Versionning Git/GitHub

Généralités

Outils de Versionning

Définition

> 📌 Un système de contrôle de version est un système qui permet à tout projet collaboratif de travailler ensemble afin de suivre les modifications apportées au projet (pas forcément développement)

Acronymes :

SCM : Source Code Management → Nom de l'outil de versioning

Ces outils permettent ainsi de prendre des "snapshots" du projet afin d'avoir une maitrise en temps réel sur un projet :

- Tracabilité
- Collaboration
- Peu de conflits et facilité de gestion
- Code fonctionnel et développement simultané$
- Création de workflow

Exemple de SCM :

- Git
- AWS CodeCommit
- Azure DevOps Server
- Subversion (apache)
- …

Technologie PtP → Peer To Peer

> 💡 Le PtP est un système décentralisé(= distribué) permettant de partager des fichiers ou autre entre plusieurs PC.

Cela permet ainsi d'avoir une sécurité pour les fichiers, qui sont stockés sur plusieurs supports à distance.

Git → SCM

GitHub → Interface graphique pour Git et avec stockage sur des serveurs distants du code

GitLab → Interface Graphique

> 💡 Repository ⇒ C'est un espace disque sur lequel on initialise une projet GitHub

Créer un répo Git à partir de 0

`git init` → Permet d'initialiser le dossier d'initialisation de git dans le dossier local afin de l'initialiser localement.

Cela créer un dossier caché `.git` qui permet ainsi de tracer tout ce qui se trouve à l'intérieur (fichiers, sous-dossier, etc)

`git status` → Permet de suivre le statut du fichier (a exécuter à la racine du projet)

> !! Git sauvegarde chaque fichier quand il y a une modification (on est donc dans un modèle de snapshot).

`git commit -m "Ajout readme"` → Permet de faire un commit sur la branche localement :

- `-m` ⇒ oblige l'ajout d'un commentaire de commit
- `"Texte"` ⇒ Titre du commit

(`-a` → permet de commit et ajouter à la volée)

`git log` → Permet de voir les commit effectués dans le repo

> 💡 CheckOut ⇒ Récupération du projet pour le mettre dans le dossier local. Synchronisation des modifications faites sur Git par les autres.

`git branch -M main` → Changement du nom de la branche principale

`git remote add origin git@github.com:Th1b0t/rep_cours.git` → Permet de faire un raccourci vers l'URL du repository distant :

- `remote` → élément de type distant

- `add origin` → ajout d'un alias "origin"
- `git@github.com:Th1b0t/rep_cours.git` → Repo distant

`git push -u origin main` → Permet de pousser la modification sur la branche "main" du repo distant.

// Envoie le code sur origin et s'assure que la branche main en local existe et se synchro avec la branche main en remote //

`git remote` → Permet de voir les alias de repo distants

`git config -l` → permet de lister la configuration du repertoire local git.

Problème clé SSH

changer la clé privé dans l'agent SSH → `eval "$(ssh-agent -s)"`
Fonctionne uniquement en Git Bash (pas powershell)

Pour ajouter la nouvelle clé SSH à l'agent → `ssh-add c:/users/nom_user/.ssh/nom_clé`

La clé s'ajoute.

