? Gérer des univers parallèles

Maintenant on va voir un autre coté de Git qui est la possibilité de gérer plusieurs version du code simultanément avec des "branches".

Créer une nouvelle branche

Imaginons que nous voulons tester de nouvelles fonctionalités sans impacter la branche principale. Pour cela on va créer une nouvelle branche et aller dessus:

```
# "checkout" est utilisé pour voyager entre les différentes branches
# Le flag "-b" est utilisée pour créer une nouvelle branche
git checkout -b ma-super-fonctionalite
```

Maintenant on peut aussi voir la liste des branches et voir sur quelle branche nous sommes actuellement

```
git branch
```

A partir de maintenant tout commit fait ne sera pas enregistré sur la branche principale. Si on veut retourner sur la branche principale on peut alors utiliser un simple checkout

```
git checkout master


# Et puis quand on veut y retourner
git checkout ma-super-fonctionalite
```

Voyager entre les branches

Vous êtes en train de faire votre projet, mais vous voullez aller voir ou rapidement modifier un truc sur la branche principale. Problème : vous n'avez pas encore fait de commit, donc vous ne pouvez pas changer de branche sinon vous allez perdre vos changements.

Git heureusement nous protège de ce sénario et affichera un message d'erreur si on souhaite changer de branche sans commit les changements.

Mais il y a une autre commande que l'on peut utiliser pour foutre tous les changements en cours dans un coin le temps de voyager entre les branches. C'est `stash`

```
# Cette commande va mettre temporairement vos changements dans un coin pour les retrouver plus tard
git stash

# Vous pouvez maintenant changer de branche sans problème
git checkout master
```

Maintenant vous avez fini vos explorations et voulez retourner à votre nouvelle branche et récupérer vos changements:

```
git checkout ma-super-fonctionalite

# "stash pop" pour retrouver les changements qui avaient été mis de coté avec stash
git stash pop
```

Si à l'inverse on souhaite ne pas les retrouver et les supprimer on peut utiliser "stash drop"

```
git stash drop
```

Fusioner des branches

Maintenant si on imagine que l'on a fait nos changements sur notre nouvelle branche et que l'on veut ajouter ces changements à la branche principale. On va utiliser `merge`

```
# Changement vers la branche principale
git checkout master
git merge ma-super-fonctionalite
```

Et en théorie ça foncitonne □□

Mais dans certains cas, quand les deux branches ont des modifications différentes pour un même fichier cela peut créer des conflits.

? Merge conflict

Pour gérer les conflits il y a plusieurs méthodes.

Revision #1
Created 26 April 2023 17:09:22 by SnowCode
Updated 26 April 2023 17:09:23 by SnowCode