**Comprehensive Guide for SonarQube with Quality Gate for Jenkins**

**Configuring JaCoCo code coverage Maven plugin**

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.7.9</version>

<configuration>

<includes>

<include>com/swayam/demo/springbootdemo/rest/\*\*</include>

</includes>

<excludes>

<exclude>com/swayam/demo/springbootdemo/rest/config/\*\*</exclude>

<exclude>com/swayam/demo/springbootdemo/rest/entity/\*\*</exclude>

<exclude>\*\*/RestFulMicroserviceApplication.class</exclude>

</excludes>

</configuration>

<executions>

<execution>

<id>prepare-agent</id>

<configuration>

<destFile>${jacoco.test.unit.dataFile}</destFile>

<propertyName>surefireArgLine</propertyName>

</configuration>

<goals>

<goal>prepare-agent</goal>

</goals>

</execution>

<execution>

<id>post-unit-test</id>

<phase>test</phase>

<goals>

<goal>report</goal>

</goals>

<configuration>

<dataFile>${jacoco.test.unit.dataFile}</dataFile>

<outputDirectory>target/jacoco/site-unit-test</outputDirectory>

</configuration>

</execution>

</executions>

</plugin>

**Things to note**

**Including or excluding packages or class**

It is a good practice to include only project-specific classes. Otherwise, it tends to have the coverage for 3rd party libraries as well. The way to include/exclude a package is to name it like: com/swayam/demo/springbootdemo/rest/config/\*\*

For including a single class: \*\*/RestFulMicroserviceApplication.class

**Prepare Agent**

First, we would need to prepare the JaCoCo Agent for instrumentation:

<http://www.jacoco.org/jacoco/trunk/doc/prepare-agent-mojo.html>

This would set a property having the agent details, which is then passed o to the surefire plugin so that the tests are run with this agent. This is how the surefire plugin is configured:

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<configuration>

<!-- Sets the VM argument line used when unit tests are run. -->

<argLine>${surefireArgLine}</argLine>

</configuration>

</plugin>

**Generating Report**

After the Junit Tests are run, we can specify the directory for generating the coverage report:

<http://www.jacoco.org/jacoco/trunk/doc/report-mojo.html>

**Configuring Sonar Maven plugin**

SonarQube’s documentation for configuring a Maven plugin is scant, misleading and difficult to decipher. Its high time that they start hiring some good Tech Writer!

<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner+for+Maven>

This is how the plugin is configured in your pom.xml. No global setting bullshit in settings.xml. Whoever suggested that is an idiot! The global settings.xml should kept as pristine as possible.

<plugin>

<groupId>org.sonarsource.scanner.maven</groupId>

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

<version>3.4.0.905</version>

<executions>

<execution>

<goals>

<goal>sonar</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

Apart from this, SonarQube also expects certain properties to be set.

<https://docs.sonarqube.org/display/SONAR/Analysis+Parameters>

I have included this under the sonar profile as below:

<profile>

<id>sonar</id>

<properties>

<sonar.projectKey>${project.artifactId}</sonar.projectKey>

<sonar.projectName>${project.artifactId}</sonar.projectName>

<sonar.projectVersion>${project.version}</sonar.projectVersion>

<sonar.host.url><http://192.168.1.4:9000/></sonar.host.url>

<sonar.verbose>true</sonar.verbose>

<sonar.showProfiling>true</sonar.showProfiling>

<!--

Sonar plugin is BUGGY!! It will not run when the below property is set

<sonar.scanner.dumpToFile>target/sonar-properties.txt</sonar.scanner.dumpToFile>

-->

<!--

<sonar.login>admin</sonar.login>

<sonar.password>admin</sonar.password>

-->

<sonar.sources>src/main/java</sonar.sources>

<sonar.exclusions>

\*\*/com/swayam/demo/springbootdemo/rest/config/\*\*,

\*\*/com/swayam/demo/springbootdemo/rest/entity/\*\*,

\*\*/RestFulMicroserviceApplication.java

</sonar.exclusions>

<sonar.sourceEncoding>UTF-8</sonar.sourceEncoding>

<sonar.language>java</sonar.language>

<sonar.tests>src/test/java</sonar.tests>

<sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>

<sonar.junit.reportPaths>target/surefire-reports</sonar.junit.reportPaths>

<sonar.java.binaries>target/classes</sonar.java.binaries>

<sonar.jacoco.reportPaths>${jacoco.test.unit.dataFile}</sonar.jacoco.reportPaths>

<sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>

</properties>

...

