Formation Git Avancée

VI - Les hooks

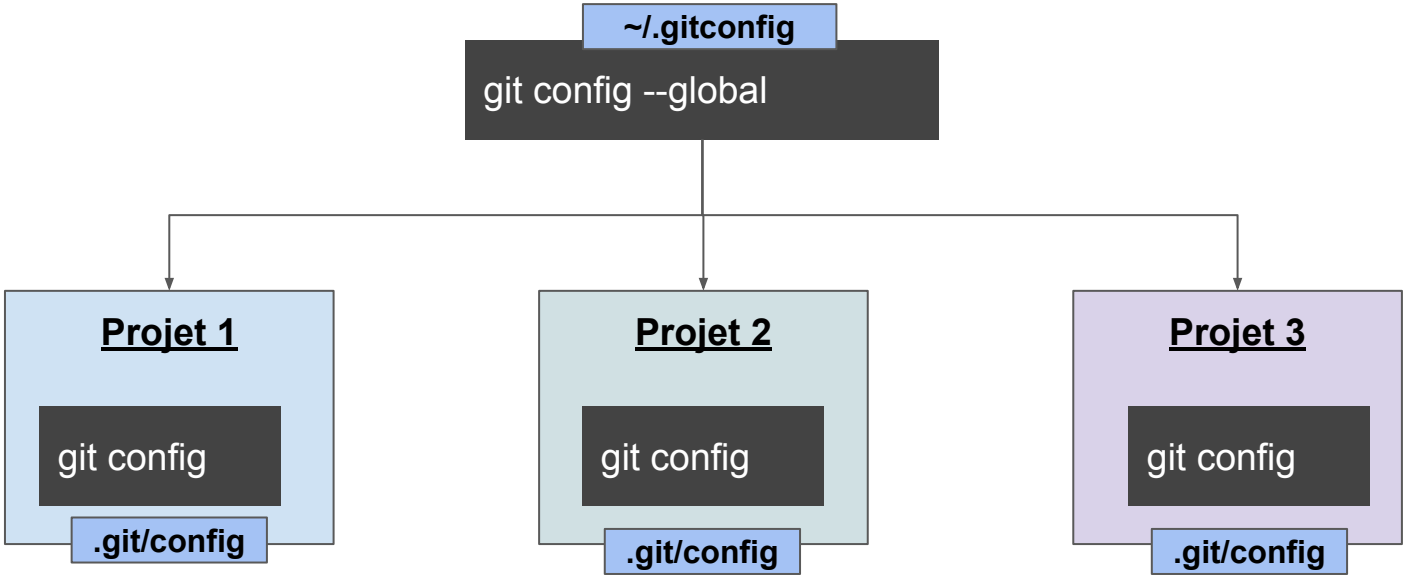
Codeur Pro

www.codeur-pro.fr

Arnaud MERCIER

arnaud.mercier.formation@gmail.com

Configuration de Git



Configurations de Git

```
$ git config user.name “nom_utilisateur”
$ git config user.email <email utilisateur>
```

```
$ git config core.editor <editor>
$ git config merge.tool <tool>
```

```
$ git config commit.template <path_file>
```

```
$ man git-config
```

