Git : Maîtriser git avec la ligne de commandes

Git est un outil de versionning qui s'utilise principalement en ligne de commandes. Tout d'abord assurez vous d'avoir installer git

Configurer git

Avant de commencer à versionner un projet git, il est obligatoire de configurer notre git. En effet, pour fonctionner, git à besoin de vous connaître (de savoir qui vous êtes).

Pour cela il suffit les commandes suivante pour dire à git qui nous sommes, nous donner une identité :

```
$ git config --global user.email mon-email.com
$ git config --global user.name "John john"
```

> Cette étape est obligatoire pour tout le monde, elle ne se fais qu'une seul fois. Votre email et user.name est entièrement libre ! À vous de choisir ce que vous souhaitez

> **Attention** Ce user.name et user.email aura un impact lors de l'utilisation de « remote » (github, bitbucket, gitlab etc ...)

Démarrer un projet git

Pour démarrer un projet versionné avec git, il faut tout d'abord choisir un dossier contenant le projet sur notre ordinateur.

> Dans ce support, nous allons démarer avec un dossier vide, cependant vous pouvez très bien démarrer un projet git dans un dossier qui n'est pas vide.

Commençons par créer un nouveau dossier et se rendre à l'intérieur :

```
$ mkdir MonProjet
$ cd MonProjet
```

Maintenant que nous avons un dossier vide, nous pouvons démarrer le versionning avec git :

```
$ git init
```

> Un petit peu de vocabulaire, un projet versionné avec git porte le nom **repository** (dépot).

> Un projet git n'est ni plus ni moins **un dossier sur votre ordinateur**

Lorsque vous lancer la commande `git init`, `git` ne fais vraiment pas grand chose:

- Il créé un dosier `.git` qui contiendra, commit, indexes et branches. Ce dossier est uniquement utilisé en interne par git, nous vous amusez pas à éditer, modifier ou ouvrir des fichiers de ce dossier.

> Entrainez-vous ! Créer votre premier projet nommé `MonPremierProjetGit` à l'emplacement de votre choix sur votre ordinateur. Utilisé la ligne de commande git (git bash).

L'index et les commits

Premièrement, commençons par créer un premier fichier :

```
$ touch monfichier.html
```

Il est possible de consulter l'état de l'index avec la commande git :

```
$ git status
```

> Il est possible de lister tout les fichiers y compris dans les sous dossier avec l'option `u` :
>
> ```
> $ git status -u
> ```

> Il n'affiche l'état de l'index c'est à dire : en **rouge** les nouveaux fichiers qui ne sont pas dans l'index en **orange** les fichiers modifié qui ne sont pas dans l'index et en **vert** les fichiers qui sont dans l'index

Pour ajouter un fichier dans l'index il faut utiliser :

```
$ git add nomDuFichier.extension
```

> Il est possible d'ajouter **tout les fichiers les rouge et orange** avec une seul commande:
>
> ```
> $ git add .
> $ git add -A
> ```

Pour enlever un fichier de l'index :

```
$ git reset nomDuFichier.extension
```

> Pour enlever tout les fichier **vert** de l'index :
>
> ```
> $ git reset --mixted
> ```
>
> Le **head** veut dire « espace de travaille en cours »

Maintenant que nous pouvons manipuler l'index, il est possible de créer notre premier étape. C'est à dire de prendre tout les fichiers **vert** (ceux qui sont dans l'index) et en faire un étape de notre projet (**commit**) :

```
$ git commit -m "Premier message"
```

Une fois le commit réaliser, les fichiers indéxé (**vert**) vont disparaitre car ils seront enregistré dans git :).

> **Entrainez-vous** !
> Créer 3 fichier :

> - index.html
> - style.css
> - src/index.js
>
> En utilisant les commande `git status`, `git add` et `git commit` réaliser un premier commit avec le message "Premier commit" avec les fichier `index.html` et `style.css`
>
> En utilisant les même commandes, réaliser un second commit avec le message "Second commit" avec le fichier `src/index.js`

Il est possible de consulter l'historique du projet avec la commande :

```
$ git log --oneline
```

> **Pour quitter l'historique, appuyer sur la touche `q`**

Les branches

Lors de la création d'un projet git avec `git init`, la branche initiale est la branche `master`.

> Il est possible de modifier ce comportement en spécifiant une option à git :
>
> ```
> $ git config --global init.defaultBranch main
> ```

Il est possible de lister toutes les branches d'un projet avec la commande :

```
$ git branch
```

> La branche sur lequel vous vous situez sera afficher en **vert** avec une étoile devant

Pour créer une nouvelle branche et se déplacer sur la branche il faut utiliser la commande suivante :

```
$ git checkout -b nom-de-ma-branche
```

> **Il est essentiel de créer sa propre branches pour tout ajout, modification, supression afin d'éviter tout problèmes de colisions**

On peut utiliser la même commande, mais sans l'option `-b` pour changer de branche

```
$ git checkout master
```

Il est aussi possible de supprimer une branche avec la commande :

```
$ git branch -d ma-branche
```

> **Il est fortement déconseillé de supprimer la moindre branche d'un projet !!! Si on supprime une branche, on supprime une partie de l'historique du projet**

L'espace de travaille

Pour pouvoir changer de branche, il faut absolument un espace de travaille vide : C'est à dire que la commande `git status` ne doit afficher aucun fichiers ! Si il existe des fichiers, alors il faudra faire un commit avant toutes opérations sur une branche.

> **Entrainez-vous**
>
> Créer une branche "ma-branche" et vous rendre dessus.
>
> Ajouter et créer un commit avec un nouveaux fichier `README.md`
> Maintenant utiliser la commande `git log --oneline` afin de consulter votre historique
>
> Changer de branche pour la branche master et lancer à nouveau la commande `git log --oneline`.
>
> Qu'est-ce qui à changé entre les deux branches ?

Fusionner des branches

Il est possible de fusionner une branche avec une autre c'est le **merge**

Pour cela, rendez-vous sur la branche qui doit accépter la fusion (hôte) et lancer la commande suivante :

```
$ git checkout master
$ git merge ma-branche
```

> **Généralement, le merge ou fusion est une opération réservé au remote (github, gitlab, bitbucket ...)**
> Il est déconseillé d'utiliser la commande **merge**.