Git

mail du prof: mnbdpro@gmail.com

Systeme de controle de version :

Git logiciel de versioning

Github Site qui permet le stockage de ces données

Git : Outils qui permet a tout projet de faire vivre la code base de la partager et de la faire vivre de manière active

SCM source code management :

- Permet de stocker les versions
- QUI, QUE, QUOI, COMMENT, POURQUOI

Les principaux SCM :

- Git
- AWS code commit
- Azure devOps serveur
- Subversion
- Mercurial
- CVS
- …

Peer to peer : C’est un lien qui permet de partager/dupliquer des fichiers sur plusieurs fichier décentraliser/distribuer. Permet de récupérer des fichier même si son PC fais BOOOOOOOOM, comme la blockchaine

GitHub est un site web → plateforme de travail collaboratif

Repository alias repo : espace disque sur lequel on stock le projet

`git init` creer un dossier .git caché (localement) un dépôt ou tout ce qui est dedans est tracé

`git status` indique le status du dépôt

ce dépôt git va permettre de suivre l’état, tout changement, tout sera trace.

Git sauvegarde tout les changement afin de pouvoir remonter facilement

Git fonctionne avec des instantané

Lorsqu’on travail sur git il créer automatiquement des branches

`git add fichier` permet de suivre un fichier ou `git add *` ajout tout les fichier non suivis

`git add .` ajoute tous les fichier de façon récursive

`git commit -m "Description de la sauvegarde"` Permet de sauvegarder pour git (Souvent et expliquer) localement pour le moment

`git log` Permet de voir les log

Tout ce fais en décentralisé donc pas besoin d’internet le seul moment ou il y a besoin c’est pour envoyer au patron et a tout le monde

Checkout : Recupere le projet pour travailler sur notre machine

`git branch -M main` Permet de modifier le nom de la branche master en main (Optionnel)

`git remote add origin git@github.com:ajean74/GitHubVersioning.git` remote une adresse distante sur lequel je peux envoyer mon code, origin est un alias, le tout creer un raccourci du nom origin pour l’url

`git push -u origin main` Envoie le code sur origin donc centralise le code, ma branche local main va a la branche distante main d’origin et s’assure que les deux branche main soit synchroniser (originssh)

`ssh-keygen -t ed25519 -C "anthony”` Crée une clé publique et une clé privé, le clé publique doit être transmise a GitHub.

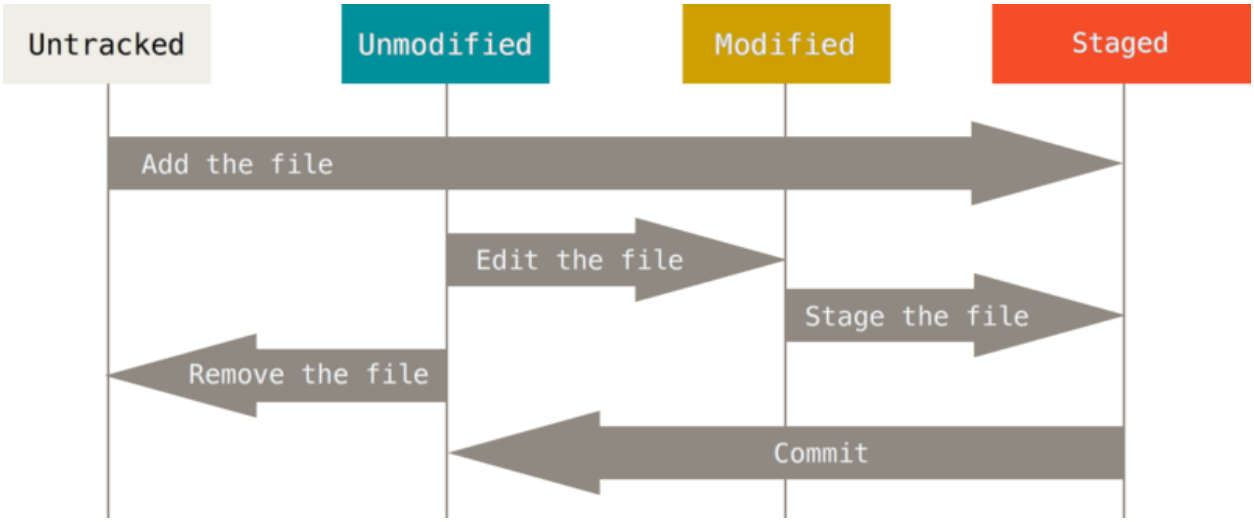
La clé correspond a la machine.