Les alias

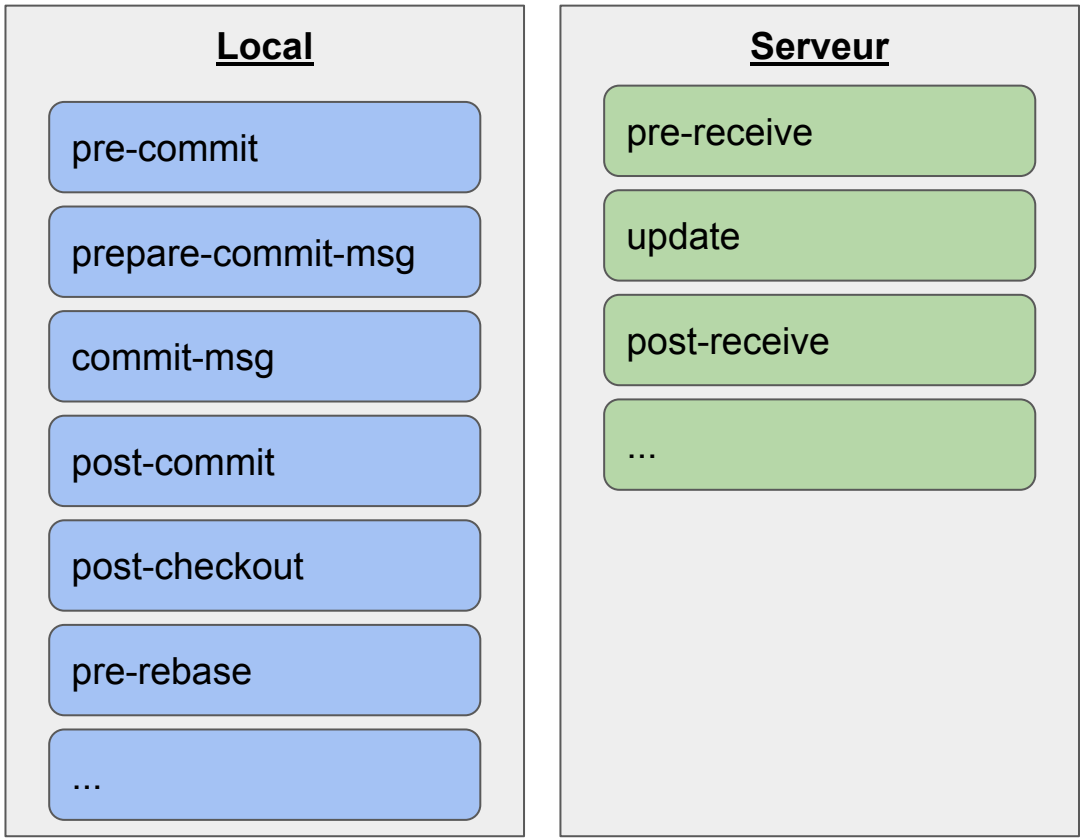
```
$ git config --global alias.slog 'log --oneline --decorate --graph --all'
```

```
$ git config --global alias.unstage 'reset HEAD'
$ git unstage hello.html

=

$ git reset HEAD hello.html
```

```
$ git config --global alias.last 'log -1 HEAD'
$ git last
commit 66938dae3329c7aebe598c2246a8e6af
Author: Arnaud MERCIER

    Ajout médaille d'or
```

```
$ git config alias.[short] 'cmd'
```

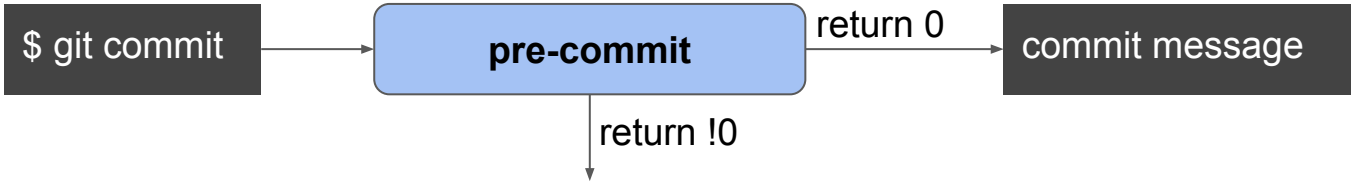
Les Hooks



- Les hooks résident dans le dossier .git/hooks.
- Les hooks doivent être exécutables. chmode +x <script>
- Les hooks peuvent être écrit dans n'importe quel langage de script.
- Des exemples sont disponibles dans .git/hooks avec l'extension .sample
- Les hooks ne sont pas dupliqué lors d'un "git clone"

