C

git

# **Exercice 1: Introduction à Git.**

Un dépôt Git permet de:

* garder toutes les versions successives des fichiers du dossier
* partager ces fichiers avec des collègues
* gérer les conflits d’édition sur ces fichiers
* gérer des dépôts distants
* et bien d’autres choses…

Dans cet exercice, nous allons commencer à voir le premier.

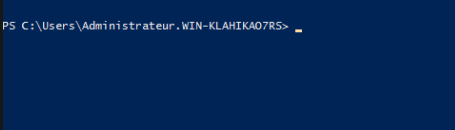
Avant de créer un dépôt Git dans notre espace de travail (« workspace »), il faudra s’assurer d’avoir Git sur sa machine et de maîtriser quelques notions sur la ligne de commande

## **Introduction au terminal.**

Si vous avez Windows, alors ouvrez Windows Powershell, qui inclut les commandes introduites dans ce paragraphe. Pour trouver Powershell sur Windows, faites une recherche dans la barre de recherche en bas à gauche.

|  |
| --- |
|  |

Vous voyez une ligne avec le nom de la machine suivi de d’un curseur.Cette ligne est l’invite de commande .



## **Lignes de commandes.**

chaque commande Linux a un nom:

|  |  |
| --- | --- |
| ls | pour afficher (**l**i**s**ter) des fichiers et dossiers |
| pwd | pour afficher le dossier courant (**p**rint **w**orking **d**irectory) |
| cd | pour **c**hanger de **d**ossier |

Certaines commandes ont besoin d’arguments. Par exemple, pour changer de dossier, il faut savoir où aller. On ajoute donc des arguments à la commande :

Ouvrir un répertoire et créer un dossier dans le repertoire nommé EXO1, ensuite utiliser la commande cd votrerepertoire pour aller vers le le votre répertoire.

Q1) Taper : cd ..↵

commenter :

|  |
| --- |
| Cd.. permet de sortir du fichier actuel et de repasser au fichier d’avant |

Q2) Tapez pwd↵

commenter :

|  |
| --- |
| Pwd sert afficher le chemin d’accès a un fichier |

Q3) tapez ls↵

commenter :

|  |
| --- |
| Ls permet d’afficher tout les documents présent dans un fichier |

Q4: comment faire la différence entre un fichier et un dossier dans la list avec la commande ls↵

commenter :

|  |
| --- |
| Les dossiers sont afficher avec un d----- alors que les fichiers sont afficher avec -a---- |

## **Installation de GIT**

[Téléchargez Git pour Windows (site en anglais)](https://git-scm.com/download/win) où vous pouvez choisir soit l’exe si vous savez où le mettre, soit un setup automatique qui va gérer tout ça pour vous

## **Configuration de GIT**

Sur le dossier TP1 avec la commande powershell tapez les commandes suivantes

1- Obtenir la version de git

|  |
| --- |
| git --version |

2- Configurer votre nom d’utilisateur sur le projet tp1

|  |
| --- |
| git config --global user.name "Your Name" |

3-Configurer votre e-mail sur le projet.

|  |
| --- |
| git config --global user.email <your email address> |

4-Configuration du proxy(ne pas faire) :

|  |
| --- |
| git config --global http.proxy [http://10.0.0.1:80](http://10.0.0.1/) |

5-Visualiser votre configuration avec la commande suivante:

|  |
| --- |
| git config --list |

/!\ Mettre le résultat sur votre compte rendu

$ git config --list

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

user.name=Oxidayzz

user.email=couton.joh@gmail.com

## **Commandes de base GIT**

|  |
| --- |
| Cette partie du TP est à rendre à la fin de la séance. |

1-Créer une fichier index.htm dans le repertoire TP1

2-Ouvrir le fichier avec vsCode ou block-note

3-Copier coller le code suivant

|  |
| --- |
| <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre </h1>  </body>  </html> |

Il faut se placer dans le répertoire en question pour faire le travail suivant:

soit avec powershell comme suit:



ou bien avec le terminal de vs code comme suit:

|  |
| --- |
|  |

4-initialisation de votre dépôt avec la commande suivant :

|  |
| --- |
| git init |

**mettre le résultat sur le document réponse.**

5- visualiser le statut de votre dépôt avec la commande suivante :

|  |
| --- |
| git status |

**mettre le résultat sur le document réponse et faire une recherche sur les mots clés “branch master” et “commit”.**

6- Ajouter tous les fichiers du répertoire sur votre dépôt GIT avec la commande suivant:

|  |
| --- |
| git add . |

**faire un ” git status” et commenter le résultat ainsi que le changement de la couleur du nom de votre document.**

|  |
| --- |
|  |

7- Attribuer un commentaire à une étape de votre projet avec la commande “git commit”.

le git commit est une capture de votre travail au moment du commit qui va s’enregistrer dans le dépôt git

|  |
| --- |
| git commit -m "premierecommit" |

**Mettre le résultat sur votre compte rendu**

**Refaire un ” git status” et commenter ce résultat en faisant votre recherche sur internet .**

8-Visualiser vos dépôts sur git avec la commande suivante:

|  |
| --- |
| git log --oneline |

**Mettre le résultat sur votre compte rendu et commenter ce résultat en faisant votre recherche sur internet . Donner l’adresse de votre dépôt. Ensuite écrire seulement “git log” sur votre commande et noter le résultat.**

|  |
| --- |
|  |

9-Ajouter un paragraphe a votre index.htm et enregistrer

|  |
| --- |
| <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre</h1>  <p>C’est un paragraphe</p>  </body>  </html> |

10- Créer un dossier dans votre répertoire nommé “template” .

