

# SONARQUBE

Salvador Torres Domínguez

## Índice:

¿Qué es? .....	2
Funciones .....	2
¿Cómo implementarlo? .....	2
Webgrafia: .....	9

## ¿Qué es?

Es una plataforma de código abierto diseñada para evaluar y mejorar la calidad del código fuente. Proporciona herramientas para realizar análisis estático del código, identificar posibles problemas, y ofrecer métricas y informes detallados sobre la salud general del código.

## Funciones

- Examina el código fuente sin ejecutarlo, identificando posibles problemas como bugs, vulnerabilidades de seguridad, código duplicado, mala práctica de codificación, etc.
- Proporciona métricas cuantitativas sobre la calidad del código, como la complejidad ciclomática, la duplicación de código, el número de problemas encontrados, entre otros.
- Puede integrarse en entornos de integración continua (CI) para analizar automáticamente el código después de cada cambio realizado, permitiendo la detección temprana de problemas.
- Admite una amplia gama de lenguajes de programación, como Java, C#, JavaScript, Python, entre otros.
- 

## ¿Cómo implementarlo?

-Necesitamos instalar el entorno de ejecución de y el kit de desarrollo de Java:

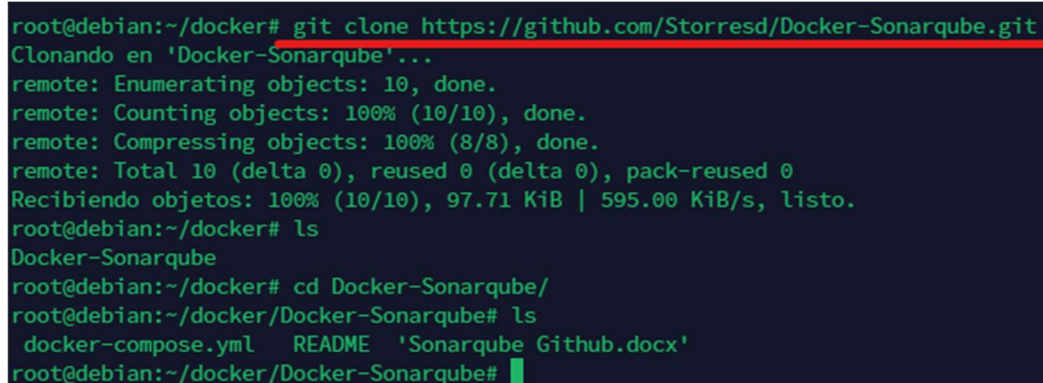
***sudo apt update && apt upgrade***

***sudo apt install default-jre***

***sudo apt install default-jdk***

-**Descargamos** el repositorio que contiene lo necesario para descargar el archivo docker-compose.yml que contiene lo necesario para levantar sonarqube:

***git clone https://github.com/Storresd/Docker-Sonarqube.git***



```
root@debian:~/docker# git clone https://github.com/Storresd/Docker-Sonarqube.git
Clonando en 'Docker-Sonarqube'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
Recibiendo objetos: 100% (10/10), 97.71 KiB | 595.00 KiB/s, listo.
root@debian:~/docker# ls
Docker-Sonarqube
root@debian:~/docker# cd Docker-Sonarqube/
root@debian:~/docker/Docker-Sonarqube# ls
docker-compose.yml  README  'Sonarqube Github.docx'
root@debian:~/docker/Docker-Sonarqube#
```

-Accedemos a la carpeta descargada, y **levantamos** los contenedores:

***cd Docker-Sonarqube/***

***docker compose up -d***

```

root@debian:~/docker/Docker-Sonarqube# docker compose up -d
[+] Running 17/24
  db 14 layers [#####] 17.73MB/109MB Pulling
  ✓ e1caac4eb9d2 Download complete
  ✓ 7a2930f13d47 Download complete
  ✓ a6c49e965138 Download complete
  ✓ ed8dc94f857d Download complete
  ✓ 1f07b4807698 Download complete
  ✓ a776288d4030 Download complete
  ✓ 7cbb4adb3448 Download complete
  ✓ b6dbd7317d5f Download complete
  * 52814b5dc710 Downloading [=====>] 17.73MB/109MB
  ✓ b68697689b55 Download complete
  ✓ 6d80681e3923 Download complete
  * 4270a9f40aee Waiting
  * d28fa0286314 Waiting
  * cb1ee5bc271e Waiting
  sonarqube 8 layers [#####] 182.8MB/462.1MB Pulling
  ✓ d66d6a6a3687 Already exists
  ✓ 18f947fdc0fc Pull complete
  ✓ 5374706a264d Pull complete
  ✓ c9aeff6b625d Pull complete
  ✓ 82faf7b7220d Pull complete
  * 6dfd4447b8f5 Downloading [=====>] 182.8MB/462.1MB
  ✓ a5f5c4c8480d Download complete
  ✓ 4f4fb700ef54 Download complete

