# AIDE - gérer plusieurs versions de fichiers (Git)

Lien : <https://www.youtube.com/watch?v=CEb_JM_hsFw>

https://github.com/jasonchampagne/FormationVideo

git remote add origin https://github.com/jasonchampagne/FormationVideo.git

git add --all\*

git commit -a -m "Update notes"

git push https://github.com/js202005082300/FormationVideo.git

Git = gestionnaire de version. Intérêt de pouvoir gérer un système de version sur des fichiers. Pratique en développement de logiciels en gérant plusieurs versions de ces fichiers et de revenir en arrière en cas d'erreurs ou de revenir à une version antérieure (TimeMachine). Git permet de faire du travail collaboratif sur du travail synchroniser avec tout le monde. Utile à toute personne qui a besoin d'un système de gestion de version (développeur, écrivain, etc.).

Git est un logiciel Open-Source.

Logiciel qui est utilisé en ligne de commandes (ou avec un client et son interface, une GUI tel que TortoiseGit avec une licence GPL et Open Source, totalement gratuit).

Github est un site qui permet d'avoir un ensemble de dépôts distants à récupérer. Il permet de récupérer une archive zip ou une version toujours à jour d'un dépôt avec le chemin vers ce dépôt afin de le cloner en local dans notre répertoire.

## Installation standard (tout par défaut)

<https://git-scm.com/download/win>

<https://git-scm.com/download/linux>

## Configurer Git

Ouvrir git-bash.exe

**git config --global user.name "js"**

**git config --global user.email "js201910271836@outlook.com"**

**git config --global user.mail "js201910271836@outlook.com"**

=> **Afficher les fichiers, dossiers et lecteurs cachés :** Options de l'Explorateur de fichiers -> Affichage (onglet) -> Fichiers et dossiers cachés -> Afficher les fichiers, dossiers et lecteurs cachés

## Gérer un dépôt

Dépôt = répertoire de travail qui contient un projet.

**cd** C:/Users/js201/OneDrive/Formations/FormationVideo/GitRepos/FormationVideo

**ls <chemin>** -> lister le contenu du répertoire <chemin>.

**mkdir <name>**  -> make directory, permet de créer un répertoire <name>.

**cd <chemin>** -> accéder à <chemin>.

## Initialiser un nouveau dépôt

**git init** -> initialiser un dépôt Git à cet endroit, quand on est dans le bon répertoire.

## Récupérer un dépôt distant

git clone https://github.com/jasonchampagne/FormationVideo.git

git clone <chemin> <name> -> cloner un dépôt distant (url, liens -> https://<depot> ou répertoire pour git-> git://<depot>). Choisir un nom pour le nommer différemment du dépôt en ligne.

## Activer l'historique sur de nouveaux fichiers

-> permet de revenir en arrière en cas de problèmes car Git sauvegarde l'historique complet des événements sur le dépôt. D'où l'intérêt de faire des commits réguliers pour revenir en arrière facilement via l'historique de Git.

-> les 3 états d'un fichier :

* État de modification : en cours de mordication quand on est en train de travailler dessus.
* État stage (git add) : fichier modifié et prêt à être sauvegardé. La stage area est la zone de la préparation à la sauvegarde de tout ce qu'il y a dans le dépôt.
* État commit (git commit) : fichier sauvegardé à l'historique de Git.

## Opération sur l'ensemble des fichiers

**git add <file>** -> ajouter un fichier en vue d'être sauvegardé (en mode stage).

git add --all / git add \* -> préparer la sauvegarde de tout.

git mv <ancien\_nom> <nouveau\_nom> -> déplacer et/ou renommer un fichier.

git rm <file> -> supprimer le fichier.

## Sauvegarde des fichiers

git commit [-m "Description"] -> sauvegarder (id -> 28401f2).

git commit -a -m "Description" -> ajouter (-a) et sauvegarder.

## Opérations de vérifications

git log -> afficher l'historique de ce qui a été sauvegardé (commit stage).

git status -> afficher l'historique de ce qui n'as pas encore été sauvegardé (stage state).

## Opérations d'analyses

git diff < id\_commit1> < id\_commit2> -> comparer la différence entre plusieurs commits.

## Synchroniser avec le dépôt distant

git pull -> se synchroniser avec le dépôt distant.

git push origin master -> pousser vers la branche principale du dépôt les modifications faites (possibilité de revenir en arrière).

## Revenir en arrière, à l'état de commit

git reset -hard HEAD -> annuler tous les changement depuis le dernier commit (roll back). HEAD est un pointeur vers la branche principale.

git reset -hard HEAD^ -> supprimer ce qui a été fait au dernier commit.

git revert <id\_commit> -> restaurer le dépôt dans l'état du commit identifié (TimeMachine).

Le dépôt = équivalent à un arbre.

Master = branche principale du dépôt.

HEAD-> MASTER = nom de la branche courante, principale. HEAD est un pointeur qui mène vers cette branche, la branche courante, celle qu'on est en train d'utiliser. La branche MASTER est suffisante pour les projets.

Créer une branche = modifier le projet sans infecter la branche principale (travailler sur une copie sans infecter l'original en travaillant sur une version parallèle) pour in fine fusionner notre branche avec la branche principale. Exemples :

-> correction d'un bug sur un projet de 2-3 personnes et quand le bug sera corrigé, il suffira de fusionner le projet avec la branche principale avec git pull.

-> tester ou développer un ajout de fonctionnalité.

-> développer un projet en parallèle (branches qui ne fusionne jamais). Par exemple, une branche pour chaque console de jeux (PC, XBOX, PS4).

git branch -> lister les branches.

git branch <name> -> créer une nouvelle branche.

git checkout <name> -> changer de branche courante. Le pointeur HEAD va pointer sur la branche spécifier.

## Fusionner avec la branche principale

git checkout <name> -> switch sur une branche. On ne peut pas changer de branche s'il y a des commits à faire.

git merge <name> -> fusionner avec une autre branche.

## Supprimer une branche

git branch -d <name> -> supprimer une branche.

## Récupérer les changements d'un dépôt distant

git remote add <name> https://<depot> -> récupérer l'accès au dépôt.

git fetch <name> -> synchroniser avec un dépôt distant.

git pull <name> -> récupérer le dernier état du dépôt en question.