</profile>

**Including or excluding packages or class**

Remember that Sonar works on Java source code. The way to include/exclude a package is to name it like: \*\*/com/swayam/demo/springbootdemo/rest/config/\*\*

Similarly, you can exclude individual class like this: \*\*/RestFulMicroserviceApplication.java

**Running SonarQube Analysis and Fetching the results of QualityGate**

Run the Maven command:

mvn clean install sonar:sonar -P sonar

This will create the below file:

target/sonar/report-task.txt

There are the below 2 urls that has to be read from here:

1. serverUrl=http://192.168.1.4:9000

2. ceTaskUrl=http://192.168.1.4:9000/api/ce/task?id=AWE3eRSZAEMe8tTgpicn

Read the response from the ceTaskUrl using curl, and save it to a file ceTask.json:

curl http://192.168.1.4:9000/api/ce/task?id=AWE3eRSZAEMe8tTgpicn -o ceTask.json

The element we are interested in is task.analysisId:

"analysisId": "AWE3eRcyxJqMzJgr501D"

We need to read the response of the url: $serverUrl/api/qualitygates/project\_status?analysisId=$analysisId

curl http://192.168.1.4:9000/api/qualitygates/project\_status?analysisId=AWE3eRcyxJqMzJgr501D -o qualityGate.json

If the Json value projectStatus.status is ERROR, the project has failed QualityGate.

**Integration with Jenkins**

To start with, we would need to install the below Jenkins plugins:

**1. Sonar Quality Gates Plugin**:

<https://plugins.jenkins.io/sonar-quality-gates>

<https://github.com/arkanjoms/sonar-quality-gates-plugin/blob/master/README.md>

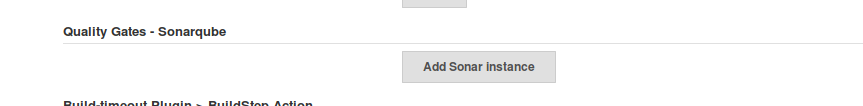
**2. SonarQube Scanner for Jenkins**:

<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner+for+Jenkins>

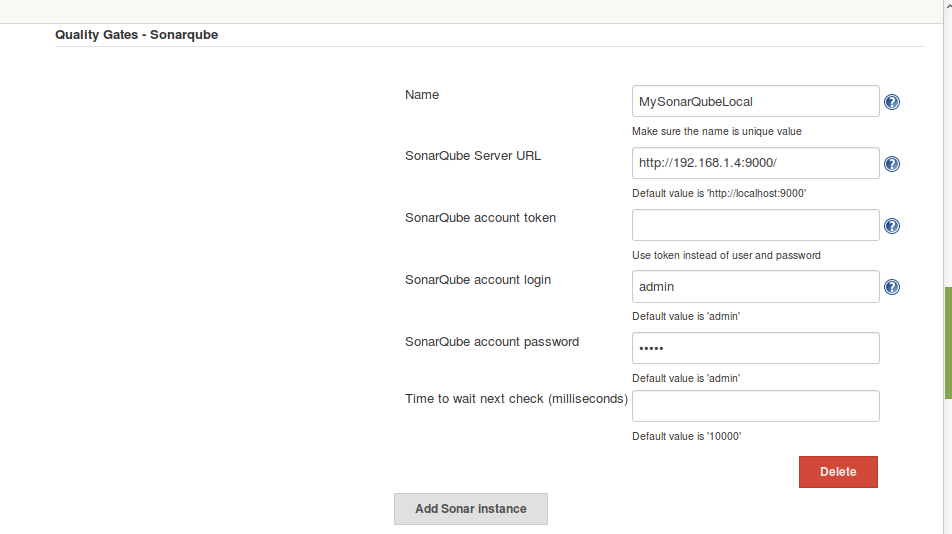
We would need to tell Jenkins about our SonarQube installation. This is done by logging into Jenkins and then navigating to Manage Jenkins -> Configure System

