pre-commit, un framework pas que reservé aux développeurs !

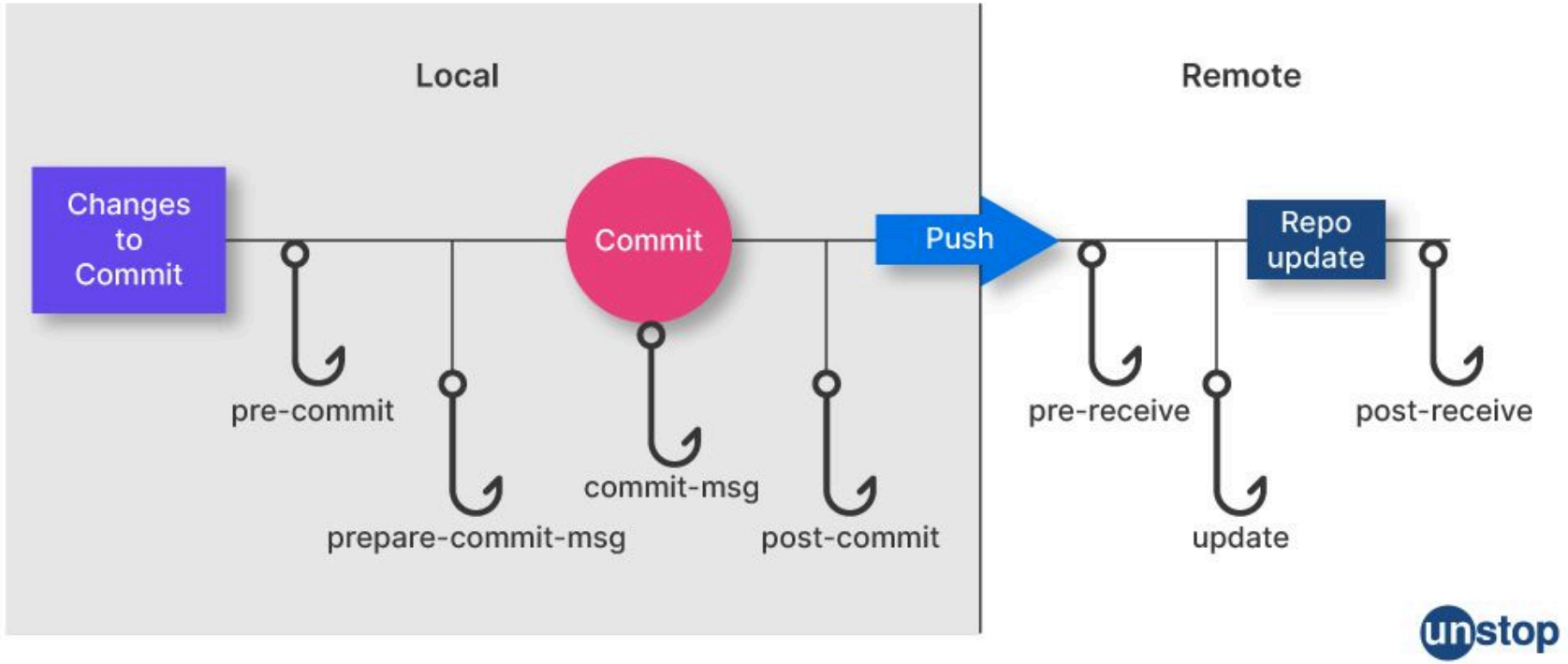
Et si nous mettions encore plus de linters dans notre Infrastructure as Code ?

Sommaire

Hooks git et cycle de vie des commits

Un hook git va permettre de :

- Détecter des erreurs ou problèmes de syntaxe (linting)
- Reformater automatiquement le code (formating)
- Réaliser des tests unitaires automatiquement
- Identifier les secrets avant qu'ils ne se retrouvent dans les commits
- Définir ses propres actions à exécuter avant le commit


Local
Remote
Changes to Commit
Commit
Push
Repo update
pre-commit
prepare-commit-msg
commit-msg
post-commit
pre-receive
update
post-receive
unstop

Démonstration sur un nouveau projet git

- Présentation des hooks
- Initialisation de pre-commit en erreur
- Test de la branche courante

```
cat .git/hooks/pre-commit
```

```
#!/usr/bin/env bash

git_branch=$(git branch --show-current)

if [[ "$git_branch" == "main" ]] || [[ "$git_branch" == "master" ]]; then
    message="Impossible de commit sur $git_branch"
    echo -e "\033[0;31m$message\033[0;0m"
    exit 1
else
    exit 0
fi
```

pre-commit, quèsaco ?

