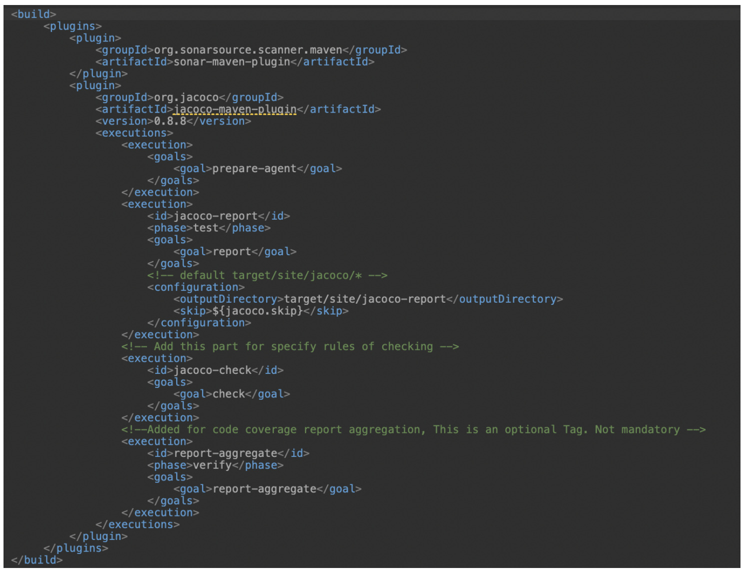
Resource	Date	Description
App.java		Replace this use of System.out or System.err by a logger.
pom.xml		Take the required action to fix the issue indicated by this "FIXME" comment.

3 files of project sonarqube_maven.eclipse (at 07/09/2022 22:07)

- You can see the **SonarLint Report using the report view** or when you **hover over your code**.
- Next, let us install our required dependencies for SonarQube. In the Package Explorer, expand all the folders in the shared project
- Open up the pom.xml file and **add** the necessary properties. You can find the maven repository metadata here:

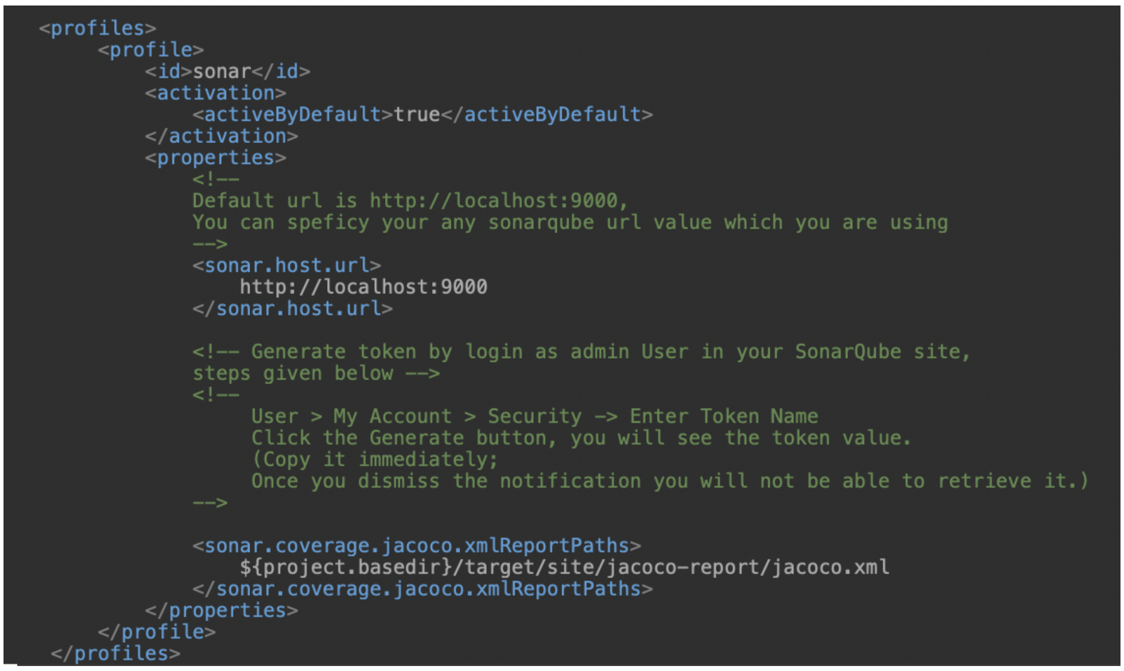
```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
 </properties>

 <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>5.8.2</version>
    </dependency>
    <dependency>
      <groupId>org.sonarsource.scanner.maven</groupId>
      <artifactId>sonar-maven-plugin</artifactId>
      <version>3.9.1.2184</version>
    </dependency>
  </dependencies>
```

