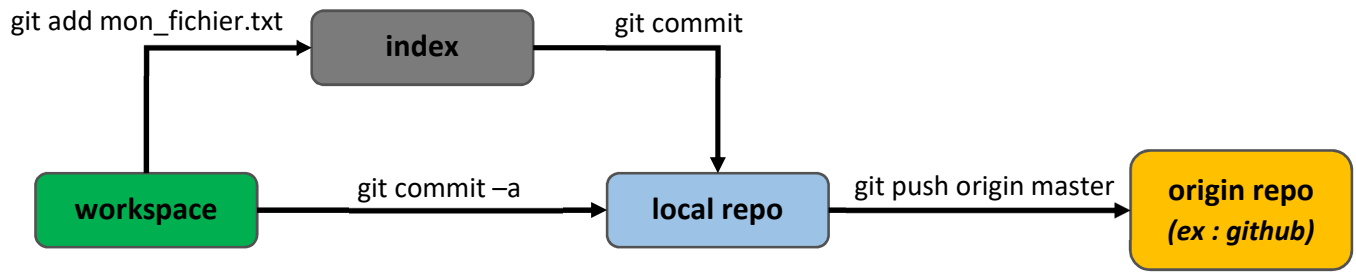
Git & GitHub - Partie 3 Activité

Introduction : schema

Commençons par différencier les différentes commandes/étapes pour valider un travail local et le mettre dans un répertoire partagé (partagé par exemple sur gitHub).

Voici un schéma simplifié pour les trois commandes : add/commit/push




```
$console : git add mon_fichier.txt
```

Cette commande va ajouter à l'index *(liste des fichiers modifiés)* le fichier mon_fichier.txt afin de préciser à git que ce fichier sera dans la liste à commit, puis peut-être dans la liste à push.

⇨ Maintenant Git sait que les modifications apportées à mon_fichier.txt feront partie du prochain commit. Il est prêt à être commit.

```
$console : git commit –m "Message d'explication des changements"
```

Cette commande va versionner notre travail sur tous les fichiers de l'index *(juste mon_fichier.txt pour le moment)*. En local, le projet est à jour. On y insère un message d'explications avec le paramètre -m

⇨ Maintenant Git connait notre dernière version « commitée »

```
$console : git push origin master
```

Cette commande va envoyer dans le répertoire origin (GitHub) tout le travail qui avait été indexé par « add » et validé en local par « commit ».

Qu'est-ce qu'un commit

Un « *commit* » est un état précis d'enregistrement des changements dans un projet référencé par Git en local. Le commit va donner un identifiant unique (le hash SHA) à cet enregistrement qui va servir de version. Ainsi un utilisateur pourra retrouver les changements proposés en fonction de la version du commit.

NOTE : un commit peut être un changement d'un mot ou carrément d'un dossier, d'un lib, etc.

À quoi sert la commande git log

```
$console : git log
```

Cette commande permet d'afficher la liste des derniers commits et donne pour chacun :

- Le hash identifiant SHA
- Auteur de la modification
- Date
- Message du commit (-m "Message d'explication des changements")

Il s'agit de l'historique des modifications. Pour en voir une précisément, il faut taper :

```
$console : git show identifiantHash
```

Où identifiantHash est le hash trouvé en tapant "git log".

Qu'est-ce qu'une branche

Une branche est une dérivation du projet principal afin de pouvoir travailler sans interférer avec les autres collaborateurs du même projet principal. La branche est un workspace séparé dans lequel il est possible d'apporter diverses évolutions sans perturber le développement de la partie principale du code.

On crée la branche, on se place dessus :

```
$console : git branch ma_branche
$console : git checkout ma_branche
```

Le workspace sera alors à l'image de la branche mais le projet principal master ne subira pas les changements sauf en cas de fusion !
Après les éventuels commit sur la branche, on se replace sur master (projet principal) :

```
$console : git checkout master
$console : git merge ma_branche
```