Commandes de base GIT : git init

git init : permet de créer un dépôt git local.

Commandes de base GIT : git add

git add : Ajouter nos fichiers dans le staging area

Utilisation :

git add mon_fichier.ext

git add mon_dossier

git add *.ext : Ajouter les fichier d'une extension spécifique.

git add -A ou git add* . : Ajouter tous les fichier

Commandes de base GIT : git status

git status : Montre l'état des fichier en montrant les changement qui sont en stage et ceux qui ne le sont pas encore.

Commandes de base GIT : git commit

git commit -m”Votre message” : C’est la commande de la création de version ou du versionnement proprement dit.

L’option -m* : Permet de définir un message qui décrit le motif du commit.

l’option -a(--all) : Quand on ajoute cette option,les fichiers sont ajoutés au stage et après faire un commit.

Commandes de base GIT : git log

git log : Cette commande vous permet de voir l'historique de vos changements

--oneline* : permet d'afficher l'historique avec une ligne par commit (plus lisible)

-n <nombre> : permet de sélectionner le nombre de commit à afficher

-p <fichier> : permet de voir l'historique des commits affectant un fichier en particulier

--author <nom_auteur> : permet de voir l'historique par rapport au nom de l'auteur

Commandes de base GIT : git checkout

git checkout <nom_branch> : C’est la commande qui vous permet de naviguer entre les branches.

git checkout <hash_commit> : Voir l’état du code à commit particulier.Tout changement effectué ici n’a aucun impact sur votre historique.

git checkout <hash_commit> <mon_fichier.html> : Permet de transformer le <fichier> tel qu'il était lors du <commit> et l'ajoute au staging.Si vous ne voulez pas revenir à ce code, il suffit de faire **git checkout main <mon_fichier>**.

Commandes de base GIT : git revert

La commande revert vous permet de défaire un commit.C’est-à-dire retirer tous les changement qui avaient été fait.Ceci ne permet donc pas de revenir à l’état du projet à cet instant sans rien perdre dans l’historique.

git revert <nom_commit>

Commandes de base GIT : git reset

Contrairement à la commande revert, la commande reset est très dangereuse, car elle nous permettre de revenir en arrière et donc modifier l'historique.

git reset HEAD <nom_fichier> : Supprimer des modification dans le stage.Si par exemple on a pas encore terminé le travail à faire sur un fichier, on peut le retirer du stage.

git reset : Va simplement retirer tous les fichiers du stage sans supprimer l'historique ou les modifications

Pour les 2 commandes ci-dessus, on peut utiliser une alternative récente de git qui est :

git restore --staged <nom_fichier> : Retirer un fichier du stage.En mettant un point à la place de **<nom_fichier>**, tous les fichiers modifiés sont retirés du stage.

Commandes de base GIT : git reset

git reset <commit>: Revenir au commit tout en ajoutant tous les autres changements d'autres commits de l'historique dans le working Area pour que le travail ne soit pas perdu.

git reset --hard <commit> : Va non seulement supprimer l'historique pour revenir au commit, mais aussi toutes les modification du stage

Si vous avez fait un soft reset, vous pourrez revenir en arrière avec :

1. git reflog : Pour voir les traces de l'historique
2. git reset ''HEAD@{1}": Choisir le commit sur lequel on souhaite de positionner

Commandes de base GIT : git branch

git branch <nom_branch> : C'est la commande qui vous permet de créer une branche local.

Sans option **<nom_branch>** cette commande affiche les branches.

Commandes de base GIT : git stash

git stash : stocke (ou stashe) temporairement les changements apportés à votre copie de travail pour que vous puissiez effectuer d'autres tâches, puis revenir et les réappliquer par la suite.Il peut aussi être utile si par erreur, vous aviez commencé à écrire du code dans une branche autre que celle qui est prévue pour le code.

Commandes de base GIT : git push

git push : C’est la commande qui vous permet d’envoyer vos changements dans le repo distant.