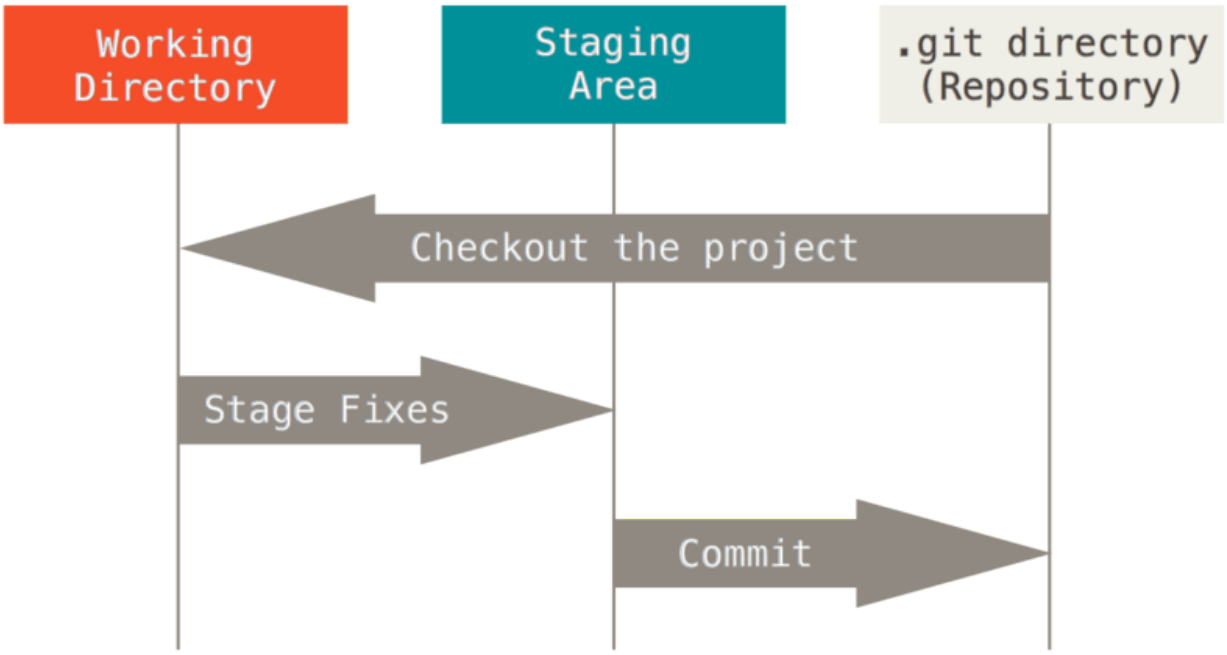
---

3 états pour les fichiers avec GIT :

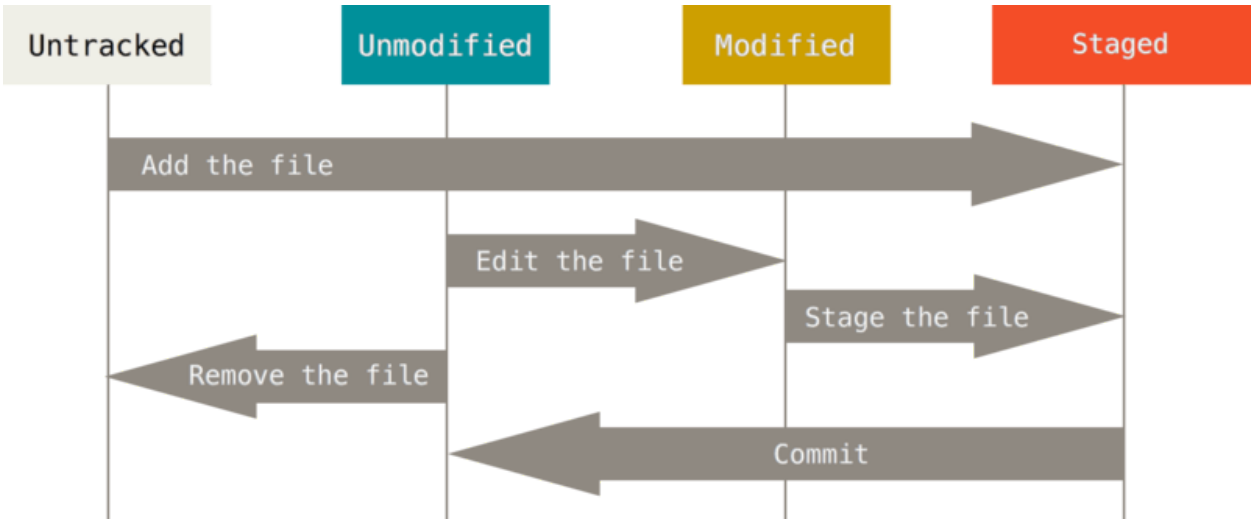
- Modifié (Modify) → Modifications apportées sur le disque uniquement
- Indexé (Staged) → quand la commande `git add` est exécutée
- Validé (Comited) → Quand la commande `git commit` est exécutée

Ordre optimisé :

- commit → Sauvegarde en local
- pull → récupère depuis le repo distant
- push → Pousse dans le repo distant



Les différentes étapes de travail avec git



Description des différents états d'un fichier lorsque les états changent

Pour supprimer un repository local → Il faut supprimer le fichier `.git` pour supprimer le repo local

> ⚠️ Cela supprime tout le suivi local de git, s'il n'est pas sauvegardé en ligne, il sera perdu.

