Check the quality of your code with SonarQube

What is SonarQube ?

SonarQube is a code review tool that automatically detects bugs, vulnerabilities and code smells in your code.

It can be integrated into your continuous integration workflow.

This tool can analyse the code quality of 27 programming languages including :

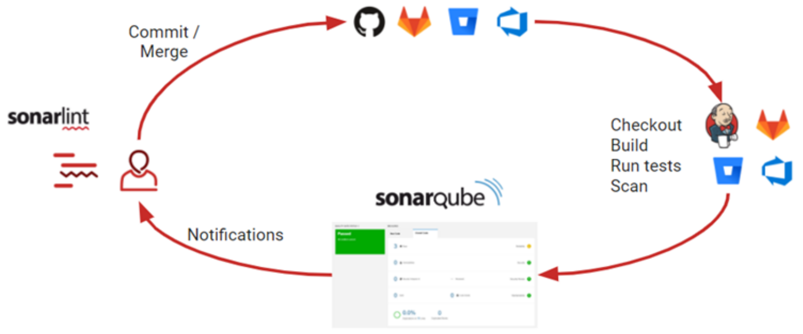
- Python
- JavaScript
- HTML5 / CSS 3
- Java
- PhP

Unfortunately it doesn't support R but plugins exist:

- sonar-r-plugin but it doesn't seem to be actively maintained.
- covr: can create a SonarQube Generic XML file for test coverage.

Overview

This section is based on 'Overview' section in sonarQube documentation (https://docs.sonarqube.org/latest/):



During the development of a project:

1. Developers write code in an IDE and then commit/merge their changes.

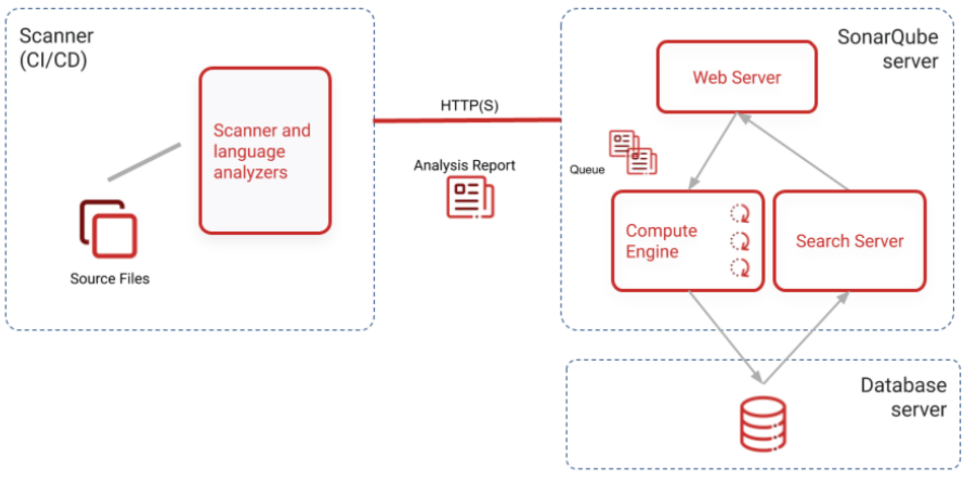
2. A continuous integration (CI) tool builds and runs unit tests, and an integrated SonarQube scanner analyses your changes

3. The scanner posts the results to the SonarQube server which provides feedback.

SonarQube installation

SonarQube instance components

A SonarQube instance is composed of three components:



1. The SonarQube Server. This server is composed of:
    - A web server that serves the SonarQube user interface
    - A search server that gets the data stored in a SonarQube database
    - A compute engine that processes code analysis report and saves it in a SonarQube database
2. A database server used to store:
    - The SonarQube configuration
    - The metrics and issues for code quality and security (supported database PostgreSQL, Microsoft SQL server and Oracle)
3. One or more scanners to analyse projects

Prerequisites to install SonarQube

To run sonarQube, you must have Java 11 installed on your machine.

A small instance of the SonarQube server requires 2GB or RAM to run efficiently.

A 64-bit system on the server side.

Free disk space to store code quality analyses.

For linux, you must ensure that:

- `vm.max_map_count` is greater than or equal to 524288
- `fs.file-max` is greater than or equal to 131072
- the user running SonarQube can open at least 131072 file descriptors
- the user running SonarQube can open at least 8192 threads

Command to check this.

```
sysctl vm.max_map_count
sysctl fs.file-max
ulimit -n
ulimit -u
```

You can set the `vm.max_map_count` and `fs.file-max` values permanently by editing `/etc/sysctl.conf`: `echo "vm.max_map_count=524288" >> /etc/sysctl.conf && echo "fs.file-max=131072" >> /etc/sysctl.conf`. Then reload the configuration with the new value with `sudo sysctl -p`.

If the user running SonarQube does not have the permission to have at least 131072 open descriptors, you must insert those lines in `/etc/security/limits.conf`:

```
sonarqube   -   nofile   131072
sonarqube   -   nproc    8192
```

Quick installation of the SonarQube server

SonarQube can be installed quickly to try it by using a docker image or an executable file. This is very useful to quickly getting started with SonarQube. However, if you want to upgrade it, you can lose all your code analyses.

If you only want to test sonarQube you can read the page Try Out SonarQube in the documentation.

Production installation of the SonarQube server

You can install SonarQube on a clustered setup or on a single node setup. To see how to install SonarQube on a clustered setup you can read this page from the documentation

Installation with docker and PostgreSQL

1. Create docker volumes

To use SonarQube with docker you have to launch the following commands:

```
$ docker volume create --name sonarqube_data
$ docker volume create --name sonarqube_logs
$ docker volume create --name sonarqube_extensions
$ docker volume create --name postgres-data
```

These commands allow to create the necessary volumes to avoid losing data when the SonarQube container is deleted. For example, when you upgrade SonarQube.

2. Create a Postgresql database

To create and configure PostgreSQL you can type the following command:

```
$ docker container run --restart=always --name postgresql -v postgres-
data:/var/lib/postgresql/data -p 5432:5432 -d -e POSTGRES_USER=sonar -e
```

```
POSTGRES_PASSWORD=sonar -e POSTGRES_DB=sonardb postgres:10-alpine
```

Command details:

- `docker container run`: Command used to launch the container
- `--restart=always`: Always restart the container when it is stopped
- `--name postgresql`: Gives the name postgresql to this container
- `-v postgres-data:/var/lib/postgresql/data`: The database will be stored on the volume postgres-data. If this container is deleted, the data will be preserved.
- `-p 5432:5432`: We link the port 5432 of the container to the port 5432 of the computer.
- `-d` : We execute the container as a background task
- `-e POSTGRES_USER=sonar`: We set up the user sonar via an environment variable
- `-e POSTGRES_PASSWORD=sonar`: We set up the password via an environment variable. Replace it with your real password.
- `-e POSTGRES_DB=sonardb`: We name the database via an environment variable
- `postgres:10-alpine`: The image containing the postgresql container.

Of course, you can create your PostgreSQL by using another method. Without docker or using a different configuration method.