Les Hooks Locals

pre-commit

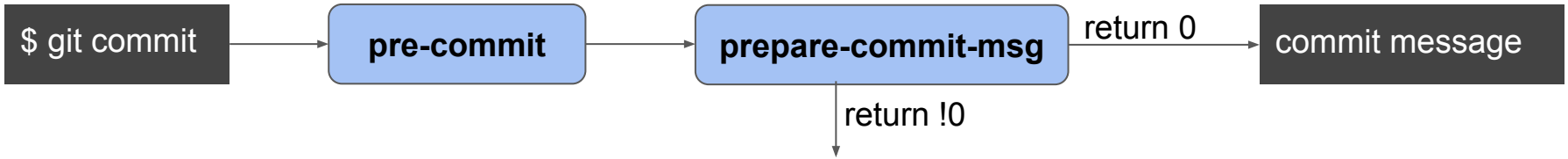


- Permet de réaliser des tests sur son code avant de faire le commit:
  - Si le retour de pre-commit est 0, alors on passe a la prochaine étapes du commit.
  - Sinon le commit est annulé

- Ce script ne prend pas d'arguments
- Peut être ignoré via l'option -n de la commande git commit

Les Hooks Locals

prepare-commit-msg

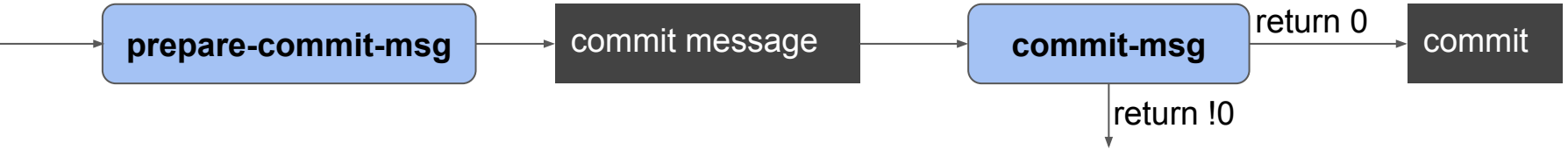


- Permet de modifier le message de commit par défaut
  - Si le retour du script est 0, alors on passe a la prochaine étapes du commit.
  - Sinon le commit est annulé

- Ce script comporte 3 arguments:
  1. Le nom d'un fichier temporaire qui contient le message. Changez le message de commit en modifiant le fichier en place.
  2. Le type de commit. Il peut s'agir de message (option -m), template (option -t), merge (si le commit est un commit de merge) ou squash (si fusion de commits).
  3. L'empreinte SHA1 du commit concerné. Uniquement attribuée si l'option -c, -C ou --amend a été ajoutée.

Les Hooks Locals

commit-msg



- Appelé après la saisie utilisateur du message de commit, il permet de vérifier le message de commit
  - Si le retour du script est 0, alors on passe a la prochaine étapes du commit.
  - Sinon le commit est annulé

- Ce script comporte 1 argument:
  1. Le nom d'un fichier temporaire qui contient le message.

Les Hooks Locals

post-commit

- Il ne peut modifier les conséquences de l'opération git commit, c'est pourquoi il est surtout utilisé à des fins de notification.

Les Hooks Locals

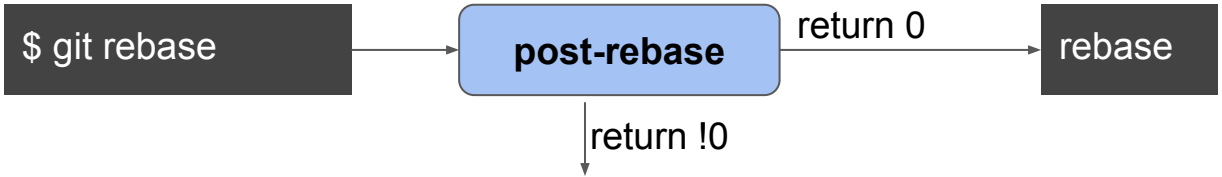
post-checkout

- Appelé immédiatement après un git checkout. Cette fonction est pratique pour supprimer des fichiers qui pourraient semer la confusion dans votre répertoire de travail. Ou encore pour modifier votre espace de travail en fonction de la branche cible.

