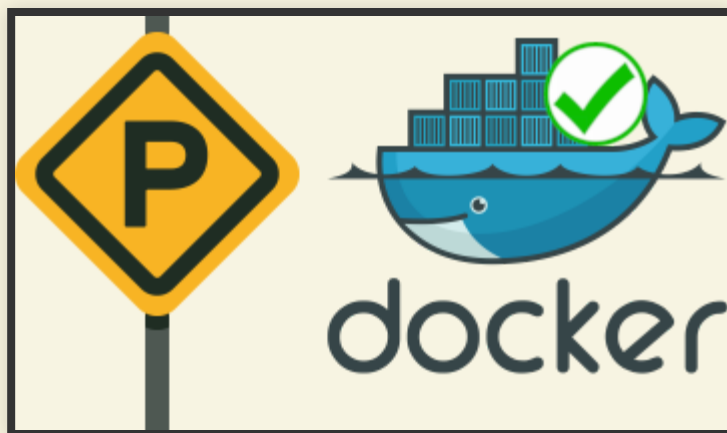


# **Safedockerignore, un hook pour pre-commit**

---

# Créer un hook pour pre-commit : vérifier son .dockerignore

---




Soirée des communautés techniques rennaises

**Luc Sorel-Giffo** — jeudi 16 mai 2024 — 20h salle i58

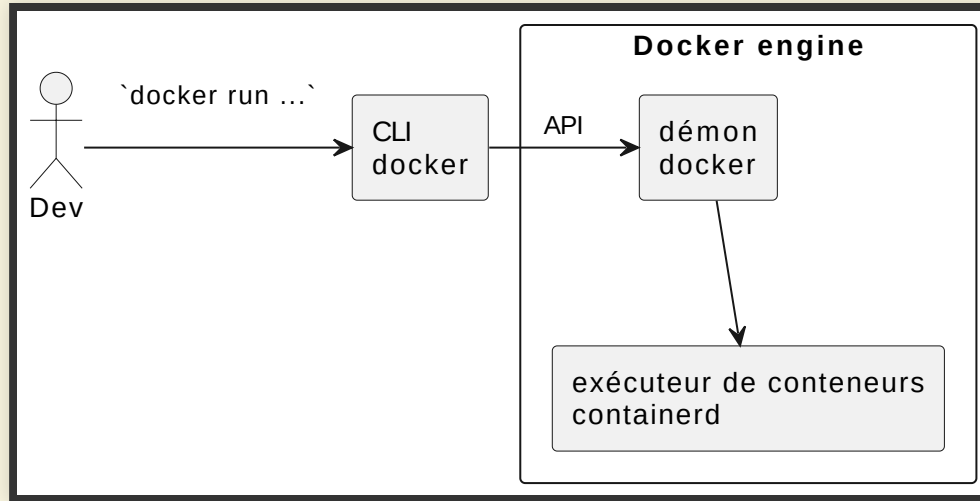
@lucsorelgiffo@floss.social (dev Python chez Purecontrol ∞)

# Au programme

---

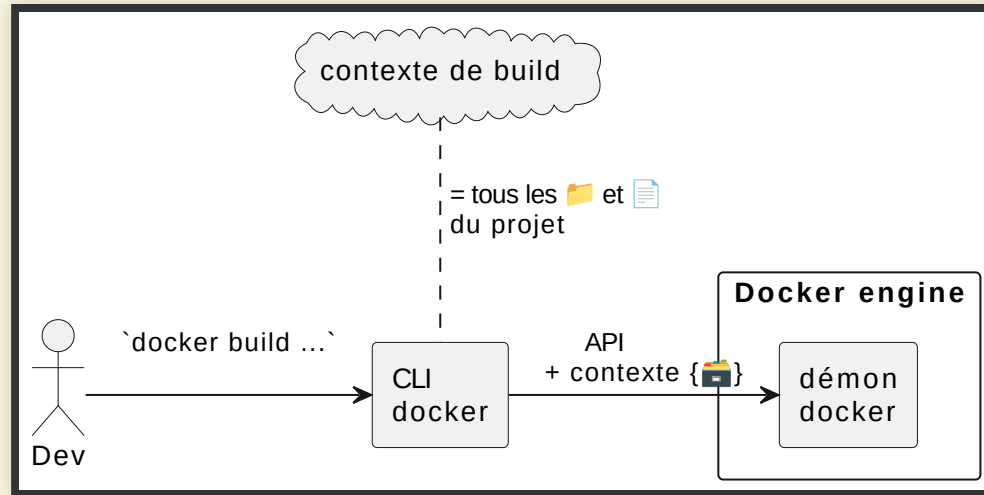
- parcours de création d'un hook pour l'outil **pre-commit**
-  infos glanées au passage sur
  - Docker
  - git
  - Python

