**Utiliser GIT**

**Les commandes pratiques**

- **git branch (git b)** : voit sur quelle branche je suis, par rapport à toutes les branches dispo

- **git branch –a (git ab)** : voit plus de branches

- **git status** (**git st**) : voit quel est l'état des fichiers

- **git lga** : voir l'état des branches entre elles et par rapport aux commits + le positionnement du HEAD.

- **git remote add <alias de ce repo distant> <url repo distant>** : permet d’ajouter un repo distant

- **git remote -v** : voir la liste des repo distant

- **git checkout -b <nomDeLaBrancheACreer>** : déplace le HEAD sur une nouvelle branche que l’on crée.

**Sauver son travail localement**

- 1/ **git checkout <branche locale de travail qui n'a pas son pareil sur github>**

- 2/ **git add -A**

- 3/ **git commit –m "mon message"**

**Sauver son travail sur github**

- 1/ sauver son travail localement

- 2/ **git checkout <branche locale qui est la copie de la branche distante>** (--> positionner le HEAD sur la branche à pusher)

- 3/ **git fetch** (--> cela vérifie si des modifications ont été faite sur la branche distante. En vrai, cela vérifie toutes les branches du projet)

- 4/ Si il y a des modifications, faire "**Récupérer des données distantes**", puis retourner à "**Sauver son travail sur github - 2/**"

- 5/ **git merge <branche locale de travail qui n'a pas son pareil sur github>** (--> je met à jour ma branche locale qui est la copie de la branche distante, grace à ma branche locale de travail qui n'a pas son pareil sur github)

- 6/ **git push <alias de l'adresse distante>** (par défaut origin) **<branche distante qui est la copie de la branche locale>** (par défaut git regarde la branche sur laquele on est)

- 7/ **git checkout <branche locale de travail qui n'a pas son pareil sur github>** (--> je suis de retour sr ma branche locale de travail pour continuer mon travail)

**Récupérer des données distantes**

- 1/ **git checkout <branche locale qui est la copie de la branche distante>** (--> positionner le HEAD sur la branche branche locale qui est la copie de la branche distante)

- 2/ **git fetch** (--> cela vérifie si des modifications ont été faite sur la branche distante. En vrai, cela vérifie toutes les branches du projet)

- 3/ **git merge <aliasAdresseDistante/branche distante qui est la copie de la branche locale>** (-->e.g. origin/dev)

- 4/ **git checkout <branche locale de travail qui n'a pas son pareil sur github>** (--> on se repositionne localement)

- 5/ **git rebase <branche locale qui est la copie de la branche distante>** (--> mettre à jour sa branche locale de travail par rapport à sa branche locale qui est la copie de la branche distante, qui est à jour depuis 3/)

**Récupérer des données distantes, en cours de travail**

- 1/ **git add - A**

- 2/ **git stash** (--> met de côté le travail actuel)

- 3/ **git checkout <branche locale qui est la copie de la branche distante>** (--> positionner le HEAD sur la branche branche locale qui est la copie de la branche distante)

- 4/ **git fetch** (--> cela vérifie si des modifications ont été faite sur la branche distante. En vrai, cela vérifie toutes les branches du projet)

- 5/ **git merge <aliasAdresseDistante/branche distante qui est la copie de la branche locale>** (-->e.g. origin/dev)

- 6/ **git checkout <branche locale de travail qui n'a pas son pareil sur github>** (--> on se repositionne localement)

- 7/ **git rebase <branche locale qui est la copie de la branche distante>** (--> mettre à jour sa branche locale de travail par rapport à sa branche locale qui est la copie de la branche distante, qui est à jour depuis 5/)

- 8/ **git stash pop** (--> remet en service le travail mis de côté)

**Revenir au commit précédent en supprimant le commit actuel**

- **git reset --hard @~1**

(le commit 'supprimé' est encore en mémoire jusqu'au passage du garbage collector.

git reflog pour voir tous les commits)

**Supprimer une branche locale**

pré-requis : ne pas être sur la branche à delete

- **git b -d <nomDeBrancheADelete>** (--> manière safe de delete. Vérifie par exemple que la branche a été merge avant de delete)

ou

- **git b -D <nomDeBrancheADelete>** (--> Delete quoi qu'il en soit)

**Supprimer une branche distante**

pré-requis : ne pas être sur la branche à delete

- **git push <alias du repo distant> :<brancheADelete>**