SonarQube with Docker compose: complete tutorial

Denis Verkhovskii · Follow
5 min read · Dec 22, 2023

Another one specific thing which I always try to give my students — the Code Review Buddy.

In this article I'm going to show how to setup a SonarQube instance using Docker and also how to use it in Java Applications.

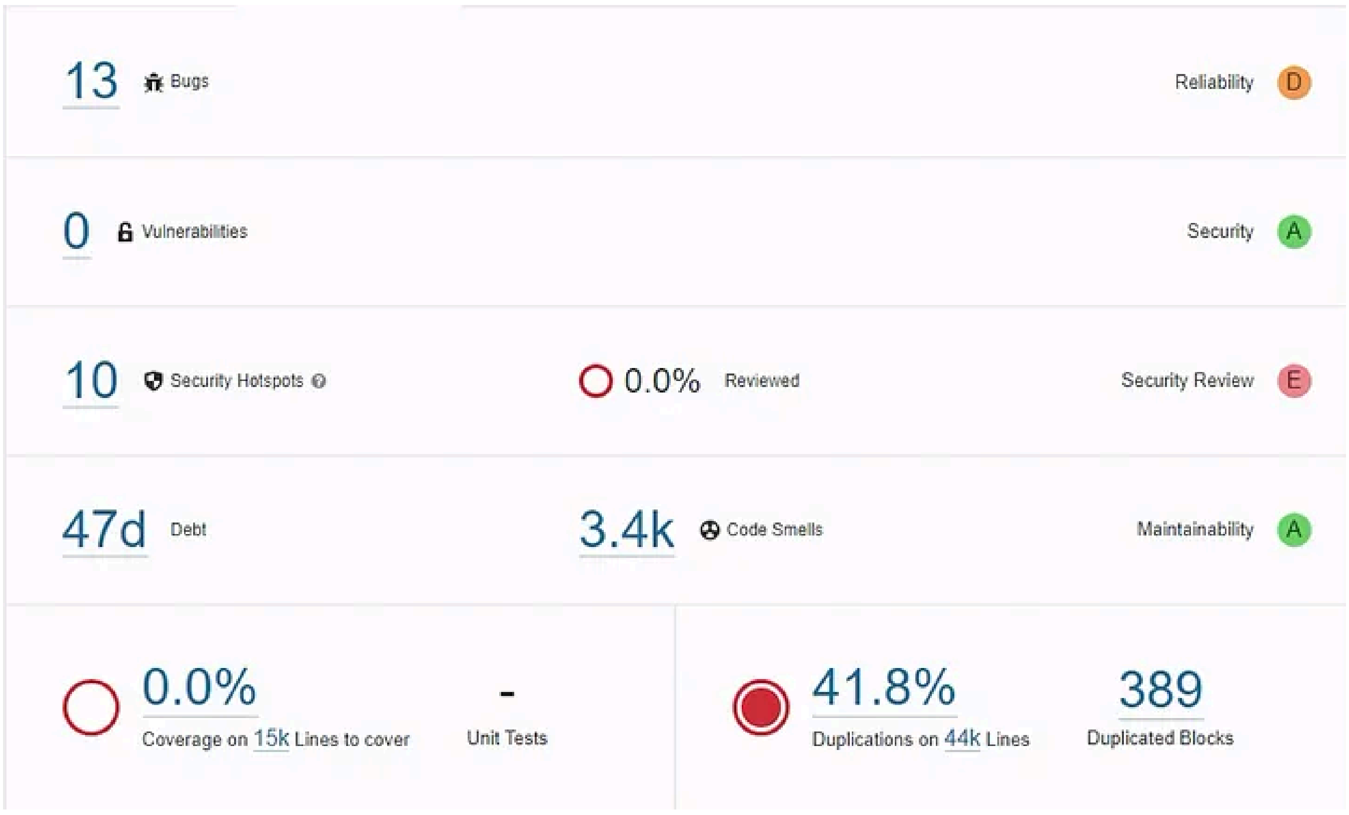
Mainly this article is written for the beginners, who want to install SonarQube locally, but the same applies for any other use case when the static code analysis is needed.

What is a SonarQube?

I like to refer to SonarQube as a Code Review Buddy, which helps your project to be nice and clean in terms of clean code, absence of silly bugs (which sometime happen even with the best developers) and vulnerabilities.

So, basically SonarQube is static analysis tool, which checks you code and signalises if somethings is wrong with it.

Here's an example of not really healthy project:



What do we see here?

- First of all SonarQube detected in the project **13 bugs**, which could cause a lot of problems.
- There are no **Vulnerabilities**, which means that your code uses safe versions of libraries. But sometimes if you're using well-known libaries there could be found a vulnerability, which makes the project insecure and could cause potential exploits by hackers. That's exactly what we want to prevent.
- **Security hotspots** also show the possible weak places in your code, which could be used by an attacker.

- **Code Smells** are the signals, that probably some code in the project is not optimal/readable or even written with bad practices.
- **Coverage** shows how many lines of the code are actually covered by tests
- **Duplications** — how many places we have with the same logic, but duplicated, obviously.

Prerequisites

First of all you'll need Docker to be installed on you machine. The complete installation guide could be found here: on the official site.

Once you've installed the Docker and Docker compose we can proceed with configuration of the SonarQube.

