

# Integración de SonarCloud con Github Actions

## SonarCloud

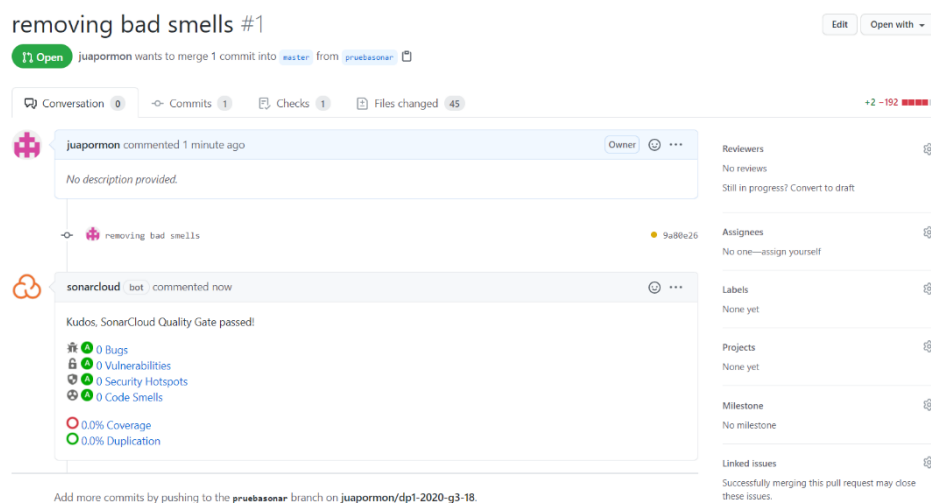
Una de las cosas más importantes a la hora de implementar código en un proyecto software es que este sea de la mayor calidad posible, para ello, se ha implementado en el proyecto la herramienta SonarCloud, la cuál realiza un analisis y nos devuelve los trozos de código en los que se puede mejorar la calidad, gracias a las métricas que implementa esta herramienta.

Podemos encontrar varios tipos de métricas:

- **Bugs:** Errores en el código que tienen mucha probabilidad de romper la aplicación y que deben ser resueltos lo antes posible.
- **Vulnerabilities:** Código que puede ser explotado por hackers.
- **Security Hotspots:** Código sensible en cuanto a seguridad que debe ser revisado para comprobar si puede generar vulnerabilidad.
- **Code Smells:** Código extraño que puede ser difícil de mantener.
- **Debt:** El tiempo que se estima que se tardará en resolver los Code Smell existentes
- **Duplicated Blocks:** Bloques de código idénticos

