Git Bash

Commandes

git clone

Clone le serveur dans le répertoire courant.

> Pour obtenir le lien aller sur l'interface web et cliquer en haut à droite sur le bouton clone puis copier le lien « clone with ssh ».

Clone branche principale

```
git clone <lien>
```

Clone branche particulière

```
git clone -b <nom de la branche> <lien>
```

Clone à partir d'un tag

```
git clone -b <tag> <lien>
```

Cloner uniquement un répertoire à partir d'un tag

```
git clone -b <tag> --filter=blob:none --sparse <lien>
cd <base>
git sparse-checkout add "répertoire"
```

git checkout

Permet de se positionner sur une branche ou sur un tag.

```
git checkout <branche>
git checkout <commit hash> -- <fichier>     # Pour récupérer une version spécifique d'un fichier à partir d'un commit particulier
git checkout HEAD -- <fichier>              # Pour revenir à la dernière
```

```
version d'un fichier
git checkout HEAD~1 -- <fichier>            # Pour revenir à la version
précédente d'un fichier
git checkout <branche> -- <nom_du_fichier>  # Pour récupérer la version d'un
fichier depuis une branche spécifique
git restore <fichier>                       # pour annuler les modifications
```

Attention de sauvegarder d'éventuelles modifications avant de lancer la commande !

git pull

Met à jour la copie locale à partir du serveur.

git add

Ajoute un (ou des) fichier(s) au projet.

```
git add <liste fichiers>
```

git restore

Efface les modifications faites sur un fichier

```
git restore <fichier>
```

git rm

Supprime un (ou des) fichier(s).

```
git rm <fichier>
```

git mv

Renome un (ou des) fichier(s). On doit rester sur le même repository.

```
git mv <oldname> <newname>
```

git reset HEAD

Supprime un fichier de la liste des fichiers à commiter

```
git reset HEAD <fichier>
```

git status

Liste l'état des fichiers.

git commit

Prise en compte de fichiers.

```
git commit -m "commentaire" <liste fichiers>
```

Possibilité de prendre en compte tous les fichiers avec l'option -a (all)

```
git commit -am "commentaire" <liste fichiers>
```

git push

Remonte les modifications sur le serveur.

git log

Lister l'historique des commits

```
git log [<fichier>]
git log -p <fichier>                                # liste les commits avec les modifications
git log --graph --oneline                           # affiche la liste des commits sur une ligne
git log --pretty=oneline [<fichier>]                # idem
git log --since="2025-07-01" --until="2025-07-31"   # commits entre 2 dates
git log <tag1>...<tag2>                             # commits entre 2 tags
git log --author="Paul Bismuth"                     # commits d'une personne donné
git log --grep="texte" -i                           # commits contenant un texte spécifique (-i pour ignore case)
git log -- chemin/vers/fichier.txt                  # commits affectant un fichier spécifique
git log --decorate --oneline                        # pour voir l'historique des commits et la pose des tags
git log --name-status                               # pour avoir les fichiers concernés (A : Added M : Modified D : Deleted R : Renamed C : Copied)
git log --date=iso --pretty=format:"%H|%an|%ad|%s"  # format personnalisé
```

Format personnalisé

```
%H  : Hash complet du commit
%h  : Hash abrégé du commit
%T  : Hash complet de l'arbre
%t  : Hash abrégé de l'arbre
%P  : Hashes des parents
%p  : Hashes abrégés des parents
%an : Nom de l'auteur
%ae : Email de l'auteur
%ad : Date de l'auteur (respecte le format --date=)
%ar : Date relative de l'auteur (ex: "il y a 2 jours")
%ai : Date de l'auteur au format ISO 8601
%cn : Nom du committer
%ce : Email du committer
%cd : Date du commit (respecte le format --date=)
%cr : Date relative du commit
%ci : Date du commit au format ISO 8601
%d  : Références (branches, tags)
%D  : Références sans les "() " autour
%s  : Sujet du commit (première ligne du message)
%f  : Sujet formaté sous forme de nom de fichier
%b  : Corps du message de commit
%B  : Message de commit complet (sujet + corps)
%N  : Notes du commit
%GG : Signature GPG
%G? : Statut de vérification de la signature GPG
%GS : Signature GPG
%GK : Clé de signature GPG
```