First thing first, we would need a docker-compose.yml

```
version: "3"

services:
  sonarqube:
    image: sonarqube:lts-community
    depends_on:
      - sonar_db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://sonar_db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    ports:
      - "9001:9000"
    volumes:
      - sonarqube_conf:/opt/sonarqube/conf
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
      - sonarqube_temp:/opt/sonarqube/temp

  sonar_db:
    image: postgres:13
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
      POSTGRES_DB: sonar
    volumes:
      - sonar_db:/var/lib/postgresql
      - sonar_db_data:/var/lib/postgresql/data
volumes:
  sonarqube_conf:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  sonarqube_temp:
  sonar_db:
  sonar_db_data:
```

What we have here?

- To run SonarQube we need two service: **SonarQube** itself and a **Database**.
- To be able to **keep our analysis results** we need to setup **volumes** as well. You might want to setup them in different location though, but for the simplicity we'll let docker itself to decide, where they would be located on you host machine. It's needed because by default if Docker container is removed for any reason, the data will be lost as well without volumes.
- OWASP dependency check plugin for SonarQube. This plugin helps to verify that your code doesn't have a vulnerabilities. The actual binary could be downloaded here.

Not it's time to hit the link!

Just type the http://localhost:9001/ in your browser and we'll be ready to start!

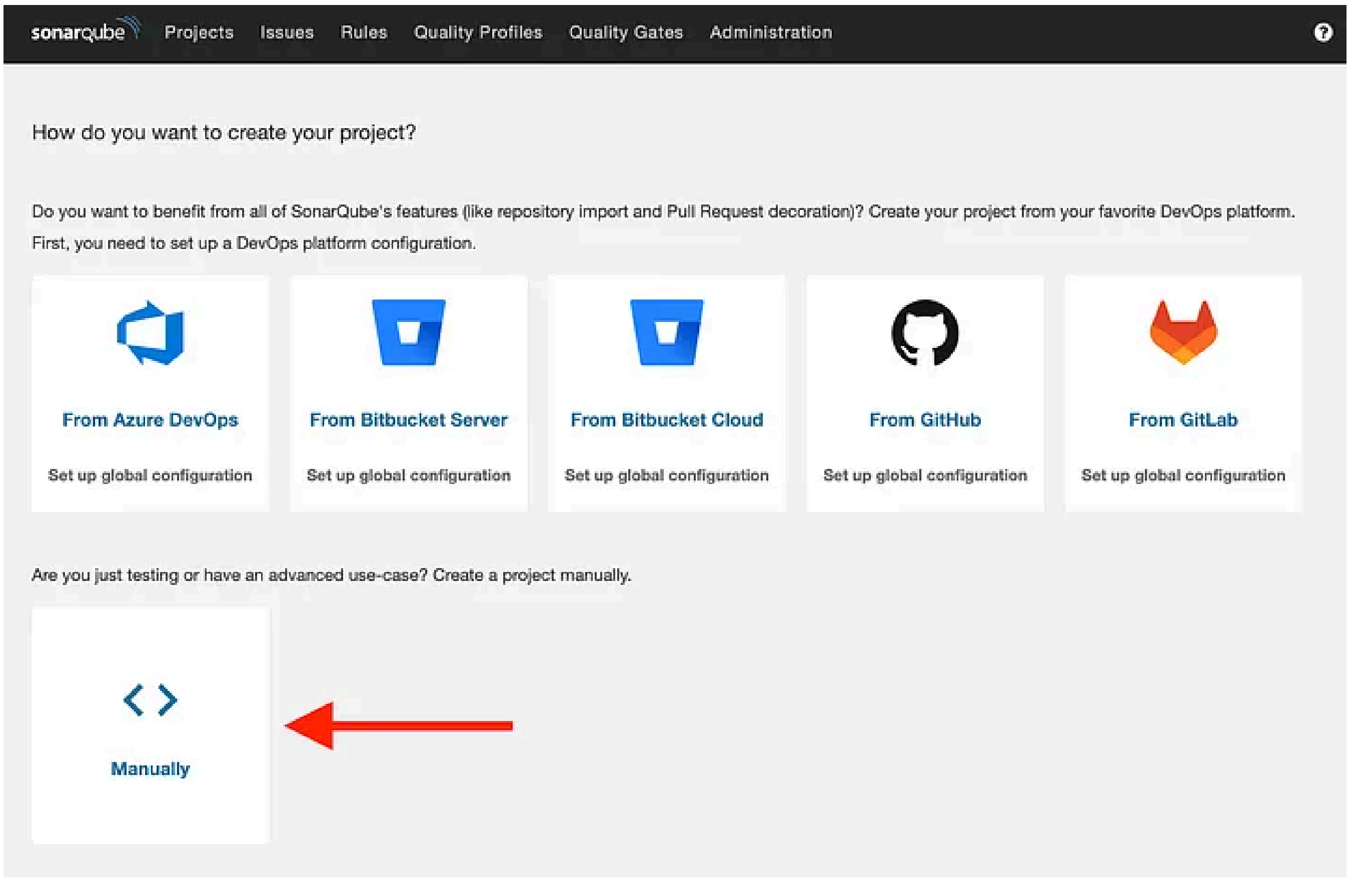
To login you'll need to use default credentials:

- **Login**: admin
- **Password**: admin

Configure project

There are several different ways, how you could setup your project in SonarQube.