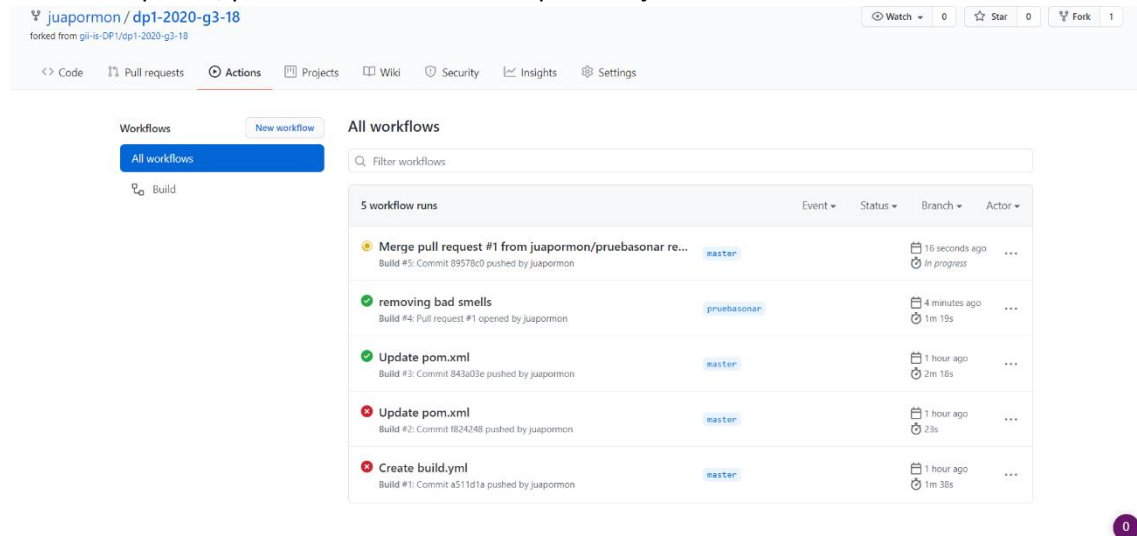
## Github Actions

Para la ejecución de SonarCloud en nuestro proyecto hemos decidido usar Github Actions como CI, ya que es una herramienta bastante reciente y muy potente, que nos permite ejecutar un analisis con cada pull request que implica a la rama master. Esto nos permite ver las posibles mejoras de calidad que tiene el código nuevo que estamos intentando mergear y tener un mayor control sobre la deuda técnica.



A parte de la ejecución del analisis en cada pull request, también se ejecutará un analisis de la rama master despues de cada mergeo, sabiendo así cual es el total de deuda de todo el proyecto y tener una vista general de las métricas.

En estas capturas, podemos ver como se empieza a ejecutar el analisis.



juapormon / dp1-2020-g3-18

forked from gii-is-DP1/dp1-2020-g3-18

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Workflows New workflow

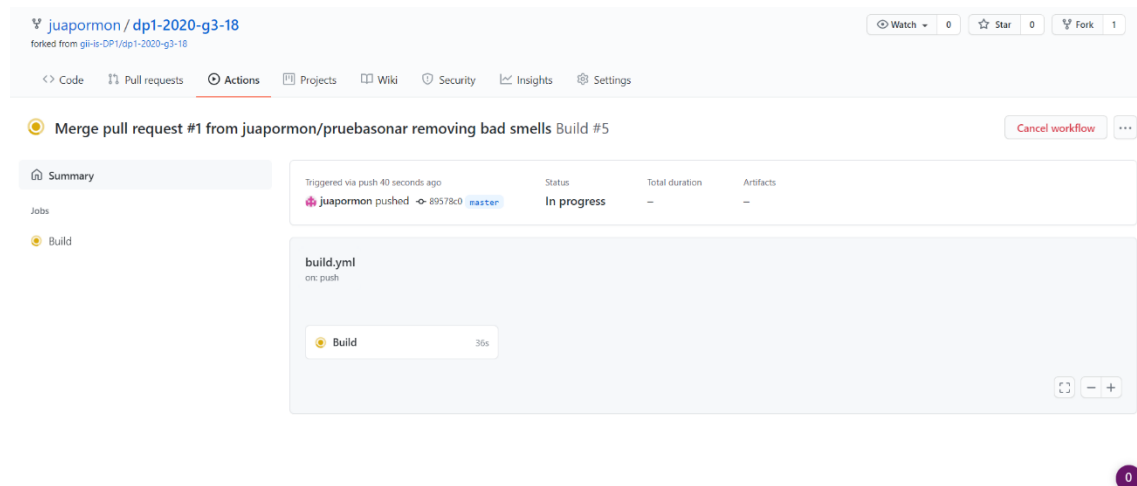
All workflows

Filter workflows

5 workflow runs

Event	Status	Branch	Actor
Merge pull request #1 from juapormon/pruebasonar removing bad smells	In progress	master	juapormon
removing bad smells	In progress	pruebasonar	juapormon
Update pom.xml	Completed	master	juapormon
Update pom.xml	Failed	master	juapormon
Create build.yml	Failed	master	juapormon

Esta sería la vista en detalle.



juapormon / dp1-2020-g3-18

forked from gii-is-DP1/dp1-2020-g3-18

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Merge pull request #1 from juapormon/pruebasonar removing bad smells Build #5