4. Configure and run the SonarQube container

To configure the SonarQube container, you can use the following commands:

```
$ # docker container run -d --name sonarserver \
    -p 9000:9000 \
    -v sonarqube_data:/opt/sonarqube/data \
    -v sonarqube_extensions:/opt/sonarqube/extensions \
    -v sonarqube_logs:/opt/sonarqube/logs \
    -v sonarqube_confs:/opt/sonarqube/conf \
    sonarqube:latest
$ # The previous command allows to launch the SonarQube server that builds
the configuration file sonar.properties.
$ docker container stop sonarserver # Stop the container
```

Details of the previous command:

- `docker container run -d --name sonarserver`: Runs a container in background and name it sonarserver
- `-p 9000:9000`: Binds the port 9000 on the computer to the same port on the container
- `-v sonarqube_data:/opt/sonarqube/data`: Mounts the volume `sonarqube_data` into the directory `/opt/sonarqube/data` in the container. Note that this volume contains the PostgreSLQ database.
- `-v sonarqube_extensions:/opt/sonarqube/extensions`: Mounts the volume `sonarqube_extensions` into the directory `/opt/sonarqube/data` in the container. Note that this volume contains the Postgresql database.

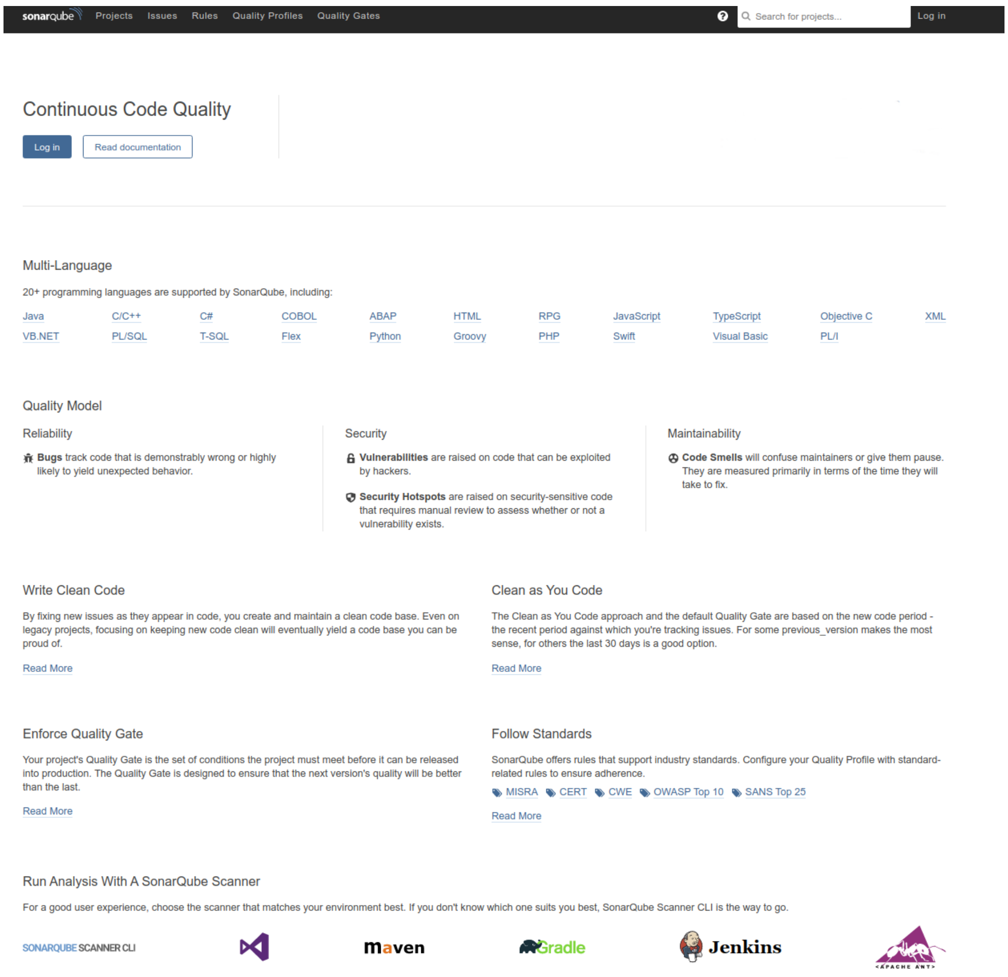
- `-v sonarqube_logs:/opt/sonarqube/logs`: Mounts the volume `sonarqube_logs` into the directory `/opt/sonarqube/logs` in the container. SonarQube logs will be stored here.
- `-v sonarqube_confs:/opt/sonarqube/conf`: Mounts the volume `sonarqube_confs` into the directory `/opt/sonarqube/conf` in the container. Configurations of SonarQube will be stored here.

```
$ docker volume inspect sonarqube_confs # Command used to inspect the
volume sonarqube_confs. This is the volume that will be used by SonarQube
to store its configuration. By inspecting it, we can see where the
configuration data are going to be stored.
[
    {
        "CreatedAt": "2020-11-19T16:01:12+01:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/sonarqube_confs/_data",
        "Name": "sonarqube_confs",
        "Options": {},
        "Scope": "local"
    }
]
$ sudo nano /var/lib/docker/volumes/sonarqube_confs/_data/sonar.properties
$ # Edit and uncomment the following lines:
$ # sonar.jdbc.username=sonar
$ # sonar.jdbc.password=sonar # replace it by the real password
$ # sonar.jdbc.url=jdbc:postgresql://192.168.1.58:5432/sonardb
```

Then, restart the container:

```
$ docker container start sonarserver
```

Type `http://localhost:9000/` and you should see the following site:



Continuous Code Quality

Log in | Read documentation

Multi-Language

20+ programming languages are supported by SonarQube, including:

Java	C/C++	C#	COBOL	ABAP	HTML	RPG	JavaScript	TypeScript	Objective C	XML
VB.NET	PL/SQL	T-SQL	Flex	Python	Groovy	PHP	Swift	Visual Basic	PL/I	

Quality Model

Reliability

Bugs track code that is demonstrably wrong or highly likely to yield unexpected behavior.

Security

Vulnerabilities are raised on code that can be exploited by hackers.

Security Hotspots are raised on security-sensitive code that requires manual review to assess whether or not a vulnerability exists.

Maintainability

Code Smells will confuse maintainers or give them pause. They are measured primarily in terms of the time they will take to fix.

Write Clean Code

By fixing new issues as they appear in code, you create and maintain a clean code base. Even on legacy projects, focusing on keeping new code clean will eventually yield a code base you can be proud of.

Read More

Clean as You Code

The Clean as You Code approach and the default Quality Gate are based on the new code period - the recent period against which you're tracking issues. For some previous_version makes the most sense, for others the last 30 days is a good option.

Read More

Enforce Quality Gate

Your project's Quality Gate is the set of conditions the project must meet before it can be released into production. The Quality Gate is designed to ensure that the next version's quality will be better than the last.

Read More

Follow Standards

SonarQube offers rules that support industry standards. Configure your Quality Profile with standard-related rules to ensure adherence.