We'll consider the local, which doesn't require an existent



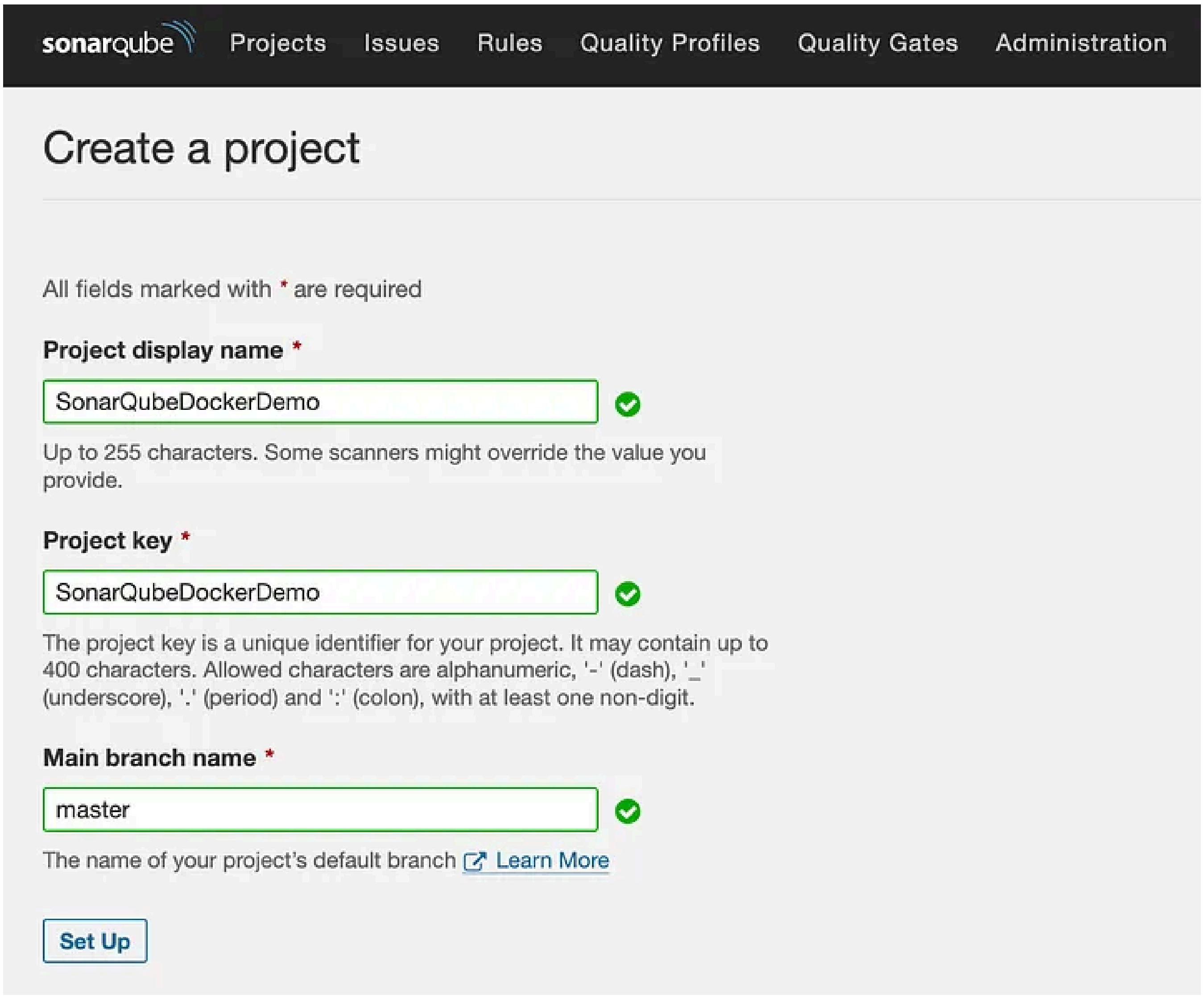
How do you want to create your project?

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 Server
Set up global configuration
From Bitbucket Cloud
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.

Just put here your project name and the default branch name. In my case it’s **master,** but in some projects it could be **main** or dev as **well**


sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Create a project

All fields marked with * are required

Project display name *

SonarQubeDockerDemo

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

Project key *

SonarQubeDockerDemo

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

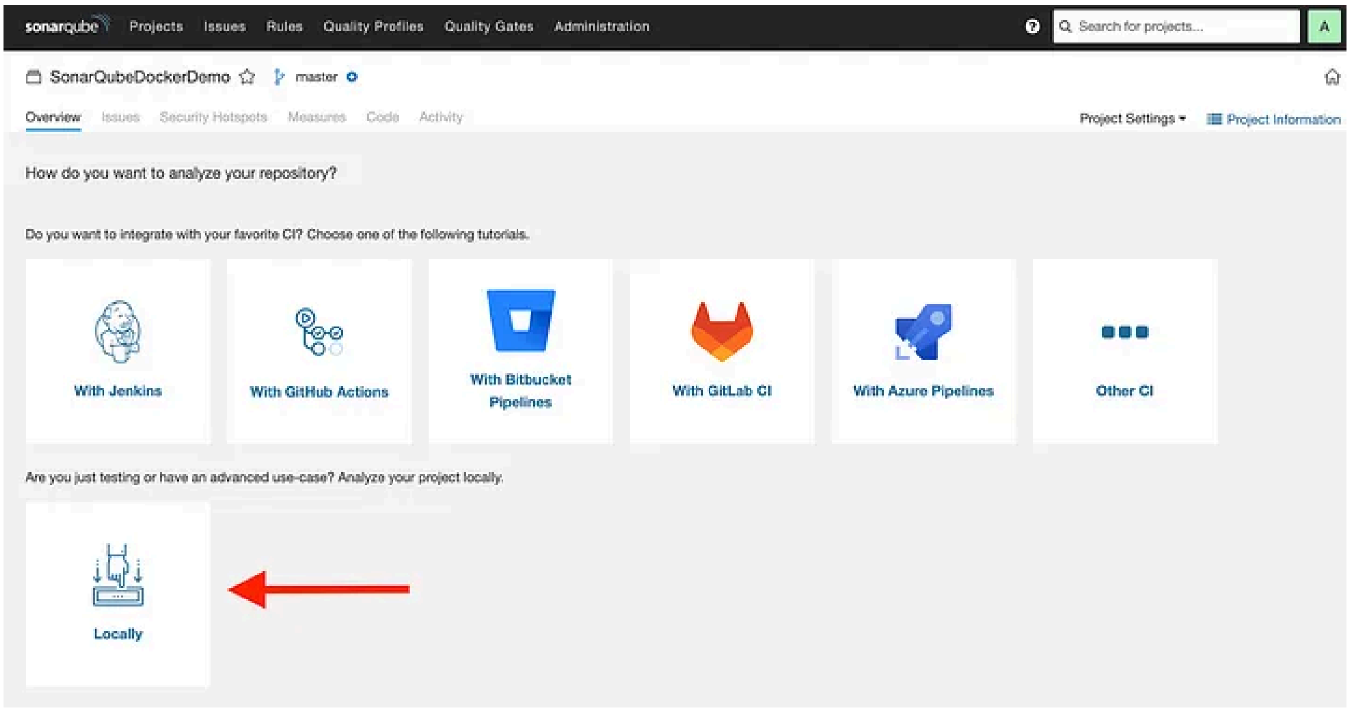
Main branch name *

master

The name of your project's default branch Learn More

Set Up

Next we'll need to setup a token. For the reason of simplicity again we'll use just Local setup, without any complex system.


sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

SonarQubeDockerDemo master

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

How do you want to analyze your repository?

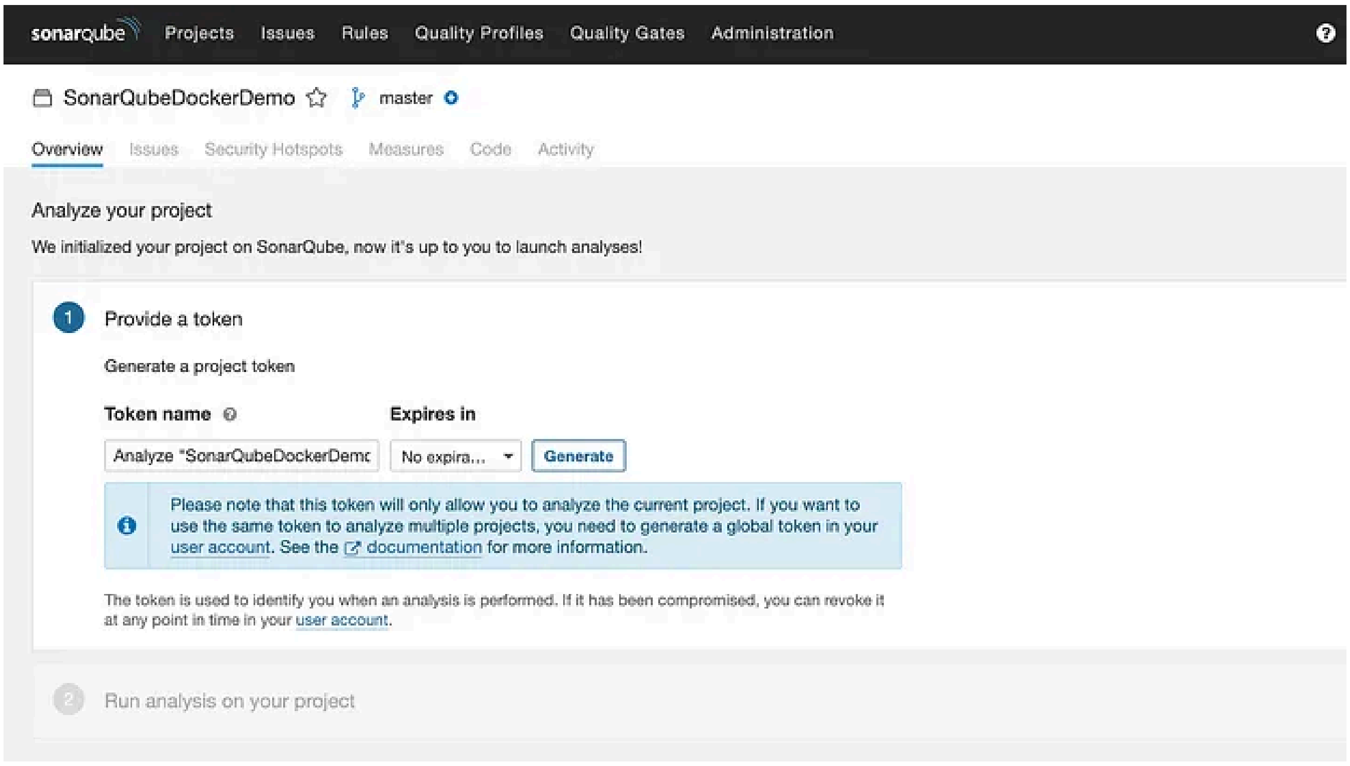
Do you want to integrate with your favorite CI? Choose one of the following tutorials.