Créer un repo Git à partir d'un projet

Pour récupérer un projet, on va faire le clone d'un repo distant :

On va sur le projet > Bouton "Code"> Clone > lien SSH ou HTTPS

On va ensuite effectuée sur le dossier où l'on souhaite cloner le projet :

`git clone lien/ssh_ou_Https`

Le clone s'effectue dans le dossier.

Commandes git :

`git add .` → Permet d'ajouter le suivi sur les fichiers locaux (`.` = tous le dossier)

`git rm --cached` → permet d'exclure un fichier du suivi git

> 📌 Dans le git, on ne pousse pas les fichiers de dépendances (node par exemple) ainsi que les fichiers de compilation non utiles

Pour faire ignorer des fichiers, on va créer un fichier `.gitignore`

On va ainsi ajouter les noms des fichiers dans ce fichier qui seront donc pas pris en comtpe par git. Cela prend en compte la récursivité des dossiers.

`git reset HEAD nom_fichier` → Permet de retirer un fichier de l'index :

- `HEAD` → Permet de l'appliquer par sécurité sur la branche du bout sur laquelle on travail.

Pour renommer un fichier, on va utiliser la même méthode que sur Linux :

`git mv ancien_nom nouveau_nom`

Cela permet ainsi d'avoir un suivi en passant par git.

Pour afficher les logs de manière plus lisible :

`git log --pretty=oneline`

Pour détruire l'historique Git, on peut utiliser la commande :

Les notions de branches :

Une branche est un espace de travail.

Quand on créé une nouvelle branche, on fait une copie du projet à partir d'un commit donné, et cela n'impactera donc pas le projet de base lorsqu'on travaillera dessus.

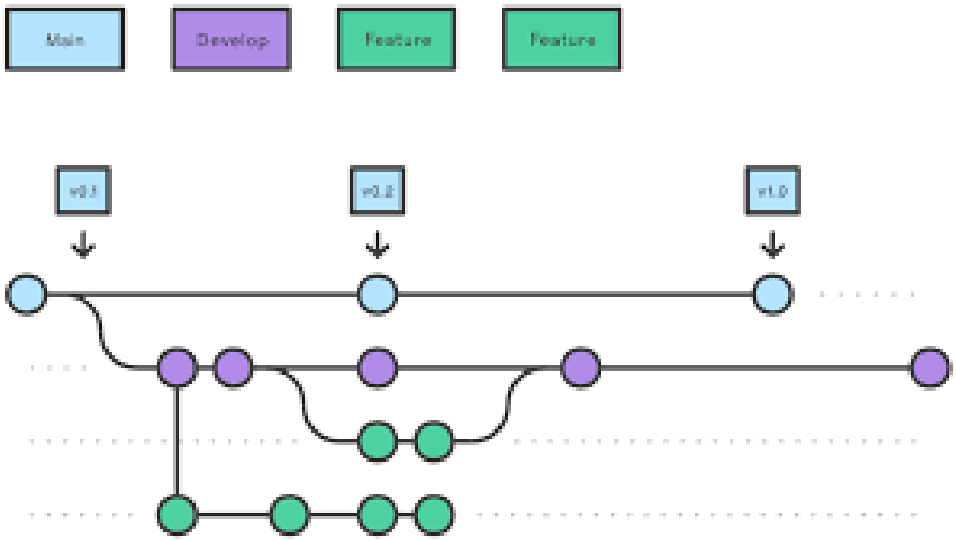
`bit branch nom_branche` → permet de créer une branche

`git branch` → Permet de lister les branches existantes

`git log` permet de savoir également sur quelle branche on est ⇒ `HEAD →` Indique la branche sur laquelle on est présente.

Pour changer de branche, on effectue la commande → `git checkout nom_branche`

Pour créer et changer à la volée de branche, on utilise → `git checkout -b nom_branche`



Les branches dans Git

Travaille sur les branches

On peut fusionner deux branches, pour ce faire :

- On se place dans la branche qui va recevoir la fusion (le merge en anglais) grâce à la commande `checkout`
- On rentre ensuite la commande pour fusionner → `git merge nom_branche_a_fusionner`

👥 Suivi projet groupe Versioning