MISRA CERT CWE OWASP Top 10 SANS Top 25

Read More

Run Analysis With A SonarQube Scanner

For a good user experience, choose the scanner that matches your environment best. If you don't know which one suits you best, SonarQube Scanner CLI is the way to go.

SONARQUBE SCANNER CLI | maven | Gradle | Jenkins | APACHE ANT

Other installation methods (SonarQube server):

- You can use docker-compose to easily set up a multi-container SonarQube application.
- You can also install SonarQube from a Zip file

To install SonarQube with one of these method, you can check the documentation: https://docs.sonarqube.org/latest/setup/install-server/

Installation of sonar-scanner

To use sonar-scanner, you have to download it first. You can download it here.

Extract the content of the `.zip` file in a given folder and then add the path of the sonar-scanner bin folder in your path permanently by editing your `~/.bashrc` file and adding:

```
export Path=$PATH:/path_to_sonar/bin
```

Then, if you open a new terminal, you will be able to run a `sonnar-scanner command`.

Note that you can also install and run `sonnar-scanner` with a docker container:

```
docker container run \
    --rm \
    -e SONAR_HOST_URL="http://localhost:9000" \
    -v "${YOUR_REPO}:/usr/src" \
    sonarsource/sonar-scanner-cli
```

- `--rm`: the container is removed once it stops
- `-e SONAR_HOST_URL="http://localhost:9000"`: setting an environment variable linking to the SonarQube Server
- `-v ${YOUR_REPO}:/usr/src` : Mount your repository on `/usr/src` in the container to perform the code analysis

Analyse your code quality with sonar-scanner.

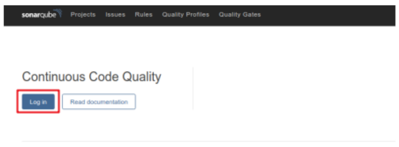
With a command line

Here is a simple project to demonstrate how to use SonarQube. This simple repository can be found here: `https://gitbio.ens-lyon.fr/nfontrod/calculator` (tag: `tag1`). In the folder `src/` there is a file named `calculator.py` which contains this code:

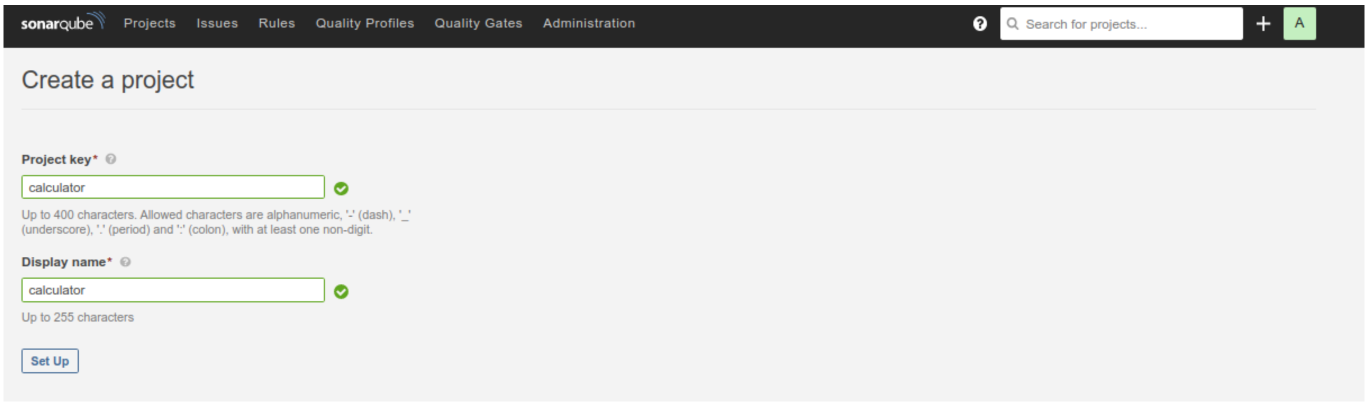
```python
#!/usr/bin/env python3

# -*- coding: UTF-8 -*-

"""
Description: calculator function
"""



def Additions(a: float, b: float):
    """
    Add two number a and b

    :param a: A first number
    :param b: A second number
    :return: The sum of a and b
    """
    return a + b


if __name__ == "__main__":
    Additions(5, 17)
```

We want to check the quality of this code:

- Go to `http://localhost:9000` and login. The default username and password are `admin`.



- Create a new project by clicking on the `+` in the top right corner and give the project a name.



Create a project

Project key*

calculator

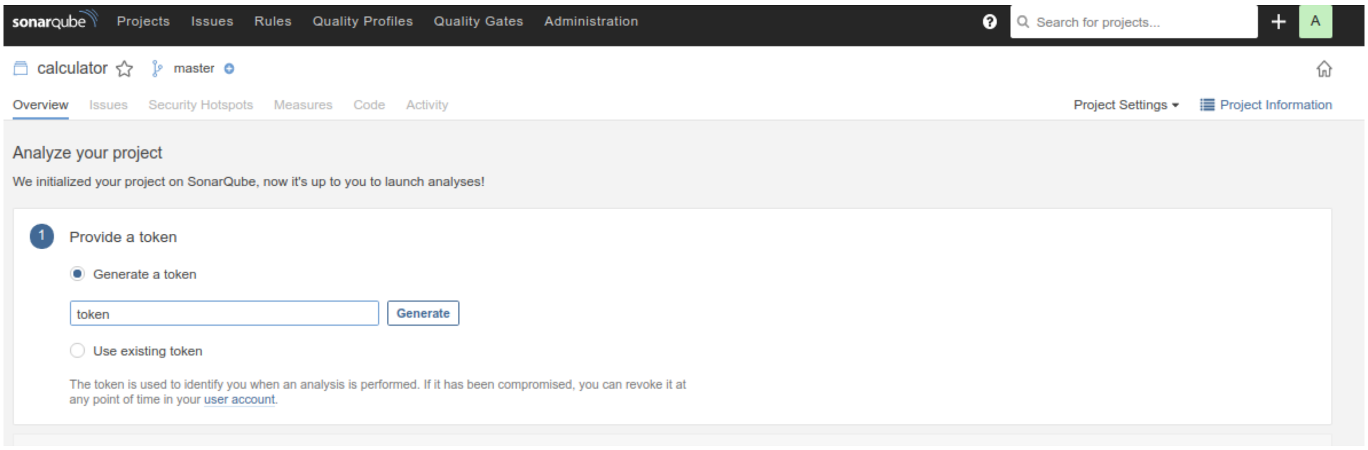
Up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Display name*

calculator

Up to 255 characters

- Generate a new token and copy it.