git diff

- Lister la liste des commits avec la commande **git log** ci-dessus

Console

```
git diff <fichier>                  # pour voir les évolutions par rapport au dernier commit
git diff <commit hash> <fichier>    # pour voir les évolutions par rapport à un commit donné
```

Graphiquement

```
git difftool HEAD~1 <fichier>                   # différence par rapport à la version précédente
git difftool <commit1> <commit2> <fichier>      # différence entre 2 commits
```

```
du fichier
git difftool master <fichier>                    # Pour comparer avec la
branche master
git difftool                                     # Pour voir tous les
changements non commités
git difftool <tag1>:<fichier> <tag2>:<fichier>  # pour comparer un fichier
entre 2 tags
```

gitk

Permet de lancer une interface graphique

```
gitk <fichier>
```

Merge

Merge d'un fichier

Exemple merge du fichier .gitlab-ci.yaml depuis la branche migration_engine vers la branche develop

```
git branch -a                                                    # pour
vérifier le nom complet de la branche
git pull origin develop                                          # pour
mettre à jour le repo avec la branche develop
git checkout remotes/origin/migration_engine -- .gitlab-ci.yaml  # on
récupère la version du fichier depuis la branche migration_engine
git add .gitlab-ci.yml                                           # optionnel
si le fichier n'existait pas dans la branche develop
git commit -m "Merge du fichier .gitlab-ci.yml"
git push origin develop
```

Gestion droits unix

Interrogation

```
git ls-files --stage
```

- 100644 : fichier classique
- 100755 : fichier avec les droits d'exécution (+x)
- 040000 : répertoire
- 120000 : lien symbolique

Modification

Rendre un fichier exécutable

```
chmod +x <fichier>
git update-index --chmod=+x <fichier>
```

Rendre un fichier non exécutable

```
git update-index --chmod=-x <fichier>
```

Ne fonctionne que depuis un vrai système linux, pas à partir de Git bash

Création d'un fichier exécutable

A partir de Git 2.9 il est possible de créer un fichier directement comme exécutable :

```
git add --chmod=+x <fichier>
```

Dépannage

En cas de problème essayer de retirer puis de remettre les droits :

```
git update-index --chmod=-x <fichier>
git update-index --chmod=+x <fichier>
```

Gestion tag

Interrogation

Lister tous les tags

```
git tag -n
git tag -n --sort=refname           # dans l'ordre des noms de tag (par défaut)
git tag -n --sort=committerdate     # dans l'ordre chronologique
```

Lister les tags suivant un patern

```
git tag -l "v1.4.*"
```

Interroger le contenu d'un tag

```
git show v1.4.5
git show --pretty= --name-only v1.4.5
```

Positionnement

Choisir quel commit tagger

```
git log
git log --author "MEYLHOC Olivier"    # pour ne voir que les commit d'une personne
```

Poser un tag localement et le remonter sur le serveur

```
git tag -a <tag> <commit_ID>
git push origin <tag>
```

Déplacer un tag localement et le remonter sur le serveur

```
git tag -f <tag> <commit_ID>
git push origin <tag> --force
```

Effacer

- Effacer un tag localement

```
git tag -d <tag>
```

- Effacer un tag sur le serveur

```
git push origin --delete <tag>
```

Checkout

Il est possible de voir les versions de fichier correspondant à un tag en faisant un checkout à partir de celui-ci :

```
git checkout <tag>
```

Eviter de faire des modifications à partir de cet état « detached HEAD ».

Pour faire des modifications il est préférable de créer une nouvelle branche à partir du tag :

```
git checkout -b <nouvelle branche> <tag>
```