## Motivation : sécuriser et accélérer "docker build" 1/2



Docker fonctionne avec une architecture client-serveur

## Motivation : sécuriser et accélérer "docker build" 2/2



💡 **.dockerignore**, pour exclure des ressources du contexte de build

- ressources inutiles (bande passante, temps)
- ressources sensibles (secrets)

# Syntaxe d'un .dockerignore

---

## Syntaxe classique :

```
# exclut un dossier
tests/
# exclut un fichier
sample-data.json
# exclut via un pattern
tests/**/*-data.json
```



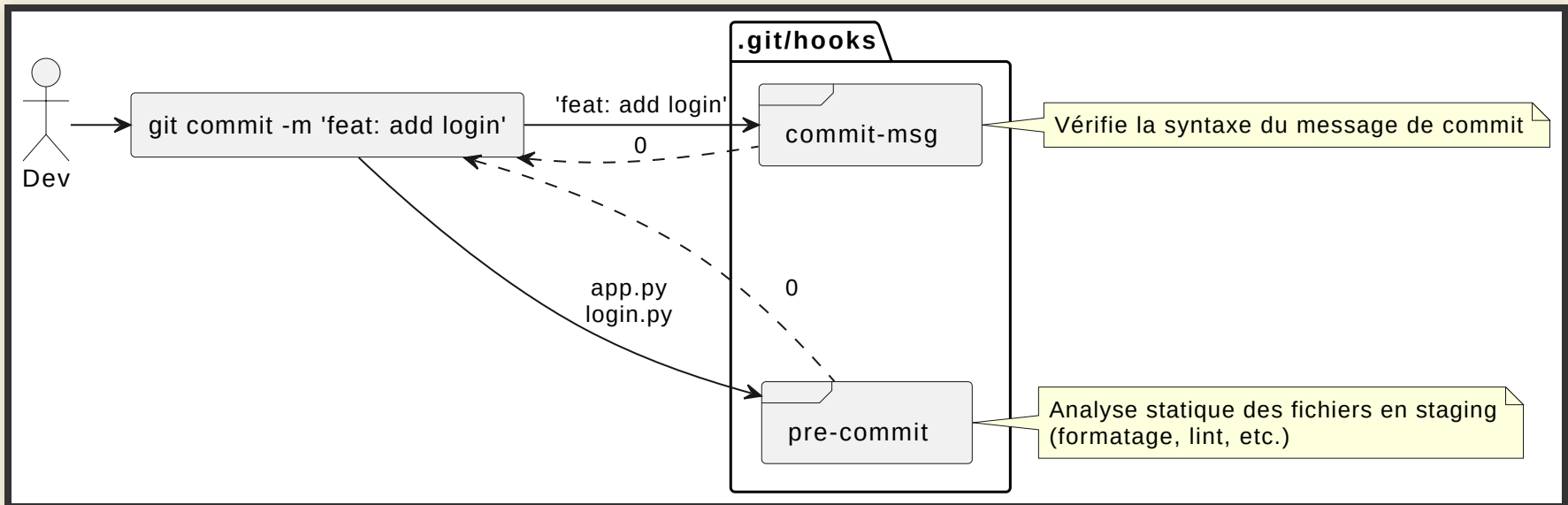
## Syntaxe agressive :

```
# exclut tout
*
# règles d'inclusion ("!..." -> "sauf ...")
!pyproject.toml
!my_app/
```

# Les hooks de git



git propose des hooks pour contrôler son cycle de vie :



- ⚠️ `.git/hooks` pas versionnable
- 🤖 écriture difficile de scripts de hook multiplateformes

*A framework for managing and maintaining multi-language pre-commit hooks.*



Figure 1. <https://pre-commit.com/>

- [github.com/pre-commit/pre-commit](https://github.com/pre-commit/pre-commit)
- 12.1k ★, 99+ releases (mai 2024)
- open-source (MIT license)

Voir [youtu.be/I0HrTE45RVM](https://youtu.be/I0HrTE45RVM) : Hook'il est beau, notre code ! Guider la qualité de code avec pre-commit (BreizhCamp 2023)


Hook'il est beau, notre code ! Guider la qualité de...






