5. Git et historisation

Consulter l'historique des modifications

On peut consulter l'historique des modification sur un projet git avec la commande `git log`.

```
git log
commit 8cc61227d7ac2ed799fe6b18cf5fb044731acf98 (HEAD
-> master)
Author: GabagoolOverHere <vicmartindev@gmail.com>
Date:   Sun Mar 12 15:35:34 2023 +0100

    ajout du fichier 2

commit 7c17df9836b97219be2dd3aad1c83412a6152943
Author: GabagoolOverHere <vicmartindev@gmail.com>
Date:   Sun Mar 12 14:54:26 2023 +0100

    titre
    première version.
```

Cela affiche la liste des commits réalisés du plus récent au plus ancien. On peut également afficher chaque commit sur une seule ligne avec l'option `--pretty`.

```
git log --pretty=oneline
8cc61227d7ac2ed799fe6b18cf5fb044731acf98 (HEAD ->
```

```
master) ajout du fichier 2
7c17df9836b97219be2dd3aad1c83412a6152943 titre
première version.
```

Documentation complète de `git log` ici.

Écraser et remplacer un commit

Pour remplacer un `commit` on peut utiliser l'option `--amend`.

Cela peut-être utile si après avoir le `commit` on se rend compte qu'on aurait oublié un fichier. On peut alors le `git add`, puis faire un `git commit --amend` pour remplacer le dernier commit avec le nouveau contenant le fichier oublié.

Annuler les modifications apportées à un fichier

Pour revenir à l'état du code à une précédente *snapshot*, on peut utiliser la commande `git checkout -- [nom_du_fichier]` ou la commande spécialisée `git restore`.

```
echo 'coucou' >> test.txt
git status
```

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be
committed)
  (use "git restore <file>..." to discard changes in
working directory)
        modified:   test.txt
cat test.txt
coucou
git restore test.txt
git status
On branch master
nothing to commit, working tree clean
```

Ici, on écrit dans le fichier `test.txt` avec la commande `echo`, et avec `git restore`, on retourne à l'état initiale en annulant les changements.

Exercice

> Dans un dossier créer un dossier "git-3"

> Initialiser un dépôt git dans le dossier "git-3" nouvellement créé

> Y créer deux fichiers texte "fichier1.txt" et "fichier2.txt"

> Ajouter "fichier1.txt" et "fichier2.txt" dans la *staging area*

> Commiter ces fichiers avec le message suivant : "commit du fichier1"

> Écrire "coucou" dans "fichier2.txt"

> Lire le contenu de "fichier2.txt" avec une commande `cat`

> Restorer "fichier2.txt"

> Lire à nouveau le contenu de "fichier2.txt"

> Ajouter "fichier2.txt" dans la *staging area*

> Commiter ces fichiers en remplacant le commit précédent avec le message suivant : "commit du fichier1 et du fichier2"

> Effectuer un `git log` avec l'option `--oneline` pour avoir un aperçu de l'historique du dépôt