Le jeudi, c'est Git

- Structure de git :
  - Working Directory (WD) : Répertoire de travail. Celui dans lequel on agit, on code.
  - Staging Area (SA) : Zone tampon, avant d'envoyer les fichiers dans le Repository
  - Repository (R) : Contient l'arborescence des commits
  - Stash : Pile pour mettre les fichiers modifiés que l'on ne souhaite pas ajouter
- Configuration de git
  - `git config --global user.name <username>`
  - `git config --global user.email <email>`
  - `git config --global core.editor <editor>` : permet de définir l'éditeur de texte qu'appellera git (pour les messages des commits, des merges, etc...)
- Commandes de git
  - `git init` : créer un projet git
  - `git add <fileName>` : ajoute le fichier du WD au SA
  - `git add -A` : ajoute tous les fichier au SA
  - `git reset <fileName>` : retire le fichier du SA et le remet dans le WD
  - `git commit -m <message>` : Met les fichiers du SA dans le R avec le message donné
  - `git commit` : Pareil que au dessus, mais ouvre l'éditeur de texte définit dans la configuration de git (cf : core.editor)
  - `git status` : indique les fichiers modifiés du WD qui ne sont pas dans le SA
  - `git diff` : indique les différences en le WD et le SA
  - `git diff <idCommit>` : indique les différences avec le commit sélectionné
  - `git diff <idCommit> <idCommit>` : entre les deux commits
  - `git log` : affiche les différentes versions
  - `HEAD~n` : pointe vers le nième commit avant HEAD
  - `git tag <tagName> <commit>` : met un tag sur le commit
  - `git stash` : met dans le stash les fichiers modifiés non validés
  - `git stash pop` : remet les modifications du stash dans le WD
  - `git reset <commitId>` : Met le HEAD sur le commit. Supprime tous ceux d'après
    - –soft : ne modifie pas le WD
    - –hard : modifie le WD pour correspondre au R
  - `git checkout` – : remet le fichier dans l'état du dernier commit
- Gestion des dépôts en ligne
  - `git clone <url>` : clone le dépôt distant
  - `git push` : envoie le dépôt courant sur le dépôt distant

  - `git pull` : récupère les modifications du dépôt distant
  - Si vous souhaitez envoyer sur un dépôt distant sans avoir cloné (remote non défini dans ce cas), il faudra réaliser ces deux commandes au début :
    * `git remote add origin <url>` : défini le remote
    * `git push -u origin master` : premier push
    * Une fois ceci fait, vous pouvez simplement faire push et pull
    * `git remote remove origin` : supprime le remote origin
- Gestion des branches
  - `git branch <nom>` : créer une branche
  - `git checkout -b <nom>` : créer une branche et la définit comme branche courante
  - `git branch` : liste des branches
  - `git branch -d <nom>` : supprime la branche
  - `git branch -D <nom>` : force la suppression de la branche (en cas de non fusion)
  - `git checkout <nom>` : change la branche courante
  - `git log --graph --decorate --all` : Arborescence du dépôt avec toutes les branches
- Gestion de la fusion
  - `git merge <branche>` : fusionne la branche sélectionné dans la branche courante
  - `git rebase <branche>` : linéarise la branche courante dans la branche sélectionnée (mais casse l'historique => utiliser que sur le dépôt local). Il faudra penser ensuite à réavancer l'étiquette de la branche sélectionné.