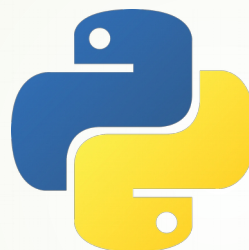


DAWAN Paris  
DAWAN Nantes  
DAWAN Lyon

11, rue Antoine Bourdelle, 75015 PARIS  
32, Bd Vincent Gâche, 5e étage - 44200 NANTES  
Bt Banque Rhône Alpes, 2ème étage - 235 cours Lafayette 69006 LYON



# Intégration Continue: Analyse statique du code

Plus d'info sur <http://www.dawan.fr> ou 0810.001.917

Formateur: Matthieu LAMAMRA

# Lint PYHTON

- Le module pylint : analyse statique de code source
  - Intérêt: évaluation de la qualité du code source python
  - Installation : **sudo pip3 install pylint** (à l'intérieur du venv)
  - Détecte et liste les **erreurs de syntaxe** et les entorses aux conventions **pep8**
  - Propose des tableaux de synthèse et assigne une note au code

```
$ pylint [fichier]
```

```
$ pylint -E [fichier] #n'affiche que les erreurs
```

```
$ pylint --reports=y [fichier] #ajoute le rapport de synthèse
```

# Lint Python

- Fichier `.pylintrc` : configuration du linter
  - Permet de régler le niveau et la précision de l'analyse
  - Permet d'ajuster certaines conventions d'écritures (camelCase, snake\_case...)
  - Permet de limiter les messages d'informations à certaines règles

```
$ pylint --generate-rcfile > .pylintrc #création du fichier
```

```
[...]  
method-naming-style=snake_case  
[...]  
disable=print-statement,...
```

# Lint PYHTON

- Analyse syntaxique dans gitlab
  - On fait exécuter pylint -E sur l'ensemble des fichiers python du dépôt

```
pylint-E:  
stage: syntax  
before_script:  
  - pip3 install -r $CI_PROJECT_DIR/requirements.txt  
script:  
  - 'find . -type f -name "*.py" -not -path "./lib/*" | xargs pylint -E'
```

# Linters PYTHON

- Analyse de la qualité du code dans gitlab
  - On veut récupérer le score de pylint pour l'afficher sur un badge dans gitlab

```
pylint:
  stage: style
  before_script:
    - pip3 install -r $CI_PROJECT_DIR/requirements.txt
  script:
    - >
      find . -type f -name "*.py" -not -path "./lib/*" |
      xargs pylint --output-format=text --exit-zero | tee pylint.txt
    - score=$(sed -n 's/^Your code has been rated at \([-0-9.]*\)V.*\1/p' pylint.txt)
    - >
      anybadge --label=Pylint
      --file=pylint.svg
      --value=$score
      2=red 4=orange 8=yellow 10=green
  artifacts:
    paths:
      - pylint.svg
```

# Lint PYHTON

- Création d'un badge dans Gitlab CI
  - On renseigne l'url de l'artefact dans **Settings** → **General** → **Badges**
  - Ce badge s'affiche sur la page d'accueil du projet

## Name

pylint

## Link

The **variables** GitLab supports: `%{project_path}`, `%{project_id}`, `%{default_branch}`, `%{commit_sha}`

`http://gitlab.myusine.fr/%{project_path}/-/commit/%{commit_sha}`

e.g. `https://example.gitlab.com/%{project_path}`

## Badge image URL

The **variables** GitLab supports: `%{project_path}`, `%{project_id}`, `%{default_branch}`, `%{commit_sha}`

`http://gitlab.myusine.fr/%{project_path}/-/jobs/artifacts/master/raw/pylint.svg?job=pylint`

e.g. `https://example.gitlab.com/%{project_path}/badges/%{default_branch}/pipeline.svg`

## Badge image preview

Pylint 4.71



# Linteur PYTHON

- Remontée du rapport pylint
- Nécessite l'installation du module python `pylint-gitlab` pour générer un rapport JSON au format « `Code Quality` »

```
pylint:
stage: style
before_script:
  - pip3 install -r $CI_PROJECT_DIR/requirements.txt
script:
  - >
    find . -type f -name "*.py" -not -path "./lib/*" |
    xargs pylint --output-format=pylint_gitlab.GitlabCodeClimateReporter --exit-zero
    > codeclimate.json
artifacts:
paths:
  - codeclimate.json
reports :
  codequality : codeclimate.json
rules :
  - if: '$CI_COMMIT_BRANCH == "master"'
  - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
```

# Linters PYTHON

- Affichage du rapport pylint dans le job

Pipeline Needs Jobs 1 Tests 0 **Code Quality**


! Found 39 code quality issues  
This report contains all Code Quality issues in the source branch.

