# Aide-mémoire Git 50 commandes Git à connaître



Git est un système de contrôle de version distribué qui aide les développeurs à collaborer sur des projets de toute envergure.

Linus Torvalds, le développeur du noyau Linux, a créé Git en 2005 pour aider à contrôler le développement du noyau Linux.

## Qu'est-ce qu'un système de contrôle de version distribué ?

Un système de contrôle de version distribué est un système qui vous aide à suivre les modifications que l’on a apportées aux fichiers de son projet.

Cet historique des modifications réside sur l’ordinateur local et permet de revenir facilement à une version précédente de son projet en cas de problème.

Git facilite la collaboration.   
Tous les membres de l'équipe peuvent conserver une sauvegarde complète des référentiels sur lesquels ils travaillent sur leur machine locale. Ensuite, grâce à un serveur externe comme BitBucket, GitHub ou GitLab, ils peuvent stocker en toute sécurité le référentiel en un seul endroit.

De cette façon, différents membres de l'équipe peuvent le copier localement et chacun possède une vue d'ensemble claire de toutes les modifications apportées par toute l'équipe.

Git a de nombreuses commandes différentes que l’on peut utiliser. En voici cinquante qui sont celle les plus utiles à retenir.

## Comment vérifier sa configuration Git …

La commande ci-dessous renvoie une liste d'informations sur sa configuration git, y compris le nom d'utilisateur et l'adresse de courriel …  
**>> git config -l**

## Comment configurer son nom d'utilisateur Git …

Avec la commande ci-dessous, il est possible de configurer son nom d'utilisateur …  
**>> git config --global user.name "Tux"**

## Comment configurer son adresse de courriel d'utilisateur Git …

Cette commande permet de configurer l'adresse de courriel de l'utilisateur que l’on utilisera pour les confirmations (*commits*) ...  
**>> git config --global user.email "tux@linux.info"**

## Comment mettre en cache ses informations de connexion …

On peut stocker les informations de connexion dans le cache afin de ne pas avoir à les saisir à chaque fois.  
On utilise simplement cette commande …  
**>> git config --global credential.helper cache**

## Comment initialiser un dépôt (référentiel) Git …

Tout commence à partir d'ici.   
La première étape consiste à initialiser un nouveau dépôt Git localement à la racine de son projet.   
On peut le faire avec la commande ci-dessous …  
**>> git init**

## Comment ajouter un fichier à la zone de préparation (*straging*) …

La commande ci-dessous ajoutera un fichier à la zone de préparation.   
On remplace simplement **<Nom du fichier>** par le nom du fichier que l’on souhaite ajouter à la zone de préparation …  
**>> git add <Nom du fichier>**

## Comment ajouter tous les fichiers dans la zone de préparation …

Si on souhaite ajouter tous les fichiers de son projet à la zone de préparation (*staging*), on peut utiliser un caractère générique tel que le point (.) et chaque fichier sera ajouté …  
**>> git add .**

## Comment n'ajouter que certains fichiers à la zone de préparation …

Avec l'astérisque dans la commande ci-dessous, il est possible d’ajouter tous les fichiers commençant par un **<Motif>** dans la zone de préparation …  
**>> git add <Motif>\***

## Comment vérifier l'état d'un dépôt …

Cette commande affichera l'état du référentiel actuel, y compris les fichiers mis en attente, non mis en zone de préparation (*staging*) et non suivis …  
**>> git status**

## Comment valider les modifications dans l'éditeur de Git …

Cette commande ouvrira un éditeur de texte dans le terminal où on pourra écrire un message de validation complet.  
Un message de validation est composé d'un bref résumé des modifications, d'une ligne vide et d'une description complète des modifications qui suivent …  
**>> git commit**

## Comment valider les modifications avec un message …

Il est possible d’ajouter un message de validation sans ouvrir l'éditeur.   
Cette commande permet de spécifier uniquement un bref résumé de son message de validation …  
**>> git commit -m "<message>"**

## Comment valider les modifications (et ignorer la zone de préparation) …

On peut ajouter et valider des fichiers suivis avec une seule commande en utilisant les options -a et -m …  
**>> git commit -a -m "<message>"**

## Comment voir son historique de commit …

Cette commande affiche l'historique des validations pour le référentiel actuel …  
**>> git log**

## Comment voir votre historique de confirmation (*commit*), y compris les modifications …

Cette commande affiche l'historique de la validation, y compris tous les fichiers et leurs modifications …  
**>> git log -p**

## Comment voir une confirmation (*commit*) spécifique …

Cette commande affiche un commit spécifique.  
On remplace <ID du commit> par l'ID de la validation (*commit*) que l’on trouve   
dans le journal de confirmation (*commit*) ...

