GIT aide-memoire

Table of Contents

Désactiver temporairement la vérification des certificats ssl

Il peut arriver lors de l'expiration des certificats que l'on ait besoin de désactiver temporairement la vérification ssl pour pouvoir clôner un dépôt git, cette solution présente l'avantage qu'elle ne prend effet que pour la commande associée:

```
GIT_SSL_NO_VERIFY=true git clone http://<url>
```

Annuler le dernier commit

```
git reset --hard HEAD~1
```

Ramènera la branche HEAD actuelle à la révision spécifiée :

- **HEAD~1** permet de revenir à celui qui précède la révision actuelle, ce qui annule effectivement le dernier commit.
- **–soft** garantit que les modifications dans les révisions annulées sont conservées.

  Après avoir exécuté la commande, on retrouvera les modifications en tant que modifications locales non validées dans la copie de travail.

Si on ne souhaite pas conserver ces modifications, utiliser simplement le drapeau –hard.

il faut être certain de ne plus avoir besoin de ces modifications avant de le faire, car cela aura pour conséquence de supprimer tous les fichiers et dossiers ajoutés depuis le dernier commit.

Annuler un git reset --hard HEAD~1

Si on viens d'exécuter git reset et qu'on a rien fait d'autre depuis, on peut revenir à la situation précédente avec :

```
git reset --hard@{1}
```

Annuler un git add

Pour annuler **git add** utiliser

```
git reset
```

Créer un mirroir repo git distant

Configurer le proxy

```
git config --global http.proxy http://proxy.infra.dgfip:3128
```

Configurer l'alias du dépôt distant

```
git remote add {alias}
http://{user}:{password}@{url-du-repo-git}/{user}/vagrant.git
```

Refaire un add global

```
git add -A
```

Faire un commit (normalement si tous les changements devraient déjà être commités si ce n'est pas le cas faire un push sur le dépôt maître avant puis recommencer le add suivi du commit)

```
git commit -m "initialisation mirroir"

Sur la branche master
Votre branche est à jour avec 'origin/master'.

rien à valider, la copie de travail est propre
```

Faire un push

```
git push -u {alias} master

Counting objects: 12, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.56 KiB | 0 bytes/s, done.
Total 12 (delta 1), reused 0 (delta 0)
To http://{user}:{password}@{url-du-repo-git}/{user}/vagrant.git
 * [new branch]      master -> master
La branche master est paramétrée pour suivre la branche distante master
depuis {alias}.
```

Aplatir le sous-répertoire git en convertissant les répertoires en nouvelles branches

Dans certaines applications, comme Gollum, la structure de répertoires n'est pas prise en charge. Cela signifie qu'en utilisant le navigateur d'arborescence de fichiers pour accéder aux fichiers, on ne peut avoir le rendu attendu.

Une sollution pour contourner cette limitation est d'aplatir l'arborescence avec git-filter-branch:

```
for branch in `find . -type d -d 1`; do
    git filter-branch --subdirectory-filter $branch --prune-empty -- --all
done
```

`git filter-branch` sert à copier les validations existantes tout en appliquant des transformations sur chacune d'elles (avant de créer la nouvelle copie). Les arguments de filter-branch fournissent les transformations et lui indiquent les noms de branche à mettre à jour pour les orienter vers les commits nouvellement copiés au lieu de leurs commits originaux (maintenant copiés). Il ne créera aucun nouveau nom de branche.

La commande `filter-branch` peut réécrire des pans entiers d'un historique, cela peut être très utile. Voici quelques usages communs pour donner une idée de ses capacités :

- **Supprimer un fichier de chaque commit**: `git filter-branch --tree-filter 'rm -f passwords.txt' HEAD`.
- **Faire d'un sous-répertoire la nouvelle racine**: `git filter-branch --subdirectory-filter trunk HEAD`.
- **Modifier globalement l'adresse mail**: 'git filter-branch –commit-filter '\n if [“$GIT*AUTHOR*EMAIL” = “schacon@localhost”];\n then\n GIT*AUTHOR*NAME=“Scott Chacon”;\n GIT*AUTHOR*EMAIL=“schacon@example.com”;\n git commit-tree “$@”;\n else\n git commit-tree “$@”;\n fi' HEAD`