- x ? Unknown - Final newline missing in `app.py:5`
- x ? Unknown - Missing module docstring in `app.py:1`
- x ? Unknown - Missing module docstring in `bank/test_client.py:1`
- x ? Unknown - Missing class docstring in `bank/test_client.py:6`




# Linters PYTHON


- Affichage de l'évolution de la qualité dans les « merge requests »




Code quality improved on 1 point

[Collapse](#)

 **Fixed:** Unknown - Method name "\_updateOverdraft" doesn't conform to snake\_case naming style in [bank/account.py:49](#)



[Merge](#)

 [Delete source branch](#)

# SONARQUBE

- Solution d'analyse qualité universelle
  - Intérêt: centraliser l'outil d'assurance qualité pour tous les langages utilisés
  - Solution open source, gratuite (community edition)
  - Permet de construire des profils d'analyses qualité
  - Permet de renseigner des seuils de validation basés sur des métriques

```
$ sudo docker pull sonarqube:lts
```

```
$ sudo docker container run -d --name sonar --restart always \
```

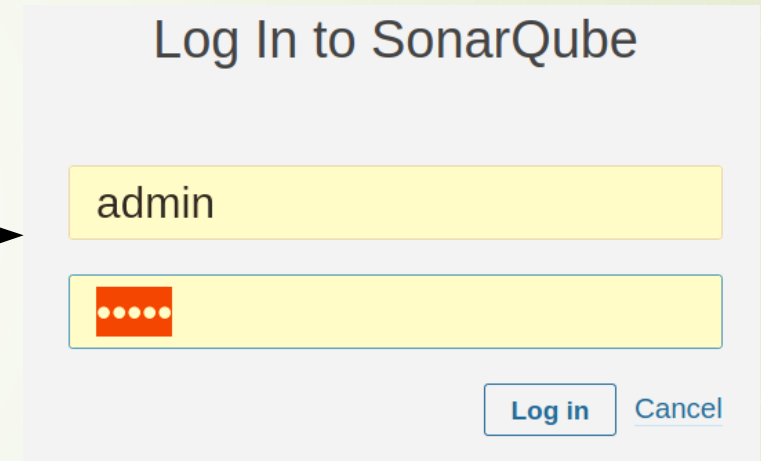
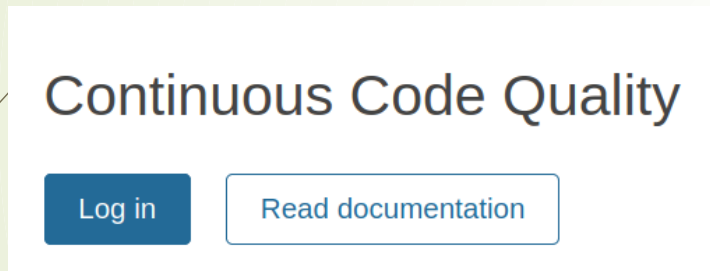
```
> -p 9000:9000 sonarqube:lts
```

# SONARQUBE : installation

- Création du projet

**http://localhost:9000**

**admin / admin**





# SONARQUBE : installation

- Création du projet

Once you analyze some projects, they will show up here.

Here is how you can analyse new projects

[Create new project](#)


# SONARQUBE : installation

- Création du projet

## Create new project

---

**Project key\*** ?



Up to 400 characters. All letters, digits, dash, underscore, period or colon.

**Display name\*** ?

Up to 255 characters

Set Up

# SONARQUBE : installation

- Création du projet

## 1 Provide a token

Generate a token

**Generate**

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](#).



# SONARQUBE : installation

- Création du projet

2 Run analysis on your project

What is your project's main language?