# Configuration avec .pre-commit-config.yaml

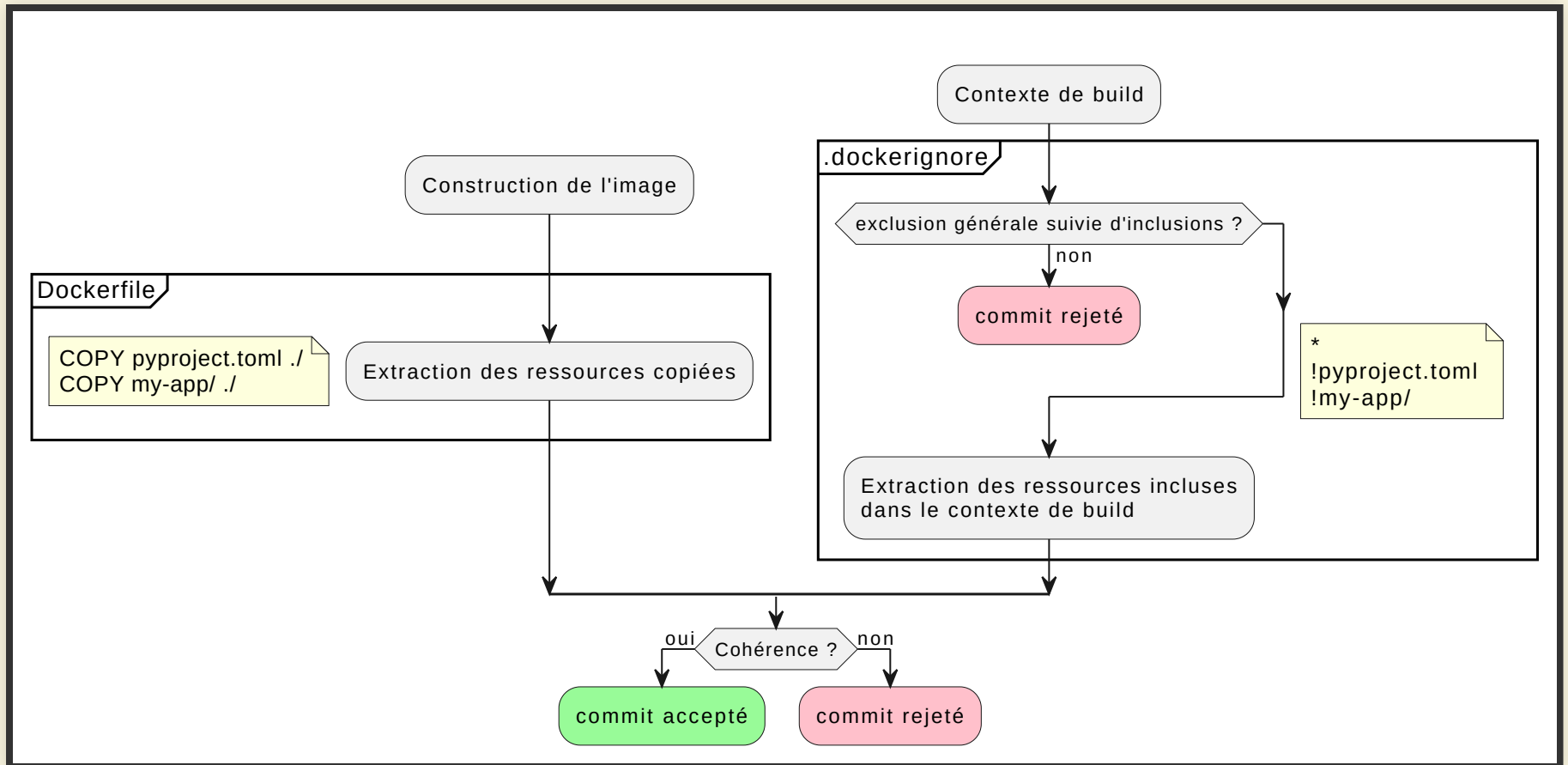
---

```
repos:
- repo: https://github.com/pre-commit/pre-commit-hooks
  rev: v4.6.0
  hooks:
    - id: check-yaml          # syntaxe des  yaml (aussi : json, xml, toml, etc.)
    - id: trailing-whitespace # nettoie les fins de ligne
    - id: end-of-file-fixer   # un seul saut de ligne de fin de fichier

    - id: check-added-large-files # évite de versionner de gros fichiers (>100 ko)
      args: [--maxkb=100]         #  configurable

- repo: https://github.com/astral-sh/ruff-pre-commit
  rev: v0.4.4
  hooks:
    - id: ruff
    - id: ruff-format
```