- Ce script accepte 3 arguments:
  1. Réf du HEAD précédent
  2. Réf du nouveau HEAD
  3. Cette option vous indique s'il s'agissait d'un checkout de branche ou de fichier. Les options seront respectivement 1 et 0.

Les Hooks Locals

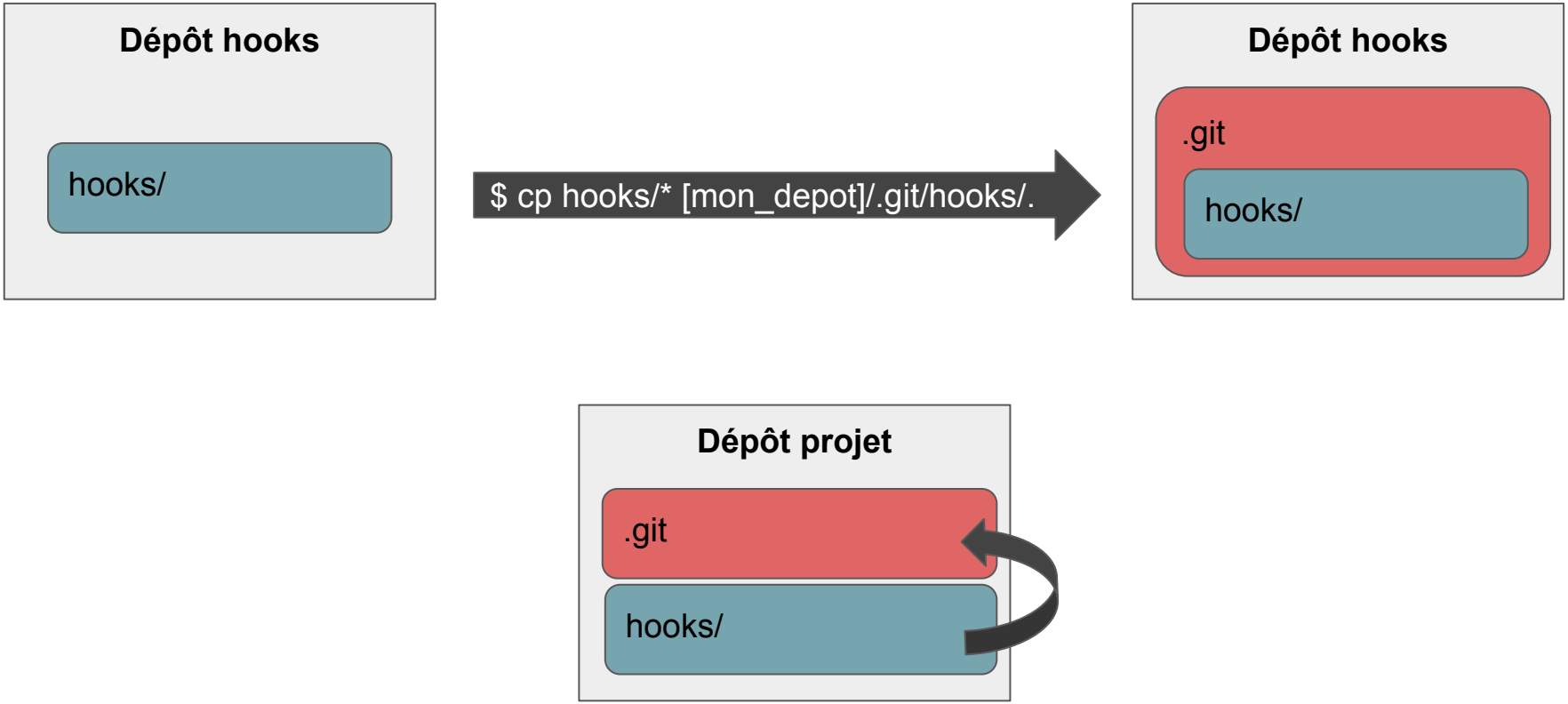
post-rebase



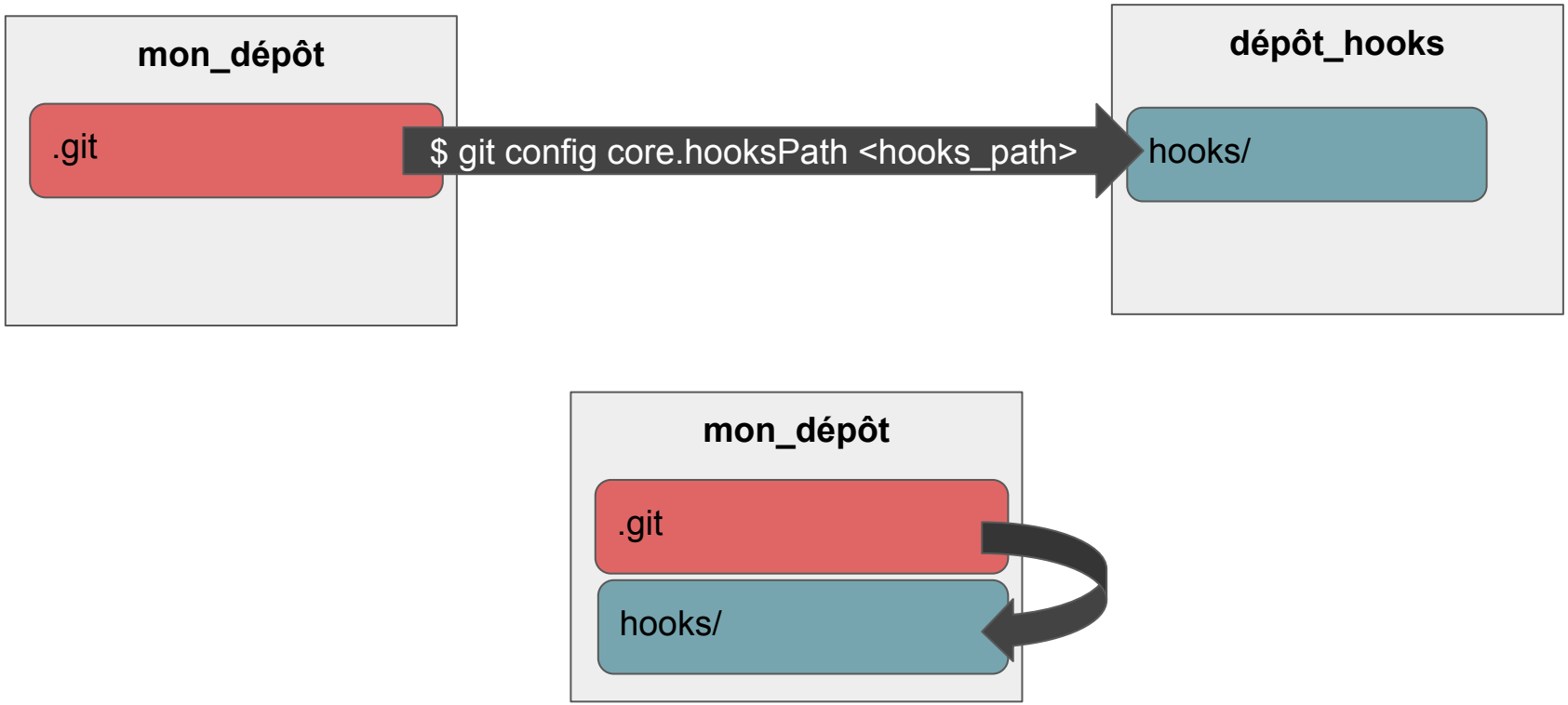
- Appelé avant que git rebase n'apporte le moindre changement. C'est le moment d'empêcher tout scénario catastrophe de se produire.

- Ce script nécessite deux paramètres:
  1. La branche upstream à partir de laquelle la série a été forkée.
  2. La branche rebasée. Si le rebase est lancé depuis cette dernière le paramètre est vide.

