Tuto GIT

La procédure à faire chaque fois que vous souhaitez modifier le projet :

1) **git pull** pour télécharger la dernière version des fichiers sur votre ordinateur (ou **git clone**, si le projet n'est pas déjà sur votre ordinateur)
2) Faites les modifcations que vous souhaitez (rajouter du code, rajouter un fichier, etc…)
3) Si il y'a des conflits il faut les résoudre
4) **git status** pour vérifier quels fichiers sont « surveillés » par git
5) **git add nomDuFichier** pour que git surveille un fichier (**git rm nomDuFichier** permet de demander a git d'arrêter de surveiller un fichier)
6) **git commit -m « Votre message »** pour créer un commit sur votre ordinateur
7) **git push** pour publier votre commit sur gitlab et le rendre accessible a tous

Note : Si jamais vous ne pouvez pas push et que l'erreur vous indique que votre historique de fichiers n'est pas bon, ça veut dire que quelqu'un a modifié les fichiers après votre pull (celui de l'étape 1), il faudra donc pull de nouveau, résoudre les conflits, commit puis enfin push

Branches

Une branche est une série de commit

Vous ne l'avez probablement pas remarqué, mais tout nos PC ce matin étaient sur la branche par défaut (qui s'appelle developp), tous nos commits avec nos coordonnées ont donc été envoyés sur cette branche

Mais du coup, si une branche c'est juste une suite de commit, a quoi ça sert ?

C'est utile, car on peut chacun créer une branche par fonctionnalité a programmer, et faire les push uniquement sur notre branche, comme ça on pourra éviter tout les conflits qu'on a eu en TP, notamment

C’est possible de créer une autre branche a partir d’une branche qui existe avec la commande **git checkout -b nomDeLaBrancheACreer**

Admettons qu’on va créer une branche « test » a partir de la branche « developp »

On fait **git checkout -b test**

La branche test est juste une copie de la branche developp, elle contient les mêmes fichiers mais est indépendante, si vous modifier des fichiers sur la branche test, ça ne modifiera pas la branche developp

Une fois qu’une fonctionnalité a fini d’être programmée par quelqu’un, on peut rajouter tout le contenu de la branche a la branche principale (developp)

On fait ça avec ce qu’on appelle une merge request, c’est-à-dire qu’on va fusionner les 2 branches (C’est une manipulation qu’on fait sur gitlab en général, je vous en reparlerais en temps voulu)

Maintenant que j’ai expliqué le principe général, on va parler des commandes

git checkout nomDeBranche vous permet de changer de branche (Attention, lorsque vous faites un commit ou un push, a bien être sur la bonne branche, attention a ne pas accidentellement faire un commit sur developp, il faudra le faire sur la branche que vous avez crée !)

git checkout -b nomDeBranche permet de créer une nouvelle branche, git vous déplacera aussi sur cette branche (donc tout vos commits iront sur cette branche)

git branch vous indique sur quelle branche vous être actuellement

Attention, il est TRES important de vérifier sur quelle branche on est avant de faire git pull, git push ou git commit, afin de ne pas modifier ou télécharger les mauvais fichiers

Sur gitlab, a gauche de l’écran, dans l’onglet « repository » il y’a une option « branches » qui vous permet de voir toutes les branches

Il y’a aussi une option « graph » qui vous permet de visualiser ses branches, et une partie des commits