With Jenkins | With GitHub Actions | With Bitbucket Pipelines | With GitLab CI | With Azure Pipelines | Other CI

Are you just testing or have an advanced use-case? Analyze your project locally.

Locally

Here we setup the **token** configurations. For the test purposes I suggest to check "**No expiration**" option.



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

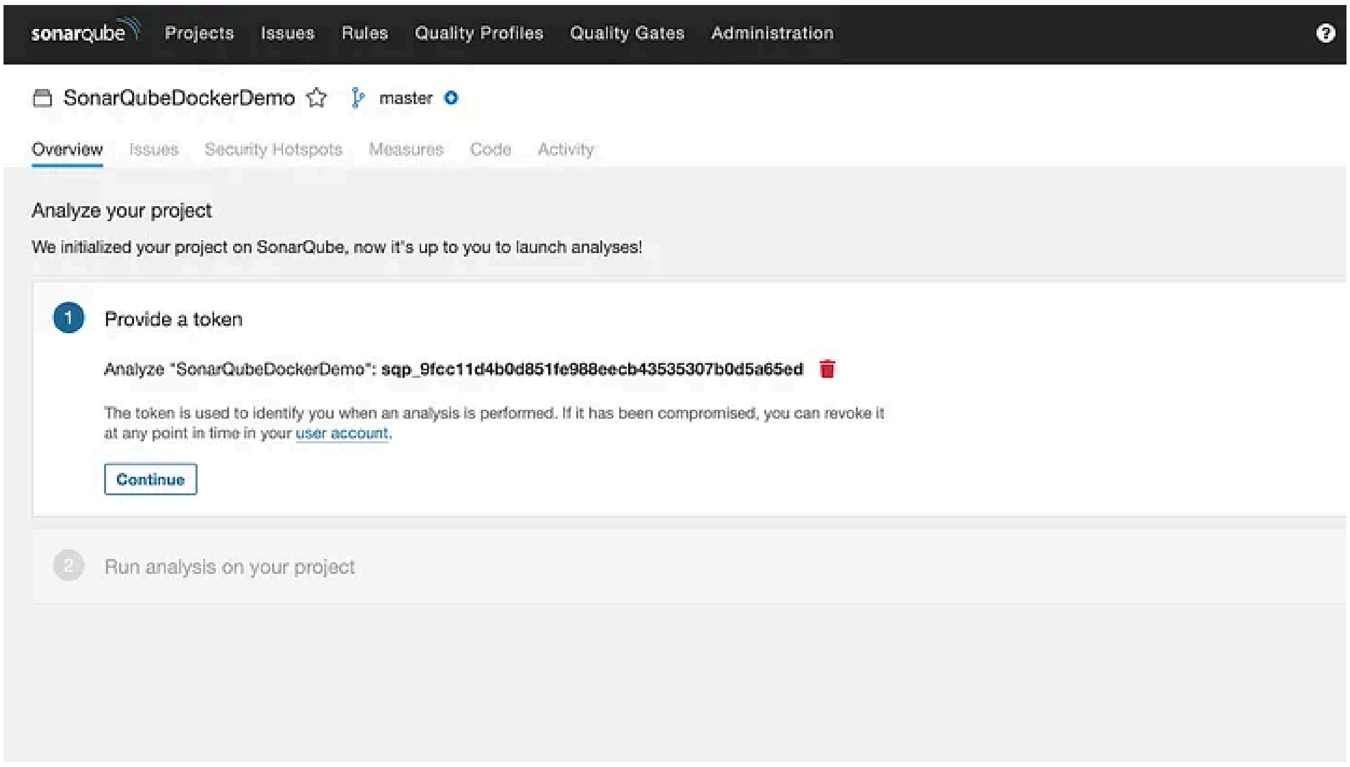
Token name: Analyze "SonarQubeDockerDemc
Expires in: No expira...
Generate

Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your user account. See the documentation for more information.

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.

2 Run analysis on your project

After previous step, you'll see the generated **token**. **Save it** somewhere to not loose. We'll need that for later.



