Fonctions GIT

**working directory 🡪 staging area 🡪 repository 🡪 Github**

**add commit push**

# Globalement

HEAD : commit le plus récent

Master : réservé pour le code stable (toujours travailler sur une branche à part)

.gitignore : fichier texte listant les fichiers à exclure du add .

# Depot local

## Configuration générale

* *git config* : Configure le depot
  + *-l :* affiche les informations du dépot
  + *–global user.name « nom »*
  + *–global user.email « mail »*
  + *--global core.editor "notepad"* : pour ouvrir un éditeur de texte lors des commits
  + credential.helper : ?????
* curl -u "username :<MYTOKEN>"https://github.com/username/ol3-1.git

## Synchronisation du dépot

* *git init* : créé un dépôt dans le repertoire courant
* git clone URLDepot: clone le dépôt en local depuis Github et créé la connexion
  + --depth=n URLDepot: clone partiel avec profondeur du projet
* git remote : répertorie les connexions actives
  + -v : inclue les URL de chaque connexion
  + add nom URLDepot : crééer une connexion au dépôt de l’URL
  + rm nom : supprime la connexion
  + rename old new: renommer la connexion
* git pull : depot local < depot Github
  + COMME PUSH
  + –force : outrepasse les interdictions en cas de litige
* git push : depot local > depot Github
  + origin nomLocalBranche : pousse juste la branche
  + -u nomLocalBranche : pousse la branche

## Staging area

* *git* *status* : affiche les éléments dans la zone de transit (en vert ceux ajoutés)
* *git add fic/dos*: déplace des éléments dans la zone de transit
  + *. / \** : sélectionne tout le contenu du dossier courant
* *git* *rm* fic: retirer des éléments de la zone de transit
  + *--cached fic* : pour garder les modifications en local
  + -f : pour supprimer les modifications en local
* *git* *reset* : retire tous les éléments de la zone de transit

## Commits

* *git commit* : transfère de la zone de transit au dépôt
  + *-m* « nom » pour donner un nom dans la commande
  + *-a* tous les fichiers qui ont été modifiés
* *git log* : affiche l’historique des commits
  + *--oneline* : plus synthetique
  + git log --pretty=format:"%h %s" --graph : affiche de manière plus visuelle
  + *-n x* : affiche les x derniers commits
  + *--author="Arthur"* : Affiche tous les commits de "Arthur" dans un projet
  + *--grep="chaine a chercher"* : renvoie tous les commits composés de la chaine
  + *--fichier* : les commits ayant modifié tel fichier
  + *--before="2021-7-1* : commits avant telle date (synthaxe tolérante)
  + *--after="1 week ago"* : commits après telle date (synthaxe tolérante)
* *git diff* : montre les modifications faites depuis le dernier commit
  + *nomDossier* : seulement sur tel fichier
  + *idCommit* : modifications faites depuis tel commmit
  + *idCommit1* idCommit2 : compare les modifications entre 2 commits
  + *fic1* *fic2* : uniquement sur tel/s fichier/s
* git tag : affiche les tags posés sur les commits
  + nomTag : tagg le commit courant
  + –delete / -d nomTag : supprime le tag
* git checkout commit\_branche\_tag: se déplacer de commit / branche (obsolète, utiliser switch)
  + -f --> forcer le retour, en effaçant les modifications apportées au bakasable (nouvelle branche crée après la modification d’un ancien commit)

## Branches

* git branch --> affiche les branches
  + branche --> créé une branche
  + -M branche--> préciser le nom
  + -a -> affiche toutes les branches du depot
  + -d --> supprime une branche en vérifiant si son contenu a bien été merge
  + -D --> supprime la branche
  + -m newName -> renomme la branche courante
* git switch branche : se déplace sur la branche
  + - : retourne sur la branche precedente
  + -c : créé la branche et s'y déplace
* git merge branche : fusionne la branche à la branche courant
* git rebase : ré-écrit les commits d’une branche
  + -i HEAD~X : permet de manipuler les X derniers commits

## Divers

* git stash : transmet toute la zone de transit dans une zone dedié, visible depuis vscode

# Depot distant

## Gestion dictateur

Programmeur 🡪 Lieutenant 🡪 Dictateur