- Next, let's add/amend the necessary plugins in the pom.xml file, dependencies section.



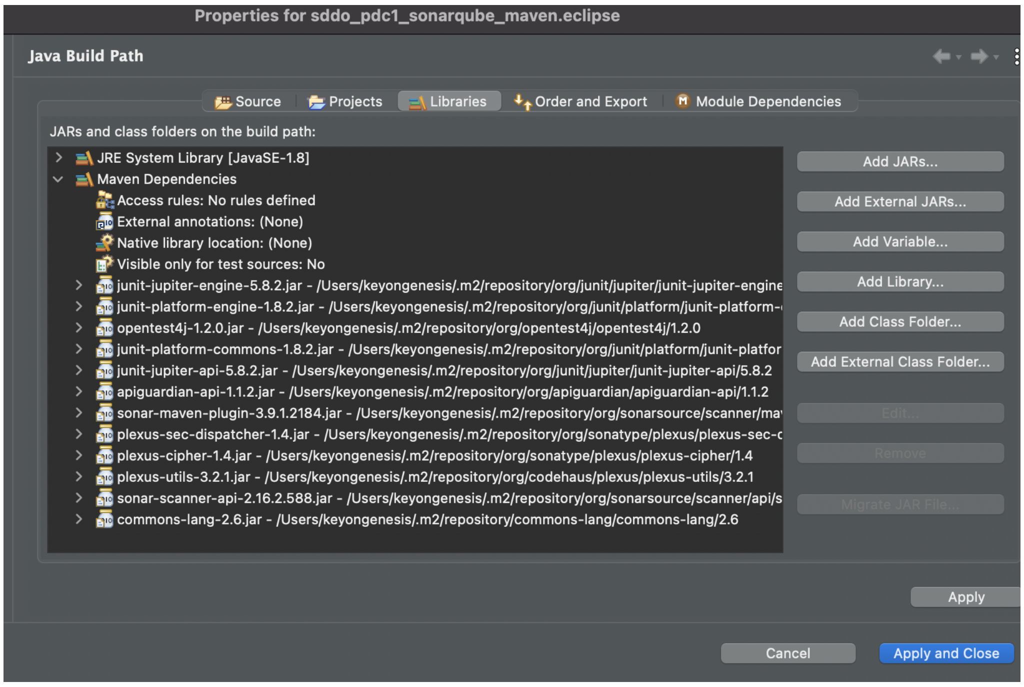
```
<build>
     <plugins>
         <plugin>
             <groupId>org.sonarsource.scanner.maven</groupId>
             <artifactId>sonar-maven-plugin</artifactId>
         </plugin>
         <plugin>
             <groupId>org.jacoco</groupId>
             <artifactId>jacoco-maven-plugin</artifactId>
             <version>0.8.8</version>
             <executions>
                 <execution>
                     <goals>
                         <goal>prepare-agent</goal>
                     </goals>
                 </execution>
                 <execution>
                     <id>jacoco-report</id>
                     <phase>test</phase>
                     <goals>
                         <goal>report</goal>
                     </goals>
                     <!-- default target/site/jacoco/* -->
                     <configuration>
                         <outputDirectory>target/site/jacoco-report</outputDirectory>
                         <skip>${jacoco.skip}</skip>
                     </configuration>
                 </execution>
                 <!-- Add this part for specify rules of checking -->
                 <execution>
                     <id>jacoco-check</id>
                     <goals>
                         <goal>check</goal>
                     </goals>
                 </execution>
                 <!--Added for code coverage report aggregation, This is an optional Tag. Not mandatory -->
                 <execution>
                     <id>report-aggregate</id>
                     <phase>verify</phase>
                     <goals>
                         <goal>report-aggregate</goal>
                     </goals>
                 </execution>
             </executions>
         </plugin>
     </plugins>
</build>
```