# Principe de fonctionnement du hook safedockerignore



# Créer un hook local "bash"

```
my-app
├── my_app/
│   └── app.py    # fichier modifié en staging
├── .pre-commit-config.yaml
├── hooks/
│   └── safedockerignore.sh
└── ...
```

```
📄 .pre-commit-config.yaml
repos:
- repo: local
  hooks:
  - id: safedockerignore
    name: Safe .dockerignore
    language: script
    entry: ./hooks/safedockerignore.sh # chmod +x
```

Voir [pre-commit.com/#new-hooks](https://pre-commit.com/#new-hooks).

```
📄 hooks/safedockerignore.sh
#!/usr/bin/env bash

echo "safedockerignore.sh parameters: $@"
echo "current working directory: $(pwd)"

exit 1
```

```
$ .venv/bin/pre-commit
Safe .dockerignore.....
- hook id: safedockerignore
- exit code: 1

safedockerignore.sh parameters: my_app
current working directory: /home/{user}
```

- 👁 paramètres = noms de fichiers
- ⚠ le dossier de travail est la racine

# Créer un hook local "python" simple

```
my-app
├── my_app/
│   └── app.py    # fichier modifié en staging
├── .pre-commit-config.yaml
├── hooks/
│   └── safedockerignore.py 🙌
└── ...
```

```
📄 hooks/safedockerignore.py
from argparse import ArgumentParser
```

```
def main():
    parser = ArgumentParser()
    parser.add_argument(
        'filenames', nargs='*',
        help='Names of changed files'
    )
    parser.add_argument(
        '--dockerfile', default='Dockerfile',
        help='Dockerfile to process'
    )
    print(parser.parse_args())
    return 1
```

```
if __name__ == '__main__':
    raise SystemExit(main())
```

```
📄 .pre-commit-config.yaml
- repo: local
  hooks:
    - id: safedockerignore
      name: Safe .dockerignore
      language: script
      entry: .venv/bin/python ./hooks/safedockerignore.py
      args: [--dockerfile=Dockerfile.prod]
```

```
$ .venv/bin/pre-commit
Safe .dockerignore.....
- hook id: safedockerignore
- exit code: 1
```

```
Namespace(
  filenames=['my_app/app.py'],
  dockerfile='Dockerfile.prod'
)
```

# Gestion des patterns de fichiers - pathlib.Path



Dockerfile

...

```
COPY config/**/*-config.json configs/
```

```
from pathlib import Path
```

```
current_working_dir = Path.cwd()
```

```
resources_rule = 'config/**/*-config.json'
```

```
resources = list(current_working_dir.glob(resources_rule))
```

```
if not resources:
```

```
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

# Gestion des patterns de fichiers - pathlib.Path

```
📄 Dockerfile
...
COPY config/**/*-config.json configs/
```


```
from pathlib import Path

current_working_dir = Path.cwd()

resources_rule = 'config/**/*-config.json'
resources = list(current_working_dir.glob(resources_rule))
if not resources:
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

- `Path.cwd()` renvoie le dossier de travail en cours (→ racine git)

# Gestion des patterns de fichiers - pathlib.Path

```
 Dockerfile
...
COPY config/**/*-config.json configs/
```


```
from pathlib import Path

current_working_dir = Path.cwd()

resources_rule = 'config/**/*-config.json'
resources = list(current_working_dir.glob(resources_rule))
if not resources:
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

- `Path.cwd()` renvoie le dossier de travail en cours (→ racine git)
- `path.glob('*.*py')` renvoie un itérateur de noms de fichier correspondant strictement au pattern

# Gestion des patterns de fichiers - pathlib.Path

```
 Dockerfile
...
COPY config/**/*-config.json configs/
```

```
from pathlib import Path


current_working_dir = Path.cwd()

resources_rule = 'config/**/*-config.json'
resources = list(current_working_dir.glob(resources_rule))
if not resources:
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

- `Path.cwd()` renvoie le dossier de travail en cours (→ racine git)
- `path.glob('*.*py')` renvoie un itérateur de noms de fichier correspondant strictement au pattern
- `path.rglob('*.*py')` recherche récursivement dans les sous-dossiers



