NOTES GIT QUENTIN MOREL :

• Commit / Pull / Push -> ordre important

• Outils versioning -> les Source Code Managment (SCM)

• Git :
- Travailler de manière collaborative
- Suivre l'avancement d'un projet
- Arborescence du projet

• SCM -> Traçabilité, Collaboration, Peu de conflits & meilleure gestion, code fonctionnel & dev simultané, création workflow

• Peer to peer
• Système décentralisé

• Repository (repo)

• Sauvegarder l'intégralité des modifications, pas seulement les différences

• Git fonctionne avec l'instantanné, on ne perd ainsi aucune donnée

< git status > -> avoir toutes les infos

• Evenement/sauvegarde = Commit

• Faire des commits régulièrement à chaque modification "cohérente"/ajout de nouvelle petite feature, mais ne pas fa ire des commits à la fin de la journée juste pour réintégrer le projet en cours

< git push -u origin main > -> Envoyer le code sur "origin" (lien qu'on a précédemment ajouté avec < git remote add origin https://github.com/Siphin/Versioning-ESGI.git >) et synchronise la branche main locale avec la branche main github

< git remote set-url origin git@github.com:Luigi1802/projetWebGroupeVersioning.git > -> Connecter un projet via SSH

<git clone URL> -> copier le projet

• Les fichiers ont 3 états dans git : Modifié (modified) -> en modifiant le fichier | Indexé (staged) -> avec < git add > | Validé (commited) -> avec < git commit >

< git commit -a > -> ajouter et commit en même temps (pas conseillé)

< rm -rf .git/ > -> supprimer le dossier .git

• Memo :
- git pull (maj/recup)
- git add .
- git commit -m "libellé modif" (ajouter)
- git push (envoyer)

< rm test.txt > -> supprimer un fichier

• Fichiers de dépendance / build / compil -> on en veut pas

< touch > -> créer un fichier

• Fichier .gitignore pour entrer des fichiers à ignorer
• Il est possible de récupérer des fichiers .gitignore préfaits afin d'avoir directement la plupart des fichiers à ignorer a u début d'un projet

• Supprimer un fichier de l'index : < git restore --staged (fichier) > OU < git rm --cached (fichier) > OU < git reset H EAD (fichier) >

< git restore > -> annuler les modifications

< git mv test.txt renamed.txt > -> renommer un fichier

• :q pour sortir

< git log > -> afficher les logs
< git log --pretty=oneline > -> afficher les logs sur une seule 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 > -> logs plus j olies

• Quand on créé une nouvelle branche -> ça copie l'ensemble du projet à partir d'un commit -> sans impact sur le pro jet de base

• Une branche est un pointeur vers un commit donné

< git branch (nom) > -> créer une nouvelle branche
< git branch > -> afficher les branches & sur quelle branche on se situe

< git checkout (nom-branche) > -> changer de branche

< git push --set-upstream origin (nom) > -> Push une branche

< git merge (nom-branche) > -> Pour fusionner (en étant positionné sur la branche sur laquelle on veut les modifs)

<git branch -d (nom-branche) > -> pour supprimer une branche

< git diff > -> voir si il y a des modifs