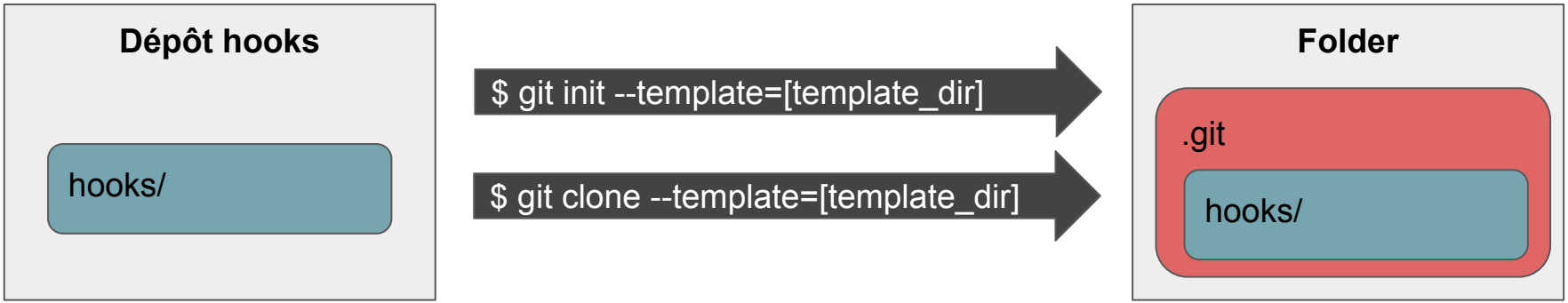
A la main


Dépôt hooks
hooks/
$ cp hooks/* [mon_depot]/.git/hooks/.
Dépôt hooks
.git
hooks/
Dépôt projet
.git
hooks/