```

En caso tuve fallos al levantar los contenedores, ejecuté los siguientes comandos:

**`echo 'vm.max_map_count=262144' | sudo tee -a /etc/sysctl.conf`**

**`sudo sysctl -p`**

```

root@debian:~/docker/Docker-Sonarqube# echo 'vm.max_map_count=262144' | sudo tee -a /etc/sysctl.conf
vm.max_map_count=262144
root@debian:~/docker/Docker-Sonarqube# sudo sysctl -p
vm.max_map_count = 262144
vm.max_map_count = 262144
vm.max_map_count = 262144
vm.max_map_count = 262144
root@debian:~/docker/Docker-Sonarqube#

```

El parámetro `vm.max_map_count` controla el número máximo de áreas de mapa de memoria que un proceso puede tener.

-Ya levantado los contenedores, podemos acceder a la **interfaz gráfica** por el puerto 9000:

**<IP del servidor>:9000**

Usuario: admin

Contraseña: admin

-Para **escanear** nuestro proyecto, tenemos que descargarnos una herramienta llamada Sonarscanner, complementaria y necesaria para Sonarqube, vamos a crear una carpeta donde descargarlo:

***sudo mkdir sonarscanner***

```
root@debian:~/docker# mkdir sonarscanner
root@debian:~/docker#
```

-Dentro de la carpeta creada ejecutamos el comando para la **descarga** de la herramienta:

**Wget** <https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip>

```
root@debian:~/docker/sonarscanner# wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip
--2024-02-25 20:48:02-- https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip
Resolviendo binaries.sonarsource.com (binaries.sonarsource.com)... 108.157.169.27, 108.157.169.34, 108.157.169.80, ...
Conectando con binaries.sonarsource.com (binaries.sonarsource.com)[108.157.169.27]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
longitud: 47123726 (45M) [binary/octet-stream]
Guardando a: sonar-scanner-cli-5.0.1.3006-linux.zip
sonar-scanner-cli-5.0.1.3006-linux.zip 100%[=====] 44,94M 30,3MB/s
2024-02-26 10:48:04 (30,3 MB/s) - sonar-scanner-cli-5.0.1.3006-linux.zip guardado [47123726/47123726]
root@debian:~/docker/sonarscanner#
```

-Extraemos Sonarscanner:

***unzip sonar-scanner-cli-5.0.1.3006-linux.zip***

```
root@debian:~/docker/sonarscanner# unzip sonar-scanner-cli-5.0.1.3006-linux.zip
Archive:  sonar-scanner-cli-5.0.1.3006-linux.zip
  creating: sonar-scanner-5.0.1.3006-linux/
  creating: sonar-scanner-5.0.1.3006-linux/jre/
  creating: sonar-scanner-5.0.1.3006-linux/jre/lib/
  creating: sonar-scanner-5.0.1.3006-linux/jre/lib/security/
  creating: sonar-scanner-5.0.1.3006-linux/jre/lib/jfr/
  creating: sonar-scanner-5.0.1.3006-linux/jre/lib/server/
```

-Eliminamos el archivo de archivo zip ya que no lo necesitamos:

***rm -r sonar-scanner-cli-5.0.1.3006-linux.zip***

```
root@debian:~/docker/sonarscanner# rm -r sonar-scanner-cli-5.0.1.3006-linux.zip
root@debian:~/docker/sonarscanner# ls
sonar-scanner-5.0.1.3006-linux
root@debian:~/docker/sonarscanner#
```

-Añadimos las **variables de entorno** para ejecutar sonarscanner más fácilmente desde cualquier directorio:

***nano ~/.bashrc***

Añadimos la siguiente línea al final del archivo:

**`export PATH=$PATH:/root/docker/sonarscanner/sonar-scanner-5.0.1.3006-linux/bin`**

```
GNU nano 7.2 /root/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.

# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022

# You may uncomment the following lines if you want `ls' to be colorized:
# export LS_OPTIONS='--color=auto'
# eval "$(dircolors)"
# alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -lA'
#
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'

export PATH=$PATH:/root/docker/sonarscanner/sonar-scanner-5.0.1.3006-linux/bin
```

Aplicamos los cambios:

**`source ~/.bashrc`**

```
root@debian:~/docker/sonarscanner# source ~/.bashrc
root@debian:~/docker/sonarscanner#
```

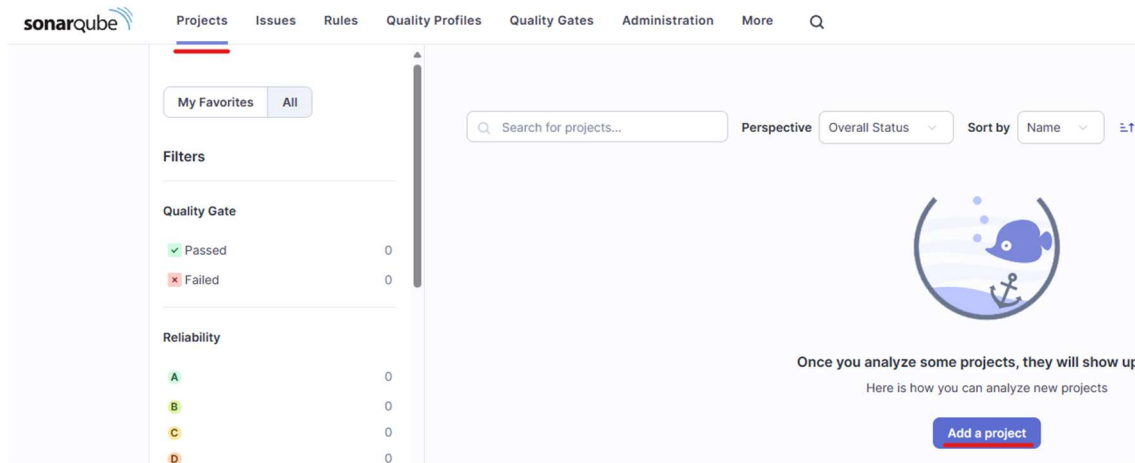
Comprobar que se ha guardado viendo la versión de la herramienta:

**`sonar-scanner -v`**

```
root@debian:~/docker/sonarscanner# sonar-scanner -v
INFO: Scanner configuration file: /root/docker/sonarscanner/sonar-scanner-5.0.1.3006-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 5.0.1.3006
INFO: Java 17.0.7 Eclipse Adoptium (64-bit)
INFO: Linux 6.1.0-18-amd64 amd64
root@debian:~/docker/sonarscanner#
```

Ahora sí, vamos a proceder con el análisis de nuestro código, para ello teniendo nuestro código (en mi caso tengo un archivo PHP con vulnerabilidad de inyección SQL), tenemos que hacer lo siguiente:

-Dentro de Sonarqube, vamos a generar un nuevo proyecto:



-Seleccionamos la opción para crear el proyecto de forma local:

### Create a local project

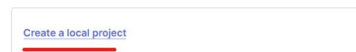
How do you want to create your project?

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.



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



-Escribimos un **nombre** para el proyecto:

1 of 2

### Create a local project

Project display name \*

detectar-php

Up to 255 characters. Some scanners might override the value.

Project key \*

detectar-php

The project key is a unique identifier for your project. It may contain alphanumeric characters, '-' (dash), '\_' (underscore), and '.' (dot). Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), and '.' (dot). At least one non-digit.

Main branch name \*

main

The name of your project's default branch. [Learn More](#)

Cancel Next

-**Seleccionamos** la opción para usar las configuraciones globales definidas en SonarQube:

## Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

### Previous version

Any code that has changed since the previous version is considered new code.  
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version

Any code that has changed since the previous version is considered new code.  
Recommended for projects following regular versions or releases.

☐ Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will be closed.  
Recommended for projects following continuous delivery.

☐ Reference branch

Choose a branch as the baseline for the new code.  
Recommended for projects using feature branches.