- Lastly, let's configure the plugin that integrates our Maven project with Sonar.



```
<profiles>
     <profile>
         <id>sonar</id>
         <activation>
             <activeByDefault>true</activeByDefault>
         </activation>
         <properties>
             <!--
             Default url is http://localhost:9000,
             You can speficy your any sonarqube url value which you are using
             -->
             <sonar.host.url>
                 http://localhost:9000
             </sonar.host.url>

             <!-- Generate token by login as admin User in your SonarQube site,
             steps given below -->
             <!--
                  User > My Account > Security -> Enter Token Name
                  Click the Generate button, you will see the token value.
                  (Copy it immediately;
                  Once you dismiss the notification you will not be able to retrieve it.)
             -->

             <sonar.coverage.jacoco.xmlReportPaths>
                 ${project.basedir}/target/site/jacoco-report/jacoco.xml
             </sonar.coverage.jacoco.xmlReportPaths>
         </properties>
     </profile>
 </profiles>
```

- After we have declared the required dependencies, save and refresh your project. Check the build path and see the libraries being added into the Maven dependency. You can do so by right-clicking on your project name and click on **Build Path > Configure the Build Path** for your project and check under the Libraries tab > Maven Dependencies of the newly opened window:



Properties for sddo_pdc1_sonarqube_maven.eclipse

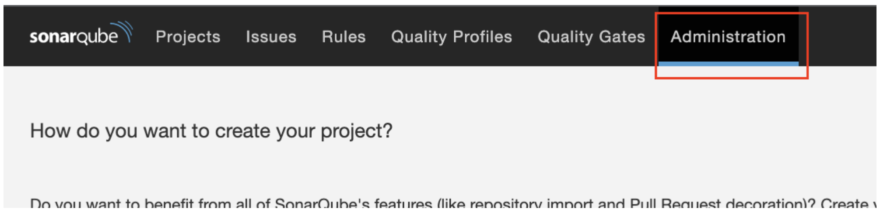
Java Build Path

JARs and class folders on the build path:

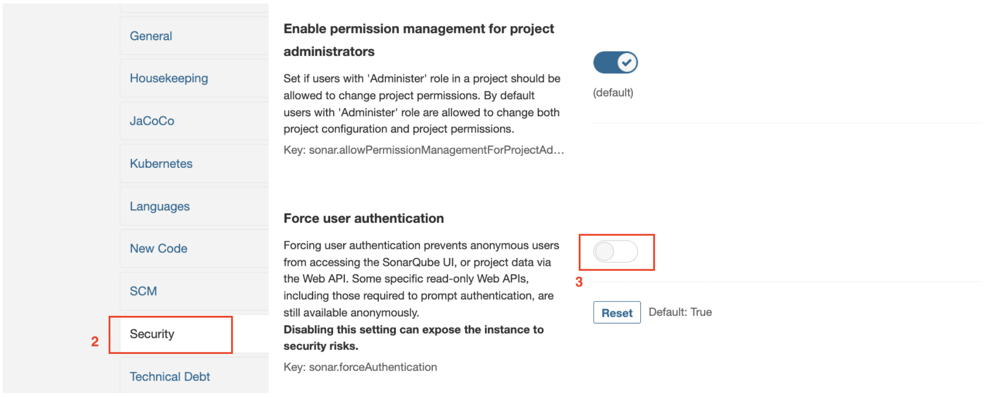
```
> JRE System Library [JavaSE-1.8]
v Maven Dependencies
    Access rules: No rules defined
    External annotations: (None)
    Native library location: (None)
    Visible only for test sources: No
  > junit-jupiter-engine-5.8.2.jar - /Users/keyongenesis/.m2/repository/org/junit/jupiter/junit-jupiter-engine
  > junit-platform-engine-1.8.2.jar - /Users/keyongenesis/.m2/repository/org/junit/platform/junit-platform-
  > opentest4j-1.2.0.jar - /Users/keyongenesis/.m2/repository/org/opentest4j/opentest4j/1.2.0
  > junit-platform-commons-1.8.2.jar - /Users/keyongenesis/.m2/repository/org/junit/platform/junit-platfor
  > junit-jupiter-api-5.8.2.jar - /Users/keyongenesis/.m2/repository/org/junit/jupiter/junit-jupiter-api/5.8.2
  > apiguardian-api-1.1.2.jar - /Users/keyongenesis/.m2/repository/org/apiguardian/apiguardian-api/1.1.2
  > sonar-maven-plugin-3.9.1.2184.jar - /Users/keyongenesis/.m2/repository/org/sonarsource/scanner/ma
  > plexus-sec-dispatcher-1.4.jar - /Users/keyongenesis/.m2/repository/org/sonatype/plexus/plexus-sec-c
  > plexus-cipher-1.4.jar - /Users/keyongenesis/.m2/repository/org/sonatype/plexus/plexus-cipher/1.4
  > plexus-utils-3.2.1.jar - /Users/keyongenesis/.m2/repository/org/codehaus/plexus/plexus-utils/3.2.1
  > sonar-scanner-api-2.16.2.588.jar - /Users/keyongenesis/.m2/repository/org/sonarsource/scanner/api/s
  > commons-lang-2.6.jar - /Users/keyongenesis/.m2/repository/commons-lang/commons-lang/2.6
```