Java

C# or VB.NET

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

# SONARQUBE : installation

- Création du projet

## Download and unzip the Scanner for Linux

And add the `bin` directory to the `PATH` environment variable

[Download](#)

## Execute the Scanner from your computer

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

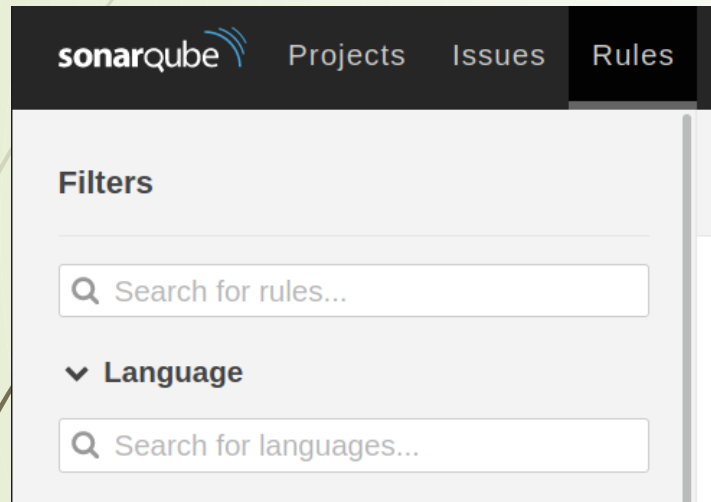
```
sonar-scanner \  
-Dsonar.projectKey=myusine \  
-Dsonar.sources=. \  
-Dsonar.host.url=http://localhost:9000 \  
-Dsonar.login=5aa127a39fba7b9702217c9479c6609f972cfc94
```

[Copy](#)

# SONARQUBE : installation

- La base de règles

## Recherche libre et par langage



sonarqube Projects Issues Rules

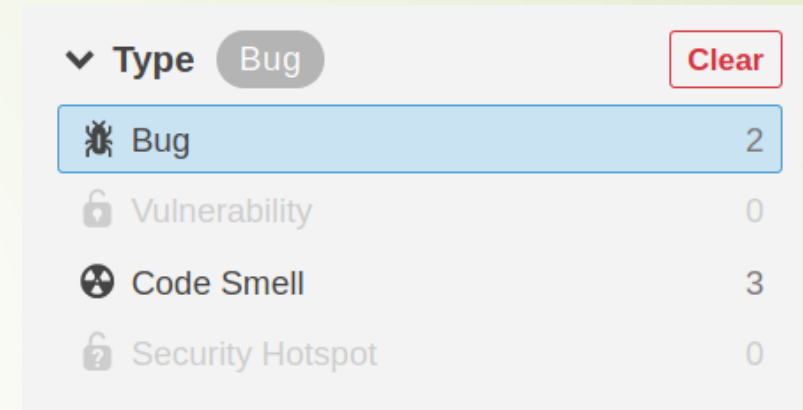
Filters

Search for rules...

Language

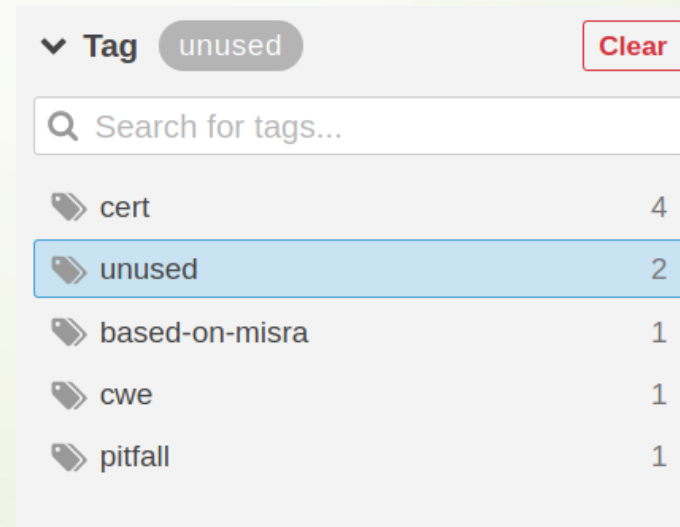
Search for languages...