[Back](#)

[Create project](#)

-Después de aceptar, seleccionamos **Locally** para seleccionar los archivos que se encuentran en nuestro equipo:

## Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

 [With Jenkins](#)

 [With GitHub Actions](#)

 [With Bitbucket Pipelines](#)

 [With GitLab CI](#)

 [With Azure Pipelines](#)

[Other CI](#)

SonarQube integrates with your workflow no matter which CI tool you're using.

[Locally](#)

Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.

-Le damos a que se genere el **token** y que expire pasado los 30 días:

Generate → Continue:



## Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

### 1 Provide a token

Generate a project token

Use existing token

Token name ?

Expires in

Analyze "detectar-php"

30 days

Generate

### 1 Provide a token

Analyze "detectar-php": `sqp_4b7b8429649b21e8a82330139b821a8b27175d85` 

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

Continue

-Seleccionamos el lenguaje del archivo que va ser analizado, como en nuestro caso es PHP seleccionamos **Other**, y el sistema operativo de nuestro equipo, **Linux**:

### 2 Run analysis on your project

What option best describes your build?

Maven

Gradle

.NET

Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux

Windows

macOS

Download and unzip the Scanner for Linux

Visit the [official documentation of the Scanner](#) to download the latest version, and add the `bin` directory to the `PATH` environment variable

Execute the Scanner

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \
-Dsonar.projectKey=detectar-php \
-Dsonar.sources=. \
-Dsonar.host.url=http://192.168.1.40:9000 \
-Dsonar.login=sqp_037f22b86870aa9735861cb04bb83e75c0193528
-Dsonar.token=sqp_4b7b8429649b21e8a82330139b821a8b27175d85
```

Please visit the [official documentation of the Scanner](#) for more details.

-Nos generará un **código**, lo copiamos y ejecutamos en el mismo directorio donde se encuentra el archivo a analizar:

```
root@debian:~/docker/Codigos# sonar-scanner \
-Dsonar.projectKey=detectar-php \
-Dsonar.sources=. \
-Dsonar.host.url=http://192.168.1.40:9000 \
-Dsonar.login=sqp_037f22b86870aa9735861cb04bb83e75c0193528
INFO: Scanner configuration file: /root/docker/sonarscanner/sonar-scanner-5.0.1.3006-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 5.0.1.3006
INFO: Java 17.0.7 Eclipse Adoptium (64-bit)
INFO: Linux 6.1.0-18-amd64 amd64
INFO: User cache: /root/.sonar/cache
```

Este comando analiza el código fuente del proyecto y lo envía a SonarQube para su evaluación, una vez terminado, nos aparecerá algo como lo siguiente:

```
INFO: Analysis total time: 11.991 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 14.173s
INFO: Final Memory: 18M/64M
INFO: -----
root@debian:~/docker/Codigos#
```

-Volvemos a Sonarqube, y vemos que nos ha detectado una vulnerabilidad:

Quality Gate Status ?

✓ Passed

Enjoy your sparkling clean code!

Measures Last analysis: 35 seconds ago

New Code Overall Code

**Security**  
1 Open issues  
1 H 0 M 0 L

**Reliability**  
0 Open issues  
0 H 0 M 0 L

**Maintainability**  
0 Open issues  
0 H 0 M 0 L

**Accepted issues**  
0  
Valid issues, but not fixed. They represent accepted technical debt.

**Coverage**  
0.0%  
On 14 New Lines to cover.

**Duplications**  
0.0%  
On 28 New Lines.

**Security Hotspot**  
1  
E

view priority: High

SQL Injection 1

Make sure that formatting this SQL query is safe here.

1 of 1 shown

`/vulnerable.php` Open in IDE

```
7 // Crear conexión
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Verificar conexión
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // Vulnerabilidad de inyección SQL
16 $id_usuario = $_POST["id"];
17 $result = $conn->query("SELECT * FROM usuarios WHERE id = $id_usuario");
```

Make sure that formatting this SQL query is safe here.

En el mensaje que nos genera sobre el código ha podido detectar la vulnerabilidad de inyección SQL.

## Webgrafia:

<https://www.digitalocean.com/community/tutorials/how-to-ensure-code-quality-with-sonarqube-on-ubuntu-18-04#step-6-setting-up-the-code-scanner>

<https://github.com/jesusmatiz/sonarqube-community/blob/main/docker-compose.yaml>

<https://github.com/Storresd/Docker-Sonarqube.git>