Run **mvn test**, the JaCoCo code coverage report will be generated at target/site/jacoco/*

5. Go to sonarqube web page, then go to administration


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
How do you want to create your project?
Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create y

After that, go to security and disable " Force User Authentication".



Enable permission management for project administrators

Set if users with 'Administer' role in a project should be allowed to change project permissions. By default users with 'Administer' role are allowed to change both project configuration and project permissions.

Key: sonar.allowPermissionManagementForProjectAd...

(default)

Force user authentication

Forcing user authentication prevents anonymous users from accessing the SonarQube UI, or project data via the Web API. Some specific read-only Web APIs, including those required to prompt authentication, are still available anonymously.
Disabling this setting can expose the instance to security risks.

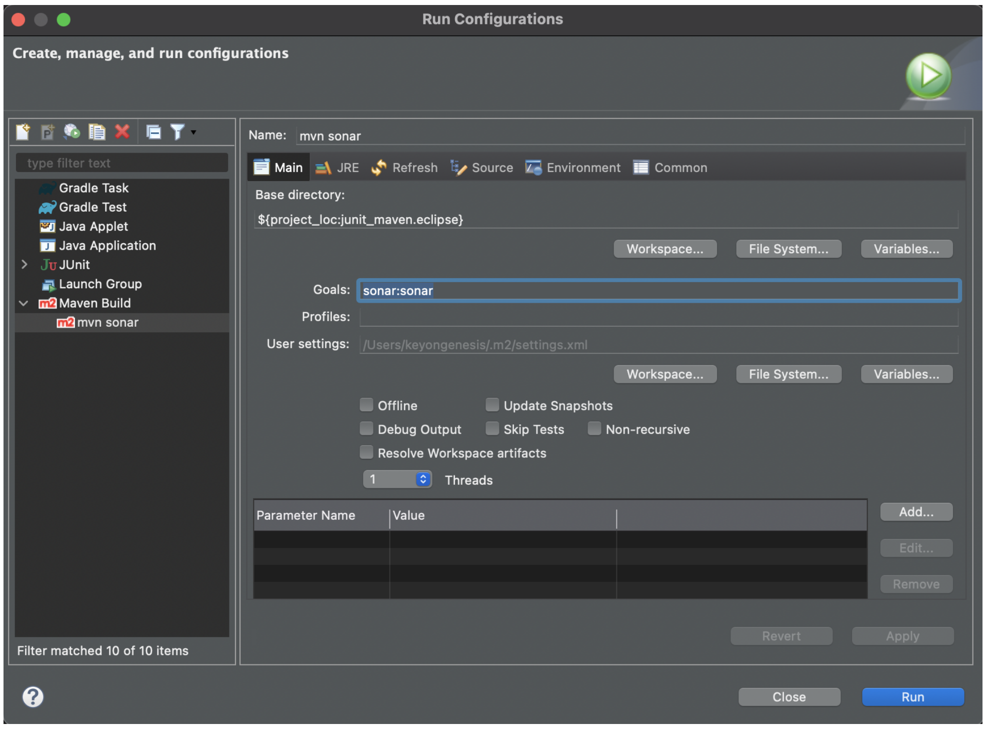
Key: sonar.forceAuthentication

Default: True