**>> git show <ID du commit>**

## Comment afficher les statistiques du journal …

Cette commande amènera le journal Git à afficher des statistiques sur les changements de chaque confirmation (*commit*), y compris les lignes modifiées et les noms de fichiers …  
**>> git log --stat**

## Comment afficher les modifications apportées avant de les valider à l'aide de "diff" …

Il est possible de transmettre un fichier en tant que paramètre pour visualiser uniquement les modifications sur un fichier spécifique.  
La commande git diff affiche uniquement les modifications non organisées par défaut.  
On peut appeler diff avec le commutateur --staged pour afficher les changements par étapes …  
**>> git diff  
>> git diff all\_checks.py  
>> git diff –staged**

## Comment afficher les changements en utilisant git add -p …

Cette commande ouvre une invite et vous demande si on souhaite ou non effectuer des modifications.  
Elle inclut également d'autres options …  
**>> git add -p**

## Comment supprimer les fichiers suivis de l'arborescence de travail actuelle …

Cette commande attend un message de validation afin d’expliquer pourquoi le fichier a été supprimé …  
**>> git rm <Nom de fichier>**

## Comment renommer des fichiers …

Cette commande met le fichier dans la zone de préparation (*staging*) les modifications, puis attend un message de validation …  
**>> git mv <Nom original> <Nouveau nom>**

## Comment ignorer des fichiers …

Il faut créer un **fichier .gitignore** et le valide.

## Comment annuler les modifications non mises en scène …

**>> git checkout <Nom du fichier>**

## Comment annuler les modifications par étapes …

On peut utiliser le commutateur -p afin de spécifier les modifications que l’on souhaite réinitialiser …  
**>> git reset HEAD <nom du fichier>  
>> git reset HEAD -p**

## Comment modifier la confirmation (commit) la plus récente …

La commande **git commit --amend** permet de modifier et d'ajouter des changements à la validation la plus récente …  
**>> git commit --amend**

**Remarque** …  
Réparer une validation (*commit*) locale avec le commutateur --amend est une fonction intéressante et on peut le pousser vers un référentiel partagé après l'avoir corrigé.   
Toutefois, on doit éviter de modifier des confirmations (*commits*) qui ont déjà été rendues publiques.

## Comment annuler la dernière validation (commit) avec Git …

La commande **git revert** va créer une nouvelle confirmation (*commit*) qui est l'opposé de toute confirmation (*commit*) donnée.  
Il est possible d’annuler la dernière confirmation (*commit*) en utilisant l'alias HEAD comme ceci …  
**>> git revert HEAD**

## Comment restaurer une ancienne confirmation (commit) …

On peut annuler une ancienne validation (*commit*) en utilisant son identifiant de confirmation.   
Cela ouvre l'éditeur afin que l’on puisse ajouter un message de validation …  
**>> git revert <ID de confirmation>  
>> git revert <*commit-ID*>**

## Comment créer une nouvelle branche …

Par défaut, on a une branche, la branche principale (*main branch*).   
Avec cette commande, on peut créer une nouvelle branche.   
Git ne le basculera pas automatiquement – on devra le faire manuellement avec la commande suivante …  
**>> git branch <Nom de la branche>**

## Comment basculer vers une branche nouvellement créée …

Lorsque l’on souhaite utiliser une branche différente ou nouvellement créée, on peut utiliser cette commande …  
**>> git checkout <Nom de la branche>**

## Comment lister les branches …

Il est possible d’afficher toutes les branches créées à l'aide de la commande **git branch**.   
La commande affichera une liste de toutes les branches et marquera la branche actuelle à l’aide d’un astérisque (\*) et la surlignera en vert …

**>> git branch**

## Comment créer une branche dans Git et y basculer immédiatement …

En une seule commande, on peut créer et basculer immédiatement vers une nouvelle branche …  
**>> git checkout -b <Nom de la branche>**

## Comment supprimer une branche …

Lorsque l’on a terminé de travailler avec une branche et qu’elle a été fusionnée, il est possible de la supprimer à l'aide de la commande suivante …  
**>> git branch -d <Nom de la branche>**

## Comment fusionner deux branches …

Pour fusionner l'historique de la branche dans laquelle on ses trouvez actuellement avec le <Nom de la branche>, on doit utiliser la commande ci-dessous …  
**>> git merge <Nom de la branche>**

## Comment afficher le journal de validation sous forme de graphique …

On peut utiliser le commutateur –graph pour afficher le journal de validation sous forme de graphique.   
De plus, le commutateur –oneline limitera les messages de validation à une seule ligne …  
**>> git log --graph --oneline**

