TP2 Introduction à Git

Durée : 4 heures maximum

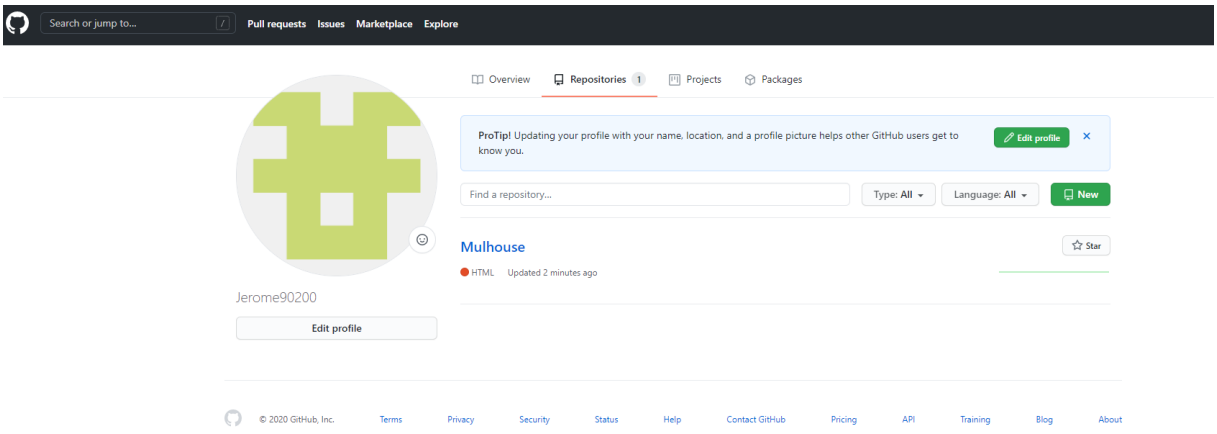
Introduction : Afin de faire ce TP, 2 solutions s'offre à vous :

- Utiliser mon repository existant
- Créer votre propre repository (sur github par exemple https://github.com/join?source=login)

Exercice 1 : Mise en place d'un dépôt git

- Dans un premier temps, vous allez placer l'ensemble de vos fichiers du TP1 dans un dossier les regroupant. Ensuite, vous allez ouvrir un invité de commande. Grâce à la commande cd (commande directory), placez-vous sous le dossier que vous venez de créer. Pour info, lorsque je veux monter dans un dossier, il faut faire « cd monDossier ». Lorsque je veux descendre d'un dossier, il faut faire « cd .. ».
- Une fois à la racine de votre dossier, vous allez pouvoir entre la commande echo "# Mulhouse" >> README.md » Cela va permettre de créer un fichier readme pour le projet Mulhouse de mon git.

```
C:\Users\User\test_mail>echo "# Mulhouse" >> README.md
```



- Pour initialiser le repository en local, il faut taper la commande « git init »

```
C:\Users\User\test_mail>git init
Initialized empty Git repository in C:/Users/User/test_mail/.git/
```

- Ensuite, vous allez ajouter les fichiers contenu dans votre dossier au repository git. Pour cela, taper la commande « git add LICENSE » et « git add . ».

```
C:\Users\User\test_mail>git add .
```

- Vous allez ensuite pouvoir créer la premiere version de votre repository GIT. Pour cela, il suffit d'entrer « git commit –m « Ma première version » »

```
C:\Users\User\test_mail>git commit -m "First commit"
[master (root-commit) fe205d6] First commit
 7 files changed, 336 insertions(+)
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/test_mail.iml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .idea/workspace.xml
 create mode 100644 test-iframe.html
 create mode 100644 test-mail.html
```

- La version étant créée, il faut maintenant la lier à une branch (car tout est une histoire de branch sur git). Pour cela, nous allons en créer une avec la commande « git branch -M main »

```
C:\Users\User\test_mail>git branch -M main
```

