

## Gérer les dépôts

- `git init [--bare] <répertoire>`  
Créer un nouveau dépôt
- `git clone <url> [répertoire]`  
Récupérer un dépôt distant

## Gérer les commits

- `git add <fichier....>`  
Ajouter des modifications à la zone de cache
- `git commit [-m <message>]`  
Committer des modifications
- `git rm <fichier...>`  
`git commit`  
Supprimer un fichier
- `git mv <src> <dst>|<rep_dst>`  
Renommer / déplacer un fichier
- `git reset <fichier>`  
Désindexer un fichier

## Défaire des commits

- `git reset --mixed <SHA1>`  
Déplacer le HEAD et tout récupérer dans la zone de travail
- `git reset --soft <SHA1>`  
Déplacer le HEAD et tout récupérer dans la zone de cache
  - `git reset --hard <SHA1>`  
Déplacer le HEAD sans rien récupérer

## Gérer les branches locales

- `git branch <branche> [SHA1]`  
Créer une branche
- `git checkout -b <branche>`  
Créer une branche et se déplacer dedans
- `git status`  
`git checkout <branche>`  
Changer de branche
- `git branch [-v]`  
Lister les branches locales
- `git merge <branche>`  
Fusionner des branches
- `git status`  
éditer le(s) fichier(s)  
`git add <fichier>`  
`git commit`  
Gérer des conflits de merge
- `git mergetool -t <tool> <fichier>`  
Editer un fichier lors d'un merge avec un outil graphique
- `git merge --abort`  
Annuler un merge en cours
- `git log --oneline <branche>..<branche>`  
Comparer l'historique de 2 branches
- `git branch -m <old> <new>`  
Renommer une branche locale
- `git branch -d|-D <branche>`  
Supprimer une branche locale

## Receueillir de l'information

- `git status`  
Etat du dépôt : zones de cache et de travail, nouveaux fichiers, conflits branche locale/distante
- `git diff [--cached]] [SHA1 SHA1]`  
Afficher le contenu des modifications entre 2 commits, dans la zone de cache ou de travail
- `git blame [SHA1] <fichier>`  
Lister les auteurs ayant apporté les dernières modifications pour chacune des lignes du fichier
- `git log [--oneline] [SHA1]`  
Historique des commits de la branche courante
- `git show <SHA1>:<fichier>`  
Consulter une version donnée d'un fichier
- `git checkout [SHA1]-- <fichier>`  
Récupérer un fichier dans la zone de travail pour une version donnée
- `git remote show origin`  
Obtenir de l'information sur le dépôt distant
- `git log --oneline -- <fichier>`  
Lister les commits concernant un fichier donné
- `git log --oneline --name-status`  
Afficher les fichiers modifiés commit par commit
- `git log --oneline -diff-filter=<F>`  
Filtrer l'historique des commits en fonction de l'action réalisée avec <F> :  
A ajouté, D supprimé, M modifié, R renommé, C copié
- `git log --oneline --grep="<chaîne>"`  
Filtrer l'historique en fonction d'une chaîne à rechercher dans le message de commit

## Gérer les branches distantes

- `git pull [serveur branche]`  
Mettre à jour le dépôt local et les branches locales
- `git fetch`  
Mettre à jour les références distantes uniquement
- `git push [serveur branche]`  
Pousser des modifications sur le serveur distant
- `git checkout <branche>`  
`git checkout -b <locale> origin/<dste>`  
Récupérer une branche distante

## Mettre de côté son travail

- `git stash [save <message>]`  
Mettre de côté son travail
- `git stash list`  
Lister les zones de stash
- `git stash show -p <stash@{x}>`  
Voir le contenu d'une zone de stash
- `git stash apply <stash @{x}>`  
Récupérer le contenu d'une zone de stash
- `git stash drop <stash@{x}>`  
Supprimer une zone de stash
- `git stash pop <stash@{x}>`  
Récupérer et supprimer une zone de stash
- `git stash branch <branche> <stash@{x}>`  
Récupérer le contenu d'une zone de stash dans une branche dédiée

## Modifier des commits

- `git commit --amend`  
Modifier le dernier commit : message de commit, contenu du commit
- `git revert [SHA1...]`  
Jouer un commit à l'envers (utilisable sur des commits déjà envoyés sur le serveur)