Analyze your project

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

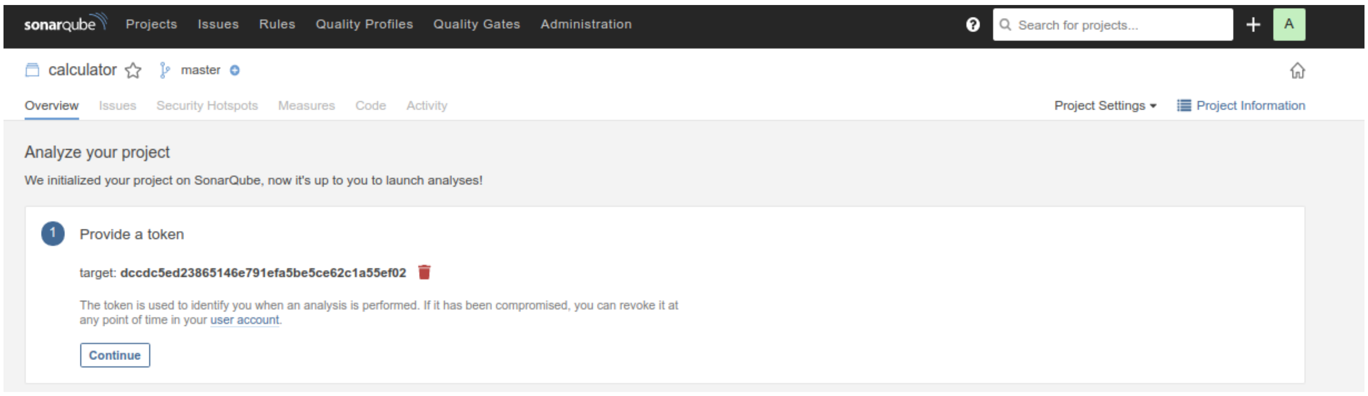
Provide a token

● Generate a token

token

○ Use existing token

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



Analyze your project

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

Provide a token

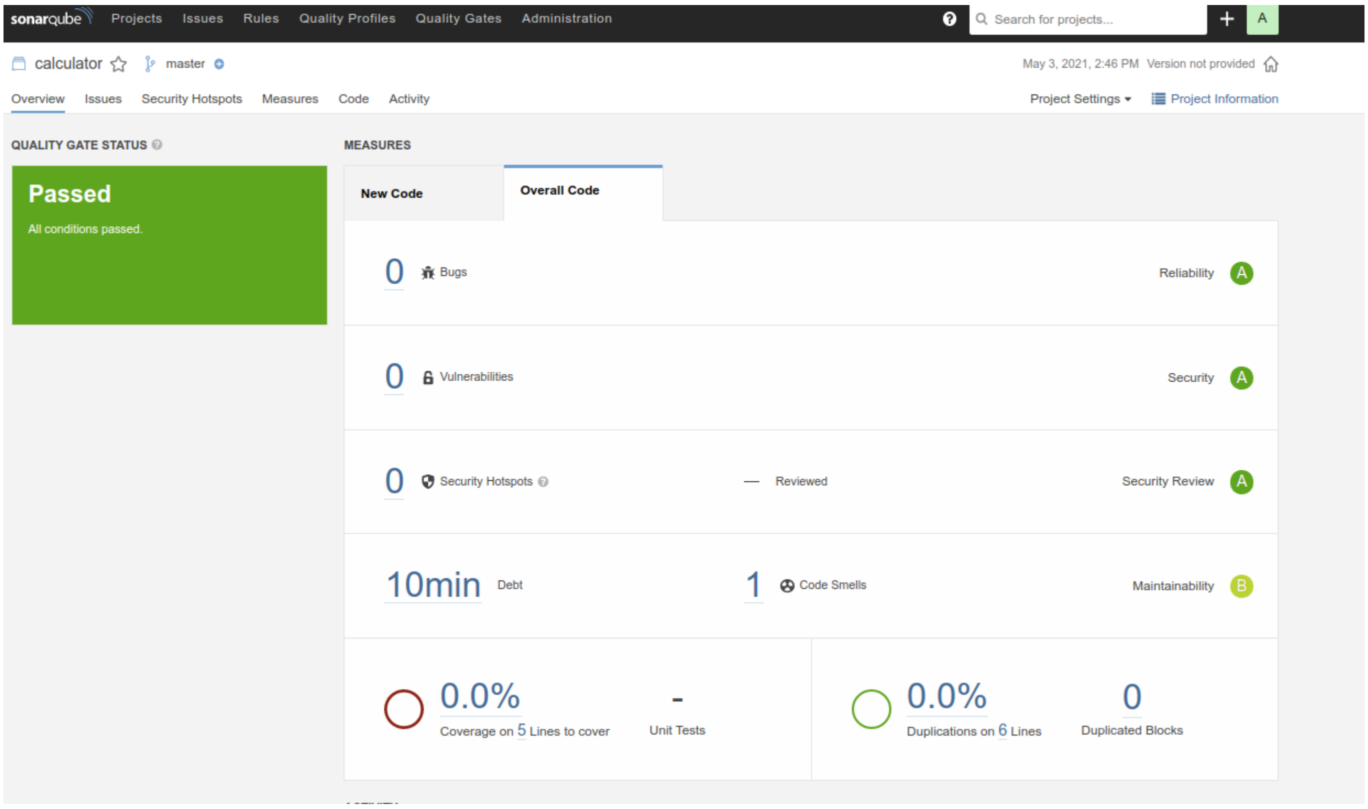
target: **dccdc5ed23865146e791efa5be5ce62c1a55ef02**

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

- Go on your project directory (`calculator` in this example) and execute the following command.

```
sonar-scanner   -Dsonar.projectKey=calculator   -Dsonar.sources=.   -
Dsonar.host.url=http://localhost:9000   -
Dsonar.login=dccdc5ed23865146e791efa5be5ce62c1a55ef02
```

- Go on the SonarQube server (http://localhost:9000) to see the result:



QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

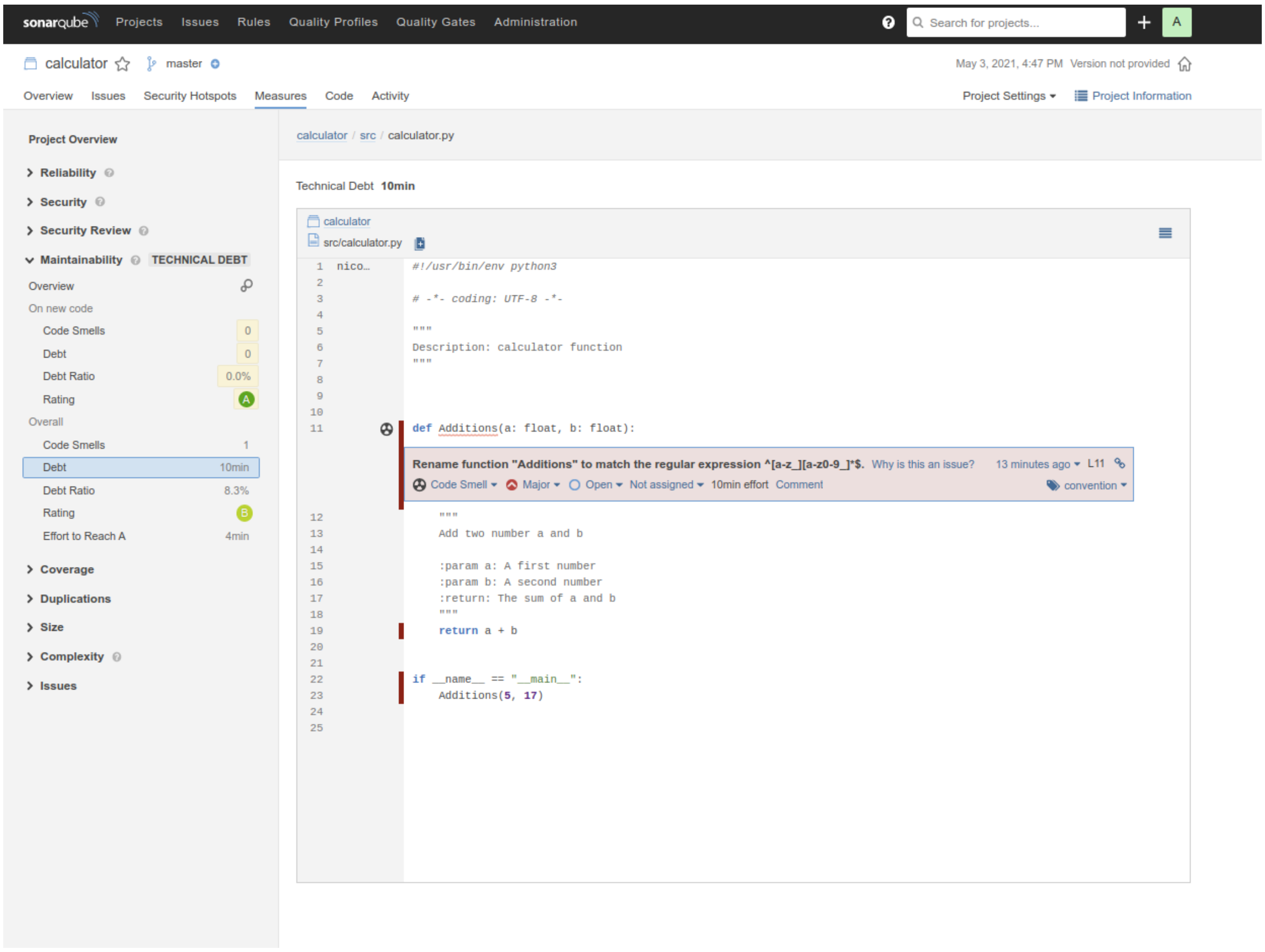
0 Security Hotspots — Reviewed — Security Review A

10min Debt — 1 Code Smells — Maintainability B

0.0% Coverage on 5 Lines to cover — - Unit Tests

0.0% Duplications on 6 Lines — 0 Duplicated Blocks

by clicking on `10min debt` then on `src/calculator.py` you shoud see something like this:



Overview	
On new code	
Code Smells	0
Debt	0
Debt Ratio	0.0%
Rating	A
Overall	
Code Smells	1
Debt	10min
Debt Ratio	8.3%
Rating	B
Effort to Reach A	4min

calculator / src / calculator.py

Technical Debt **10min**

calculator
src/calculator.py

```
#!/usr/bin/env python3

# -*- coding: UTF-8 -*-

"""
Description: calculator function
"""



def Additions(a: float, b: float):
```

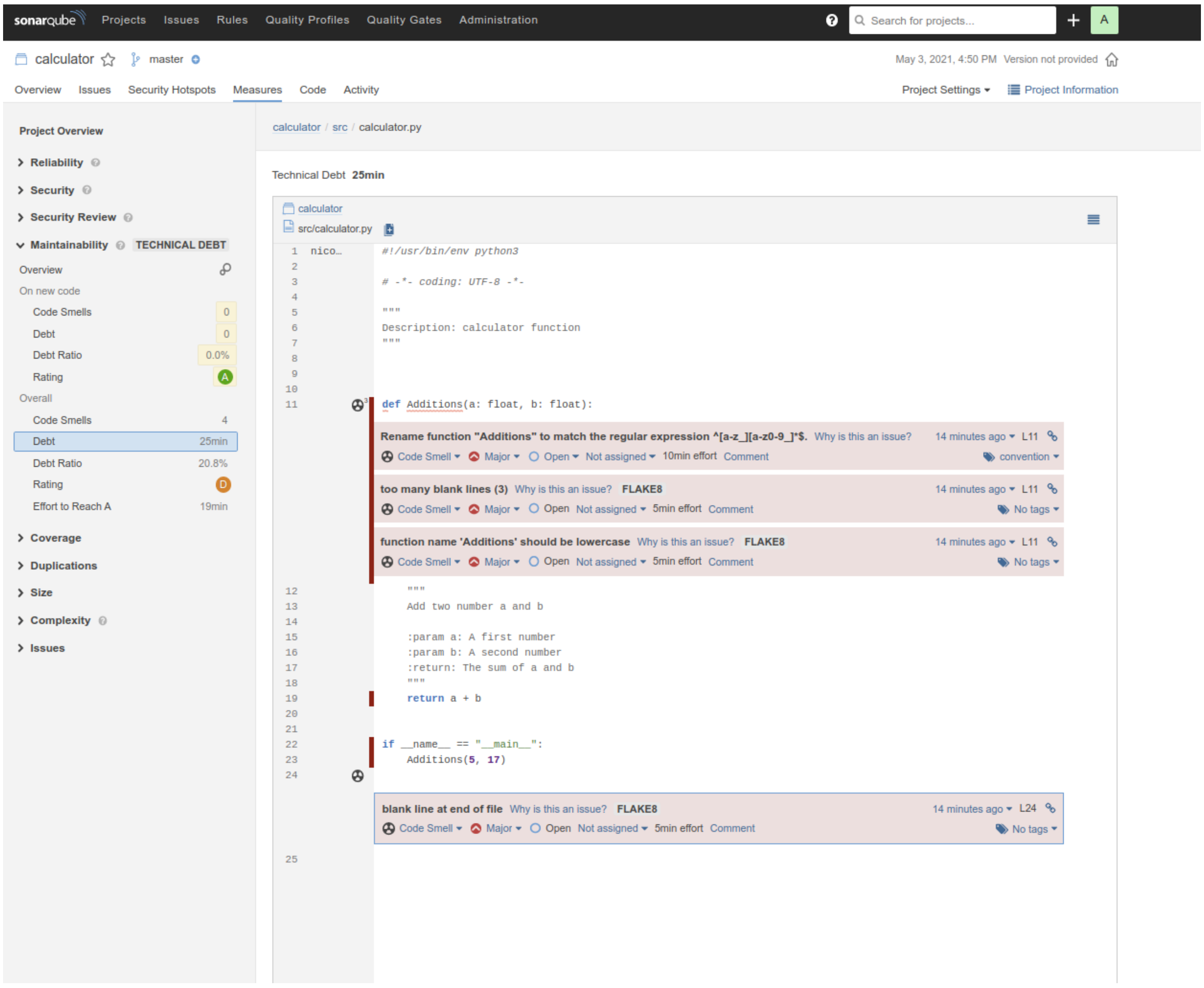
Rename function "Additions" to match the regular expression ^[a-z_][a-z0-9_]*$. Why is this an issue? 13 minutes ago L11
Code Smell · Major · Open · Not assigned · 10min effort · Comment · convention

```
    """
    Add two number a and b

    :param a: A first number
    :param b: A second number
    :return: The sum of a and b
    """
    return a + b


if __name__ == "__main__":
    Additions(5, 17)
```

