# Git et Github

Avant-propos : Ce tutoriel est tiré du cours réalisé par OpenClassroom. Il a pour but de synthétiser les informations essentielles pour utiliser **Git** et **Github** dans vos projets. Si vous souhaitez suivre le cours entièrement, voilà le lien : https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-developpement/5641728-decouvrez-la-magie-du-controle-de-versions

## I - Git

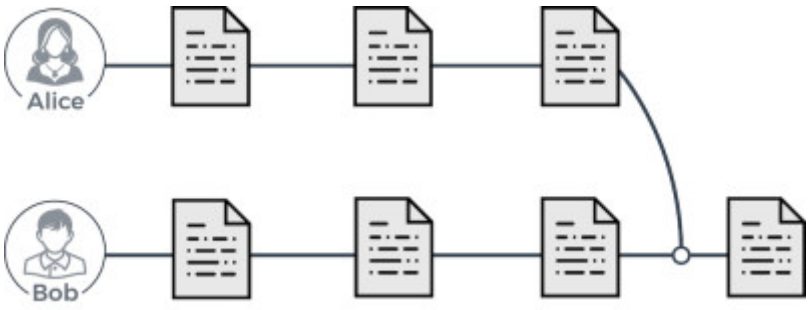
### 1. Introduction à Git

**Git** est un outil de contrôle de version. C'est un programme qui permet aux développeurs de conserver un historique des modifications et des versions de tous les fichiers d'un projet. Ces outils ont trois grandes fonctionnalités :

- revenir en arrière en cas de problème : vous avez supprimé une fonction, une classe pour les remplacer par une nouvelle fonctionnalité qui ne fonctionne pas ? Pas de problème, Git possède une sauvegarde de votre travaille avant modification.
- suivre l'évolution étape par étape d'un code source pour retenir les modifications effectuées sur chaque fichier
- travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs

Prenons un exemple de l'intérêt de l'usage de Git dans un travail de groupe.

*Alice et Bob travaillent sur un même projet et ont initialisé Git pour leur projet. Grâce à Git, chacun modifie ses fichiers, et chacun peut envoyer et recevoir les mises à jour des fichiers à n'importe quel moment, et cela sans écraser les modifications de l'autre. Des modifications même en urgence n'auront aucun impact sur le travail de l'autre !*




Maintenant que nous avons présenté l'intérêt de travailler avec Git, voyons comment l'installer et l'utiliser.

### 2. Installer et initialiser Git

### 1. Sur Linux

1. Ouvrir le terminal (Ctrl + Alt + T)
2. Taper la commande : sudo apt-get install git
3. Fini #LinuxMasterRace)

### 2. Sur Mac OS

Demande à Google et fait le tuto ici.

### 3. Sur Windows

Télécharger Git et l'installer.

### 4. Initialisation de Git

Ouvrez un terminal et tapez les commandes suivantes:

- git config –global color.diff auto
- git config –global color.status auto
- git config –global color.branch auto
- git config –global user.name “John Doe”
- git config –global user.email johndoe@example.com

## 3. Utilisation de Git

### 1. Les dépôts

Un dépôt Git est un entrepôt virtuel de votre projet. Il vous permet d'enregistrer les versions de votre code et d'y accéder au besoin. Il en existe de deux types : local et distant.

Le premier est assez intuitif. Les différentes versions sont enregistrées localement sur votre ordinateur et chaque nouvelle version est une modification de la précédente.

Le dépôt distant permet de stocker certaines versions qu'on lui aura envoyées, afin de garder un historique délocalisé. En plus de les stocker, vous pouvez aussi les rendre publics, et chacun pourra alors venir y ajouter ses évolutions. Afin que vous puissiez collaborer sur des projets, il est nécessaire de disposer de dépôts distants. Le dépôt distant est un historique de votre projet hébergé sur Internet ou sur un réseau. Nous présenterons et utiliserons GitHub dans la suite de ce tutoriel mais il en existe bien évidemment d'autres.

Pour initialiser un dépôt Git, tapez la commande suivante dans le dossier souhaité : *git init*. Le dossier est maintenant un dépôt Git !

### 2. Les branches

Le principal atout de Git est son système de branches. Les différentes branches correspondent à des copies de votre code principal à un instant T, où vous pourrez tester toutes vos idées les plus folles sans que cela impacte votre code principal.

Sous Git, la branche principale est appelée la branche main (précédemment master, ça vient de changer). C'est celle-ci, où au final, vous aurez à la fin toutes vos modifications. Le but est de ne surtout pas réaliser les modifications directement sur cette branche, mais de réaliser les modifications sur d'autres branches, et après tests, les intégrer sur la branche master.

Reprenons l'exemple de Bob et Alice plus haut. Bob et Alice ont tous les deux travaillés sur des branches différentes avant de les fusionner. Ils peuvent ainsi travailler chacun de leur côté sans contraintes et avec contrôle de version avant de regrouper leurs travaux respectifs. Le plus beau dans cette histoire ? Git s'occupe automatiquement de la fusion comme un grand sans que vous ayez besoin de faire quoi que ce soit.

Quelques commandes utiles :

- **git branch** : affiche les branches existantes. L'étoile signifie que c'est la branche sur laquelle vous vous situez et que c'est sur celle-ci qu'actuellement vous réalisez vos modifications.
- **git branch branch_name** : crée la branche branch_name. Attention, cette commande ne vous bascule pas de branche.
- **git checkout branch_name** : vous bascule sur la branche branch_name.

### 3. Les commits

Un commit est tout simplement un enregistrement de votre travail à un instant T sur la branche courante où vous êtes.

**git add file_name** : ajoute le fichier *file_name* au prochain commit. A réaliser avant chaque commit sur les fichiers qui vous intéressent.

**git commit -m "description du commit"** : réalise un commit avec comme description *description du commit*. La description est très importante pour retrouver le fil de vos commits, et revenir sur un commit en particulier. Ne la négligez pas !

## GitHub

GitHub est un outil de communication et de collaboration entre plusieurs développeurs (ou tout autre personne qui écrit du texte). C'est une interface web créée pour faciliter l'interaction avec Git en hébergeant ses dépôts distants.

**git remote add name github_url** : lie le dépôt local au dépôt GitHub à l'url *github_name*

**git clone github_url** : télécharge les fichiers du dépôt GitHub à l'url *github_name*

**git push** : actualise le dépôt GitHub en y ajoutant vos commits. Se limiter à une fois par jour

**git pull** : actualise le dépôt local. A réaliser avant chaque push