3. Le commit parfait

Lorsqu'on travaille en groupe il est important d'emettre des messages de `commit` pertinents. Pour cela il est bien vivement conseillé de ne mettre dans un `commit` que les fichiers qui ont un changement qui touche au même sujet.

Répartir ses `commit` en fonction du sujet

La *staging area* va nous permettre de selectionner des fichiers et des parties de fichiers qui sont liés par un même sujet et donc seront rassemblés sous un même `commit`.

Supposons qu'on ait 2 fichiers `test1.txt` et `test2.txt`. Le contenu de `test1.txt` expose un sujet `topic1` et `test2.txt` par expose un sujet `topic2` et `topic1`.

Pour faire un `commit` au sujet du `topic1` on va avoir besoin de `commit` `test1.txt` et une partie de `test2.txt`. On peut le faire en utilisant `git add -p`.

```
mkdir part && cd part
git init
touch test1.txt && touch text2.txt
git add .
```

```
echo 'topic 1' | tee text1.txt text2.txt
echo 'topic2' >> text2.txt
git add -p text2.txt
+topic1
+topic2
(1/1) Stage this hunk [y,n,q,a,d,e,?]? h
y - stage this hunk
n - do not stage this hunk
q - quit; do not stage this hunk or any of the
remaining ones
a - stage this hunk and all later hunks in the file
d - do not stage this hunk or any of the later hunks
in the file
e - manually edit the current hunk
? - print help
@@ -0,0 +1,2 @@
+topic1
+topic2
(1/1) Stage this hunk [y,n,q,a,d,e,?]? e
```

On va pouvoir ici commenter les lignes de codes qui ne sont pas pertinentes pour notre `commit` qui va porter sur le `topic1`.

```
git status
n branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   text1.txt
        new file:   text2.txt
```

```
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:   text1.txt
        modified:   text2.txt
```

Le message de commit parfait

Le sujet doit être concis et clair. Si aucun message de ce type de nous vient à l'esprit, alors les fichiers ajoutés dans votre commit ne sont probablement pas les bons.

En effectuant un `git commit` dans argument ni option, on peut choisir le titre du `commit` et le corps dans lequel on pourra donner plus de détails.

```
git commit
Titre de mon commit
Corps de mon commit dans lequel je vais pouvoir
donner plus de détails.
# Please enter the commit message for your changes.
Lines starting
# with '#' will be ignored, and an empty message
aborts the commit.
#
```

```
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   text1.txt
#       new file:   text2.txt
#
# Changes not staged for commit:
#       modified:   text1.txt
#       modified:   text2.txt
#
```

Exercice

> Dans un dossier créer un dossier "git-2"

> Initialiser un dépôt git dans le dossier "git-2" nouvellement créé

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

> Ajouter ces fichiers dans la *staging area*

> Écrire "sujet 1" dans les deux fichiers avec une commande `echo`

> Écrire "sujet 2" dans "fichier2.txt"

> Ajouter dans la *staging area* les fichiers ou parties de fichiers ne parlant que du "sujet 1"

> Effectuer un `git status` pour constater que "fichier2.txt" est en partie dans la *staging area* et en parti non

> Commiter ce fichier avec le message suivant : "commit de mes fichiers"