via git config



Les templates



le dossier template doit contenir un dossier hooks à la racine

Dépôts serveur

dépôt git de travail

code source

Dossier .git

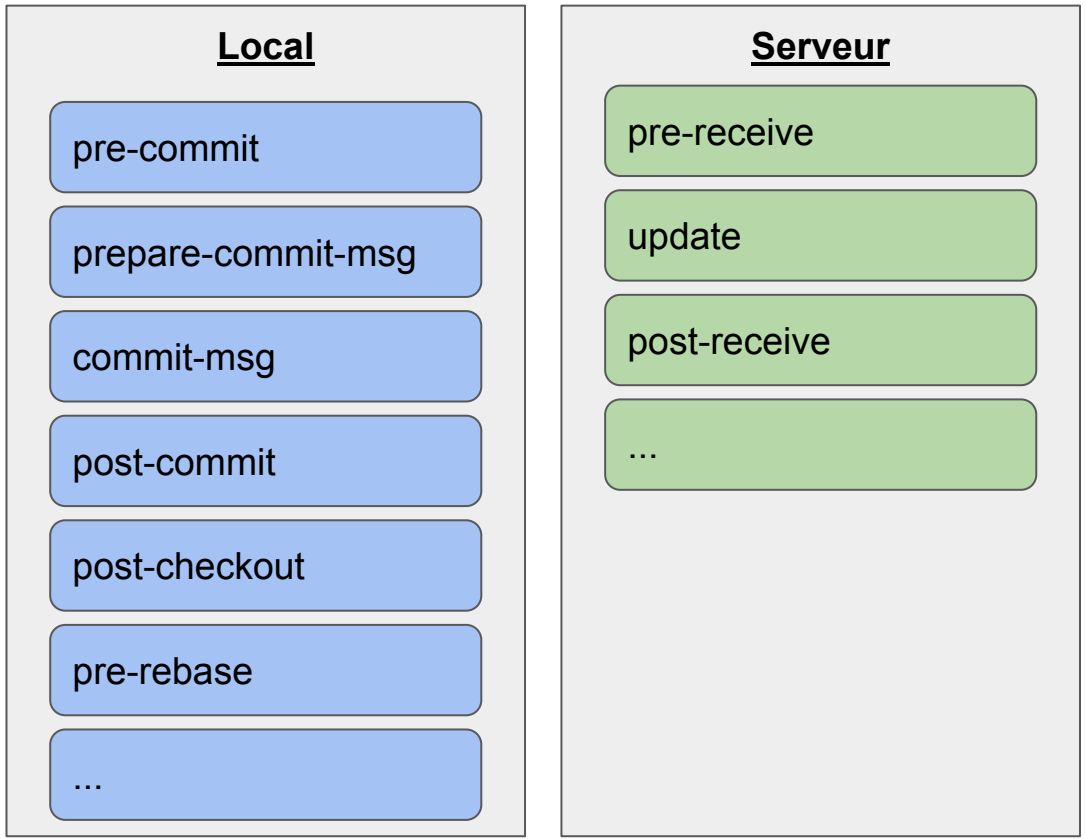
$ git init <repo_name>

dépôt git serveur

$ git init --bare <repo_name>

le dossier .git contient l'historique, ne pas le supprimer !

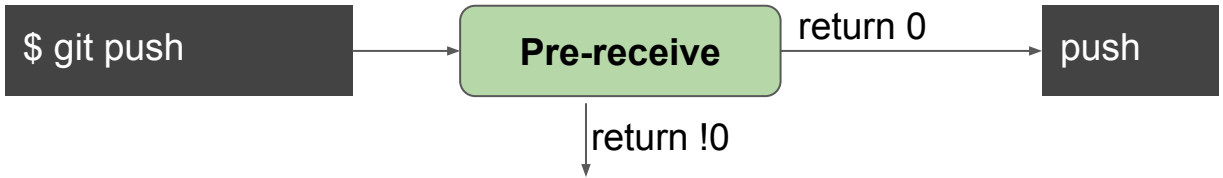
Les Hooks



- Les hooks résident dans le dossier .git/hooks.
- Les hooks doivent être exécutables. chmode +x <script>
- Les hooks peuvent être écrit dans n'importe quel langage de script.
- Des exemples sont disponibles dans .git/hooks avec l'extension .sample
- Les hooks ne sont pas dupliqué lors d'un "git clone"

Les Hooks Serveurs

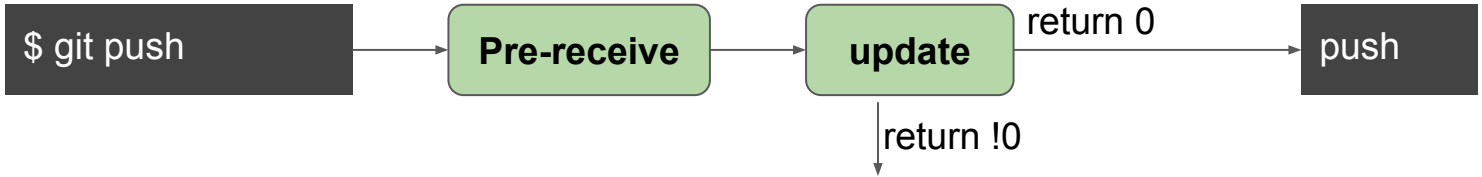
Pre-receive



- Appelé à chaque fois qu'un développeur utilise git push pour pusher des commits vers le dépôt.
- Si le retour du script est 0, alors le push est accepté, sinon il est refusé

Les Hooks Serveurs

update



- Appelé après pre-receive.
- Appelé pour chaque référence mises à jours.
- Si le retour du script est 0, alors le push est accepté, sinon il est refusé

- Ce script accepte 3 arguments:
  1. Le nom de la réf mise à jour
  2. Le nom de l'ancien objet stocké dans la réf
  3. Le nom du nouvel objet stocké dans la réf.