## Recherche par type



Type	Bug	Clear
Bug	2	
Vulnerability	0	
Code Smell	3	
Security Hotspot	0	

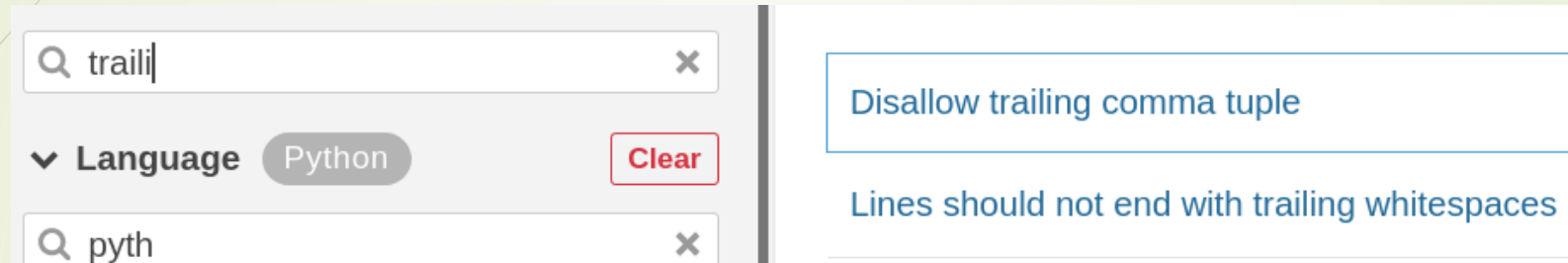
## Recherche par mot clé



Tag	unused	Clear
Search for tags...		
cert	4	
unused	2	
based-on-misra	1	
cwe	1	
pitfall	1	

# SONARQUBE : installation

- La base de règles : exemple



A screenshot of the SonarQube search interface. It features a search bar with the text 'trailing' and a dropdown menu for 'Language' set to 'Python'. There is a 'Clear' button next to the language filter. Below the search bar, there is a search bar with the text 'pyth'. To the right of the search bar, there are two search results: 'Disallow trailing comma tuple' and 'Lines should not end with trailing whitespaces'.

## Lines should not end with trailing whitespaces

python:S113

Code Smell Minor Main sources convention Available Since May 28, 2020 SonarAnalyzer (Python)

Constant/issue: 1min

Trailing whitespaces are simply useless and should not stay in code. They may generate noise when comparing different versions of the same file.

If you encounter issues from this rule, this probably means that you are not using an automated code formatter - which you should if you have the opportunity to do so.

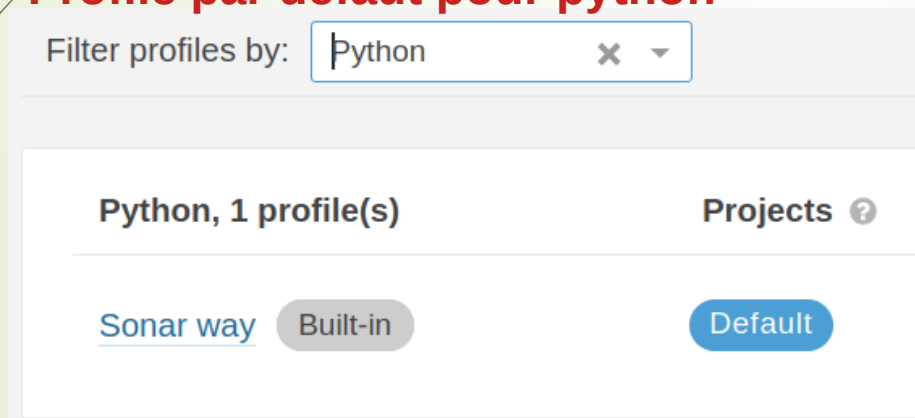
[Extend Description](#)

