# CheatSheet Git

12 mars 2018

### Create

Cloner un repo existant distant en local

```
$ git clone ssh://user@domain.com/repo.git
```

Créer un repo local simple

```
$ git init
```

### Local Change

Afficher les fichiers modifiés dans le repo local

```
$ git status
```

Afficher les modifications faites sur les fichiers trackés

```
$ git diff
```

Afficher les modifications faites seulement sur un seul fichier

```
$ git diff <fichier>
```

Ajouter des fichiers à la staging area

```
$ git add -v <fichier1> <fichier2> etc.
```

Ajouter tous les fichiers modifiés à la staging area (trackés + non trackés)

```
$ git add -Av
```

Ajouter seulement une partie des modifications d'un fichier dans la staging area

```
$ git add -p <fichier>
```

- y : Oui, ajouter le morceau présenté à la staging area.
- n : Non, ne pas ajouter.
- d : Non, ne pas ajouter ni celui-ci, ni les suivants du fichier en cours.
- s : Découper le groupe de modification en modifications unitaires.
- e : Éditer à la main le morceau sélectionné.
- q : Arrêter les ajouts

Ajouter tous les changements sur les fichiers trackés et commiter immédiatement

```
$ git commit -A
```

Commiter le contenu de la staging area

```
$ git commit
```

Ajouter au commit précédent le contenu actuel de la staging area et modification du message (- -no-edit pour ne pas le modifier)

```
$ git commit --amend
```

Mettre de coté les modifications faites sur le repertoire de travail et le ramener à l'état du commit pointé par HEAD

```
$ git stash
```

Restaure dans le repertoire de travail le dernier lot de modifications mis de coté par la commande stash

```
$ git stash pop
```

### Commit history

Afficher l'historique des commits

```
$ git log
```

Afficher l'historique des modifications pour un seul fichier (Ajouter - - avant <file> pour inclure les fichiers supprimés, ou dont le nom a été modifié)

```
$ git log -p <file>
```

Afficher les auteurs des modifications d'un fichier

```
$ git blame <file>
```

### Branch & Tag

Afficher l'ensemble des branches existantes (-v pour avoir le détail du dernier commit de chaque branche)

```
$ git branch --all
```

Faire pointer HEAD sur le pointeur d'une branche

```
$ git checkout <branch>
```

Créer une nouvelle branche

```
$ git branch <newbranch>
```

Supprimer une branche locale

```
$ git branch -d <branch>
```

Apposer un tag sur le commit pointé par HEAD

```
$ git tag <tag-name>
```

Faire pointer une branche sur un commit

```
$ git branch -f <branch> <commit>
```

### Update & Publish

Lister les remote existants

```
$ git remote -v
```

Lister les informations d'un remote spécifique

```
$ git remote show <remote>
```

Ajouter un nouveau remote

```
$ git remote add <short-name> <url>
```

Télécharger toutes les modifications présentes sur un remote sans les intéger au repo local

```
$ git fetch --all <remote>
```

Télécharger les modifications présentes sur une branch du remote et les intégrer au repo local

```
$ git pull <remote> <branch>
```

Publier sur le remote le contenue d'une branche (–tags pour push les tags en même temps)

```
$ git push <remote> <branch>
```

Supprimer une branche présente sur un remote

```
$ git branch -dr <remote/branch>
```

### Merge & Rebase

Merge une branche dans le HEAD

```
$ git merge <branch>
```

Rebase le HEAD sur une branche (ne jamais rebase des commits déjà publiés !)

```
$ git rebase <branch>
```

Annuler un rebase

```
$ git rebase --abort
```

Continuer un rebase après avoir résolu les conflits

```
$ git rebase --continue
```

Ignorer un commit dans durant le rebase

```
$ git rebase --skip
```

Choisir les commits qui doivent être rebase sur une branche (Rebase interactif)

```
$ git rebase -i <branch>
```

- p, pick = utiliser le commit
- r, reword = utiliser le commit, mais éditer le message du commit
- e, edit = utiliser le commit, mais supend la procédure pour faire des corrections (git commit –amend)
- s, squash = utiliser le commit, mais le fusionne avec le commit précédent

Ouvrir l'outils de merge configuré dans git (fonctionne uniquement si il est configuré)

```
$ git mergetool
```

### Undo

Supprime le contenu de la staging area

```
$ git reset HEAD
```

Ramène l'état du répertoire de travail à l'état du dernière commit

```
$ git reset --hard HEAD
```

Ramène l'état d'un fichier du répertoire de travail à son état du dernière commit

```
$ git checkout HEAD <fichier>
```

Ramène l'état d'un fichier du répertoire de travail à son état dans un commit précédent

```
$ git checkout <commit> <fichier>
```

Ajoute à la staging area l'inverse du contenu d'un commit

```
$ git revert <commit>
```

### Submodules

Ajouter un submodule au répertoire de travail

```
$ git submodule add <url>
```