Cheatography

Git Basics Cheat Sheet

by Gaëlle (gaelle3182) via cheatography.com/26108/cs/7216/

Configuration

Configuration système (pour tous les utilisateurs et tous les dépôts)

```
$ git config --system [param etre] [valeur]
```

Configuration utilisateur spécifique

```
$ git config --global [param etre] [valeur]
```

Configuration dépôt spécifique

```
$ git config [param etre] [valeur]
```

Vérifier la configuration

```
$ git config --list
```

Chaque fichier écrase les valeurs du fichier précédent. Les valeurs dans la configuration du dépôt sont les plus spécifiques et priment donc sur les valeurs de la configuration utilisateur qui elles-mêmes l'emportent sur les valeurs de la configuration système.

Récupérer un dépot git

Initialiser un nouveau dépôt

```
$ cd projec t_d ire ctory
$ git init
```

Cloner un dépôt existant

```
$ git clone [url]
```

Cloner un dépôt existant en changeant le nom du répertoire

```
$ git clone [url] myproject
```

Explorer l'historique

Voir tout l'historique

```
$ git log
```

Formater l'affichage des logs : affichage des logs sur une seule ligne

```
$ git log --pret ty= oneline
```

Limiter le nombre de logs affichés : avoir les 2 derniers logs

Explorer l'historique (cont)

```
$ git log -2
```

Options pour formater les logs

Option	Description
-p	Affiche les changements introduits dans chaque commit
--stat	Affiche les statistiques des fichiers modifiés dans chaque commit
--shortstat	Affiche des statistiques abrégées
--name-only	Affiche juste la liste des fichiers modifiés
--name-status	Affiche la liste des fichiers modifiés mais aussi ajoutés et supprimés
--abbrev--commit	Affiche seulement les quelques premiers caractères du checksum de commit
--relative-date	Affiche la date dans un format relative (ex depuis 2 jours)
--graph	Affiche un graphe en ASCII des branches et des fusions en plus des informations de log
--pretty	Affiche les logs dans un format alternatif. Plusieurs options possibles : oneline, short, full, fuller et format

Options pour limiter les logs affichés

-(n)	Afficher les n derniers commits
--since,--after	Afficher les commits faits depuis la date indiquée
--until,--before	Afficher les commits faits avant la date indiquée
--author	Afficher les commits dont l'entrée author correspond à la chaîne indiquée
--commiter	Afficher les commits dont l'entrée commiter correspond à la chaîne indiquée
--grep	Afficher les commits dont le message contient la chaîne indiquée
-S	Afficher les commits qui ajoutent ou enlèvent du code comportant la chaîne indiquée

Travailler avec le dépôt local

Vérifier l'état des fichiers

```
$ git status
```

Vérifier l'état des fichiers dans un format compact

```
$ git status -s
```

Suivre un nouveau fichier (versionner un nouveau fichier) ou indexer un fichier modifié

```
$ git add [filename]
```

Suivre tous les fichiers du répertoire de travail

```
$ git add .
```




By **Gaëlle** (gaelle3182)

cheatography.com/gaelle3182/

Published 28th April, 2016.
Last updated 8th May, 2016.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Git Basics Cheat Sheet

by Gaëlle (gaelle3182) via cheatography.com/26108/cs/7216/

Travailler avec le dépôt local (cont)

Voir les changements entre le répertoire de travail et la zone d'indexation (staged area) = voir les changements qui ne sont pas indexés

```
$ git diff
```

Voir les changements entre la zone id'indexation et le dépôt = les changements qui n'ont pas été commité

```
$ git diff --changed
```

Enregistrer les changements dans le dépôt

```
$ git commit
```

```
$ git commit -m " [me ssa ge] "
```

Enregister les changements dans le dépôt sans passe la par la zone d'indexation

```
$ git commit -a -m " [me ssa ge] "
```

Ne plus versionner un fichier et le supprimer du répertoire

```
git rm [filename]
```

Ne plus versionner un fichier sans le supprimer du répertoire

```
$ git rm --cached [filename]
```

Renommer un fichier

```
$ git mv [file_ from] [file_to]
```

Revenir en arrière

Refaire un commit (pour changer le message ou ajouter des fichiers oubliés)

```
$ git --amend
```

Enlever un fichier de la zone d'indexation

```
$ git reset HEAD [filename]
```

Supprimer les derniers changements d'un fichier (revenir au fichier comme il était au dernier commit)

```
$ git checkout -- [filename]
```