# Gestion des patterns de fichiers - pathlib.Path

```
 Dockerfile
...
COPY config/**/*-config.json configs/
```


```
from pathlib import Path

current_working_dir = Path.cwd()

resources_rule = 'config/**/*-config.json'
resources = list(current_working_dir.glob(resources_rule))
if not resources:
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

- `Path.cwd()` renvoie le dossier de travail en cours (→ racine git)
- `path.glob('*.*py')` renvoie un itérateur de noms de fichier correspondant strictement au pattern
- `path.rglob('*.*py')` recherche récursivement dans les sous-dossiers
- (🙏 ne plus utiliser `os.path` pour gérer les chemins de fichiers)

# Gestion des patterns de fichiers - pathlib.Path

```
 Dockerfile
...
COPY config/**/*-config.json configs/
```

```
from pathlib import Path

current_working_dir = Path.cwd()

resources_rule = 'config/**/*-config.json'
resources = list(current_working_dir.glob(resources_rule))
if not resources:
    print(f'Dockerfile error: COPY {resources_rule} instruction matches no resource')
```

- `Path.cwd()` renvoie le dossier de travail en cours (→ racine git)
- `path.glob('*.*py')` renvoie un itérateur de noms de fichier correspondant strictement au pattern
- `path.rglob('*.*py')` recherche récursivement dans les sous-dossiers
- (🙏 ne plus utiliser `os.path` pour gérer les chemins de fichiers)
- → [docs.python.org/3/library/pathlib.html](https://docs.python.org/3/library/pathlib.html)

## Conclusion

---

- hook pre-commit = outil CLI
  - argparse, click, fire, etc. ( → [j1el.github.io/c-est-cli-qui-gagne](https://j1el.github.io/c-est-cli-qui-gagne/))
  - return 0 (succès) ou un code d'erreur
- configuration projet dans `.pre-commit-config.yaml`
- hook git → dossier de travail = racine git

## **Dans une prochaine session Python Rennes ?**

---

- hook python plus complexe (imports, dépendances, etc. un sous-projet à part entière)
- REX plus détaillé
  - Python
  - pre-commit
- publication du hook dans un dépôt git indépendant

# Rejoignez Python Rennes !

---

Communauté "services numériques" complétant les communautés datascience existantes.

357 membres 🎉



Figure 2. Groupe meetup :  
<https://www.meetup.com/fr-FR/python-rennes/>

Rejoignez  
[pythonrennes.slack.com](https://pythonrennes.slack.com)  
(actualités, entraide, orga).

71 membres



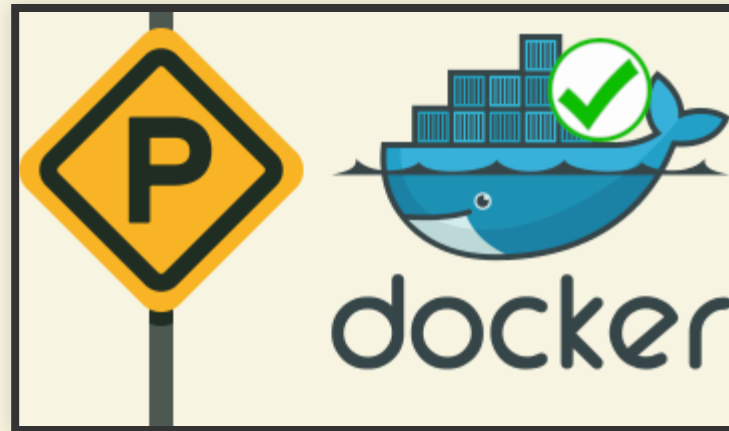
Figure 3. Invitation slack :  
[https://join.slack.com/t/pythonrennes/shared\\_invite/zt-1yd4yioap-IBAngm3Q0jxAKLP6fYJR8w](https://join.slack.com/t/pythonrennes/shared_invite/zt-1yd4yioap-IBAngm3Q0jxAKLP6fYJR8w)

Compte Twitter X 🧑 :  
[@PythonRennes](https://twitter.com/PythonRennes)

83 personnes abonnées

# Merci !

---



## Des questions ?

Diaporama à retrouver sur [github.com/lucsorel/conferences/tree/main/python-rennes-2024.05.16-safedockerignore-quickly](https://github.com/lucsorel/conferences/tree/main/python-rennes-2024.05.16-safedockerignore-quickly)