TD #1

GIT TD#1

commandes de base

INSTALLATION

=> utiliser la ligne de commande git

- si vous venez de réinstaller votre OS et que git n'est pas disponible, suivre la procédure :
  - sudo apt install git
  - https://git-scm.com/download/win
  - nous n'utiliserons pas d'interface graphique
- lancer un terminal :
  - gitbash
  - via win-R, recherche, programmes, raccourci, ...
- à défaut lancer :
  - powershell
  - via les mêmes méthodes

CLONE

=> copier un projet existant en local en utilisant une connexion ssh sans saisie de mot de passe

- s’appuyer sur des clés privée et publique
- https://github.com/settings/keys
- ssh-keygen -t ed25519 -C "commentaire, changer moi"
- garder les noms de fichiers proposés
- passphrase vide (c’est des TD/TP), attention perte de sécurité
- more ~/.ssh/id_ed25519.pub
- copier la ligne entière
- coller ici : https://github.com/settings/ssh/new
- faire le clone du projet : git clone git@github.com:XXX/YYY.git

FAIRE UN COMMIT

=> créer un lot à versionner

- une fois le clone fait : git clone git@github.com:XXX/YYY.git
- les commandes se font dans le répertoire projet :
  - cd YYY
- créer des fichiers :
  - touch journal.md
  - touch TODO.md
- vérifier leur état git :
  - git status
- ajouter ces fichiers :
  - git add .
  - git add journal.md TODO.md
  - git add *.md ; git add <...>

FAIRE UN COMMIT

=> envoyer notre lot sur le serveur

- vérifier leur nouvel état git :
  - git status
- ajouter un commentaire :
  - git commit -m “Create .md files”
- envoyer ces fichiers sur le serveur, sans que cela fonctionne :
  - git push
- commande pour le premier push :
  - git push --set-upstream origin master
- vérifier que ça fonctionne sans options et que tout est synchronisé :
  - git push
- recommencer :
  - nano journal.md # ajouter une ligne : création le <date>
  - git add *.md ; git commit -m “Update journal.md” ; git push

FAIRE UNE BRANCHE

=> travailler dans un “tiroir” spécifique

- vérifier l’état git :
  - git status
- créer une branche :
  - git checkout -b first-branch
- modifier des fichiers :
  - nano journal.md # ajouter une ligne : branche 1 le <date>
  - git add *.md ; git commit -m “Update journal.md”
  - git push
    - => trouvez la bonne commande !
- faire un 2e commit sur la branche
- vérifier sur le web :
  - https://github.com/XXX/YYY

FAIRE UNE COPIE DE COMMIT

=> copier un lot de code

- vérifier l'état git :
  - git status
- changer de branche :
  - git checkout main ; git status
- copier le 2e commit de la branche first-branch :
  - git log --graph --all
  - git cherry-pick ZZZZ
- vérifier l'état git :
  - git status
- faire un commit et pousser sur le serveur

DÉPLACER LA RACINE D'UNE BRANCHE

=> changer “d'armoire” notre “tiroir” spécifique

- vérifier l'état git :
  - git status
- vérifier la synchronisation avec le serveur :
  - git pull
- déplacer la racine de la branche first-branch sur le dernier commit de main :
  - git log --graph --all
  - git rebase ZZZZ
- vérifier l'état git :
  - git status
- pousser sur le serveur

DÉPLACER INTÉRACTIVEMENT UNE BRANCHE

=> synchroniser nos changements sur le serveur

- vérifier l'état git :
  - git status
- vérifier la synchronisation avec le serveur :
  - git pull
- déplacer la racine de la branche first-branch sur le 2e commit de main :
  - git log --graph --all
  - git rebase -i ZZZZ
  - faire un commit expliquant les différentes options du 'rebase -i' dans journal.md
- vérifier l'état git :
  - git status
- pousser sur le serveur
- corriger l'erreur

CHERCHER LE COUPABLE

=> vérifier la chronologie et l'historique d'un fichier

- vérifier l'état git :
  - git status
- lister les commits :
  - git log --graph --all
- afficher les détails d'un fichier :
  - git blame journal.md
- trouver les infos :
  - de chronologie
  - d'utilisateur
  - copier un exemple dans journal.md, l'expliquer en commentaire dans ce même fichier
  - commiter, pousser