Un framework pour gérer des hooks de pre-commit, pour différents languages.

- Centralisation des hooks de pre-commit dans un fichier de configuration
- Catalogue de hooks proposé par la communauté
- Version 1.0.0 introduite en septembre 2017
- En janvier 2025 : version 4.1.0

Démonstration

- Installation de pre-commit
- Configuration de pre-commit dans le projet git
- Exécution de pre-commit

Présentation des hooks et de son catalogue de scripts

https://pre-commit.com/hooks.htmlhttps://github.com/pre-commit/pre-commit-hooks

Linters

Détecter les erreurs de syntaxe et validation du code

- check-added-large-files
- check-yaml
- check-toml
- check-executables-have-shebangs
- check-shebang-scripts-are-executable

Formatters

Corriger automatiquement les fichiers.

- end-of-file-fixer
- trailing-whitespace

Git

Vérifier si des conflits de merge sont présents

- check-merge-conflict

Github actions

- rhysd/actionlint

Documentation

Analyser la syntaxe des fichiers markdown

- markdownlint/markdownlint:

Langage python

Ruff : Linter et formatter Python, écrit en rust

- astral-sh/ruff-pre-commit

Scripts bash

Shellcheck : l'outil d'analyse pour les scripts bash

- shellcheck-py/shellcheck-py

Gitleaks

- Identifier les mots de passe, tokens, clés API, certificats
- Générer un rapport au format json, pour lister tous les secrets
- Possibilité de personnaliser finement la détection des secrets
- Ignorer des fichiers ou des secrets spécifiques
- https://github.com/gitleaks/gitleaks

```
→  ~/code(master) gitleaks git -v

Finding:     "export BUNDLE_ENTERPRISE__CONTRIBSYS__COM=cafebabe:deadbeef",
Secret:      cafebabe:deadbeef
RuleID:      sidekiq-secret
Entropy:     2.609850
File:        cmd/generate/config/rules/sidekiq.go
Line:        23
Commit:      cd5226711335c68be1e720b318b7bc3135a30eb2
Author:      John
Email:       john@users.noreply.github.com
Date:        2022-08-03T12:31:40Z
Fingerprint: cd5226711335c68be1e720b318b7bc3135a30eb2:cmd/generate/config/rules/sidekiq.go:sidekiq-secret:23
```

Framework pre-commit dans un contexte DevOps

Projets Ansible

Linter Ansible et ansible-lint

Fichier de configuration .ansible-lint

```
---
profile: production

# exclude_paths included in this file are parsed relative to this file's location and not relative to the CWD of execution.
exclude_paths:
- .cache/

# Ansible-lint does not fail on warnings from the rules or tags listed below
warn_list:
- command-instead-of-module
- command-instead-of-shell

# Ansible-lint will skip and ignore the rules or tags listed below
skip_list:
  - yaml
```

Configuration pre-commit

```yaml
repos:
  - repo: https://github.com/ansible/ansible-lint.git
    rev: v25.1.3
    hooks:
      - id: ansible-lint
        files: \.(yaml|yml)$
        name: Ansible Lint
        description: Run configurations on .ansible-lint file
        verbose: true
        # args: [-p, ./ansible/*]
```

Projets Terraform

Exemple de hooks disponibles avec les repositories **antonbabenko/pre-commit-terraform** et **terraform-docs**

- terraform fmt : formater les fichiers de configuration avec une convention de nommage recommandée
- terraform validate : vérifier la configuration terraform au format HCL
- tflint : vérifier si il existe des erreurs
- tfsec : identifier des failles de sécurité dans la configuration terraform
- terraform_docs : générer des fichiers README

Configuration pre-commit

```
repos:
- repo: https://github.com/terraform-docs/terraform-docs
  rev: "v0.19.0"
  hooks:
    - id: terraform-docs-go
      args: ["markdown", "table", "--output-file", "README.md", "./"]
- repo: https://github.com/antonbabenko/pre-commit-terraform
  rev: "v1.97.4"
  hooks:
    - id: terraform_fmt
    - id: terraform_tflint
    - id: terraform_validate
    - id: terraform_tfsec
```

pre-commit dans des pipelines ?

Avantages

- Utiliser la même configuration pre-commit qu’en local
- Simplifier la maintenance

Inconvénients

- Gestion des rapports au format attendu
- Utiliser une configuration spécifique pour les pipelines
- Migrer la CI existante

Conclusion

- Avantages à l’intégrer dès la mise en place d’un nouveau projet
- Élargir au fil de l’eau sur un projet existant
- Permet de mettre en place des linters et formatters simplement