Cancel workflow

Summary

Jobs

Build

Triggered via push 40 seconds ago

juapormon pushed -> 89578c0 master

Status In progress

Total duration 1m 36s

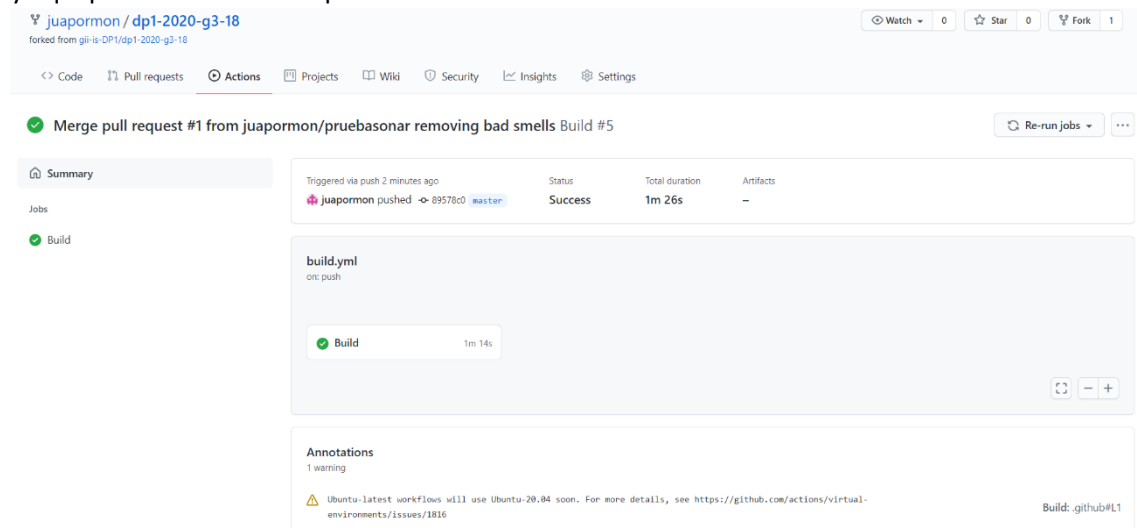
Artifacts

build.yml

on: push

Build 36s

y aquí podemos ver como pasa el build.



juapormon / dp1-2020-g3-18

forked from gii-is-DP1/dp1-2020-g3-18

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Merge pull request #1 from juapormon/pruebasonar removing bad smells Build #5

Re-run jobs

Summary

Jobs

Build

Triggered via push 2 minutes ago

juapormon pushed -> 89578c0 master

Status Success

Total duration 1m 26s

Artifacts

build.yml

on: push

Build 1m 14s

Annotations

1 warning

Ubuntu-latest workflows will use Ubuntu-20.04 soon. For more details, see https://github.com/actions/virtual-environments/issues/1816

Build: github#L1

y todavía existiría una vista con aún mas detalle en los pull request hechos.

improving code quality #4

Open juapormon wants to merge 1 commit into master from fixbugs

Conversation 0 Commits 1 Checks 3 Files changed 22

improving code quality 10e1201

Travis CI

SonarCloud

Build on pull\_request

Build

Build succeeded now in 1m 14s

Set up job

Run actions/checkout@v2

Set up JDK 11

Cache SonarCloud packages

Cache Maven packages

Build and analyze

Post Cache Maven packages

Post Cache SonarCloud packages

Post Set up JDK 11

Post Run actions/checkout@v2

```
1 Post job cleanup.
2 /usr/bin/git version
3 git version 2.30.0
4 /usr/bin/git config --local --name-only --get-regexp core.sshCommand
5 /usr/bin/git submodule foreach --recursive git config --local --name-only --get-regexp 'core.sshCommand' && git config --local --unset-all 'core.sshCommand' {}
6 /usr/bin/git config --local --name-only --get-regexp http.https://github.com/.extraheader
7 http.https://github.com/.extraheader
8 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader
9 /usr/bin/git submodule foreach --recursive git config --local --name-only --get-regexp 'http.https://github.com/.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' {}
10 Complete job
```

## Nuestro proyecto

Este sería el estado de nuestro proyecto tras la ejecución del analisis.

sonarcloud My Projects My Issues

February 8, 2021, 7:39 PM Version 2.2.0.BUILD-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity Administration

Quality Gate Not computed

We can't display your Quality Gate without a New Code definition.  
The quality gate helps you know if your project is production-ready.