- L'objectif va maintenant de lier nos fichier avec le repository distant. Pour cela, il faut entre la commande « git remote add origin https://github.com/Jerome90200/Mulhouse.git » Vous pouvez changer mon repository avec votre repository personnel.

```
C:\Users\User\test_mail>git remote add origin https://github.com/Jerome90200/Mulhouse.git
```

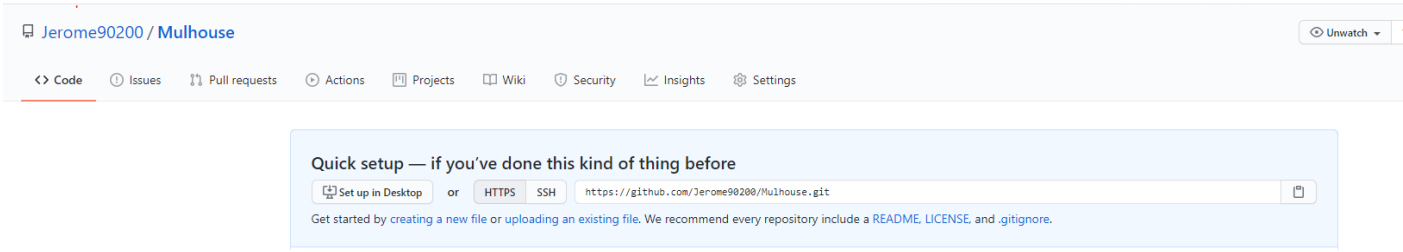
- Enfin, il ne vous restera plus qu'à envoyer vos fichier sur la branch de votre repository distant grâce à la commande « git push -u origin main »

```
C:\Users\User\test_mail>git push -u origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 5.46 KiB | 931.00 KiB/s, done.
Total 13 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Jerome90200/Mulhouse.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Exercice 2 : Cloner un repository existant

Pour cloner un repository existant sur votre machine en local, il vous suffit de taper la commande « git clone https://github.com/Jerome90200/Mulhouse.git »

L'url ci-dessus est celle du repository (voir capture d'écran)




```
C:\Users\User>git clone https://github.com/Jerome90200/Mulhouse.git
Cloning into 'Mulhouse'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 13 (delta 2), reused 13 (delta 2), pack-reused 0
Unpacking objects: 100% (13/13), 5.44 KiB | 40.00 KiB/s, done.
```

Vous avez maintenant le dossier Mulhouse qui s'est créé sur votre ordinateur en local. Pour vérifier (sous windows) :

```
C:\Users\User>dir Mulhouse
 Le volume dans le lecteur C n'a pas de nom.
 Le numéro de série du volume est ECE5-EA94

 Répertoire de C:\Users\User\Mulhouse

19/11/2020  23:07    <DIR>          .
19/11/2020  23:07    <DIR>          ..
19/11/2020  23:07    <DIR>          .idea
19/11/2020  23:07                15 README.md
19/11/2020  23:07               477 test-iframe.html
19/11/2020  23:07             9 487 test-mail.html
               3 fichier(s)            9 979 octets
               3 Rép(s)  74 457 317 376 octets libres
```

Vous pouvez maintenant entrer dans le dossier de développement :

```
C:\Users\User>cd Mulhouse

C:\Users\User\Mulhouse>
```

Et vous avez accès au document du dépôt git.

```
C:\Users\User\Mulhouse>dir
 Le volume dans le lecteur C n'a pas de nom.
 Le numéro de série du volume est ECE5-EA94

