HOOKS DE PRE-COMMIT POUR GIT

Lucas Cimon
@ PyConFr 2017

BONJOUR !

Lucas Cimon

Voyages-Sncf.com Technologies
@ Nantes

https://chezsoi.org +

PRE-COMMIT ??

Extrait de git-scm.com

"Le crochet pre-commit est lancé en premier [...]. Il est utilisé pour inspecter l'instantané qui est sur le point d'être validé, pour vérifier si vous avez oublié quelque chose, pour s'assurer que les tests passent ou pour examiner ce que vous souhaitez inspecter dans le code."

L'AÏEUL

`lint`:

- existait déjà 1979
- vérifiait du code C

CASSE-PIED ?

CommitStrip du 2015/09/18

- à VSCT, avaient initialement une mauvaise réputation
- 2 questions:
  - qui en a déjà utilisé ?
  - à qui ça a déjà cassé les pieds ?

POURQUOI LES UTILISER ?

- assurance qualité
- gain de temps
- historique `git` plus propre
- validation côté client **ET** serveur

+ "un outil pour les gouverner tous..."

- QA: code style, tests, bugs finding

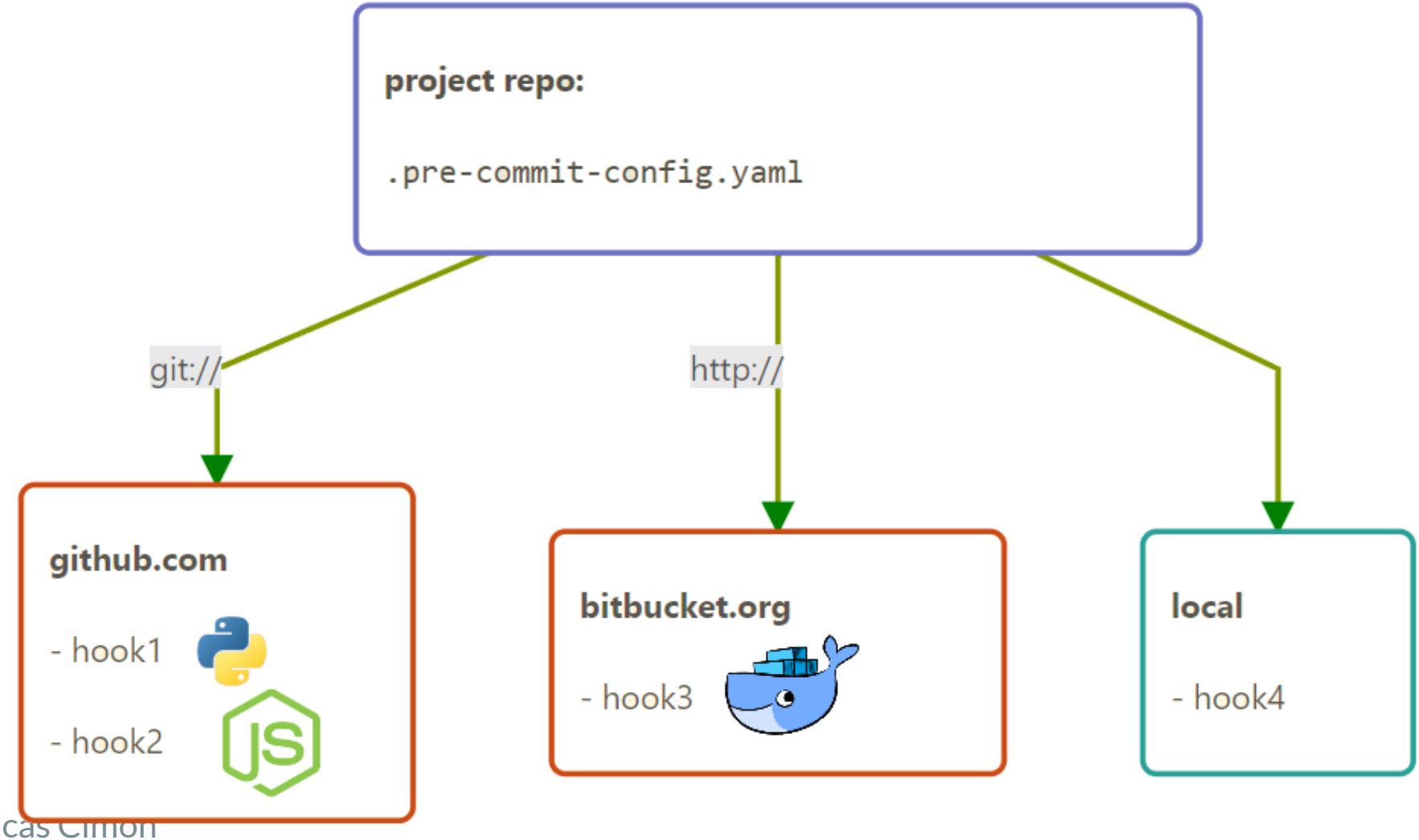
POURQUOI CELUI-CI ?

