# Git et GitHub

**Git** est un système de gestion de versions décentralisé open-source, conçu pour gérer des projets de tous types et de toutes tailles. Il permet aux développeurs de travailler sur un même projet simultanément sans écraser le travail des autres, grâce à la fonctionnalité de branches qui permet de travailler sur des versions différentes d'un même projet et en même temps.  
  
Le problème, avec Git, c’est que tous nos données sont enregistrées sur notre ordinateur. Donc, le jour que notre ordinateur tombe en panne, nous pourrions perdre tous nos travaux. Ça serait donc mieux si tous les changements étaient enregistrer quelque part en ligne et que nous puisons y accéder sur n’importe quelle machine et quel que soit notre position grâce à une simple connexion internet ; d’où le développent de GitHub pour répondre à ce genre de problème.

**GitHub** est donc un service complémentaire à Git. C’est une plateforme web qui utilise Git pour le contrôle de version et le partage de code source. Elle permet aux développeurs de collaborer sur des projets, de partager des codes sources, des idées et des projets.   
GitHub fournit également des outils pour suivre les problèmes et les demandes de mise à jour. Elle permet également de partager son code source avec le public ou de le garder privé.

**Idée reçue**

Git et GitHub n'est pas conçus que pour des projets informatique.  
Il est bien possible d'utiliser Git et GitHub pour d'autres types de projets en dehors du développement de logiciels. En effet, Git et GiHub peut être utilisé pour gérer les versions de tout type de fichiers, y compris des documents texte, des projets de recherche, des livres, des présentations, des portfolios, des feuilles de calcul, des images, des vidéos, etc. En utilisant Git, vous pouvez enregistrer l'historique des modifications de ces fichiers, effectuer des sauvegardes, revenir en arrière, suivre les modifications apportées.

En résumé, Git est un système de gestion de version décentralisé, tandis que GitHub est une plateforme web basée sur Git, qui fournit des outils de collaboration pour des projets de tout type, en particulier de développement de logiciels

Dans un premier temps, nous allons voir les commandes de base de Git et GitHub, comment créer un dépôt Git puis le mettre en ligne sur GitHub. Ensuite, nous allons voir des commandes plus avancées ; l’utilisation du système des branches ce qui permet à des développeurs différent de travailler sur des versions différentes d'un même projet simultanément sans interférer avec le travail des autres.

## Installation & Connexion

Installer Git et créer un compte sur GitHub.

## Les Commandes Git (à exécuter sur Git Bash)

Dans cette section, nous allons lister les commandes les plus utilsées dans Git Bash.

**Configuration et initialisation**  
git --version : Pour voir la version de Git que vous uttilisez.  
  
git --help/ : Liste des commandes Git.  
  
git init : Pour initialiser un dépot Git.  
  
git config --global user.name "votre nom"  
  
git config --global user.email "votre email" : (facultatif).  
  
Il est possible de visualiser votre configuration ainsi que les répertoires de travail dans le fichier .gitconfig qui se trouve généralement dans le dossier Utilisateurs « Users ».

**Dans le répertoire de travail**

echo "# description" : Pour créer un fichier de description pour votre projet.  
  
git add file\_name : Pour ajouter un fichier nommé file\_name dans le stage area (l'index.)  
  
git add doc/ : Ajout d'un dossier nommé doc.  
  
git add \* : ajouter tout (fichiers et/ou dossiers).  
  
git add -A : Pour ajouter que les nouveaux fichiers.  
  
.gitignore : S’il y a des fichiers ou des dossiers que vous ne voulez pas suivre, il suffit de les mettre dans un fichier .gitignore puis sauvegarder ce fichier.

**Index ou stage area**

git mv nom1\_fichier nom2\_fichier : Pour renonnmer un fichier.  
  
git rm -f file\_name : Supprimer un fichier.  
  
git rm --cached file\_name : Masquer un fichier dans l'index.  
  
pour qu'il ne soit pas sauvegardé.

**Sauvegarde**

git commit : Pour sauvegarder simplement.  
  
git commit -m "description..." : Sauvegarde et description.  
  
git log : Pour voir l'historique des sauvergardes.  
  
git commit -a -m "description..." : Raccourci pour sauvegarder les fichiers sans passer par l’Index.

**Dépot distant**

git remote : Pour voir les dépots distants auquels le dépôt local est lié.  
  
git remote add origin « url\_du\_dépôt\_distant » : Lié le dépot local à un dépot distant qu'on va appeler ici origin.  
  
git push -u origin main : Sauvegarder les données de la branche main vers un dépot distant nommé ici origin.

**Les branches**

git branch : Pour voir les différentes branches.  
  
git branch -M new\_name : Renommer la branche courante.  
  
git branch dev : Créer une nouvelle barnche nommée dev.  
  
git checkout dev : Se placer sur la branche nommée dev. Il faut sauvegarder les fichiers avant de pouvoir changer de branche.  
  
git merge app : fusionner une branche nommée app avec la branche courante.  
  
git diff branch\_1 branch\_2 : Apperçus de la fusion entre les barnches branch\_1 et branch\_2.  
  
git branch -d dev : Suprimer la branche nommée dev.  
  
git branch --help : Aide sur les branches.

**Autres commandes**

cd doc/ : Ce placer dans un dossier.  
  
ls : Voir les fichier et/ou dossiers contenus dans le dossier courant.  
ls -al : Pour voir en plus les fichiers qui sont cachés.  
  
touch file\_name : Pour créer un fichier nommé file\_name.  
  
vim file\_name : Pour modifier le fichier file\_name.  
 i : Pour commencer la modification.  
 alt (ou esc) puis :wq -> Pour enregistrer.