 Répertoire de C:\Users\User\Mulhouse

19/11/2020  23:07    <DIR>          .
19/11/2020  23:07    <DIR>          ..
19/11/2020  23:07    <DIR>          .idea
19/11/2020  23:07                15 README.md
19/11/2020  23:07               477 test-iframe.html
19/11/2020  23:07             9 487 test-mail.html
               3 fichier(s)            9 979 octets
               3 Rép(s)  74 455 175 168 octets libres
```

Si je souhaite modifier un fichier et le partager à mon groupe :

- Je modifie mon fichier en local dans le dossier du projet
- Ensuite je vais indexer les fichiers différents du répertoire git actuel grâce à la commande « git add . »
- Les fichiers étant indexés, je vais maintenant créer la nouvelle version en y ajoutant mes modifications avec la commande « git commit –m « ce_que_je_viens_de_modifier » »
- Enfin, il ne me reste plus qu'à envoyer mes modifications sur le répertoire git via la commande « git push »

Si l'un de mes camarades modifie de répertoire git, j'utilise la commande git pull pour mettre à jour

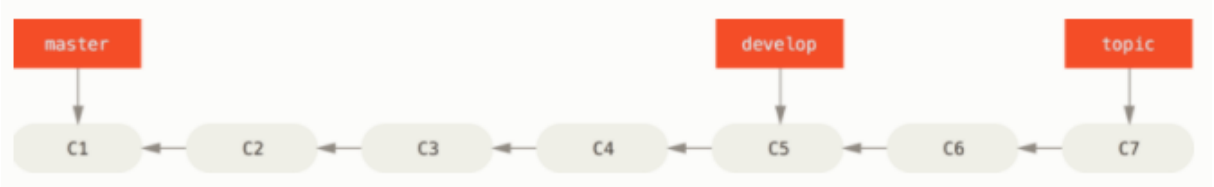
Exercice 3 : Mise en pratique

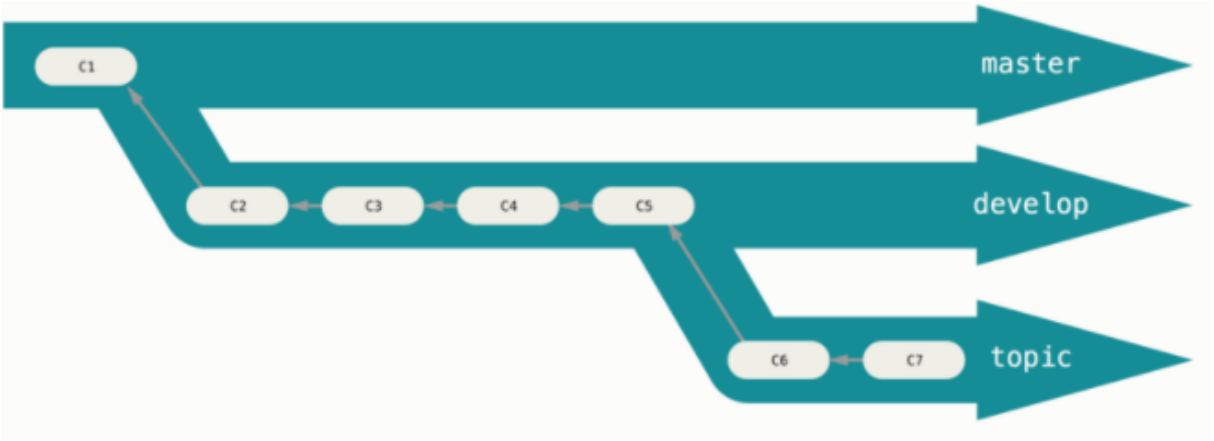
En groupe de 2 ou 3, vous allez chacun créer un repository sur votre compte github. Vous demanderez ensuite aux autres de cloner votre projet.

Ensuite, chacun d'entre vous ajoutera un fichier sur le repository. Vous pourrez ensuite modifier ses fichiers et envoyer vos modifications sur le serveur.

Exercice 4 : Les branches git

Suivi d'un commit dans les différentes branches





L'objectifs des branches git est d'avoir différents environnement de travail (branche master équivalent au code en prod, branche develop équivalent au code qui est terminé par le développeur et qui doit être testé par le chef de projet (souvent sur un serveur de pré-prod) et les branches topic qui sont les branches de travail des développeurs).

Pour créer une branche, il faut avant toutes choses effectuer la commande « git pull » pour avoir un local à jour avec le répertoire git.

