# Astuces Git

Voici plusieurs astuces qui vont aider le travail avec Git.

## Astuce 01 Correction automatique

Si la fonction de **correction automatique de Git est activée**, on peut laisser **Git corriger automatiquement une sous-commande mal saisie**.

Comme exemple …  
On veut vérifier l'état à l’aide de la commande git status mais on entre la commande git stats par accident.   
Dans des circonstances normales, Git affiche que 'stats' n'est pas une commande valide …

**>> git stats  
git: 'stats' n'est pas une commande git. Voir 'git --help'.  
La commande la plus similaire est   
status**

Pour éviter des scénarios similaires, on peut **activer la correction automatique Git** dans sa configuration Git …  
**>> git config --global help.autocorrect 1**

Si on souhaite que **cela s'applique uniquement au référentiel actuel, on omet l'option --global**.

Au lieu de suggérer une sous-commande alternative, Git exécute maintenant simplement la suggestion, qui dans ce cas était git status.

## Astuce 02 Optimisation du référentiel

Il est possible de garder son référentiel propre et à jour avec quelques pratiques simples.  
 L'une des meilleures pratiques consiste à utiliser le **fichier .gitignore** .

En utilisant ce fichier, on indique à Git de **ne pas stocker de nombreux fichiers indésirables** tels que des binaires, des fichiers temporaires, ...

Afin d’optimiser davantage son référentiel, on peut utiliser le **garbage collection Git** …  
**>>**  **git gc --prune=now --aggressive**

Cette commande est **utile lorsque l’on utilise fréquemment les commandes pull ou push**.  
Cette commande est un **utilitaire interne qui nettoie les objets Git inaccessibles ou orphelins** dans le référentiel.

## Astuce 03 Sauvegarde des fichiers non suivis

La plupart du temps, il est **prudent de supprimer tous les fichiers non suivis**. Mais souvent, il existe une situation dans laquelle on souhaite supprimer, mais aussi **créer une sauvegarde de ses fichiers non suivis** au cas où on en aurait besoin plus tard.

Git, avec quelques commandes Bash, facilite la **création d'une archive zip pour les fichiers non suivis** …  
**>> git ls-files --others --exclude-standard -z | xargs -0 tar rvf ~/backup-untracked.zip**

Cette commande crée une archive (et exclut les fichiers répertoriés dans .gitignore) .

## Astuce 04 Dossier .git

Chaque référentiel possède un dossier .git.   
Il s’agit d’un dossier caché spécial …  
**>> ls -a  
. .. .git**

Git fonctionne principalement avec deux éléments …

* l'**arbre de travail** (l'état des fichiers du dossier actuel) ;
* le chemin du référentiel Git   
  (en particulier, l'emplacement du dossier .git, qui contient les informations de version).

Ce **dossier stocke toutes les références et autres détails importants** tels que …

* les configurations ;
* les données du référentiel ;
* l'état de HEAD ;
* les journaux
* et bien plus encore.

**Si on supprime ce dossier, l'état actuel du code source n'est pas supprimé**, mais ses informations distantes, telles que l'historique du projet, le sont.   
La **suppression de ce dossier signifie que le projet** (au moins, la copie locale) **n'est plus sous contrôle de Git**.   
Cela signifie que l’on ne peut pas suivre les modifications; Il **n’est plus possible d’effectuer un git pull ou un git push**.

En général, il n'y a pas grand-chose à faire au dossier .git.   
Il est géré par Git et est considéré comme hors-contrôle.   
Cependant, il y a des informations intéressantes dans ce répertoire, y compris l'état actuel de HEAD …  
**>> cat .git/HEAD  
ref: refs/heads/main**

Il contient également, potentiellement, une description du référentiel …  
**>> cat .git/description  
Unnamed repository; edit this file 'description' to name the repository.  
-- Ceci est un référentiel sans nom; éditez ce fichier 'description' pour nommer le référentiel.**

Le dossier Git hooks est également ici, avec des exemples de fichiers hooks.

## Astuce 05 Affichage de fichier d'une autre branche

Parfois, on aimerait **afficher le contenu du fichier à partir d'une autre branche**.   
Cela est possible avec une simple commande git, et ce, sans changer de branche.

Pour afficher un fichier appelé README.md qui se trouve dans la branche main …  
**>>**  **git show main:README.md**

Une fois cette commande exécutée, il est possible d’afficher le contenu du fichier.

## Astuce 06 Recherche dans Git

Il est possible de rechercher dans Git avec une simple commande.   
Il est même possible de rechercher dans Git même si on ne sait pas quel commit - ou même quelle branche – on a effectué ses modifications …  
**>> git rev-list --all | xargs git grep -F ‘’**

Par exemple, si on souhaite rechercher la chaîne "font-size: 52 px;" dans un référentiel …  
**>>** **git rev-list –all | xargs git grep -F 'font-size: 52 px;'  
F3022… 9e12: HtmlTemplate / style.css: taille de la police: 52 px;  
E9211… 8244 : RR.Web / Contenu / style / style.css: taille de la police: 52 px;**

## Astuce 07 Décompte des *commits*

Il existe de nombreuses raisons pour lesquelles on pourrait avoir besoin de compter ses *commits*.   
De nombreux développeurs comptent le nombre de *commits* afin de juger du moment où incrémenter le numéro de *build*, par exemple, ou simplement pour avoir une idée de la progression du projet.

Pour compter les *commits* …  
**>>** **git rev-list --count**

Dans la commande suivante, le nom de la branche doit être un nom de branche valide dans le référentiel actuel …  
**>>** **git rev-list –count master**  
**32**  
**>>** **git** **rev-list –count dev**  
**34**

# Feuillet de reference

# 

# Autres trouvailles

## Astuces méconnues

<https://www.smashingmagazine.com/2021/02/getting-the-most-out-of-git/>

**Tirer le meilleur parti de Git**

Dans cet article, Tobias explore certaines des fonctionnalités les moins connues mais très utiles de Git.   
il présente des fonctionnalités comme …

* récupérer des *commits* supprimés ;
* nettoyer l’historique de *commits* ;
* utiliser des sous-modules afin de gérer le code tiers   
  et
* composer des commits avec précision - avec une aide- mémoire Git conviviale

**Référence** …  
<https://www.git-tower.com/blog/git-cheat-sheet/?utm_source=smashingmagazine&utm_medium=guestpost&utm_campaign=TOM>

## Exercice

**Projet Hello-World**

Le projet Hello World est une tradition séculaire en programmation informatique.

Voici un exercice simple qui permet d'apprendre quelque chose de nouveau.

Ce guide permet …

* Créer et utiliser un référentiel ;
* Démarrer et gérer une nouvelle branche ;
* Apporter des modifications à un fichier et les transférer sur GitHub en tant que commits ;
* Ouvrir et fusionner à l’aide d’une requête pull.

**Référence pour ce guide** …  
<https://guides.github.com/activities/hello-world/>

# Références

Adaptation des documents …

* <https://opensource.com/article/20/10/advanced-git-tips?utm_campaign=intrel>
* <https://opensource.com/article/20/8/dont-ignore-gitignore>