SonarQube detects that the function `Additions` should not start with a capital letter. It's good but SonarQube does not detect pep8 problems: indeed, there are 3 line breaks between the description and the function. To detect them, flake8 can be used. flake8 can be installed by typing `pip install flake8`

- To display flake8 output in SonarQube you can enter:

```
$ flake8 --exit-zero --output-file .flake8-report.txt # Generate a flake8
report
$ sonar-scanner  -Dsonar.projectKey=calculator  -Dsonar.sources=.  -
Dsonar.host.url=http://localhost:9000  -
Dsonar.login=dccdc5ed23865146e791efa5be5ce62c1a55ef02 -
Dsonar.python.flake8.reportPaths=.flake8-report.txt # sonnar-scanner
command that uses the flake8 report
```



Overview	
On new code	
Code Smells	0
Debt	0
Debt Ratio	0.0%
Rating	A
Overall	
Code Smells	4
Debt	25min
Debt Ratio	20.8%
Rating	D
Effort to Reach A	19min

calculator / src / calculator.py

Technical Debt 25min

calculator
src/calculator.py

```
#!/usr/bin/env python3

# -*- coding: UTF-8 -*-

"""
Description: calculator function
"""



def Additions(a: float, b: float):
```

Rename function "Additions" to match the regular expression ^[a-z_][a-z0-9_]*$. Why is this an issue? 14 minutes ago L11
Code Smell Major Open Not assigned 10min effort Comment convention

too many blank lines (3) Why is this an issue? FLAKE8 14 minutes ago L11
Code Smell Major Open Not assigned 5min effort Comment No tags

function name 'Additions' should be lowercase Why is this an issue? FLAKE8 14 minutes ago L11
Code Smell Major Open Not assigned 5min effort Comment No tags

```
    """
    Add two number a and b

    :param a: A first number
    :param b: A second number
    :return: The sum of a and b
    """
    return a + b


if __name__ == "__main__":
    Additions(5, 17)

```

blank line at end of file Why is this an issue? FLAKE8 14 minutes ago L24
Code Smell Major Open Not assigned 5min effort Comment No tags

Now, we have pep8 code smells ! But we still don't test anything, so no code is covered by tests ! Let's add some tests. In `tag2` of the project, an new `tests/test.py` file has been created:

```
#!/usr/bin/env python3

# -*- coding: UTF-8 -*-
```

```
"""
Description:
"""

import doctest
import unittest


def load_tests(loader, tests, ignore):
    tests.addTest(doctest.DocTestSuite('src.calculator'))
    return tests


if __name__ == "__main__":
    unittest.main()
```

A docstring test has been added in `src/calculator.py`

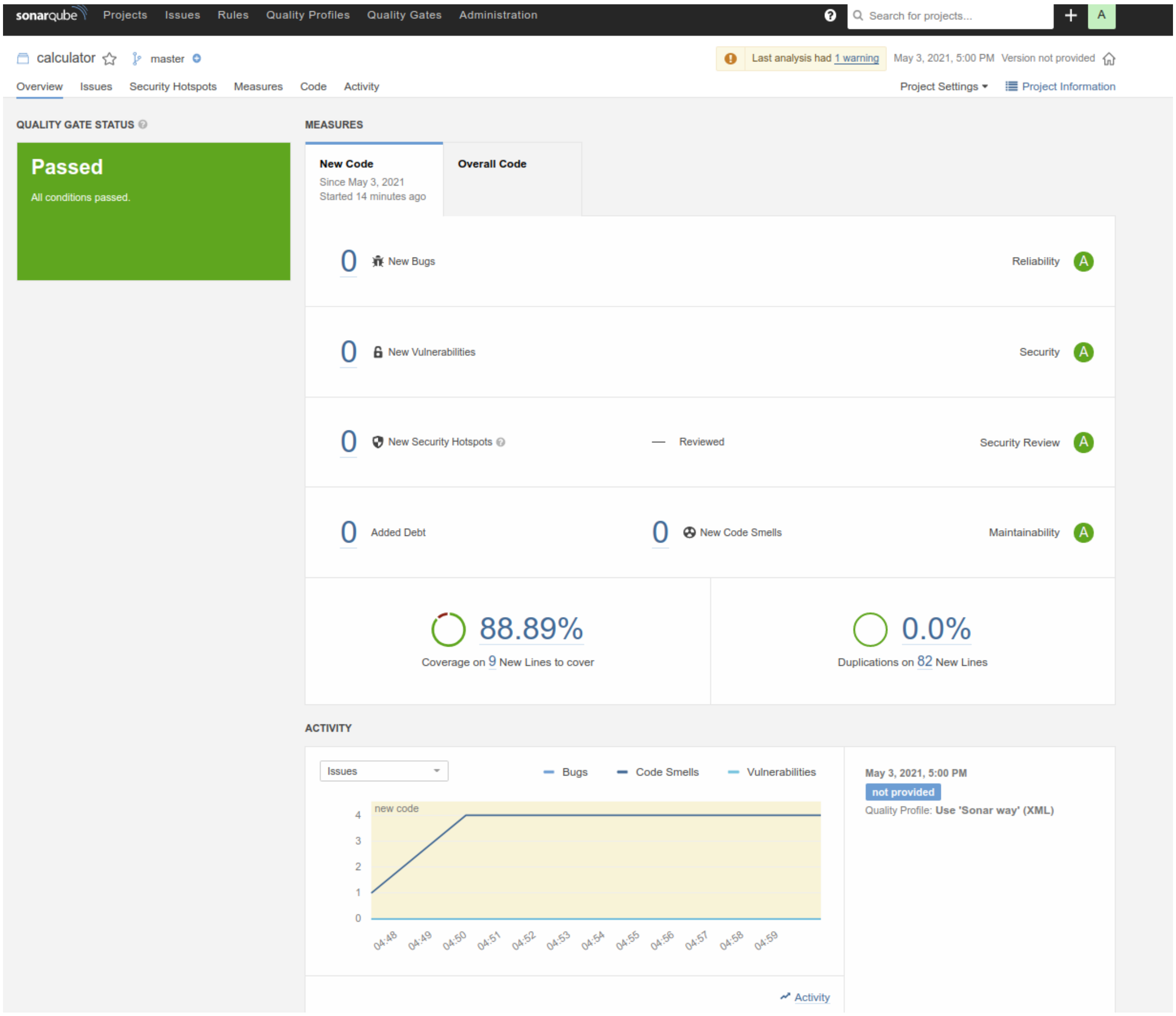
```
def Additions(a: float, b: float):
    """
    Add two number a and b

    :param a: A first number
    :param b: A second number
    :return: The sum of a and b

    >>> Additions(5, 15)
    20
    """
    return a + b
```