6. Now that we've defined the required dependency and plugin let us test it out by executing ***mvn clean Install*** to build our project and ***mvn sonar:sonar*** to perform the scan
   - Right-click on your project, run as Maven clean Install to build our project.

   Note: Ensure that you did not fail any unit test here, or the code coverage reports will not be generated!

   - Right-click on your project > Run As > Run Configuration; in the pop-up, right-click on Maven Build on the left-hand panel and select New Configuration. Input the Base directory as your project directory and Goals as ***sonar:sonar***
   - Select the **JRE tab** and **select Workspace default JRE (1.8).**
   - Click Apply > Run to start the Scanning.



Name: mvn sonar

Base directory: ${project_loc:junit_maven.eclipse}

Goals: sonar:sonar

Profiles:

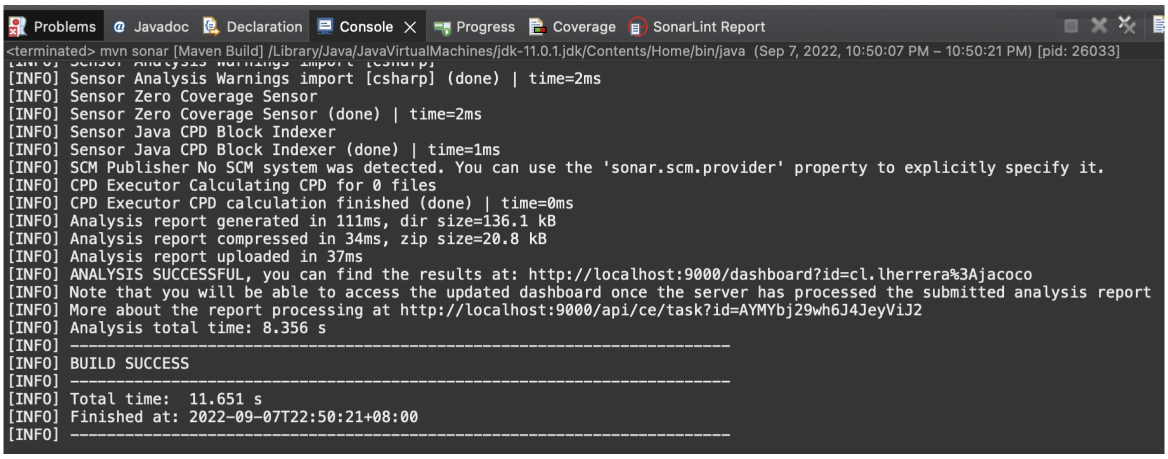
User settings: /Users/keyongenesis/.m2/settings.xml

- [ ] Offline
- [ ] Update Snapshots
- [ ] Debug Output
- [ ] Skip Tests
- [ ] Non-recursive
- [ ] Resolve Workspace artifacts

1 Threads

Parameter Name	Value

Once you see in the console "ANALYSIS SUCCESSFUL," your project has been scanned, and a link to your SonarQube report dashboard has been successfully generated.