**Specifying Sonar Qube Installation**

**Step 1**

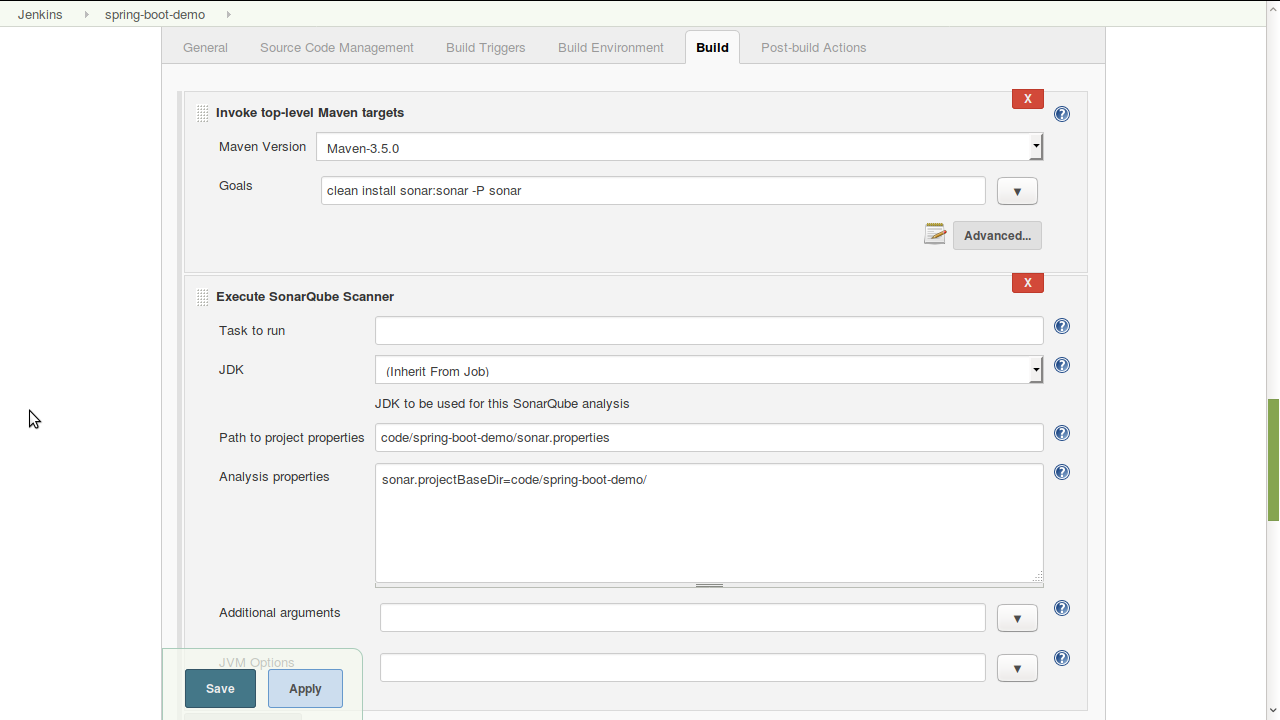


**Step 2**



**Integration with Jenkins Freestyle Project**

Go to the **Build** section -> **Add build step** -> **Execute SonarQube Scanner**.



You can specify the sonar properties either as a separate file in the section **Path to project properties** or in the section **Analysis properties**. Both of these approaches are equally bad. Instead, the properties should be passed through a Maven plugin in the build step as shown above:

clean install sonar:sonar -P sonar

In the Jenkins Freestyle Project, thats all the configuration you need.

**Integration with Jenkins Pipeline Project**

The **SonarQube Scanner** plugin has support for Jenkins Pipeline. It can be used as follows:

withSonarQubeEnv('MySonarQubeLocal') {

*sh* "/var/jenkins\_home/apache-maven-3.5.0/bin/mvn clean install sonar:sonar -P sonar"

}

To determine whether the project has passed the QualityGate, we need to use the logic stated in the section **Running SonarQube Analysis and Fetching the results of QualityGate**. Since we are using Groovy DSL, it is very easy for us to do this. I am pasting the full content of the Jenkinsfile:

**pipeline** {

**agent** {

*label* 'master'

}

**stages** {

**stage**('Build') {

**steps** {

withSonarQubeEnv('MySonarQubeLocal') {

dir('restful-microservice') {

*sh* "/var/jenkins\_home/apache-maven-3.5.0/bin/mvn clean install sonar:sonar -P sonar"

**script** {

**def** sonarProps = readFile encoding: 'utf-8', file: 'target/sonar/report-task.txt'

*echo* "sonarProps: " + sonarProps

**def** ceTaskUrl = 'ceTaskUrl'

sonarProps.split('\n').each { line ->

**if** (line.startsWith(ceTaskUrl)) {

env.SONAR\_CE\_TASK\_URL = line.substring(ceTaskUrl.length() + 1)

*echo* "env.SONAR\_CE\_TASK\_URL: " + env.SONAR\_CE\_TASK\_URL

}

**if** (line.startsWith('serverUrl')) {

**def** sonarServerUrl = line.split('=')[1]

**if** (!sonarServerUrl.endsWith('/')) {

sonarServerUrl += '/'

}

env.SONAR\_SERVER\_URL = sonarServerUrl

*echo* "env.SONAR\_SERVER\_URL: " + env.SONAR\_SERVER\_URL

}

}

}

}

}

}

}

**stage**('Quality Gate') {

**steps** {

dir('restful-microservice') {

**script** {

sleep time: 3000, unit: 'MILLISECONDS'

