

Utilisation de git via le terminal

Version du 3 mars 2022

Utiliser un terminal

Utiliser git à partir du terminal permet d'avoir une méthode indépendante du langage et du système d'exploitation. Dans le cadre de ce guide, le seul concept requis par rapport au terminal est le `cd` (choose directory). La commande `cd` permet de parcourir des dossiers un peu comme on le ferait dans un explorateur de fichier. Dans la situation où le terminal est ouvert dans le chemin suivant: `"C:\Users\osamu>"`, et que nous voulons entrer dans le dossier Downloads, la commande à effectuer serait `"cd Downloads"`. Le terminal serait désormais dans le dossier `"C:\Users\osamu\Downloads>"`. Pour revenir dans le dossier "osamu", il faut simplement effectuer la commande `"cd .."`. Il est donc possible de se déplacer d'un dossier à l'autre. Lorsqu'on veut faire un grand bond d'un seul coup, il est possible de donner le chemin absolu. Considérant l'ordre de dossier suivant; `"C:", "Users", "osamu", "PycharmProjects", "progfest"`, il est possible de directement voyager à ce dossier particulier en effectuant la commande `"cd C:\Users\osamu\PycharmProjects\progfest"`. Le terminal passera donc de `"peu\importe\le\path>"` à `"C:\Users\osamu\PycharmProjects\progfest"`. Un path absolu dans Windows débute par `"C:"` tandis que sur Linux et mac, ça débute par `"/"`. Pour voir tous les fichiers présents dans le dossier où se trouve le terminal, il faut simplement faire la commande `"dir /a"` sur Windows et `"ls -a"` sur Linux et mac.

Utiliser les commandes git de base dans le terminal

En considérant un emplacement de terminal quelconque `"absolute\path>"` en effectuant la commande

```
git clone <github_url>
```

Un dossier ayant le nom du projet sera créé dans le dossier "path" qui lui est dans le dossier "absolute". Dans le nouveau folder créé, il y aura tout votre code et un fichier `.git` (visible que par la commande `dir /a` ou `"ls -a"` dans le terminal). Ce fichier `.git` sert à stocker toutes les informations relatives à git concernant le projet. Pour que les commandes git du terminal opèrent sur votre projet, il faut s'assurer que le terminal opère dans le même dossier que le fichier `.git` (qui devrait être visible en effectuant la commande `dir /a` ou `"ls -a"`). Avant de faire quoi que ce soit d'autre, il est bon d'effectuer un "pull" avant toute séance de développement. Pour ce faire, il faut simplement effectuer la commande suivante:

```
git pull
```

Une fois que c'est fait, et que certaines modifications ont été apportées au code, il est nécessaire de spécifier quelles modifications doivent être ajoutées dans une commit en utilisant la commande `"add"`. Avant de faire la commande `"add"`, c'est une bonne pratique de faire le fichier `.gitignore` afin que seulement les choses non ignorées soient ajoutées avec la commande `"add"`. Pour ajouter l'ensemble des modifications sauf ce qui est dans le `.gitignore`, il faut faire la commande:

```
git add *
```

Une fois que c'est fait, il faut simplement faire le commit avec le message associé en effectuant la commande suivante:

```
git commit -m "message de commit"
```

La seule étape restante est de "push" les commits sur le serveur git en effectuant la commande:

```
git push
```

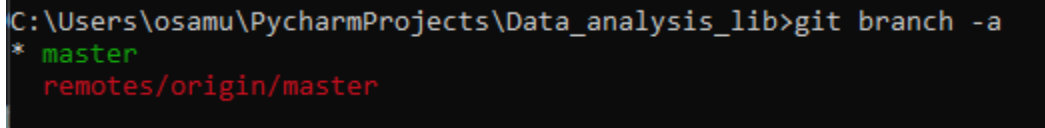
Travailler avec les branches

La prochaine étape est d'intégrer les branches aux développements. La première étape est d'être en mesure de voir les différentes branches. Pour voir l'ensemble des branches, il faut effectuer la commande:

```
git branch -a
```

Les branches qui sont sur le serveur git sont écrites en rouge tandis que les branches présentes localement sont en vert. L'étoile indique sur quelle branche vous travaillez actuellement. Pour créer une nouvelle branche localement, il faut simplement effectuer la commande

```
git branch <nom_de_la_branche>
```



```
C:\Users\osamu\PycharmProjects\Data_analysis_lib>git branch -a
* master
remotes/origin/master
```

Figure 1: Exemple d'affichage de la commande "git branch -a"

Pour que la nouvelle branche créée se retrouve aussi sur le serveur git, il faut effectuer un "push":

```
git push --set-upstream origin <nom_de_la_branche>
```

En suite, il faut sélectionner la branche en effectuant

```
git checkout <nom_de_la_branche>
```

Après cette étape, l'étoile devrait être sur la branche désirée lorsqu'on effectue "git branch -a". Lorsque le checkout est fait, il faut simplement travailler comme indiqué dans la section précédente de ce document. C'est seulement lorsque le développement de la fonctionnalité associée à la branche est terminé que le merge s'impose. Pour merger la branche x dans la branche y, il faut premièrement sélectionner la branche y en effectuant un checkout,

```
git checkout y
```

Une fois sur la branche y, il est possible de merger n'importe quelle des autres branches dans la brancher actuel. Dans ce cas nous voulons merger la branche x. La commande est donc:

```
git merge x
```

Les mises à jour de la branche x se sont donc ajoutées à la branche. La dernière étape est d'envoyer les dernières étapes effectuées sur le serveur git avec un "push":

```
git push
```

Si la branche x n'est plus utile, il est possible de l'effacer avec la commande:

```
git branch -d x
```

Pour ensuite effacer la branche x sur le serveur git,

```
git push origin --delete x
```