## Réorganiser l'historique, préparer un push

- `git rebase <branche>`  
Réorganiser l'historique pour le mettre à jour (pas de commit de merge)
- `git pull --rebase`  
Réorganiser l'historique pour le mettre à jour par rapport au serveur (pas de commit de merge)
- `git rebase -i <SHA1>`  
Remanier l'historique avant de le partager
  - `reword` : réécrire le message de commit
  - `drop` : supprimer un commit
  - `squash` : fusionner des commits et réécrire le message
  - `fixup` : fusionner des commits sans réécrire le message
- **édition des fichiers**  
`git add <fichiers...>`  
`git rebase --continue`  
Gérer un conflit lors d'un rebase
- `git rebase --abort`  
Annuler un rebase en cours

## Autres commandes

- `git cherry-pick <SHA1>...`  
Rejouer un ou plusieurs commits sur le HEAD courant
- `git add -p <fichier>...`  
Ajouter au cache les modifications bloc par bloc
- `git clean`  
Supprimer tous les nouveaux fichiers
- `git cat-file -p <SHA1>`  
Afficher le contenu d'un objet
- `git grep <string> -- <path>`  
Rechercher une chaîne de caractères dans les fichiers contenus dans la zone de travail
- `git reflog`  
Afficher l'historique des mouvements du commit de HEAD

## Gérer les tags

- `git tag`  
Lister les tags
- `git tag [-a] <tag> <SHA1>`  
Créer un tag
- `git tag -d <tag>`  
Supprimer un tag local
- `git push --delete origin <tag>`  
Supprimer un tag distant
- `git push origin <tag>`  
Partager des tags

## Gérer les sous-modules

- `git submodule add <url> <répertoire>`

Ajouter un submodule

- `git clone --recursive <url>`

Récupérer un dépôt exploitant des sous-modules

- `git fetch`  
`git log --oneline origin/<branche>`  
`git checkout <SHA1 | tag>`

Récupérer une mise à jour dans un submodule

- `git commit`  
`git push`

Dans le répertoire du super-projet

- `git submodule deinit <sous-modules>`  
`git submodule update --init`

Désactiver / Ré-activer un sous-module

- `git submodule status`

Status des sous-modules

- `git submodule summary`

Liste des écarts

- `git checkout --recurse-submodules`

Changer de branche et mettre à jour les sous-modules

- `git fetch --recurse-submodules`

Exécuter le fetch sur le dépôt principal et les sous-modules

- `git submodule foreach`

Appliquer une commande shell dans tous les sous-modules

## LFS (Large File Storage)

- `git lfs install`

Ajouter les hooks nécessaires à LFS

- `git lfs track <pattern>`

Ajouter un type de fichier à gérer avec LFS

- `git lfs untrack <pattern>`

Supprimer un type de fichier à gérer avec LFS

- `git lfs fetch --recent`

Télécharger des objets supplémentaires

### Configuration pour prise en charge des sous-modules

- `status.submoduleSummary true`

Prendre en compte les sous-modules pour git status

- `diff.submodule log`

Prendre en compte les sous-modules pour git diff

- `submodule.recurse true`

Travailler en récursif pour les commandes fetch, checkout

## Rejouer la résolution d'un conflit

- `git config --global rerere.enabled true`

Activer rerere

## Gestion de notes

- `git notes add <SHA1>`

Associer une note à un commit

- `git notes append <SHA1>`

Compléter une note existante

- `git notes --ref <catégorie> add <SHA1>`

Ajouter une note et l'associer à une catégorie

- `git log --oneline --notes=<catégorie>`

Afficher l'historique et les notes de la catégorie citée

- `git push origin refs/notes/*`

Partager les notes de toutes les catégories

## Filter-repo

- `git filter-repo --invert-paths \`  
`--path <fichier, répertoire> ...`

Supprimer des fichiers, des répertoires

- `git filter-repo \`  
`--subdirectory-filter <répertoire>`

Splitter un dépôt, définir une nouvelle racine

- `git filter-repo \`  
`--strip-blobs-bigger-than <size>`

Supprimer des fichiers d'une taille supérieure à