- To run a coverage analysis, you can install the coverage module with `pip install coverage`. Then, to build a coverage report, enter these two commands:

```
$ coverage run -m unittest discover # Find and run unittest in a file named
test.py inside a module. This creates a .coverage file
$ coverage xml # Create a coverage.xml file from a .coverage file
```

Then, we can restart an analysis by typing:

```
sonar-scanner   -Dsonar.projectKey=calculator   -Dsonar.sources=.   -
Dsonar.host.url=http://localhost:9000   -
Dsonar.login=dccdc5ed23865146e791efa5be5ce62c1a55ef02 -
Dsonar.python.flake8.reportPaths=.flake8-report.txt -
Dsonar.python.coverage.reportPaths=./coverage.xml
```



calculator master

Last analysis had 1 warning May 3, 2021, 5:00 PM Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code
Since May 3, 2021
Started 14 minutes ago

Overall Code

0 New Bugs — Reliability A

0 New Vulnerabilities — Security A

0 New Security Hotspots — Reviewed — Security Review A

0 Added Debt — 0 New Code Smells — Maintainability A

88.89% Coverage on 9 New Lines to cover

0.0% Duplications on 82 New Lines

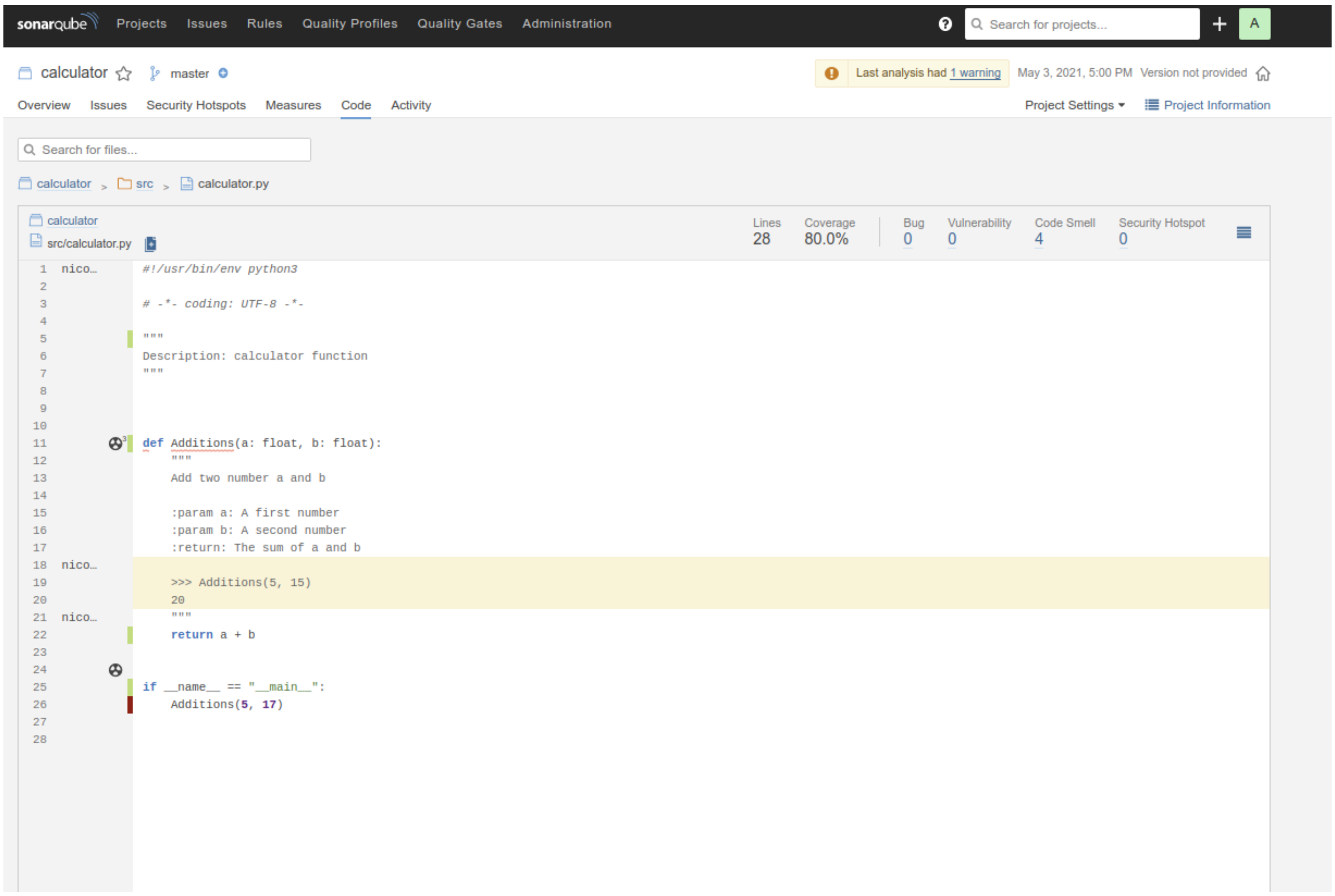
ACTIVITY

Bugs — Code Smells — Vulnerabilities

new code

May 3, 2021, 5:00 PM
not provided
Quality Profile: Use 'Sonar way' (XML)

Here, we can see that we have a coverage of 88.89 %. We can display the coverage line by line by clicking on the `code` tab and then in the `src` and `calculator.py` file. Green lines are covered/tested, red lines are not.



```
nico…   #!/usr/bin/env python3

        # -*- coding: UTF-8 -*-

        """
        Description: calculator function
        """



        def Additions(a: float, b: float):
            """
            Add two number a and b

            :param a: A first number
            :param b: A second number
            :return: The sum of a and b
nico…
            >>> Additions(5, 15)
            20
nico…       """
            return a + b


        if __name__ == "__main__":
            Additions(5, 17)
```

With a command line and a configuration file

It can be very annoying to type a command line with a lot of parameters to run sonar-scanner. Fortunately, you can launch sonar-scanner without any parameters if you use a configuration file.

To do this, create a file named `sonar-project.properties` at the root of your project and write the sonar-scanner settings in it.

Here is the `sonar-project.properties` file of the calculator repository (see `tag3`).

```
# must be unique in a given SonarQube instance
sonar.projectKey=calculator
# defaults to project key
sonar.projectName=calculator
# Path is relative to the sonar-project.properties file. Defaults to .
sonar.sources=.
# Encoding of the source code. Default is default system encoding
sonar.sourceEncoding=UTF-8
# Path the coverage file
sonar.python.coverage.reportPaths=./coverage.xml
# Description of the project
sonar.projectDescription=A to to make additions.
# SonarQube server Token
sonar.login=a03f52695660de4b8aa274d2904df75da9d47933
# Flake8 report
sonar.python.flake8.reportPaths=.flake8-report.txt
```

```
#----- Default SonarQube server
#sonar.host.url=http://localhost:9000
```

> Note that, in this file, the parameters are the same as those used in the command line but without the prefix -D.

