Aide Mémoire Git

Michel Casabianca
casa@sweetohm.net

Voici un petit aide mémoire pour Git. Ceci n'est pas une formation et pour utiliser les commandes listées ici, vous devez savoir ce que vous faites !

Une archive avec les alias et les scripts est disponible ici.

Initialisation

Pour initialiser un repository Git :

```
$ git init
$ git add *
$ git commit -m "Initial import"
```

Il peut être utile d'ajouter un fichier *.gitignore* afin d'ignorer certains fichiers dans Git.

Commiter

Lorsqu'on veut commiter son travail sur un fichier :

```
$ git add my-file
$ git commit -m "Commit message" my-file
```

Pour envoyer les modifications sur le repository :

```
$ git push
```

Pour récupérer les modifications du repository :

```
$ git pull --rebase
```

Revert et reset

Préférer le revert au reset (qui est définitif).

Pour faire un revert du dernier commit :

```
$ git revert HEAD~1
```

Pour faire un revert d'un commit en particulier *cafebabe* :

```
$ git revert cafebabe
```

Pour faire un reset du dernier commit en gardant les modifs non commitées :

```
$ git reset --soft HEAD~1
```

Pour faire un reset du dernier commit sans garder les modifs :

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

Il faudra après faire un `push -f` pour forcer la modification de l'historique.

Fusion

Pour fusionner plusieurs commits, la meilleure façon est de le faire lors du merge d'une branche.

Par exemple, pour merger le branche *foo* sur *master* en squashant les commits en un seul :

```
$ git checkout master
$ git pull
$ git merge --squash foo
$ git commit
$ git push origin master
```

Ceci présente l'avantage de ne pas modifier l'historique.

Pour squasher les commits jusque celui précédent *cafebabe* :

```
$ git rebase -i cafebabe
```

Il faudra ensuite faire un `git push -f`. Ceci modifie l'historique, il faut donc éviter de le faire sur une branche partagée. Si c'est le cas, un pull doit être fait avec `git pull --rebase`.

Branches

Pour lister les branches en local :

```
$ git branch
```

Pour lister aussi celles du repository :

```
$ git branch -a
```

Pour se placer sur une branche (en local ou sur le repository) :

```
$ git checkout my-branch
```

Pour créer une branche :

```
$ git branch my-branch
```

Pour créer une branche et se placer dessus :

```
$ git checkout -b my-branch
```

Pour effacer une branche :

```
$ git branch -d branch-to-delete
```

Parfois Git se plaint (souvent à tord) que la branche que l'on veut effacer n'a pas été mergée, dans ce cas on pourra l'effacer avec :

```
$ git branch -D branch-to-delete
```

Pour pousser une branche vers le repository :

```
$ git push -u my-branch origin
```

Pour effacer une branche en local et sur le repository :

```
$ git branch -d branch-to-delete
$ git push origin --delete branch-to-delete
$ git fetch --all --prune
```

Pour renommer une branche :

```
$ git branch -m old-branch new-branch
```

Pour renommer une branche en local et sur le repository :

```
$ git branch -m old-branch new-branch
$ git push origin :old-branch
$ git push --set-upstream origin new-branch
```

Pour mettre à jour la liste des branches de l'origine :

```
$ git remote update origin --prune
```

Tags

Pour créer un tag :

```
$ git tag my-tag
```

Pour effacer un tag :

```
$ git tag -d my-tag
```

Pour effacer un tag en local et sur le repository :

```
$ git tag -d foo
$ git push origin :refs/tags/foo
```

Pour se placer sur le tag :

```
$ git checkout tag-name
```

Créer un repo

Pour ajouter un projet GIT existant à un répository :

- Effacer l'origine du projet :

```
$ cd my-project
$ git remote rm origin
```

- Faire un clone bare du projet :

```
$ cd ..
$ git clone --bare my-project
```

Cela va créer un répertoire *my-project.git* à placer dans le répertoire du repository. On pourra accéder à ce repository par l'URL *ssh://user@address:/path/to/repo/project.git*, pourvu que cette machine soit accessible par SSH.

Alias

J'utilise un certain nombre d'alias pour faciliter l'utilisation de Git sur la ligne de commande.

```
# statut du projet
alias gs='git fetch && git status'
# pull avec rebase (évite beaucoup de problèmes)
alias gl='git pull --rebase'
# simple push
alias gh='git push'
# commiter toutes les modifications avec un commentaire
alias gc='git commit -a -m'
# commiter un fichier avec un commentaire
alias gm='git commit -m $2 $1'
# lister les branches
alias gb='git branch'
# lister toutes les branches
alias gba='git branch -a'
```

Scripts

D'autre part, pour des tâches plus complexes, j'utilise un certain nombre de scripts qui voici.

git-branch-delete

Efface une branche localement et sur le repository. Il faut lui passer le nom de la branche.

```
#!/bin/sh

set -a

if [ "$#" -ne 1 ]; then
  echo "Pass branche to delete"
  exit 1
fi

BRANCH=$1

git branch -d ${BRANCH}
git push origin --delete ${BRANCH}
git fetch --all --prune
```

git-branch-rename

Renomme une branche localement et sur le repository. Il faut lui passer l'ancien et le nouveau nom de la branche.

```
#!/bin/sh

set -e

if [ "$#" -ne 2 ]; then
  echo "Pass OLD and NEW branches on command line"
  exit 1
fi

OLD=$1
NEW=$2

git branch -m ${OLD} ${NEW}
git push origin :${OLD}
git push --set-upstream origin ${NEW}
```

git-branch-squash

Merge la branche courante dans *master* en squashant les commits.

```
#!/bin/sh

set -e

CURRENT=$(git rev-parse --abbrev-ref HEAD)
git diff-index --quiet HEAD -- || (echo "ERROR There are uncommitted changes" && ex
test "$CURRENT" != "master" || (echo "ERROR You already are on branch master" && ex
git checkout master
git pull
git merge --squash $CURRENT
git commit
git push origin master
```

git-tag-delete

Efface un tag localement et sur le repository. Il faut lui passer le nom du tag à effacer.

```
#!/bin/sh

set -e

if [ "$#" -ne 1 ]; then
  echo "Pass tag to delete on command line"
  exit 1
fi
```

```
TAG=$1

git tag -d ${TAG}
git push origin :refs/tags/${TAG}
```

git-tag-rename

Renomme un tag localement et sur le repository. Il faut lui passer l'ancien et le nouveau nom du tag.

```
#!/bin/sh

set -e

if [ "$#" -ne 2 ]; then
  echo "Pass OLD and NEW tags on command line"
  exit 1
fi

OLD=$1
NEW=$2

git tag ${NEW} ${OLD}
git tag -d ${OLD}
git push origin :refs/tags/${OLD}
git push origin ${NEW}
```

Enjoy!