```
C:\Users\User\BoatOn\boaton-pro-public>git pull
```

Ensuite vous pouvez créer votre nouvelle branche en local avec la commande git checkout -b [nom de votre branche]

```
C:\Users\User\test_mail>git checkout -b develop
Switched to a new branch 'develop'
```

Votre branche étant maintenant créée en local, il faut l'envoyer sur le répertoire git. Pour cela, il faut utiliser la commande « git push origin [nom de votre branche] ».

```
C:\Users\User\test_mail>git push origin develop
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/Jerome90200/Mulhouse/pull/new/develop
remote:
To https://github.com/Jerome90200/Mulhouse.git
 * [new branch]      develop -> develop
```

A noter que pour effectuer un changement de branches déjà créés, il suffit d'exécuter la commande « git checkout [la branche sur laquelle je veux aller] »

```
C:\Users\User\test_mail>git checkout main
Switched to branch 'main'
M       .idea/workspace.xml
Your branch is up to date with 'origin/main'.
```

Exercice 5 : Mise en pratique

Sur le répertoire git que vous avez créé, vous allez mettre en place un environnement de travail comprenant : une branche master, une branche develop et 3 branches de travail (vous pouvez les nommer comme vous le souhaitez). Ces 3 branches doivent toutes contenir le même code.

Vous allez ensuite ajouter des commentaires sur plusieurs fichiers html, en créer une version et l'envoyer sur une de vos branches de travail.

```
C:\Users\User\test_mail>git add .
warning: LF will be replaced by CRLF in .idea/workspace.xml.
The file will have its original line endings in your working directory

C:\Users\User\test_mail>git commit -m "new modif"
[develop 1c1f049] new modif
 2 files changed, 6 insertions(+), 4 deletions(-)

C:\Users\User\test_mail>git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin develop


C:\Users\User\test_mail>git push origin develop
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 598 bytes | 299.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jerome90200/Mulhouse.git
   4e8b26b..1c1f049  develop -> develop
```

Exercice 6 : Les merges (ou fusions en français) => envoyer le travail d'une branche à l'autre

Pour envoyer le travail d'une branche à l'autre, il faut tout d'abord bien avoir envoyé son travail sur le répertoire git (avec la commande git push) et avoir effectué la commande git pull pour récupérer les modifications faites par les autres membres du groupes.

Ensuite, il faut vous placer sur la branche sur laquelle vous souhaitez envoyer votre travail avec la commande « git checkout [la branche sur laquelle je souhaite me placer] »

```
C:\Users\User\test_mail>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Vous pouvez maintenant effectuer la fusion avec la commande « git merge [la branche sur laquelle est le travail que je veux fusionner] »

```
C:\Users\User\test_mail>git merge develop
Updating 4e8b26b..1c1f049
Fast-forward
 .idea/workspace.xml | 8 +++++---
 test-iframe.html    | 2 +-
 2 files changed, 6 insertions(+), 4 deletions(-)
```

Exercice 7 : Mise en pratique

Vous allez maintenant envoyer votre modification faites sur votre branche de travail sur la branche develop et ensuite, vous enverrez vos modifications qui sont actuellement sur la branche develop sur la branche master.

Exercice 8 : Mise en pratique finale

Par groupe de 2 ou de 3, vous allez créer un nouveau répertoire sur votre espace github (ou autres) qui va s'appeler « Cours Intégration web ».

Une fois le dépôt initialisé, vous allez créer 2 dossiers dans ce répertoire, un dossier « TP1 » et un dossier « correction TP1 » qui contiendront respectivement vos exercices et les corrigés que je vous ai fourni. Vous pouvez ensuite envoyer ses fichiers dans sur votre répertoire git. Vous allez ensuite mettre en place un espace de développement avec les branches : master, dévelop, feature1 et feature2. Chacune de ses branches devra avoir les mêmes fichiers et dossiers.

Vous allez ensuite créer un nouveau dossier à partir de la branche feature1 que vous allez nommer TP_CSS. Vous pouvez maintenant mettre à jour chacune de vos branches avec ce dossier.