Git: Résumé des commandes

Juvigny Xavier

Mars 2017

1 Configuration

- Nom de l'utlisateur

```
git config --global user.name "Olivier.Martin"
```

- E-mail de l'utilisateur

```
git config --global user.email "Olivier.Martin@ensta.fr"
```

- Choix de l'éditeur utilisé pour git

```
git config --global core.editor subl
```

- Choix de l'outil de comparaison de deux fichiers :

```
git config --global merge.tool diff
```

- Affiche la configuration choisie pour git :

```
git config --list
```

2 Commandes principales

- État des fichiers (modifiés, effacés, soumis, ...) :

```
git status
```

- Affichage des diverses branches :

```
git branch
```

- Créer une nouvelle branche :

```
git branch nom_de_ma_branche
```

- Changer de branche :

```
git checkout nom_de_ma_branch
```

- Première soumission :

```
git add .
git commit -m "initial commit"
```

- Soumissions suivantes :

```
git add chemin_vers_mon_fichier
git commit -m "message du commit"
```

- Annule la dernière soumission et les modifications associées :

```
git reset --hard md5_commit
git push --force
```

- Mettre à jour le dépot local :

```
git pull
```

- Envoyer sa soumission au dépot distant :

```
git push
```

- Supprimer un fichier du répertoire et de l'index des fichiers gérés par git :

```
git rm nom_du_fichier
```

- Supprimer un fichier uniquement de l'index des fichiers gérés par git :

```
git rmg --cached nom_du_fichier
```

3 Afficher les différences entre diverses versions

- Affiche la différence entre le contenu du dernier commit et celui du répertoire de travail. Cela correspond à ce qui serait commité par git commit -a.

```
git diff HEAD
```

- Affiche la différence entre le contenu pointé par A et celui pointé par B.

```
git diff A B
```

- Diff entre un dossier présent sur deux branches

```
git diff master..MA_BRANCH chemin/vers/mon_dossier
```

4 Afficher les dernières soumissions

- Afficher les dernières soumissions :

```
git log
```

- Affiche les X dernières soumissions :

```
git log -n X
```

- Affiche un ensemble de commits par date :

```
git log --since=date --until=date
```

- Représentation de l'historique à partir de `HEAD` (soumission/branche) :

```
git log --oneline --graph --decorate
```

- Représentation de l'historique à partir d'un fichier (commit / branch) :

```
git log --oneline --graph --decorate nom_du_fichier
```

5 Annuler des commits

5.1 Version soft

Seul le commit est retiré de Git : vos fichiers, eux, restent modifiés. Vous pouvez alors à nouveau changer vos fichiers si besoin est et refaire un commit.

```
git reset HEAD^
```

Pour indiquer à quel commit on souhaite revenir, il existe plusieurs notations :

- `HEAD` : dernier commit ;
- `HEAD^` : avant-dernier commit ;
- `HEAD^^` : avant-avant-dernier commit ;
- `HEAD~2` : avant-avant-dernier commit (notation équivalente) ;
- `d6d98923868578a7f38dea79833b56d0326fcba1` : indique un numéro de commit précis;

5.2 Version hard

Si vous voulez annuler votre dernier commit et les changements effectués dans les fichiers, il faut faire un reset hard. Cela annulera sans confirmation tout votre travail !

- Annuler les commits et perdre tous les changements

  ```
  git reset --hard HEAD^
  ```

- Annuler les modifications d'un fichier avant un commit
  Si vous avez modifié plusieurs fichiers mais que vous n'avez pas encore envoyé le commit et que vous voulez restaurer un fichier tel qu'il était au dernier commit :

  ```
  git checkout nom_du_fichier
  ```

- Annuler/Supprimer un fichier avant un commit
  Supposer que vous venez d'ajouter un fichier à Git avec git add et que vous vous apprêtez à le "commiter". Cependant, vous vous rendez compte que ce fichier est une mauvaise idée et vous voulez annuler votre git add.
  Il est possible de retirer un fichier qui avait été ajouté pour être « commité » en procédant comme suit :

  ```
  git reset HEAD -- nom_du_fichier_a_supprimer
  ```

6 Travailler avec Github

Récuperer le repo sur Github :

```
git clone https://github.com/JuvignyEnsta/Projet
```

Mon repo est composé d'au moins deux branches.

develop : dédié au développement et résolution de bug.

master : reflète le code en production. Personne ne doit travailler directement sur cette branche.

Pour récupérer votre branche develop

```
git branch -a
git checkout origin/develop
git checkout -b develop origin/develop
git branch
```

Développement : Branche develop
Production : Branche Master

Pour développer sur la branche develop puis soumettre sur la branche master :

```
# On se met sur la branche master
git checkout master
# On merge la branche develop
git merge develop
# Pour pousser les changements
git push origin master
# Penser à revenir sur develop
git checkout develop
```