Travailler avec les dépôts distants

Lister les noms des dépôts distants référencés localement

```
$ git remote
```

Lister les dépôts distants avec leur url

```
$ git remote -v
```

Ajouter/Référencer un dépôt distant

```
$ git remote add [short name] [url]
```

Récupérer les informations d'un dépôt distant

```
$ git fetch [short name]
```

Récupérer les informations de tous les dépôts distants

```
$ git fetch --all
```

Lister les informations du dépôt local concernant l'état entre branches locales/branches distantes

```
$ git branch -vv
```

Récupérer les données d'un dépôt distant et les fusionner au dépôt local

```
$ git pull
```

Envoyer données sur un dépôt distant (pour lequel on a les droits d'écriture)

```
$ git push [short name] [branc hname]
```

Inspecter un dépôt distant

```
$ git remote show [short name]
```

Renommer la référence (le shortname) d'un dépôt distant

```
$ git remote rename [current shortname] [new shortname]
```

Enlever/Déréférencer un dépôt distant

```
$ git remote rm [short name]
```

Travailler avec les dépôts distants (cont)

La commande `fetch` récupère les informations du serveur distant et répercute les changements sans toucher aux branches locales. Il faut ensuite faire des fusions ou créer des branches locales pour pouvoir éditer les branches distantes qui n'existent pas localement.
La commande `pull` est en quelque sorte une commande magique qui fait un `fetch` suivi d'un `merge` : `pull` regarde quel branche sur quel serveur est suivie par la branche locale puis récupère et fusionne les changements.

Les tags

Lister les tags

```
$ git tag
```

Lister les tags selon un filtre

```
$ git tag -l [pattern]
```

Voir les informations à propos d'un tag

```
$ git show [tagname]
```

Créer un tag annoté

```
git tag -a [tagname] -m "[message]"
```

Créer un tag simple

```
$ git tag [tagname]
```

Taguer des commit précédents

```
$ git tag -a [tagname] [checksum c
```

Transférer un tag sur un dépôt distant

```
$ git push [short name] [tagname]
```

Transférer tous les tags sur un dépôt distant

```
$ git push [short name] --tags
```

Récupérer les fichiers correspondant à un tag création d'une branche à partir de ce tag

```
$ git checkout -b [branc hname] [t
]
```




By **Gaëlle** (gaelle3182)

cheatography.com/gaelle3182/

Published 28th April, 2016.
Last updated 8th May, 2016.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

Git Basics Cheat Sheet

by Gaëlle (gaelle3182) via cheatography.com/26108/cs/7216/

Les branches

Lister les branches

```
$ git branch
```

Lister les branches qui sont fusionnées avec la branche courante

```
$ git branch --merged
```

Lister les branches non fusionnées

```
$ git branch --no-m erged
```

Créer une branche

```
$ git branch [branc hname]
```

Voir sur quelle branche se situe le pointeur de branche (HEAD)

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

Changer de branche

```
$ git checkout [branc hname]
```

Créer une branche et se positionner dessus

```
$ git checkout -b [branc hname]
```

Fusionner des branches (à partir de la branche sur laquelle on est)

```
$ git merge [branc hname]
```

Supprimer une branche

```
$ git branch -d [branc hname]
```

Supprimer une branche diatante

```
$ git push [short name] --delete [branc hnam
e]
```

Ignorer des fichiers

Pour empêcher git de suivre automatiquement des fichiers ou même de montrer les fichiers non suivis, il faut éditer un fichier nommé **.gitignore** dans le dépôt. Ce fichier liste un ensemble de motifs qui permettent de cibler les fichiers à exclure.

Ignorer des fichiers (cont)

En général ce sont des fichiers générés automatiquement comme des fichiers de log ou provenant du système de build.

Les règles pour construire ce fichier sont :

- les lignes vides ou commençant par # sont ignorés
- les glob patterns standards sont utilisables
- un motif qui commence par un slash (/) empêche la récursivité
- un motif qui se termine par un slash (/) indique un répertoire
- un motif qui commence par un point d'exclamation (!) annule le motif

Quelques exemples :

```
# ignorer les fichiers .log sauf le fichier
# lib.log
 *.log
!lib.log
# ignorer les fichiers .tmp dans le dossier
# racine mais pas dans les sous-d ossiers
/*.tmp
```

Recommandations pour construire un bon fichier **.gitignore** : https://github.com/github/gitignore

Sources

Livre Git Pro (fr) : https://git-scm.com/book/fr/v2

Livre Git Pro (en) : https://git-scm.com/book/en/v2

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com