# SONARQUBE : installation

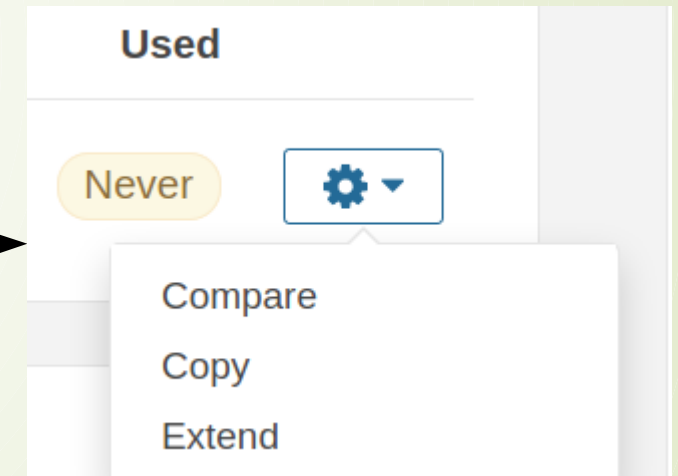
- Création du Profile qualité

**Profile : ensemble de règles à appliquer**

**Profile par défaut pour python**



**Extend: créer un nouveau profile à partir du profile par défaut**



# SONARQUBE : installation

- Création du Profile qualité

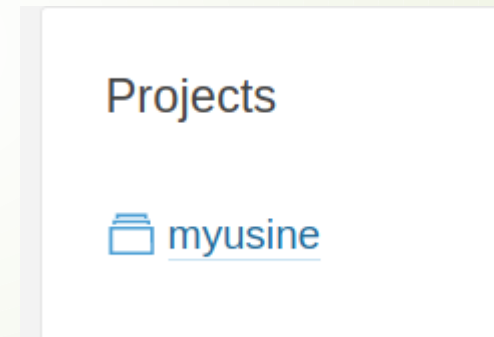
## Nouveau profile

My Sonar Way

Rules	Active	Inactive
<b>Total</b>	<b>0</b>	<b>439</b>
 Bugs	0	12
 Vulnerabilities	0	0
 Code Smells	0	417
 Security Hotspots	0	10




[Activate More](#)

## Association du projet au profile



## Ajout d'une nouvelle règle

Lines should not end with trailing whitespaces

Python  Code Smell  convention 

[Activate](#)



# SONARQUBE : installation

- « quality gates » : seuils de validation

The screenshot shows the 'Quality Gates' configuration page in SonarQube. On the left, there's a sidebar with 'Quality Gates' and a 'Create' button. Below it, 'My sonar gate' is shown with 'Sonar way' selected and 'Default' and 'Built-in' buttons. The main area is titled 'Sonar way' with a 'Built-in' button. Below this, the 'Conditions' section explains that only project measures are checked against thresholds. A table lists the default conditions:

Metric	Operator	Error
Coverage on New Code	is less than	80.0%
Duplicated Lines on New Code	is greater than	3.0%
Maintainability Rating on New Code	is worse than	A
Reliability Rating on New Code	is worse than	A
Security Rating on New Code	is worse than	A