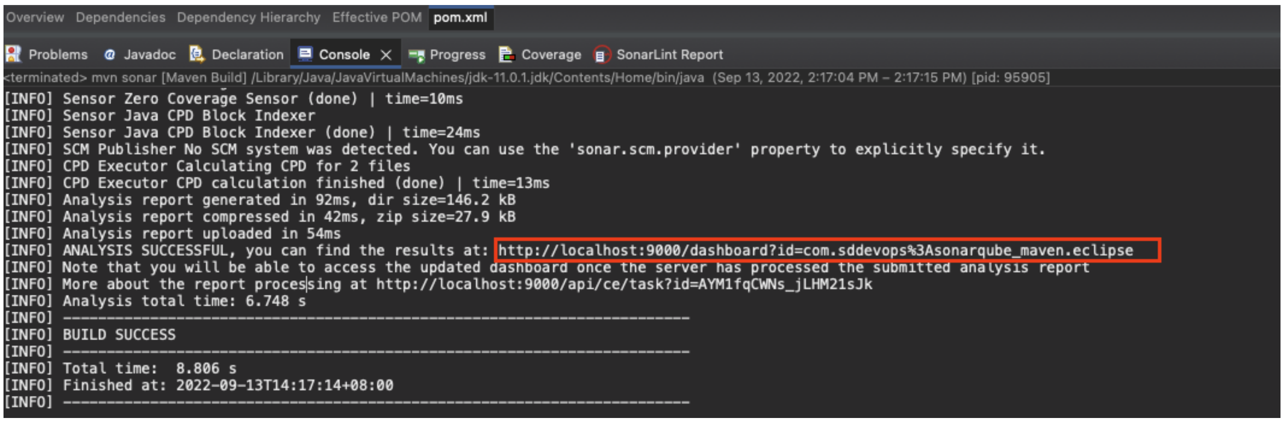


```
<terminated> mvn sonar [Maven Build] /Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java  (Sep 7, 2022, 10:50:07 PM – 10:50:21 PM) [pid: 26033]
[INFO] Sensor Analysis Warnings import [csharp] (done) | time=2ms
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=2ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=1ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=0ms
[INFO] Analysis report generated in 111ms, dir size=136.1 kB
[INFO] Analysis report compressed in 34ms, zip size=20.8 kB
[INFO] Analysis report uploaded in 37ms
[INFO] ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=cl.lherrera%3Ajacoco
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AYMYbj29wh6J4JeyViJ2
[INFO] Analysis total time: 8.356 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.651 s
[INFO] Finished at: 2022-09-07T22:50:21+08:00
[INFO] ------------------------------------------------------------------------
```

B. Examine SonarQube dashboard and identify the different components

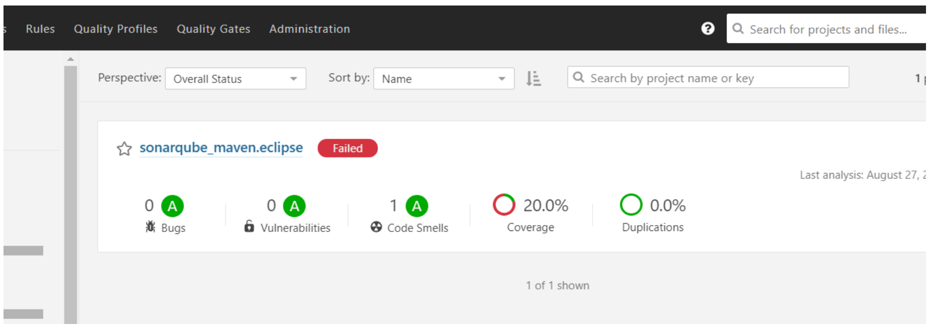
Now that we have analyzed our project and generated the SonarQube + JaCoCo analysis report, let's navigate back to our SonarQube server on http://localhost:9000 to see the updated version dashboard.

Alternatively, you may view the console and view the results of the dashboard via this link.


Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Problems Javadoc Declaration Console Progress Coverage SonarLint Report

```
<terminated> mvn sonar [Maven Build] /Library/Java/JavaVirtualMachines/jdk-11.0.1.jdk/Contents/Home/bin/java (Sep 13, 2022, 2:17:04 PM – 2:17:15 PM) [pid: 95905]
[INFO] Sensor Zero Coverage Sensor (done) | time=10ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=24ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor Calculating CPD for 2 files
[INFO] CPD Executor CPD calculation finished (done) | time=13ms
[INFO] Analysis report generated in 92ms, dir size=146.2 kB
[INFO] Analysis report compressed in 42ms, zip size=27.9 kB
[INFO] Analysis report uploaded in 54ms
[INFO] ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=com.sddevops%3Asonarqube_maven.eclipse
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AYM1fqCWNs_jLHM21sJk
[INFO] Analysis total time: 6.748 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.806 s
[INFO] Finished at: 2022-09-13T14:17:14+08:00
[INFO] ------------------------------------------------------------------------
```

- Login to SonarQube server on http://localhost:9000 using the default credentials:
  a. login: **admin**
  b. password: **admin**



- From the screen above, you should see our project SonarQube dashboard along with our JaCoCo Code coverage %.
- Click on your project name "**sonarqube_maven.eclipse**" to access the project dashboard.