Analyze your project

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

1 Provide a token

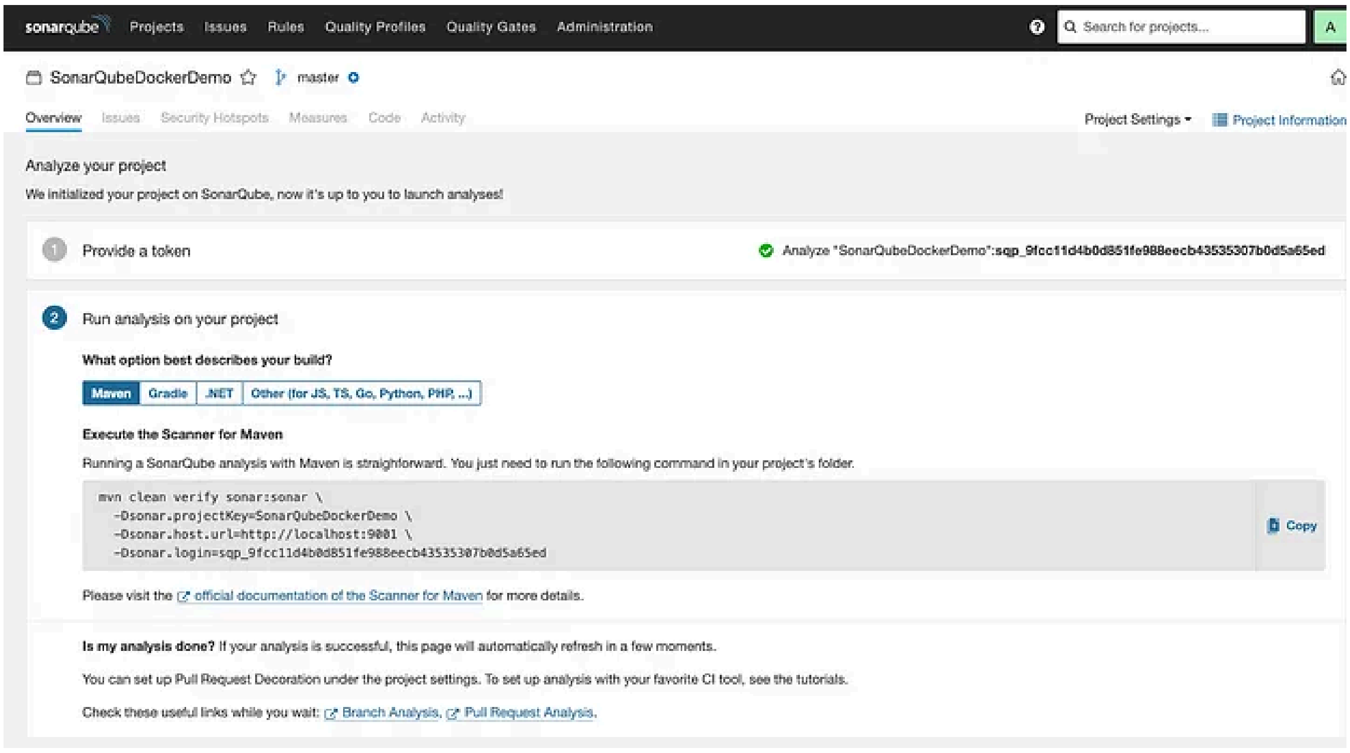
Analyze "SonarQubeDockerDemo": **sqp_9fcc11d4b0d851fe988eecb43535307b0d5a65ed**

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

2 Run analysis on your project

The last step is select the build system. In my case it's **Maven** and you could see, that SonarQube already suggests you the command, which would help you execute analysis later.



Analyze your project

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

1 Provide a token

✓ Analyze "SonarQubeDockerDemo":sqp_9fcc11d4b0d851fe988eecb43535307b0d5a65ed

2 Run analysis on your project

What option best describes your build?

[✓ Maven] Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

Execute the Scanner for Maven

Running a SonarQube analysis with Maven is straighforward. You just need to run the following command in your project's folder.

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=SonarQubeDockerDemo \
  -Dsonar.host.url=http://localhost:9001 \
  -Dsonar.login=sqp_9fcc11d4b0d851fe988eecb43535307b0d5a65ed
```

Copy

Please visit the official documentation of the Scanner for Maven for more details.

Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

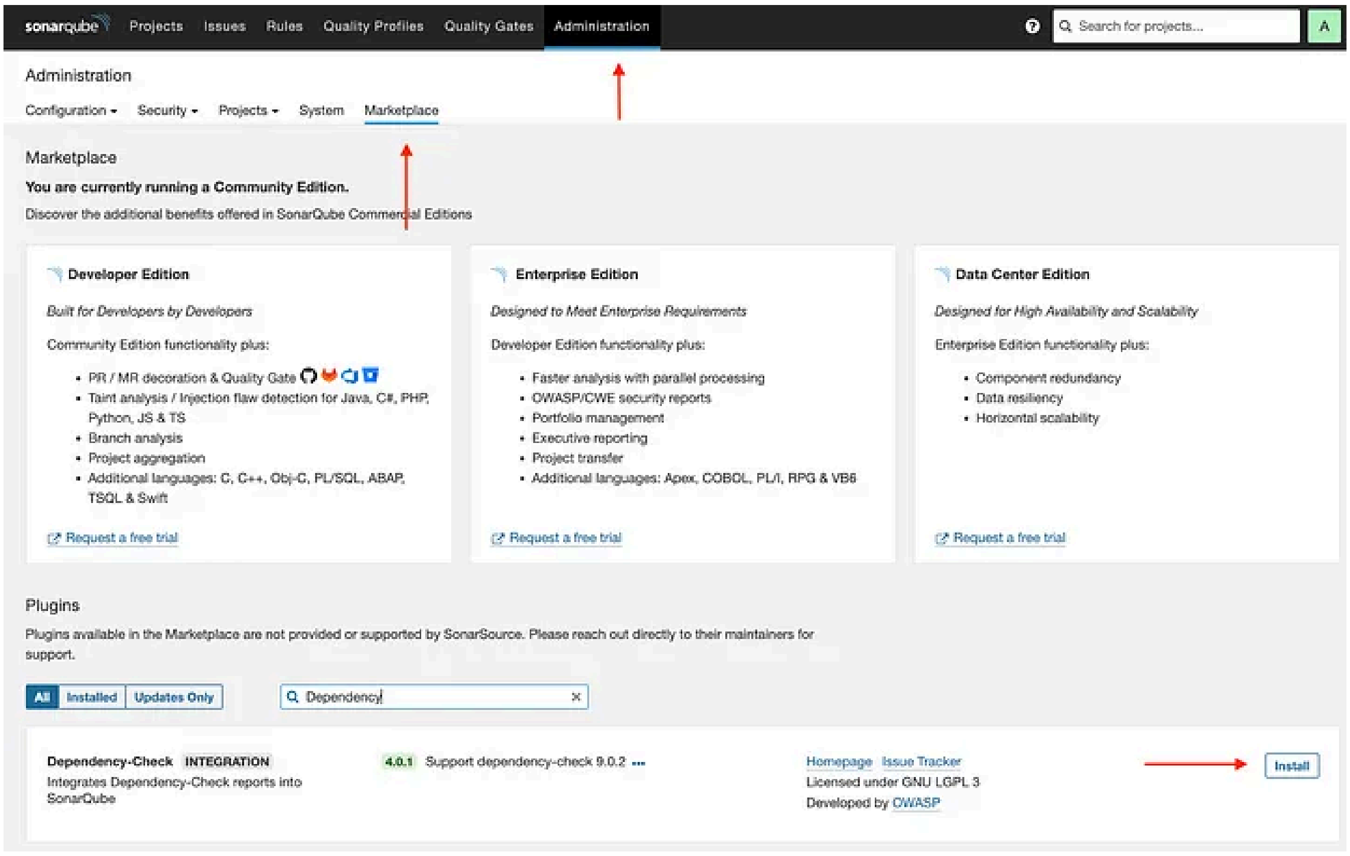
You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the tutorials.

Check these useful links while you wait: Branch Analysis, Pull Request Analysis.

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=SonarQubeDockerDemo \
  -Dsonar.host.url=http://localhost:9001 \
  -Dsonar.login=sqp_9fcc11d4b0d851fe988eecb43535307b0d5a65ed
```