## Comment afficher le journal de validation sous forme de graphique de toutes les branches …

On entre la commande précédente en ajoutant le commutateur --all ...  
**>> git log --graph --online --all**

## Comment annuler une fusion conflictuelle …

Si on souhaite supprimer une fusion et recommencer, on peut exécuter la commande suivante …  
**>> git merge –abort**

## Comment ajouter un référentiel distant …

Cette commande ajoute un référentiel distant à son référentiel local.  
On remplace simplement **<https://Référentiel>** par l'URL de son référentiel distant ...  
**>> git add remote <https://Référentiel>**

## Comment voir les URL distantes …

Il est possible de visualiser tous les référentiels distants de son référentiel local avec cette commande …  
**>> git remote -v**

## Comment obtenir plus d'informations sur un dépôt distant …

On remplace simplement **origin** par le nom de la télécommande obtenue en exécutant la commande **git remote -v** …  
**>> git remote show origin**

## Comment envoyer des modifications à un dépôt distant …

Lorsque tout le travail est prêt à être enregistré vers un référentiel distant, on peut appliquer toutes les modifications à l'aide de la commande ci-dessous …  
**>> git push**

## Comment extraire les modifications d'un référentiel distant …

Si d'autres membres de l'équipe travaillent sur le même référentiel, il est possible de récupérer les dernières modifications apportées au référentiel distant avec la commande suivante …  
**>> git pull**

## Comment vérifier les branches distantes que Git suit …

Cette commande affiche le nom de toutes les branches distantes que Git suit pour le référentiel actuel …  
**>> git branch -r**

## Comment récupérer les modifications du dépôt distant …

Cette commande téléchargera les modifications à partir d'un dépôt distant mais n'effectuera pas de fusion sur sa branche locale (comme git pull le fait) …  
**>> git fetch**

## Comment vérifier le journal des validations actuel d'un dépôt distant …

Confirmations (*commit*) après confirmations, Git crée un journal.   
On peut retrouver le journal du référentiel distant en utilisant cette commande …  
**>> git log origin/main**

## Comment fusionner un dépôt distant avec son dépôt local …

Si le référentiel distant contient des modifications que l’on souhaite fusionner avec le fichier local, cette commande le fera pour soi …  
**>> git merge origin/main**

## Comment obtenir le contenu des branches distantes sans fusionner automatiquement …

Cela permet de mettre à jour la télécommande sans fusionner aucun contenu dans les branches locales.   
on peut appeler la commande git merge ou git checkout pour effectuer la fusion ...

**>> git remote update**

## Comment pousser une nouvelle branche vers un dépôt distant …

Si on souhaite pousser une branche vers un référentiel distant, on doit utiliser la commande suivante.   
Il est important de ne pas oublier d'ajouter le commutateur -u pour créer la branche en amont …  
**>> git push -u origin <Nom de la branche>**

## Comment supprimer une branche distante …

Si on n’a plus besoin d'une branche distante, on peut la supprimer à l'aide de la commande …  
**>> git push --delete origin <Nom de la branche>**

## Comment utiliser Git rebase

On peut transférer le travail terminé d'une branche à une autre à l'aide de la commande git rebase …  
**>> git rebase <Nom de la branche>**

La commande **git rebase** peut devenir vraiment compliqué si on ne le fait pas correctement.   
Avant d'utiliser cette commande, il est suggéré de relire la documentation officielle.

## Comment exécuter rebase de manière interactive …

On peut exécuter la commande **git rebase** de manière interactive en utilisant le commutateur -i.  
Cela va ouvrir l'éditeur et présentera un ensemble de commandes que l’on peut utiliser …  
**>> git rebase -i main  
# p, pick = use commit  
# r, reword = use commit, but edit the commit message  
# e, edit = use commit, but stop for amending  
# s, squash = use commit, but meld into previous commit  
# f, fixup = like "squash", but discard this commit's log message  
# x, exec = run command (the rest of the line) using shell  
# d, drop = remove commit**

## Comment forcer une requête push …

Cette commande forcera une demande push.   
C'est généralement bien pour les branches de demander l'extraction car personne d'autre n'aurait dû les cloner.  
Toutefois, ce n'est pas quelque chose que l’on veut faire avec les dépôts publics …  
**>> git push -f**

## Conclusion

Ces commandes peuvent considérablement améliorer votre productivité dans Git. Vous n'avez pas à vous souvenir de tous - c'est pourquoi j'ai écrit cette feuille de triche. Ajoutez cette page à vos favoris pour référence future ou imprimez-la si vous le souhaitez.

Document adapté de l’article …  
<https://www.freecodecamp.org/news/git-cheat-sheet/>