Mini-Activity

Identify the various component in the SonarQube project dashboard and match them to their description:

Components	**Description**
	Percentage of lines of code covered by tests
	Identical lines of code or same blocks of code
	Coding error that will break your code immediately
	Security-sensitive code
	Code that hackers can exploit
	Set of conditions to enforce quality policy for your organization
	Code that can be confusing or difficult to maintain
	Estimated time takes to fix all code smells

- Take some time to look through the different screens and identify the various component in the dashboard:

  a. Vulnerabilities
  b. Security Hotspots
  c. Issues
  d. Measures
  e. Code
  f. Rules
  g. Quality Profile
  h. Quality Gates
  i. Bugs
  j. Technical debt
  k. Coverage
  l. Code Smells
  m. Duplications and Duplicated Blocks

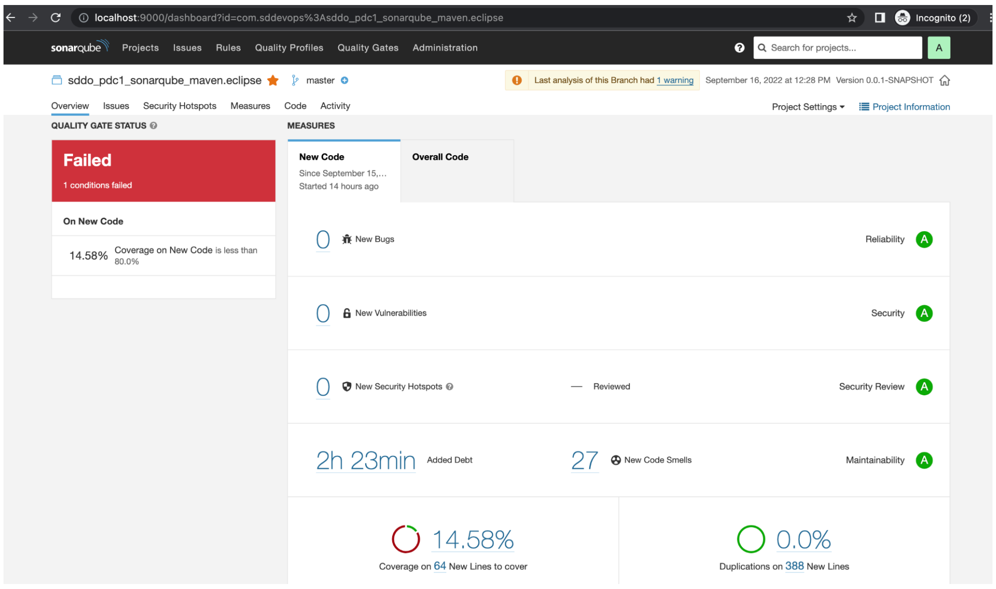
- For this exercise, we will **not** be configuring the quality gate or profiles. But take some time to look through the different screens and their description to appreciate the tool better.

Part 2: Implement SonarQube into your Project

1) In your previous project **junit_maven_eclipse**, there are three files: Song.java, SongCollection.java, and SongCollectionTest.java. Import those three files into your current SonarQube project and use SonarQube to analyze the project.

Note: Ensure that there are no fail test cases in your SongCollectionTest.java file or that the code coverage report will not be generated.

