

Dyma

Accueil

FORMATION

Formation initiale

COMMUNAUTÉ

Général

Algorithmes

Flutter

Git

HTML&CSS

Java

JavaScript

PHP

React

Symfony

Bugs et corrections

Formations

Vidéo

Valider le cours

Quiz

Leçon suivante

Git > 5. Les branches > 23. Lister et créer des branches avec git branch

3 minutes

Créer une branche

Lister les branches

Modifier le nom d'une branche

Supprimer une branche

HEAD

master

A

B

C

Arbre A

Arbre B

Arbre C

Nous sommes à la version correspondant au **commit C** et nous souhaitons démarrer un nouveau développement qui va prendre du temps, par exemple développer un système d'authentification.

Nous allons donc créer une branche en faisant :

```
git branch auth
```

HEAD

master

A

B

C

Arbre A

Arbre B

Arbre C

auth

Nous avons pour le moment simplement créé un nouveau pointeur sur le **commit C**, qui est notre nouvelle branche.

Nous pouvons vérifier simplement :

```
cat .git/refs/heads/auth
cat .git/refs/heads/master
```

Qui donneront toutes les deux le même `hash`, qui est celui du `commit C` dans notre exemple.

Ces deux références pointent donc exactement sur le même `commit`.

Attention ! Pour le moment `HEAD` pointe toujours sur `master`, nous ne nous sommes pas positionné sur la nouvelle branche `auth`, nous l'avons simplement créée.

D'ailleurs si vous faites :

```
git log
```

Vous verrez bien :

```
commit 9e633d56381d9f0335320f22ccf3f1562d1f80d8 (HEAD -> master, auth)
```

Il est bien indiqué que `HEAD` pointe sur `master`, et que `master` et `auth` pointe sur le `commit`.

Lister les branches

Avec `Git` il est très simple de visualiser toutes les branches locales d'un projet en une commande :

```
git branch
```

[Copier](#)

Si vous ne passer pas de nom, `git branch` va simplement lister tous les noms des branches. Dans notre exemple, nous aurons :

```
auth
* master
```

L'étoile indique que `HEAD` pointe actuellement sur la branche `master`.

Cela signifie que si vous faites des modifications et que vous les sauvegardez dans un `commit`, ce dernier se placera sur la branche `master`.

Vous pouvez également obtenir plus d'informations avec `git branch -v`.

```
git branch -v
```

Cette option vous donnera notamment le début du `hash` du `commit` sur lequel pointe chaque branche, ainsi que le début du message de validation, par exemple :

```
auth    d9d259c nouveau commit sur auth
* master 9e633d5 Premier commi
```

Nous allons voir comment basculer entre les branches dans la prochaine leçon !

Modifier le nom d'une branche

Modifier le nom d'une branche est très simple puisque c'est simplement le nom d'un fichier situé dans `.git/refs/heads`.

Il suffit de faire :

```
git branch -m "nouveauNom"
```

Supprimer une branche

Pour supprimer une branche il suffit de faire :

```
git branch -d branche
```

Si la branche contient des modifications qui n'ont pas été fusionnées (nous verrons en détails la fusion dans les prochaines leçons), il faudra forcer la suppression (mais attention tous les changements et tous les `commits` de la branche seront perdus) :

```
git branch -D branche
```

L'option `-D` est un raccourci pour `--delete --force` et signifie "forcer la suppression de la branche".

