GIT

Table des matières

https://git-school.github.io/visualizing-git/

Git est un logiciel de versioning, qui permet de gérer les modifications d'un projet de développement.
Pour chaque enregistrement de modification du projet (Ajout, suppression ou modification de fichier), on créé un commit.
Un commit correspond donc à une version d'un projet à un instant précis.
L'ensemble des commits constitue l'historique d'un projet.

Les logiciels de gestion de version peuvent être basés sur deux modèles :

- Modèle centralisé : un serveur central contrôle toute la base de code du logiciel.
  Ex : SVN, CVS
- Modèle distribué : tout les postes ont accès à la base de code sans passer par un serveur central.
  Ex : Git, Mercurial, Bazaar
  Les avantages du modèle distribué :
  - Moins de risque de perdre le code source
  - On peut travailler plus rapidement et sans être connecté à Internet

Un remote est une sauvegarde complète d'un projet (code + commits) sur un autre ordinateur ou service en ligne (remote externe partagé comme GitHub, Bitbucket ...)

Configuration

fichier .gitignore

```
# Ignore l'intégralité du répertoire node_modules
node_modules/
# Ignore le fichier test.txt situé à la racine du
# projet (même niveau que le fichier .gitignore)
test.txt
# Indique le chemin d'un fichier à exclure se
# trouvant dans un sous répertoire
sousRep/autreFichierAexclure.txt
```

Définir le nom et l'email pour lier l'environnement de dev au compte Github **`git config --global user.name ''Steph''`** **`git config --global user.email ''stephchevalier38@gmail.com''`** `Sans l'option` **`--global =>`** `définit pour l'environnement ciblé`

`git --version`	Affiche la version de git
`git config --list`	Affiche la config list
`git config user.name`	Affiche le nom utilisateur git
`git help`	Affichage de l'aide
`git status -s`	Affiche le statut des fichiers M = modifié A = nouveau fichier Added mais pas commité ?? = fichier pas encore pris en compte par git

Commandes courantes

`git status`	Connaître le statut du Répertoire
`git init`	Rendre le répertoire « repository Git »
`git add fichier.txt (unAutreFichier.txt)`	Ajouter le fichier dans l'index Git (liste des fichiers gardés en mémoire par Git)
`git add .`	Ajouter tous les fichiers non indexés dans l'index Git
`git commit -m ''description du commit''`	Valider un commit
`git commit -am ''description du commit''`	Ajoute le fichier dans l'index + valide le commit (fusion des 2 commandes précédentes si fichier déjà dans l'index)
`git commit --amend`	Permet de modifier la description du dernier commit (avant d'être « pushed »)

RM/MV

`git rm fichier.txt`	Supprime un fichier
`git rm --cached fichier.txt`	Retire le fichier de l'index git sans le supprimer

`git mv fichier.txt fic.txt`	Modifie le nom du fichier → fic

Historique git

`git log`	Afficher l'historique des commits
`git log -1`	Affiche le dernier commit
`git log --oneline`	Affichage réduit des commits
`git log --stat`	Affichage complet des commits
Chaque commit est identifiable par son **SHA (Secure Hash Algorithm)**	
`git checkout` *`SHA`*	Se positionner sur un commit précis
`git checkout master`	Se repositionner sur le dernier commit
`git blame fichier.txt`	Liste toutes les modifications du fichier ligne par ligne
`git show sha(du blame)`	Affiche les modifications précises du commit

Password

Profil Settings > Developer Settings > Personal access tokens

repo local vers repo github (première fois)

`git init`
`git add .`
`git commit -m ''Premier commit''`
`git remote add origin` *`lienfournipargithub`*
`git push -u origin master`

repo github vers local (synchro)

`git init`
`git remote add origin` *`lienfournipargithub`*
`git pull origin master`

Gestion du repository

`git clone` *`lienfournipargithub`*	Permet de cloner le repository d'un projet
`git push origin master`	Permet d'envoyer les modifications sur le repository

	distant (origin => ordinateur par défaut / master => branche principale)
git pull origin master	Récupérer les modifications du repository distant

Gestion des branches

Les branches permettent de travailler sur des versions de code qui divergent de la branche principale contenant le code courant.
Lorsqu'on initialise un repository Git, la branche par défaut est appelée master.

git branch	Permet de visualiser les branches du repo
git branch *nouvellebranche*	Permet de créer une nouvelle branche
git branch -d *nouvellebranche*	Permet de supprimer la branche
git checkout *nouvellebranche*	Permet de se positionner dans la nouvelle branche
git checkout -b *nouvellebranche*	Fusion des 2 commandes précédentes (création et positionnement dans une nouvelle branche)
git merge *nouvellebranche*	Permet de fusionner (rajouter) nouvelleBranche dans la branche actuelle (celle où l'on se trouve)
git commit	Un commit sans message suite à une résolution de conflit résultant d'un merge

git stash	Sauvegarde de modifications en attente (pour avoir une branche clean mais qui comporte quand même des modifications en attente)
git stash list	
git stash show	

git reset --soft	
git reset (--mixed) *SHA*	
git reset --hard	Supprime toutes les modifications