Set New Code definition

Reliability Measures

0 A Bugs

started yesterday

Security Measures

29 D Vulnerabilities

4 E Security Hotspots

Maintainability Measures

2d A Debt

210 Code Smells

Coverage Measures

0.0% Coverage

Duplications Measures

0 0% 0

About This Project

Parent pom providing dependency and plugin management for applications built with Maven

No tags

3k Lines of Code

Java 2.7k

XML 297

Project Activity

February 8, 2021, 7:39 PM

2.2.0.BUILD-SNAPSHOT

February 7, 2021, 1:42 AM

Project Analyzed

February 7, 2021, 12:51 AM

Project Analyzed

Show More

Quality Gate

(Default) Sonar way

Quality Profiles

(Java) Sonar way

(XML) Sonar way

Last analysis method

Analyzed by Github Actions

External Links

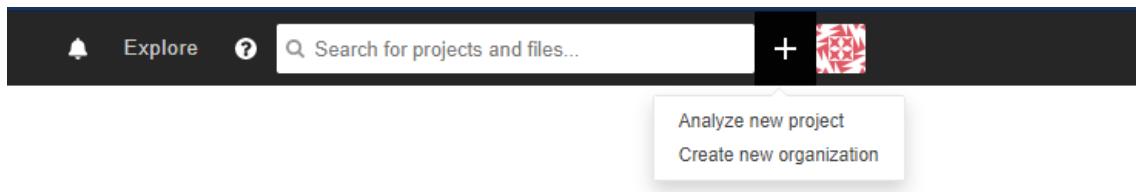
Project's Website

Sources

## Cómo integrar SonarCloud en un proyecto Maven con github Actions:

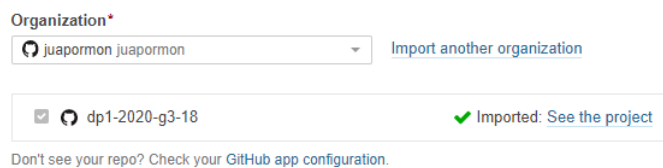
El primer paso es Crear una cuenta en SonarCloud, lo más facil es crearla a partir de nuestra cuenta de github, teniendo así ambas herramientas conectadas.

Una vez creada la cuenta, debemos dar permiso a Sonarcloud a el repositorio en el que queremos ejecutar los analisis



Solo le podremos dar permiso para acceder a los repositorios en los que nuestra cuenta tenga permiso.

### Analyze projects - Select repositories



Una vez importado, aparecerá una opción para ejecutar el analisis con Github Actions, seleccionamos esa opción y añadiremos al pom de nuestro proyecto la siguiente propiedad:

<properties>

```
<!-- Generic properties -->
<java.version>1.8</java.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

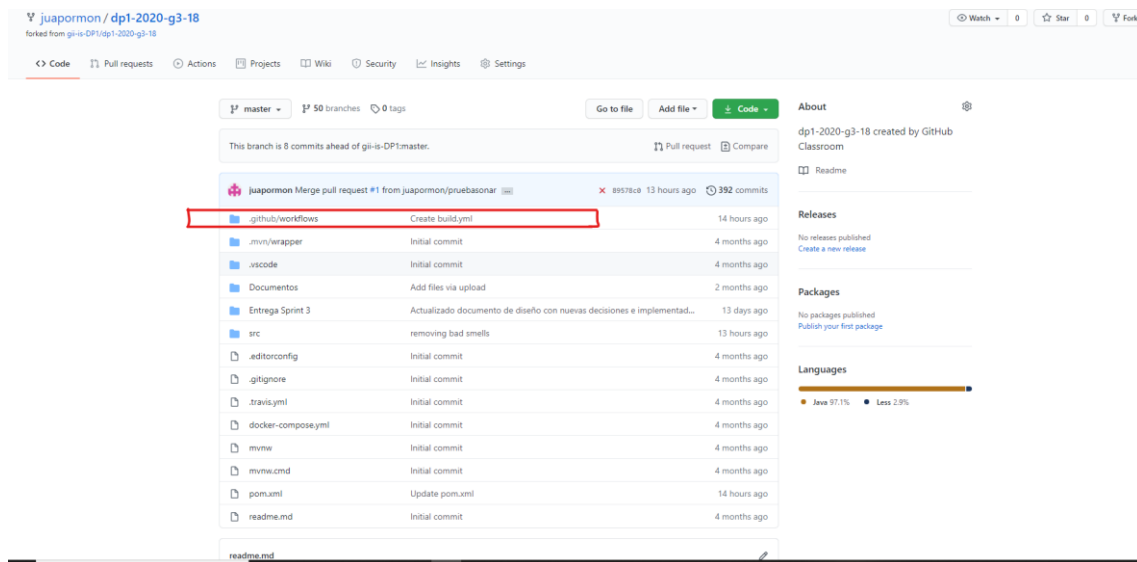
<!-- Web dependencies -->
<webjars-bootstrap.version>3.3.6</webjars-bootstrap.version>
<webjars-jquery-ui.version>1.11.4</webjars-jquery-ui.version>
<webjars-jquery.version>2.2.4</webjars-jquery.version>
<wro4j.version>1.8.0</wro4j.version>

<jacoco.version>0.8.5</jacoco.version>
<spring-format.version>0.0.19</spring-format.version>
```

```
<sonar.projectKey>juapormon_dp1-2020-g3-18</sonar.projectKey>
<sonar.organization>juapormon</sonar.organization>
<sonar.host.url>https://sonarcloud.io</sonar.host.url>
```

</properties>

Después crearemos en la raíz del proyecto una carpeta “.github” y dentro de esta otra carpeta “workflows”



Una vez ahí creamos el archivo “build.yml” con el siguiente contenido:

```

36 lines (36 sloc)  1.09 KB
1  name: Build
2  on:
3    push:
4      branches:
5        - master
6    pull_request:
7      types: [opened, synchronize, reopened]
8  jobs:
9    build:
10     name: Build
11     runs-on: ubuntu-latest
12     steps:
13       - uses: actions/checkout@v2
14         with:
15           fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
16       - name: Set up JDK 11
17         uses: actions/setup-java@v1
18         with:
19           java-version: 11
20       - name: Cache SonarCloud packages
21         uses: actions/cache@v1
22         with:
23           path: ~/.sonar/cache
24           key: ${{ runner.os }}-sonar
25           restore-keys: ${{ runner.os }}-sonar
26       - name: Cache Maven packages
27         uses: actions/cache@v1
28         with:
29           path: ~/.m2
30           key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
31           restore-keys: ${{ runner.os }}-m2
32       - name: Build and analyze
33         env:
34           GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
35           SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
36       run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar

```

En el cuál tenemos que destacar que dentro de secrets (del proyecto) tenemos ya dado por github actions el token “secrets.GITHUB\_TOKEN” del que se puede obtener más información aquí <https://docs.github.com/en/actions/reference/authentication-in-a-workflow> . y también tenemos el “secrets.SONAR\_TOKEN” el cuál tendremos que crear nosotros de la siguiente forma:

Dentro de nuestro proyecto en github accedemos a la pestaña settings y dentro de esta buscamos “Secrets”, una vez aquí, crearemos un nuevo “repository secret” con nombre “SONAR\_TOKEN” y clave la que nos proporciona sonarcloud en el paso anterior.

Una vez pusheemos este archivo, si todo se ha hecho correctamente, se empezará a hacer una build de github actions, si esta build pasa, directamente en la pestaña de sonarcloud nos aparecerá el análisis del código de nuestro proyecto, que se actualizará automáticamente con cada build tras mergear en master.