OWASP Dependency check

Now we need to install OWASP Dependency check plugin. To do that you just need to go **Administration -> Marketplace** and search for the plugin, then click 'install'


sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Marketplace

You are currently running a Community Edition.

Discover the additional benefits offered in SonarQube Commercial Editions

Developer Edition

Built for Developers by Developers

Community Edition functionality plus:

- PR / MR decoration & Quality Gate
- Taint analysis / Injection flaw detection for Java, C#, PHP, Python, JS & TS
- Branch analysis
- Project aggregation
- Additional languages: C, C++, Obj-C, PL/SQL, ABAP, TSQL & Swift

Request a free trial

Enterprise Edition

Designed to Meet Enterprise Requirements

Developer Edition functionality plus:

- Faster analysis with parallel processing
- OWASP/CWE security reports
- Portfolio management
- Executive reporting
- Project transfer
- Additional languages: Apex, COBOL, PL/I, RPG & VB6

Request a free trial

Data Center Edition

Designed for High Availability and Scalability

Enterprise Edition functionality plus:

- Component redundancy
- Data resiliency
- Horizontal scalability

Request a free trial

Plugins

Plugins available in the Marketplace are not provided or supported by SonarSource. Please reach out directly to their maintainers for support.

All | Installed | Updates Only

Dependency

Dependency-Check INTEGRATION
Integrates Dependency-Check reports into SonarQube

4.0.1 Support dependency-check 9.0.2 ...

Homepage Issue Tracker
Licensed under GNU LGPL 3
Developed by OWASP

Install

Final step: configure project itself

First of all we'll need to setup properties:

```
<properties>
        <java.version>17</java.version>
        <jacoco.version>0.8.11</jacoco.version>

        <sonar.dependencyCheck.basePath>
            ${project.basedir}/owasp-dependency-check-logs
        </sonar.dependencyCheck.basePath>
        <sonar.dependencyCheck.htmlReportPath>
            ${sonar.dependencyCheck.basePath}/dependency-check-report.html
        </sonar.dependencyCheck.htmlReportPath>
        <sonar.dependencyCheck.jsonReportPath>
            ${sonar.dependencyCheck.basePath}/dependency-check-report.json
        </sonar.dependencyCheck.jsonReportPath>
        <sonar.dependencyCheck.summarize>true</sonar.dependencyCheck.summarize>

        <sonar.coverage.exclusions>**/controller/debug/**/*</sonar.coverage.exc

        <sonar.exclusions>src/test/**/*</sonar.exclusions>
        <sonar.sources>src,pom.xml</sonar.sources>
        <sonar.test.inclusions>src/test/**/*</sonar.test.inclusions>
        <sonar.tests>src</sonar.tests>
```

```
    </properties>
```

We also need to Jacoco as a dependency:

```
        <dependency>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.6</version>
        </dependency>
```

The build sections should have plugins for SonarQube, Jacoco and reports:

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.sonarsource.scanner.maven</groupId>
                <artifactId>sonar-maven-plugin</artifactId>
                <version>3.9.1.2184</version>
            </plugin>
            <plugin>
                <groupId>org.jacoco</groupId>
                <artifactId>jacoco-maven-plugin</artifactId>
                <version>${jacoco.version}</version>
                <executions>
                    <execution>
                        <id>jacoco-initialize</id>
                        <goals>
                            <goal>prepare-agent</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>jacoco-site</id>
                        <phase>package</phase>
                        <goals>
                            <goal>report</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>report</id>
                        <phase>test</phase>
                        <goals>
                            <goal>report</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.1.2</version> <!-- Use the latest version -->
            </plugin>
        </plugins>
    </build>
```

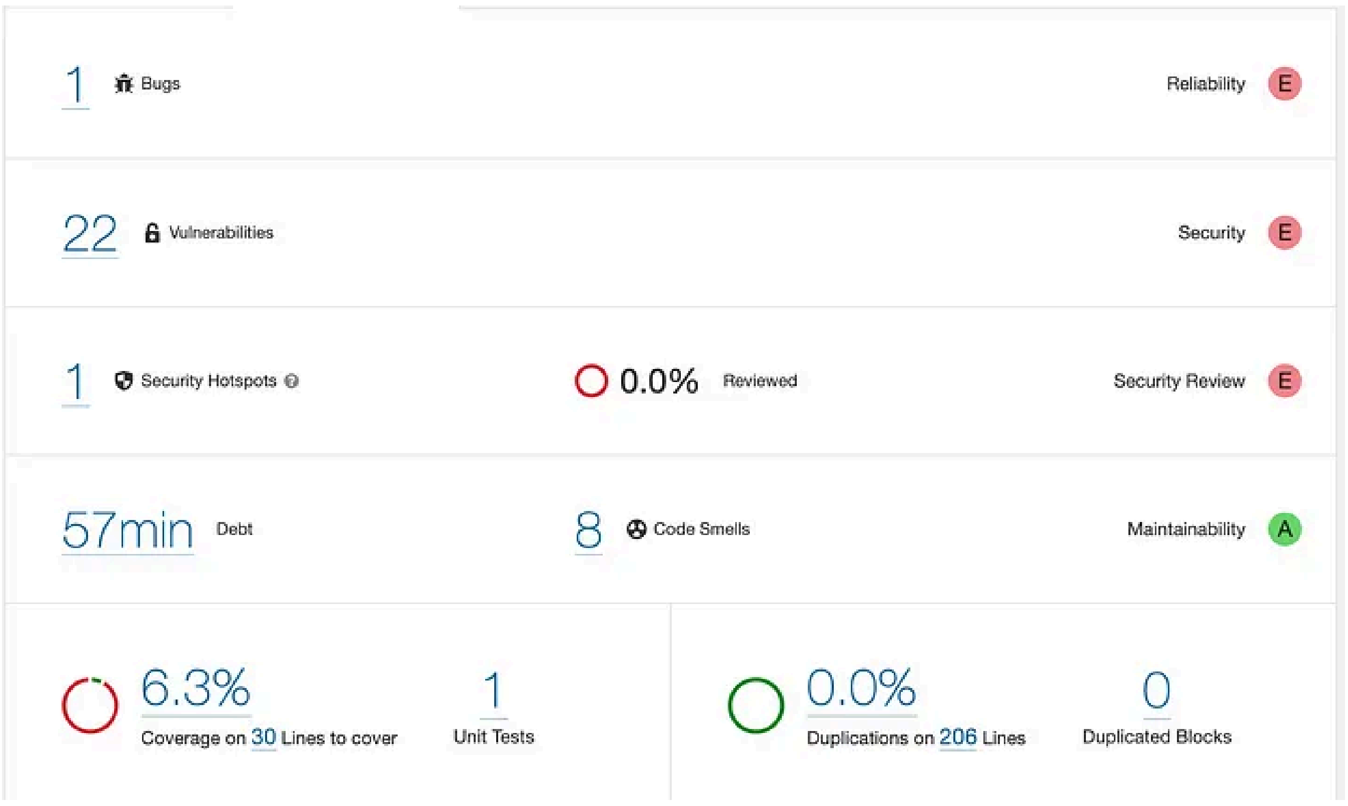
Finally we need setup a profile to generate OWASP reports:

```
<profiles>
    <profile>
        <id>sonarReports</id>

        <activation>
            <activeByDefault>false</activeByDefault>
        </activation>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.owasp</groupId>
                    <artifactId>dependency-check-maven</artifactId>
                    <version>6.5.3</version>

                    <configuration>
                        <mavenSettingsProxyId>https-p</mavenSettingsProxyIc
                        <outputDirectory>${sonar.dependencyCheck.basePath}<

                        <formats>
                            <format>html</format>
                            <format>json</format>
                        </formats>
                    </configuration>

                    <executions>
                        <execution>
                            <id>generate-dependency-check-report</id>

                            <goals>
                                <goal>aggregate</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
</profiles>
```

And now we would be ready to run the analysis:

```
# generate OWASP reports
mvn clean install
# generate OWASP reports
mvn dependency-check:aggregate -PsonarReports
# actual SonarQube analysis
mvn verify sonar:sonar \
  -Dsonar.projectKey=SonarQubeDockerDemo \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=sqp_9d4b556af1f2ca64db6e681fd7a7b14fc49af76c
```

For the project which I prepared specifically for this article, the results would be next:



If you like to download the project and test it out you can do it from Github repository here: https://github.com/DenisVerkhovsky/SonarQubeDockerDemo

Some useful links:

- SonarQube example project
- Docker images
- SonarQube site
- OWASP Dependency check plugin

Sonarqube Docker Compose Docker Sonar Owasp

D

Follow

Written by Denis Verkhovskii

22 Followers · 1 Following

Senior Software Engineer at T-Systems International. Java School Mentor. https://www.linkedin.com/in/dverkh/

No responses yet

What are your thoughts?

Respond