As this file is present, you can type `sonar-scanner` to the root of the `calculator` directory to start an analysis.

If the file has a different name than `sonar-project.properties` then you must indicate the name of the file when you run `sonar-scanner`: `sonar-scanner -Dproject.settings=./chemin/vers/fichier/sonar.properties`.

Inside your continuous integration workflow

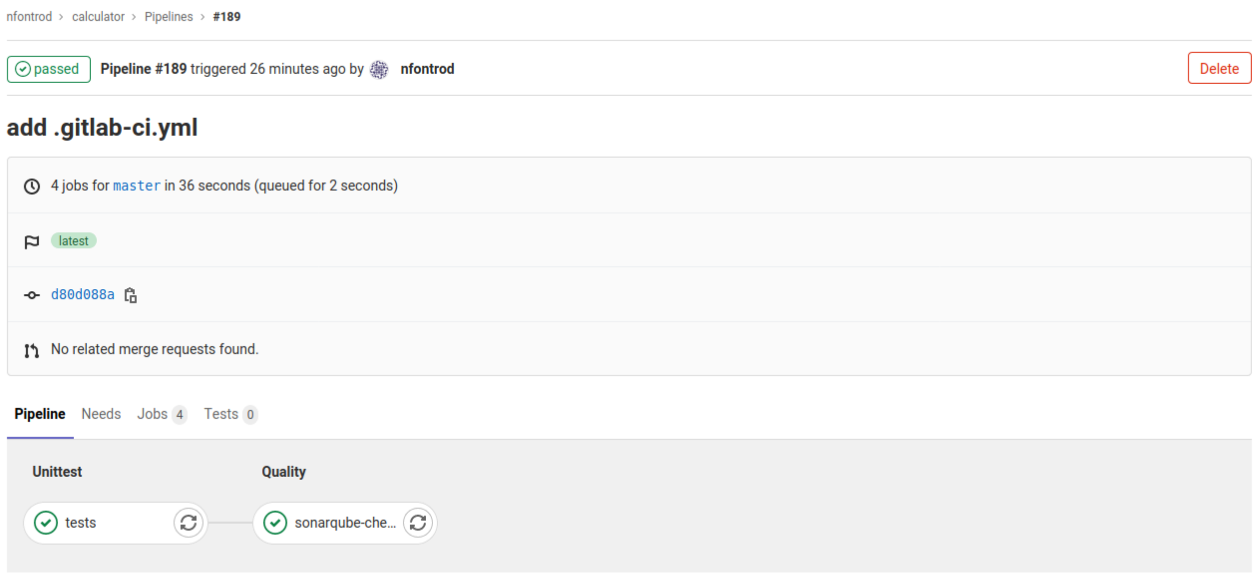
Running `sonar-scaner` every time a change is made can be time-consuming, and it's very likely that you will forget to run it from time to time. To avoid this, you can integrate SonarQube to your continuous integration workflow.

This is the `.gitlab-config.yml` file of the calculator project (see `tag3`).

```
stages: # List of the stages, for the jobs to run sequencially
  - unittest # stage that perform unit tests.
  - quality # stage that check the code quality

tests: # Defines a job named tests. It includes the scripts and settings
you want to apply to that specific task.
  stage: unittest # Reference to the stage name
  image: "python:3.8-slim" # Docker image used to run the job tests
  before_script: # Commands to execute before the script section
    - apt-get update
    - pip install coverage
    - pip install flake8
  script: # commands that trigger the tests
    - coverage run -m unittest discover
    - coverage xml
    - flake8 --exit-zero --output-file .flake8-report.txt
  artifacts: # Use for stage results that are passed between stages
    untracked: true # Every untracked files produced by this job can be
shared between different stages


sonarqube-check: # Defines a job named sonarqube-check
  stage: quality # Reference to the stage name
  image: sonarsource/sonar-scanner-cli:latest # Docker image used to run
the job tests
  variables: # Envrionment variables to use
    SONAR_TOKEN: a03f52695660de4b8aa274d2904df75da9d47933
    SONAR_HOST_URL: http://192.168.1.58:9000/
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the location of
the analysis task cache
```

```
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of the project, required by the analysis task
  dependencies: # Restrict which artifacts are passed to a specific job by providing a list of jobs to fetch artifacts from.
    - tests
  cache: # Caches are used to speed up runs of a given job in subsequent pipelines
    key: "${CI_JOB_NAME}" # The cache is shared accross pipelines only for this job.
    paths:
      - .sonar/cache # choose which files or directories to cache. Paths are relative to the project director
  script: # Command to run sonar quality analysis
    - sonar-scanner -Dsonar.qualitygate.wait=true # This option tells gitlab-ci to wait for the quality-gate report and to fail the job if the new code doesn't meet the minimum quality metrics required.
  allow_failure: true # The CI pipeline is not stopped if the script section fails
  only: # The job runs only for the master or dev branch
    - master
    - dev
```

With this file, at each commit your unit-tests will be performed, and a quality analysis of your new code will be done.


nfontrod › calculator › Pipelines › #189

passed Pipeline #189 triggered 26 minutes ago by nfontrod

add .gitlab-ci.yml

4 jobs for master in 36 seconds (queued for 2 seconds)

latest

d80d088a

No related merge requests found.

SonarLint

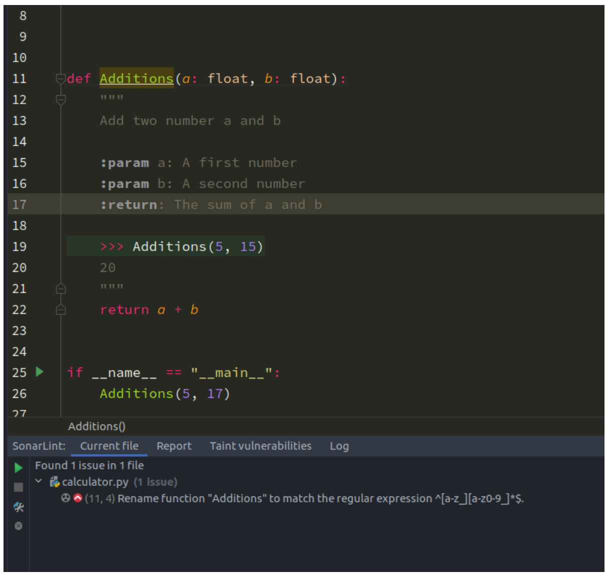
SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Using it will give you immediate quality feedback before you even commit your code.

SonarLint can be installed on those IDEs:

- Eclipse
- IntelliJ IDEA
- Visual Studio

- VS code
- CLion

Example of a SonarLint analysis on Pycharm.



```python


def Additions(a: float, b: float):
    """
    Add two number a and b

    :param a: A first number
    :param b: A second number
    :return: The sum of a and b

    >>> Additions(5, 15)
    20
    """
    return a + b


if __name__ == "__main__":
    Additions(5, 17)
```

```
Found 1 issue in 1 file
calculator.py (1 issue)
    (11, 4) Rename function "Additions" to match the regular expression ^[a-z_][a-z0-9_]*$.
```