2) Look at the updated dashboard code coverage percentage, go to the following link: http://localhost:9000/component_measures?id=com.sddevops%3Asonarqube_maven.eclipse&metric=new_coverage&view=list (Project > Code Coverage) to identify the code coverage for your project:


localhost:9000/dashboard?id=com.sddevops%3Asddo_pdc1_sonarqube_maven.eclipse

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

sddo_pdc1_sonarqube_maven.eclipse master

Last analysis of this Branch had 1 warning September 16, 2022 at 12:28 PM Version 0.0.1-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Failed
1 conditions failed

On New Code
14.58% Coverage on New Code is less than 80.0%

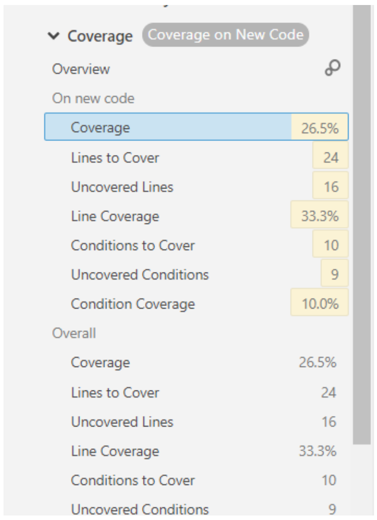
MEASURES

New Code
Since September 15,...
Started 14 hours ago

Overall Code

0 New Bugs — Reliability A
0 New Vulnerabilities — Security A
0 New Security Hotspots — Reviewed — Security Review A
2h 23min Added Debt — 27 New Code Smells — Maintainability A
14.58% Coverage on 64 New Lines to cover
0.0% Duplications on 388 New Lines

3) Based on the code coverage analysis, modify your code to increase the coverage to > 80%.
Optional (Advance Topic)



Coverage | Coverage on New Code

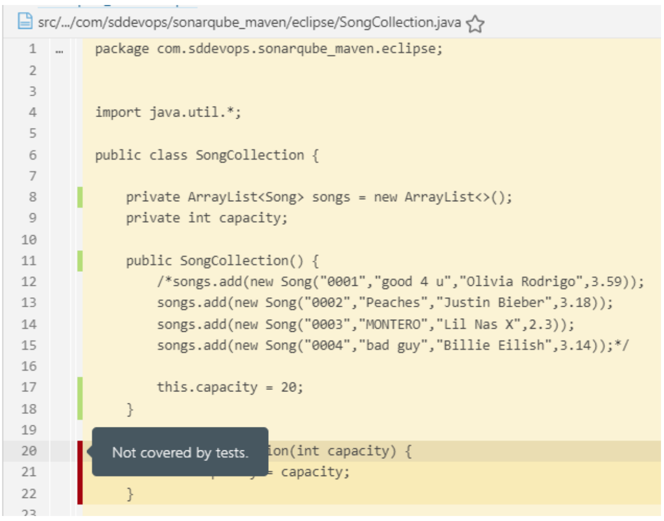
Overview

On new code

Coverage	26.5%
Lines to Cover	24
Uncovered Lines	16
Line Coverage	33.3%
Conditions to Cover	10
Uncovered Conditions	9
Condition Coverage	10.0%

Overall

Coverage	26.5%
Lines to Cover	24
Uncovered Lines	16
Line Coverage	33.3%
Conditions to Cover	10
Uncovered Conditions	9



src/.../com/sddevops/sonarqube_maven/eclipse/SongCollection.java

```
package com.sddevops.sonarqube_maven.eclipse;


import java.util.*;

public class SongCollection {

    private ArrayList<Song> songs = new ArrayList<>();
    private int capacity;

    public SongCollection() {
        /*songs.add(new Song("0001","good 4 u","Olivia Rodrigo",3.59));
        songs.add(new Song("0002","Peaches","Justin Bieber",3.18));
        songs.add(new Song("0003","MONTERO","Lil Nas X",2.3));
        songs.add(new Song("0004","bad guy","Billie Eilish",3.14));*/

        this.capacity = 20;
    }

                  ion(int capacity) {
                      = capacity;
    }
```

Not covered by tests.

Congratulations, in this lesson, you have successfully:

- Setup and **configure locally hosted SonarQube server**
- Identify how to use the **SonarLint** plugin and **practice code as you clean** using Eclipse IDE
- Examine **SonarQube dashboard** and identify the different components