11- Dans le dossier template créer un fichier “test.html” le code ci-dessous.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre</h1>  <p>c’est un paragraph</p>  <p>c’est un deuxième paragraph</p>  </body>  </html> |

12-faire un ‘git status’

**commenter le résultat sur le document réponse.**

13-faire un “git add .”

14-faire un “git commit -m “deuxiemecommit”

15-Modifier votre fichier index.html comme suit:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre</h1>  <p>C’est un paragraph</p>  <p>Je suis un autre paragraph</p>  </body>  </html> |

16- Faire un git add

## **Revenir sur un code précédent (checkout).**

1-faire un ‘git log --oneligne’ et noter le code et la première et la deuxième commit

|  |  |
| --- | --- |
| code commit 1 | <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre</h1>  <p>c’est un paragraph</p>  <p>c’est un deuxième paragraph</p>  </body>  </html> |
| code commit 2 | <!DOCTYPE html>  <html>  <head></head>  <body>  <h1>C’est un titre</h1>  <p>C’est un paragraph</p>  <p>Je suis un autre paragraph</p>  </body>  </html> |

2- revenir sur le index.html de la première commit en tapant le code suivant sur la console

|  |
| --- |
| git checkout le code de la deuxième commit index.html |

3- Pour annuler l'effet de l'opération précédente et restaurer index.html dans son état à la fin du troisième commit, tapez:

|  |
| --- |
| git reset HEAD index.html |

ensuite :

|  |
| --- |
| git checkout -- index.html |

|  |
| --- |
|  |

## 

2-TRAVAIL À FAIRE .

|  |  |
| --- | --- |
| Pour réaliser le diagramme, utiliser le lien ci-dessous.  <https://app.diagrams.net/> | **Dans ce tp vous allez chercher comment déposer et importer votre travail sur GITHUB**  1- Ouvrir un compte GITHUB  <https://github.com/>  2-Chercher comment créer un dépôt (repository)  2-Créer votre dépôt privé.  3-prendre le lien du dépôt.  4-chercher la commande git pour configurer les dépôts à distance.  5-chercher la commande qui permet de déposer votre travail  6-chercher comment visualiser vos commits sur github  7- créer un nouveau dossier sur votre ordinateur nommé ‘clone’  8-chercher la commande qui permet d’importer votre travail de github au dossier ’clone’.  9-compléter le diagramme ci-dessous : |

3-LES BRANCHES

1-Créer une nouvelle branche **my-css** avec la commande suivante .

|  |
| --- |
| git branch my-css |

**2-Faire un** **git branch** **et afficher les résultat sur le document réponse**

3-Se positionner sur la branche my-css avec la commande suivante:

|  |
| --- |
| git checkout my-css |

4-Mettre le code css suivant après la balise head :

|  |
| --- |
| <style type="text/css">  body {  background-color: #bfd0d390;  }  </style> |

**5-Faire un git commit avec le message background.**

6-Revenir à la branche master avec la commande suivante :

|  |
| --- |
| git checkout master |

**7-Rafraîchir la page Web et commenter sur le document réponse.**

8-Revenir sur la branche my-css et ajouter le code suivant à votre css.

|  |
| --- |
| h1 {  font-size: 100;  text-align: center;  color:#4b080890  } |

9-faire un git commit avec le message “h1”.

10-Revenir sur la branche master et enlever le paragraphe (balise p).

**11-Faire un git commit avec le message “changement”**

12-Comparer les deux branches avec la commande suivant:

|  |
| --- |
| git diff master..my-css |

**13-Mettre le résultat sur document réponse.**

14-Rapatrier la branche my-css dans master avec la commande suivante :

|  |
| --- |
| git diff master..my-css |

**15-faire un git log --oneline et mettre le résultat sur le document réponse.**

**16-Déposer le travail sur github et mettre le lien sur le document réponse.**

**git remote add origin https://github.com/user/repo.git**

**git remote -v**

**git push origin master**

**4-TRAVAIL A FAIRE.**

1- Chercher l'intérêt d’utiliser la commande git pull avant de faire un git push.

2-Cliquer sur le lien ci-dessous:

<https://git-school.github.io/visualizing-git/#free-remote>

Refaire toutes les étapes des branche avec cet outil de visualisation et faire une capture d'écran de votre travail .

Ouvrir votre dépot github aller sur l’onglet  ensuite  et c²omparer le graph que vous avez réalisé avec le graph sur github.

3-Visualiser la vidéo suivant <https://www.youtube.com/watch?v=V6Zo68uQPqE> de 46:16 a 1:09.

a- Expliquer les indicateurs Star et Fork.

Sur **GitHub**, ces deux indicateurs servent à évaluer la popularité et l’utilisation d’un projet open source.

b- Comment choisir les meilleurs librairies sur github pour faire votre projet ?

regardez le nombre d’étoiles, la derniere mise a jour, nombre de forks et la documentation

.

c-Expliquer comment travailler en collaboration sur un projet sur github.

Crée un repo, crée plusieurs branche, faire des push et des merges afin de réunir la vdef sur le main