[sabbehsyrine@gmail.com](mailto:sabbehsyrine@gmail.com)

//mot de passe s17111997S

//mot de passe github s09865592S

***GIT COMMANDES***

* **Initialiser git**

$ git config --global user.name "John Doe"

$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)

Grâce à l’option --global, vous n’aurez besoin de le faire qu'une fois.

$ git config –list

$ git config --global color.diff auto

$ git config --global color.status auto

$ git config --global color.branch auto

$ git config --global core.editor notepad++

$ git config --global merge.tool vimdiff

$ cd Documents/Fichiers/Git/PremierProjet

johndoe ~/Documents/Fichiers/Git/PremierProjet

$ git init

Initialized empty Git repository in c:/users/JohnDoe/Documents/Fichiers/Git/PremierProjet/

Retournez sur Git Bash, et tapez la commande suivante :colone depot a partir de github

**accéder à un dépôt distant**

git remote add OC https://github.com/OpenClassrooms-Student-Center/ProjetOpenSource.git

**coloner un dépôt en local**

git clone https://github.com/OpenClassrooms-Student-Center/ProjetOpenSource.git

**système de branches**

$git branch//Afficher branche principale(master)

git branch cagnotte //(créer une branche)

**basculé sur celle-ci.**

git checkout cagnotte

**Après :**

git branch

master

\* cagnotte

**Réaliser un Commit**

git commit -m “Réalisation de la partie cagnotte côté front end”

**//enregistrer les modifications avec la description “Réalisation de la partie cagnotte côté front end”**

**La commande Git push permet d'envoyer les modifications que l'on a réalisées en local sur le dépôt à distance, alors que la commande Git pull permet de récupérer en local le projet distant !**

**Corrigez vos erreurs sur votre dépôt local**

**Fonctionnement de git**

git add PremierFichier.txt

git commit -m ’test’

**Nous allons maintenant créer une branche (autre que la master).**

git branch brancheTest



**Supprimer une branche**

git branch -d brancheTest

git branch

**J’ai modifié la branche principale**

git status

Nous allons donc créer une remise.

git stash

$ git status

On branch master

nothing to commit, working tree clean

Parfait ! C'est exactement ce que nous voulions. Nous allons maintenant créer notre branche "brancheCommit".

git branch brancheCommit

git checkout brancheCommit

Et finalement, nous allons pouvoir appliquer la remise, afin de récupérer nos modifications sur notre nouvelle branche.

git stash apply

Nous allons d'abord regarder la liste de nos remises. Pour ce faire :

git stash list

Il suffira alors d'appeler la commande git stash  en indiquant l'identifiant.

git stash apply stash@{0}

**Maintenant que nous sommes sur la bonne branche, nous allons de nouveau faire un git reset, mais celui-ci va permettre d'appliquer ce commit sur notre nouvelle branche ! Il n'est pas nécessaire d'écrire l'identifiant en entier. Seuls les 8 premiers caractères sont nécessaires.**

$ git log

git reset --hard HEAD^

**changer le message de commit**

git commit --amend -m "Votre nouveau message de commit"

git commit --amend -m "Votre nouveau message de commit"

**j’ai oublié un fichier dans mon dernier commit**

git add FichierOublie.txt

git commit --amend --no-edit

**vous avez pushé des fichiers erronés,**Annuler son commit public avec la commande **Git revert**.

git revert HEAD^

git revert  pour annuler des changements apportés à une branche publique, et git reset  pour faire de même, mais sur une branche privée.