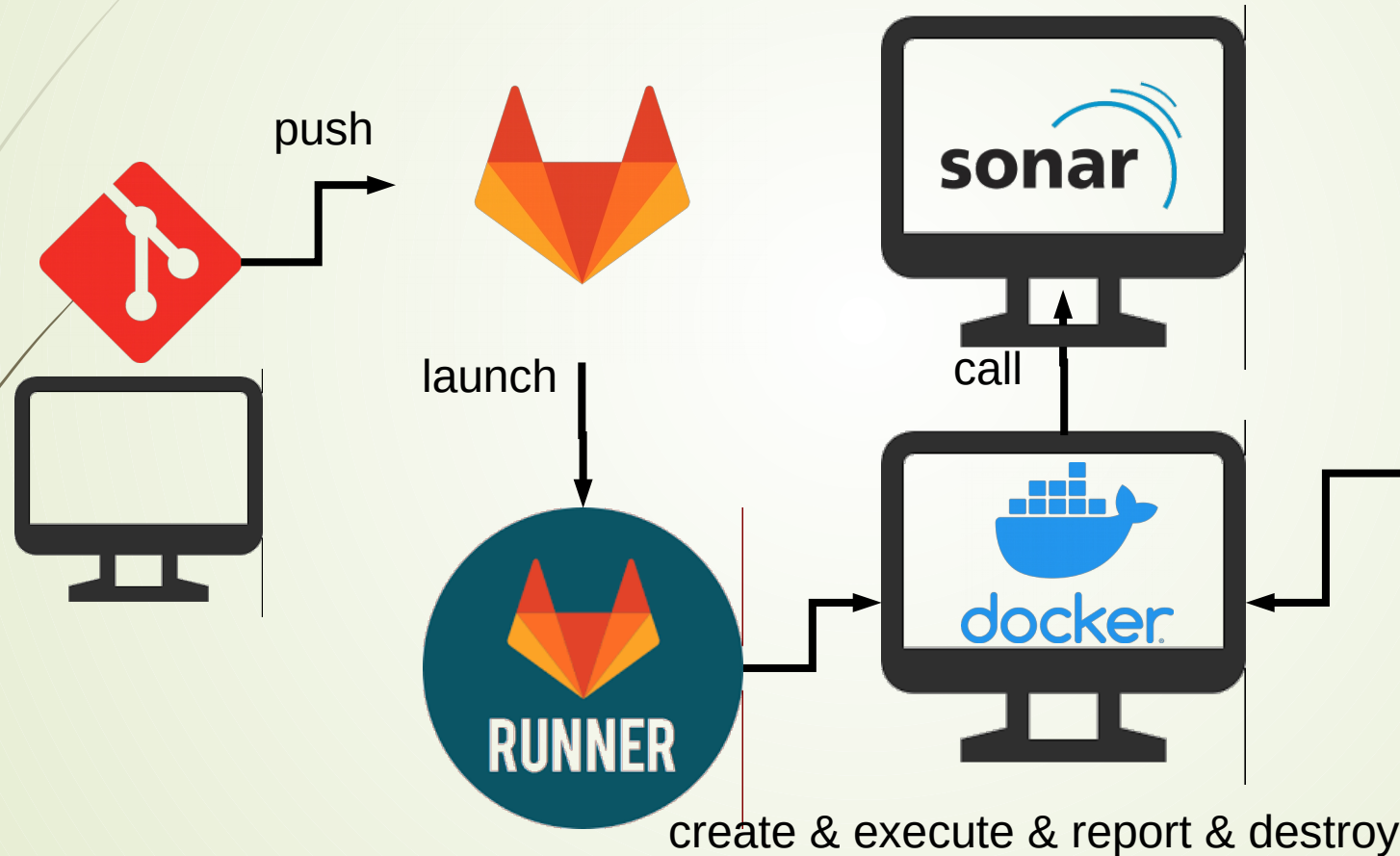
# SONARQUBE : scanner

- Sonar-scanner : l'outil de scrutation du code
  - Logiciel à télécharger sur sonarqube, préselectionné en fonction du projet
  - Fichier de configuration .properties (monde java) à la racine du dépôt local
  - Permet de distinguer le code à analyser et le type d'analyse

```
sonar.projectKey=myusine  
sonar.projectName=myusine  
sonar.host.url=http://172.17.0.1:9000  
sonar.login=5aa127a39fba7b9702217c9479c6609f972cfc94  
sonar.sources=app  
sonar.qualitygate.wait=true
```

# SONARQUBE: intégration

- Déclenchement d'un job sonarqube



```
sonar:  
  stages :  
    - quality  
  image:  
    name :  
    sonarsource/sonar-  
    scanner-cli:latest  
    ...  
  script:  
    - sonar-scanner  
    ...
```

# SONARQUBE : intégration

- Configuration du job gitlab-ci.yml
  - Image dédiée au job : **sonarsource/sonar-scanner-cli:latest**
  - Clé « **variables** » : Variables d'environnement spécifiques au job
  - Clé « **allow\_failure** » : job non bloquant

```
sonar:
  stage: style
  image:
    name: sonarsource/sonar-scanner-cli:latest
    entrypoint: [""]
  variables:
    SONAR_TOKEN: 4ad2cdccf773cb8fa5b28ef76baefab366fcbe24
    SONAR_HOST_URL: http://172.17.0.1:9000
  script:
    - >
      sonar-scanner -Dsonar.projectKey=myusine
      -Dsonar.sources=.
      -Dsonar.exclusions=lib/**/*,lib64/**/*
      -Dsonar.host.url=$SONAR_HOST_URL
      -Dsonar.login=$SONAR_TOKEN
      -Dsonar.scm.provider=git
  allow_failure: true
```

# SONARQUBE : intégration

- Protection des données sensibles
  - On peut déclarer les variables d'environnement à l'extérieur de la configuration
  - [http://gitlab.myusine.fr/root/myusine/-/settings/ci\\_cd](http://gitlab.myusine.fr/root/myusine/-/settings/ci_cd) (volet « Variables »)
  - Les données sont protégées en lecture une fois renseignées
  - Les données sont disponibles directement dans les conteneurs de CI exécutés

Type	Key	Value	Protected	Masked
Variable	SONAR_HOST_URL	*****	✓	✗
Variable	SONAR_TOKEN	*****	✓	✗



# SONARQUBE : analyse

- Résultats sur le serveur