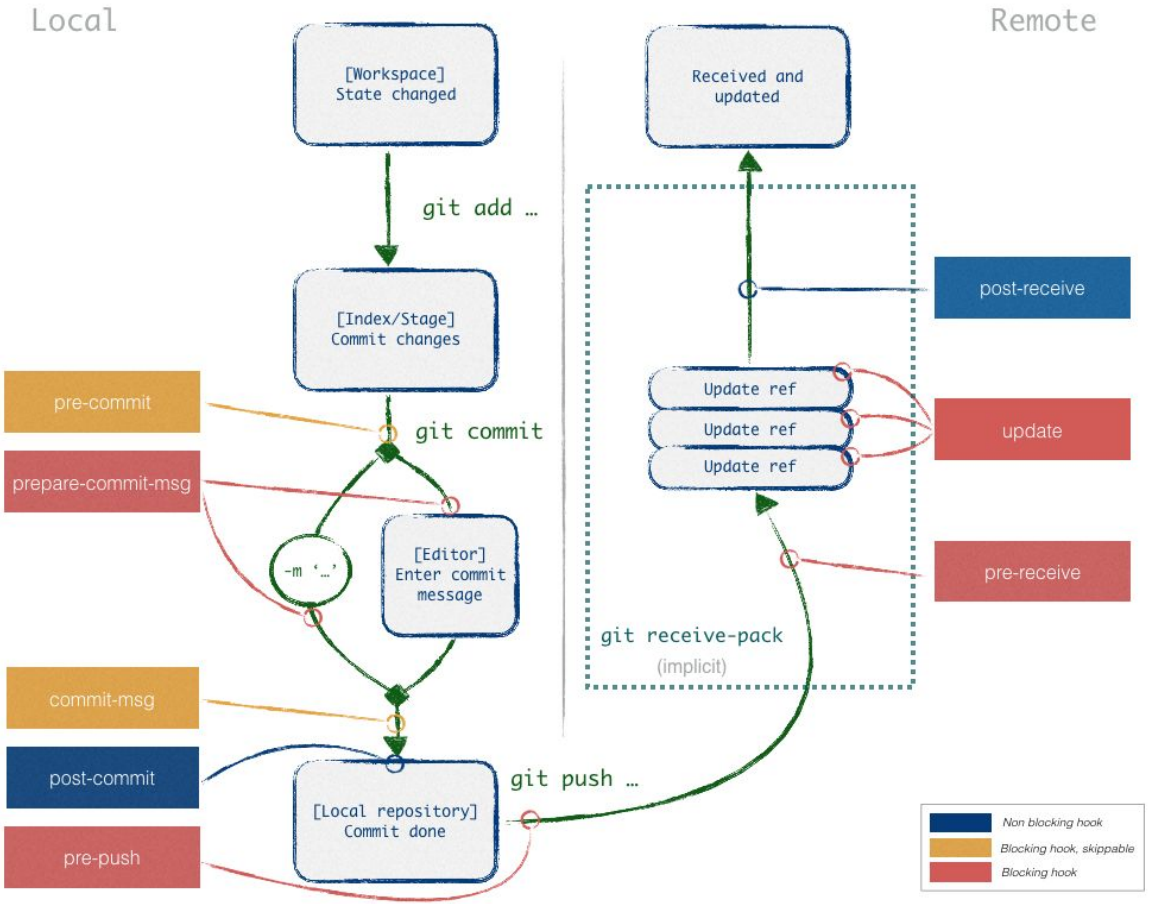
Les Hooks Serveurs

post-receive

- Le hook post-receive est appelé après un push réussi. C'est un emplacement idéal pour effectuer des notifications. Bon nombre de workflows préfèrent cet emplacement à post-commit, car les changements sont accessibles sur un serveur public et n'apparaissent pas uniquement sur l'ordinateur local de l'utilisateur. Le hook post-receive est souvent utilisé pour envoyer des e-mails à d'autres développeurs ou pour enclencher un système d'intégration continue.

- Ce script n’accepte pas d’arguments.

Bilan


Local
Remote
[Workspace]
State changed
git add …
[Index/Stage]
Commit changes
pre-commit
git commit
prepare-commit-msg
-m '…'
[Editor]
Enter commit
message
commit-msg
post-commit
pre-push
[Local repository]
Commit done
git push …
Received and
updated
post-receive
Update ref
Update ref
Update ref
update
pre-receive
git receive-pack
(implicit)
Non blocking hook
Blocking hook, skippable
Blocking hook