



# Prise de note Versioning-GIT

## Outils de versioning

Git est un outil qui permet à tout projet de faire vivre sa code base et de pouvoir travailler en collaboration avec d'autres développeurs. C'est un logiciel de versioning.

Git-Hub lui est un site web qui permet de publier des projets sur des serveurs distants

Git est un SCM

GitLab est une interface graphique

SCM (Source code Management) :

Intérêt de Git	Permission d'une maitrise total du code source	Fonctionnement
Traçabilité	Git	Chaque version sauvegarde l'intégralité du projet
Collaboration	AWS	Snapshot
Peu de conflits	CVS	Permet facilement de récupérer des versions antérieur du code
Code fonctionnel		
Création		

Workflow		

## Pear to pear :

Système décentralisé : chaque client a la possibilité d'assumer des fonctions serveur (on parle alors de "*noeuds*" plutôt que de machines clientes ou serveurs).

Système centralisé : des machines "serveurs" fournissent des services (applications et données) à des machines "clientes".

Ce qui permet d'obtenir une sécurité pour les fichiers qui sont stockés sur différents supports à distance.

Server Based Network



Peer to Peer Network



## GitHub

Un repository est un espace disque dans lequel nous pouvons initialiser un projet GitHub.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---


### Repository template

Start your repository with a template repository's contents.

**No template** ▾

---


**Owner \*** **Repository name \***


 maxousql ▾ /

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-octo-sniffle?](#)

**Description** (optional)

---

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**.gitignore template:** None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

Création d'un clef SSH :

```
ssh-keygen -t ed25519 -C "maxlaiyio@outlook.fr"
```

Avec une clef on identifie une machine, elle permet une connexion plus sécurisée.

La récursivité permet l'application à tous les sous dossiers.

Principe d'arbre.

Pas d'exécutable.

.gitignore est toujours à la racine du projet.

Les fichiers possédant des variables d'environnement sont à exclure du suivi git.

### **Vocabulaire :**

repository : espace disque dans lequel est stocké chaque version du projet, on a initialisé le projet git.

ReadMe : c'est une documentation permettant l'utilisation et la compréhension du code (Notice)

-m = message

Commit : cela signifie sauvegarde en local, snapshot qui nous permet de revenir à une version

antérieure de la sauvegarde.

Origin = un alias

git remote origin (url)

git remote : voir les alias

git push -u origin main : permet d'envoyer les fichiers

Untracked : Fichier non suivi(

Staged : Signifie ajouter au suivi

Unmodified : un fichier non modifié on peut l'enlever du suivi

Netlify : site web qui permet d'héberger une branch git.

### **Commandes :**

`$ ssh-keygen -t ed25519 -C "maxlaiyio@outlook.fr"` : Generate key SSH

`git add .` : Ajoute tous les documents à partir de la racine du fichier dans lequel on se trouve.

`git commit -m "txt"` : Créez un nouveau commit contenant le contenu actuel de l'index et le message de journal donné décrivant les modifications.

`git branch -M main` : Avec une option `-m` ou `-M`, `<oldbranch>` sera renommé en `<newbranch>`.

`git remote add <name> <url>` : Créez une nouvelle connexion à un référentiel distant. Après avoir ajouté une télécommande, vous pourrez l'utiliser `<name>` comme raccourci pratique pour `<url>` d'autres commandes Git.

`git remote rm <name>` : Supprimez la connexion au référentiel distant appelé `<name>` .

`git remote rename <old-name> <new-name>` : Renommer une connexion à distance de `<old-name>` à `<new-name>`