pre-commit.com
par Anthony Sottile

- en Python (alt: overcommit en Ruby, git-hooks en Go)
- une bibliothèque de **126** *hooks*, dont: `autopep8`, `flake8`, `prettier`, `prospector`, `pylint`, `yapf`...
- 5 lignes pour **créer** un nouveau *hook*
- projet open-source très actif

- nom pas top -> je vais l'appeler "pre-commit Python"
- historiquement développé à Yelp

FONCTIONNEMENT (1/2)



Etapes d'installation (en cache local):

1. la configuration est lue
2. les repos `git` contenant les *hooks* en Python / Go / NodeJS / Ruby / Swift / Docker sont clonés
3. les éventuelles `additional_dependencies` sont installées (pour Python: dans un `venv`)

FONCTIONNEMENT (2/2)

Exécution:

- les modifications non indexées de fichiers sont remisées (`git stash`)
- les fichiers modifiés sont passés aux *hooks*, selon:
  - le `type` du *hook*: `text`, `executable`, `python`...
  - si leur nom correspond à la *regex* `files` du *hook*

== ce qui se passe au `git commit`

DÉMO

montrer contenu du YAML

INSTALLATION

```
pip install pre-commit
# -> donne accès à la commande `pre-commit`
pre-commit install
# -> génère .git/hooks/pre-commit
```

CLI BASICS

```
pre-commit run $hook_id
# -> exécute un unique hook sur tous les fichiers "addés"
pre-commit run --files $file1 $file2
# -> exécute tous les hooks sur des fichiers spécifiques
pre-commit run --all-files
# -> exécute tous les hooks sur tous les fichiers versionnés
```

LOCAL HOOKS

```
-   repo: local
    hooks:
    -   id: check-bash-syntax
        name: Check Shell scripts syntax corectness
        language: system
        entry: bash -n
        files: \.sh$
```

BONUS

Intégration continue **super** simple.

`.travis.yml`

```
language: python
python: "3.6"
install: pip install pre-commit
script: pre-commit run --all-files
```

! hooks nécessitant Go/NodeJS/Java

BONUS

`pre-commit autoupdate`

→ modifie `.pre-commit-config.yaml`
avec dernier tag

SUPPORT WINDOWS ?

- Cygwin, MSYS2, Git For Windows, Windows 10 bash
  → **100%** supporté
- TortoiseGit: ☹

pour Cygwin, j'ai testé les hooks Go & NodeJS

ASTUCES

🚫😐🚫

```
git commit --no-verify
```

```
export SKIP=$hook_id1,$hook_id2
```

---

pour inspecter les repos / venv des hooks:
`~/.cache/pre-commit/`

cf. aussi fonctions shell

EN RÉSUMÉ

- simple d'utilisation
- puissant
- rapide à mettre en place