Lab1 - Git

Définir la configuration globale et initialiser dépôt local

1. Définissez votre nom global et votre email
   git config --global user.name "Hamdi Brahim"
   git config --global user.email "brahim.hamdi.consult@gmail.com"
2. Créer un nouveau répertoire (GitTraining)
3. Initialiser un dépôt GIT sous le répertoire crée.
   git init

Premier projet

Sur le même dépôt :

1. Créer l'arborescence suivante :
   README
   MyConsoleApp
   | ***console.txt***
   MyWebApp
   | ***web.txt***
   | ***code.txt***
2. Vérifier l'état du dépôt local
   git status
3. Indexer README.txt
   git add README.txt
   Puis, vérifier de nouveau l'état du dépôt local
   git status
4. Créer le premier commit
   git commit -m 'Ma première soumission'
5. Ecrire quelques lignes dans les fichiers de la nouvelle arborescence
6. Ajouter tous les fichiers au dépôt local (en une seule commande)
   git commit -am « Projet 1 – commit 1 »
   Y a-t-il un problème ? Comment le résoudre ?
7. Ajouter la ligne suivante à chaque fichier :
   Ajouter durant le Workshop Git
8. Valider toutes les modifications en une seule commande.
   git commit -am « Projet 1 – commit 2 »
9. Exécuter la commande suivante :
   git log
   Qu'affiche cette commande ?
10. Créer les fichiers « .gitignore » et « doc ».

Ecrire « doc » dans la première ligne du fichier « .gitignore », puis vérifier l'état du dépôt local.
Que remarquez-vous ? Pourquoi ?
Valider les changements.

Supprimer un fichier d'un dépôt

Sur le même dépôt :

1. Supprimez le fichier, puis indexer le changement :
   git rm MyConsoleApp/console.txt
   git add .
   Voir comment le statut a changé.
   Puisque `` ``` MyConsoleApp``` devrait être vide, il devrait disparaître du disque, vérifiez avec cette commande:
   ls -l
2. A quoi rassemble la zone d'indexe ?
   git status
3. Pour annuler la suppression du fichier avant la validation, utiliser la commande suivante:
   git reset HEAD
   Suivie par :
   git checkout MyConsoleApp/console.txt
4. Re-supprimer le même fichier, puis valider
   git rm MyConsoleApp/console.txt
   git commit -am « Projet – commit de suppression de fichier »
5. Revenir à la dernière validation (dernière version).
   git revert HEAD

Travailler avec les branches

Avec le dépôt :

1. Lister les branches :
   git branch
   L'étoile marque la branche active
2. Crée une nouvelle branche:
   git branch my_aple_app
   Lister les branches.
3. Renommer la branche :
   git branch -m my_aple_app my_apple_app
   Lister les branches
4. Supprimer la branche :
   git branch -d my_apple_app
5. Créer et passer à la nouvelle branche
   git checkout -b my_apple_app

git branch

6. Faire des changements sur la branche et valider
   mkdir MyAppleApp
   echo "OS X implementation" > MyAppleApp/osx.txt
   git add MyAppleApp
   git commit -m "OS X version"
7. Revenir à la branche master
   git checkout master
   git branch
   Is `MyAppleApp` folder and its content in the working tree ? Why ?
   ls -l

Historique des commit

1. Preparation:
   git checkout my_apple_app
   git checkout master
2. Afficher l'historique des commit (en une seule ligne)
   git log --oneline
3. Historique d'une branche
   git log --oneline my_apple_app
4. Historique d'un fichier
   git log --oneline MyWebApp/code.txt
5. Historique d'un fichier
   git log --oneline MyWebApp

Fusionner les branches

1. Preparation
   git clone https://github.com/DevTrainings/test_merge_conflict.git
   cd test_merge_conflict
   git checkout bar
   git checkout cherry-pick
   git checkout foo
   git checkout master
2. Historique des commits
   git log --all --graph --oneline --decorate
3. Le fichier `file` a été modifié dans les deux branches` bar` et `foo` et nous voulons récupérer ces modifications dans la branche master.
   git branch
   cat file
4. Fusionner 'bar' dans 'master'
   git merge bar

git log --all --graph --oneline --decorate
cat file

5. Fusionner ‘foo’ dans ‘master’
   git merge foo
   conflit !!
   cat file
6. Résolution du conflit, puis continuer la fusion
   git add file
   git commit -m "Merge branch 'foo'"
7. Vérifier la fusion ?
   git log --all --graph --oneline –decorate

Travailler avec un dépôt distant

1. Preparation:

   git clone https://github.com/DevTrainings/premade_remote.git

   cd premade_remote

   git checkout documentation

   git checkout master

   git remote remove origin
2. Créer un dépôt vide sur GitHub
3. Lier le dépôt distant au dépôt local:

   git remote add origin https://github.com/brahimhamdi/repos_remote
4. Pousser l’ensemble du projet au dépôt distant

   git push origin --mirror
5. Supprimer le lien avec le dépôt distant

   git remote remove origin

   origin définit le nom de dépôt distant