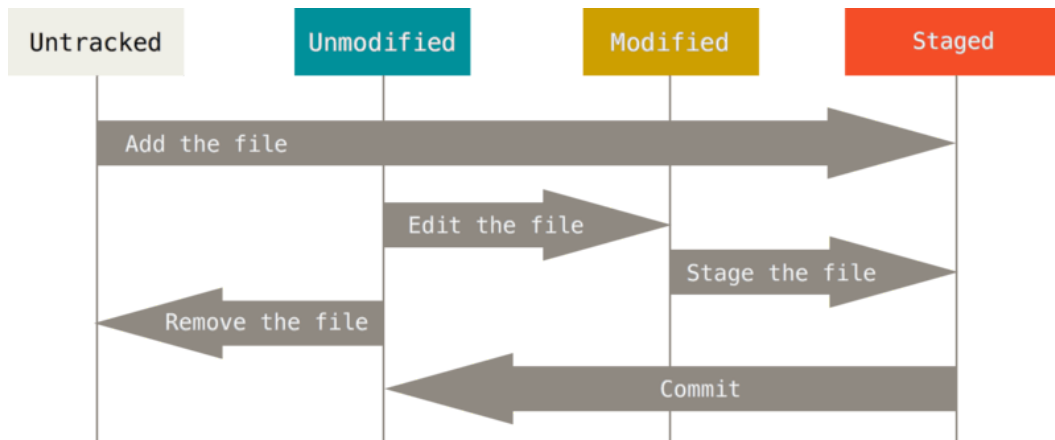
`git rm --cached <name file>` : permet de détacher un fichier du suivi git.

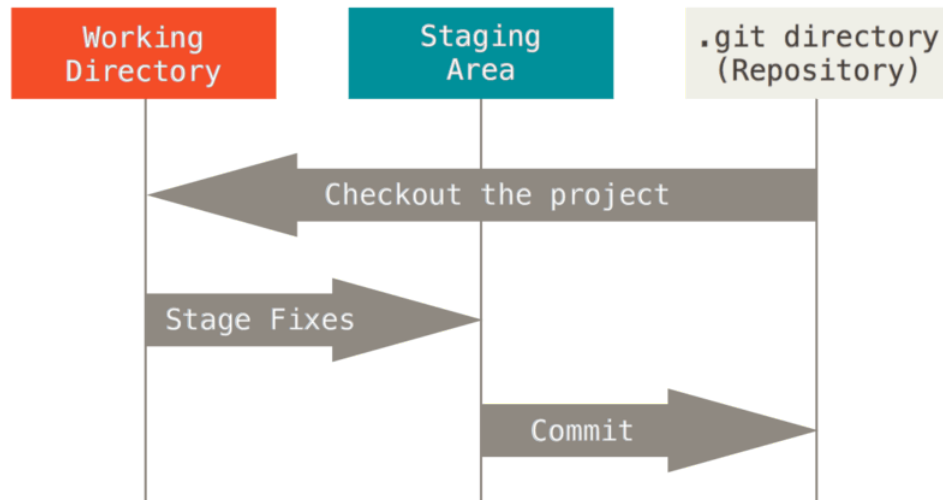
`git checkout <name file>` = `git switch <name file>`

`git diff` Afficher les changements entre les commits, le commit et l'arbre de travail

`git tag` : Les tags sont généralement utilisés pour capturer un point de l'historique utilisé pour une version marquée.

Les tags légers et les tags annotés diffèrent dans la quantité de métadonnées associées qui sont stockées. Les tags annotés sont stockés comme des objets complets dans la base de données Git.





Si création d'une nouvelle branch copie du projet à partir d'un commit donné, permet de ne pas impaqueter le projet de base

une branch est un pointeur

Ordre d'exécution a toujours respecter `commit → pull → push`

## Commandes Versionning GitHub

### Les fichiers ont trois états dans git :

- Modifié (modified) checker avec la commande "git status"
- Indexé (staged) utiliser la commande "git add"
- Validé (committed) utiliser la commande "git commit"
- m est pour ajouter un msg de commit après le -m "ajouter msg"
- push

### Création et indexage

- git add .
- git commit -m "update"

- `git push -u origin main`
- si la modification ne se fait pas il faut rajouter l'url "`git remote add origin git@github.com:maxousql/Versioning-GitHub.git`"

## Clonage

- `git clone https://github.com/OukhtySama/VersioningGithub.git`

## Gestion fichier

- `touch test.txt(nom fichier)` : touch permet de créer un fichier
- `git add .`
- `git commit -m "Update"` ( il faut commit le plus souvent possible 1 tache un commit)
- `git rm --cached test.txt (nom fichier)`
- `git add .`
- `git status`
- `git reset HEAD`
- `git log` : liste tous les commits
- `git mv` : rename en le déplaçant
- `git log --pretty=oneline`
- `git branch -d` : delete une branch