`eval "$(ssh-agent -s)”` ajoute la clé (pas obligatoire)

Chercher des commande et dire le site (stackoverflow)

`git commit -a` ne pas faire : permet de add et commit tout les fichier d’un coup

`git commit -l` affiche la config du repo git loal



Décrit les différant état d’un fichier

`rm -rf .git/*` supprime le repo local

Clonning :

`git clone https://github.com/mnbdpro/Tuto-Git.git` permet de cloner

`touch text.txt` creer un fichier faire le add puis commit puis push après

`git rm nomfichier` supprime un fichier faire le add puis commit puis push après

`git rm --cached test.txt` retire le fichier du suivi git. Les modifications ne seront pas envoyé

récursive descend en cascade dans les dossier, s'applique a tout les sous dossier

fichier .gitignore enlève les fichier que l'on ne veut pas envoyé

`git restore nomfichier` pour onstage un fichier

`git reset nomfichier` retire un fichier du suivi

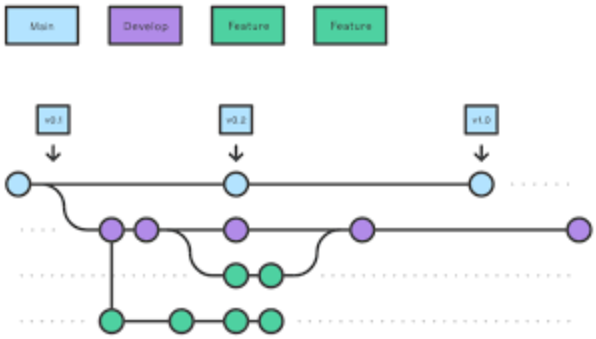
`git reset HEAD nomfichier` HEAD : pointe vers le commit le plus récent sur lequelle on travaille

`git mv nomfichier newnomfichier` Deplace et renome le fichier

`git log --pretty=oneline` affiche les lois sur une ligne

`git log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all` cosmétique jolie beau

Les branches:



Quand on crée une branche on créer une copie du projet a partir du commit donné et lorsqu'on travaille sur cette branche on n'impacte pas le projet de base

Ce que l'on modifie sur un branche n'a pas d'impact sur les autres

Par contre si on modifie un document commun au a plusieurs branche cela peut provoquer un conflit, tout comme travailler a deux sur un même document et le mettre créera un conflit

`git branch nom` creer une branche

`git branch` liste les branche et montre avec une étoile celle ou on est

`git checkout nom` change la branche active

`git checkout -v nom` creer et change la branche a la volée

`git merge nomdelabranche` fusionne deux branche mais d'abord il faut vérifier qu'il n'y est rien a commit sur les deux branche (Fusionne la branche nomdelabranche sur la branche sur laquelle on est)

`git branch -d nomdelabranche` pour supprimer la branche

`git diff` permet de voir si il y a des modification (ne pas trop utilisé)

commit → pull → push

`git pull` récupère le travaille sur GitHub depuis la dernier fois

si il y a un conflit avec un merge auto après un pull on regarde le document on choisi les truc qu'on veut et on commit

On doit commenter les commandes si la commande est utilise 15 fois on la décrit une fois et après voir truc

On doit prendre des screen