*timeout*(time: 1, unit: 'MINUTES') {

waitUntil {

**def** jsonOutputFile = 'target/sonar/ceTask.json'

*sh* 'curl $SONAR\_CE\_TASK\_URL -o ' + jsonOutputFile

**def** jsonOutputFileContents = readFile encoding: 'utf-8', file: jsonOutputFile

**def** ceTask = **new** groovy.json.JsonSlurper().parseText(jsonOutputFileContents)

env.SONAR\_ANALYSIS\_ID = ceTask['task']['analysisId']

**return** 'SUCCESS'.equals(ceTask['task']['status'])

}

**def** qualityGateUrl = env.SONAR\_SERVER\_URL + 'api/qualitygates/project\_status?analysisId=' + env.SONAR\_ANALYSIS\_ID

*echo* "qualityGateUrl: " + qualityGateUrl

**def** qualityGateJsonFile = 'target/sonar/qualityGate.json'

*sh* 'curl ' + qualityGateUrl + ' -o ' + qualityGateJsonFile

**def** qualityGateJsonFileContents = readFile encoding: 'utf-8', file: qualityGateJsonFile

**def** qualityGateJson = **new** groovy.json.JsonSlurper().parseText(qualityGateJsonFileContents)

*echo* 'qualityGateJson: ' + qualityGateJson

**if** ("ERROR".equals(qualityGateJson['projectStatus']['status'])) {

error "Quality Gate Failure"

}

*echo* "Quality Gate Success"

}

}

}

}

}

**stage**('Code Coverage') {

**steps** {

jacoco()

}

}

}

}

## **Sample report-task.txt**

**projectKey=spring-boot-demo**

serverUrl=http://192.168.1.4:9000

serverVersion=6.7.0.33306

dashboardUrl=http://192.168.1.4:9000/dashboard/index/spring-boot-demo

ceTaskId=AWBlpOuTAmeRAKOg4E\_E

ceTaskUrl=http://192.168.1.4:9000/api/ce/task?id=AWBlpOuTAmeRAKOg4E\_E

## **Sample ceTask.json**

**{**

"task": {

"id": *"AWBlpOuTAmeRAKOg4E\_E"*,

"type": *"REPORT"*,

"componentId": *"AWBg3FFS8C0jOgq5kO1m"*,

"componentKey": *"spring-boot-demo"*,

"componentName": *"spring-boot-demo"*,

"componentQualifier": *"TRK"*,

"analysisId": *"AWBlpPPI5a3\_SsAmFvXW"*,

"status": *"SUCCESS"*,

"submittedAt": *"2017-12-17T18:03:15+0000"*,

"submitterLogin": *"admin"*,

"startedAt": *"2017-12-17T18:03:17+0000"*,

"executedAt": *"2017-12-17T18:03:18+0000"*,

"executionTimeMs": 1169,

"logs": false,

"hasScannerContext": true,

"organization": *"default-organization"*

}

**}**

## **Sample qualityGate.json for PASS**

**{**

"projectStatus": {

"status": *"OK"*,

"conditions": [

{

"status": *"OK"*,

"metricKey": *"new\_security\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"OK"*,

"metricKey": *"new\_reliability\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"OK"*,

"metricKey": *"new\_maintainability\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"OK"*,

"metricKey": *"new\_duplicated\_lines\_density"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"3"*,

"actualValue": *"0.0"*

}

],

"periods": [

{

"index": 1,

"mode": *"previous\_version"*,

"date": *"2017-12-16T19:50:29+0000"*

}

],

"ignoredConditions": false

}

**}**

## **Sample qualityGate.json for FAIL**

**{**

"projectStatus": {

"status": *"ERROR"*,

"conditions": [

{

"status": *"OK"*,

"metricKey": *"new\_security\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"OK"*,

"metricKey": *"new\_reliability\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"OK"*,

"metricKey": *"new\_maintainability\_rating"*,

"comparator": *"GT"*,

"periodIndex": 1,

"errorThreshold": *"1"*,

"actualValue": *"1"*

},

{

"status": *"ERROR"*,

"metricKey": *"coverage"*,

"comparator": *"LT"*,

"errorThreshold": *"80"*,

"actualValue": *"12.6"*

}

],

"periods": [

{

"index": 1,

"mode": *"previous\_version"*,

"date": *"2017-12-31T18:00:19+0000"*

}

],

"ignoredConditions": false

}

**}**

## **Sources**

The complete sources can be found here: <https://github.com/paawak/spring-boot-demo/tree/master/restful-microservice>