Utilisation de git cherry-pick

Temps de lecture : 2 minutes

La commande `git cherry-pick`

La commande `git cherry-pick` permet de sélectionner un `commit` par référence et de l'appliquer sur la branche pointée par `HEAD`.

Elle permet d'appliquer un `commit` d'une branche sur une autre branche.

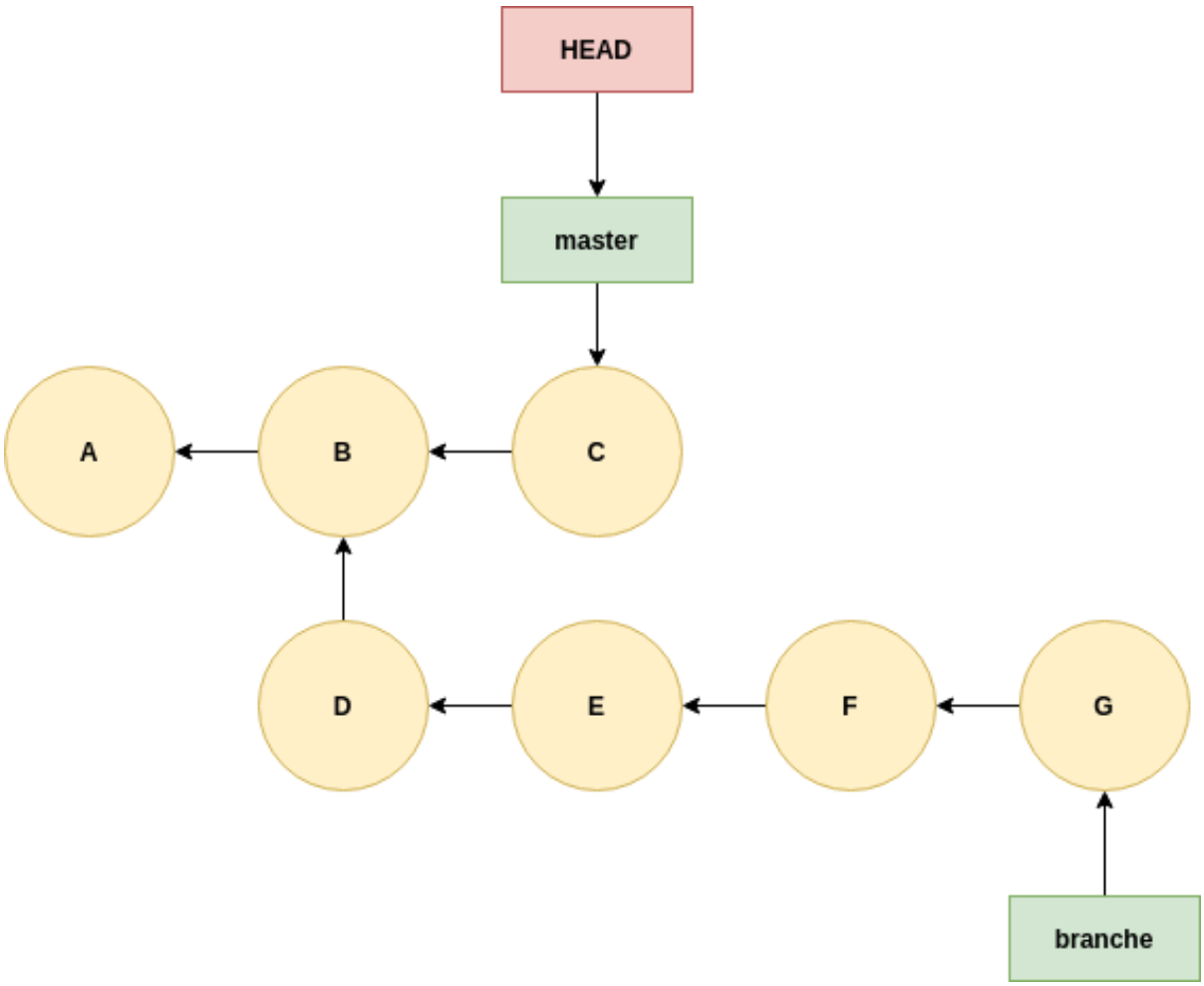
Nous allons maintenant voir des cas d'utilisation de cette commande.

Nous verrons ensuite des cas où il vaut mieux utiliser une autre commande.

Cas d'utilisation de `cherry-pick`

Prenons un exemple de cas d'utilisation de la commande `git cherry-pick`.

Prenons la situation suivante :



Nous voulons récupérer les modifications du `commit D` sur `main` et ne pas conserver le `commit E`. Nous pourrions faire un rebasage interactif, mais un moyen plus rapide est :

```
git cherry-pick D
```

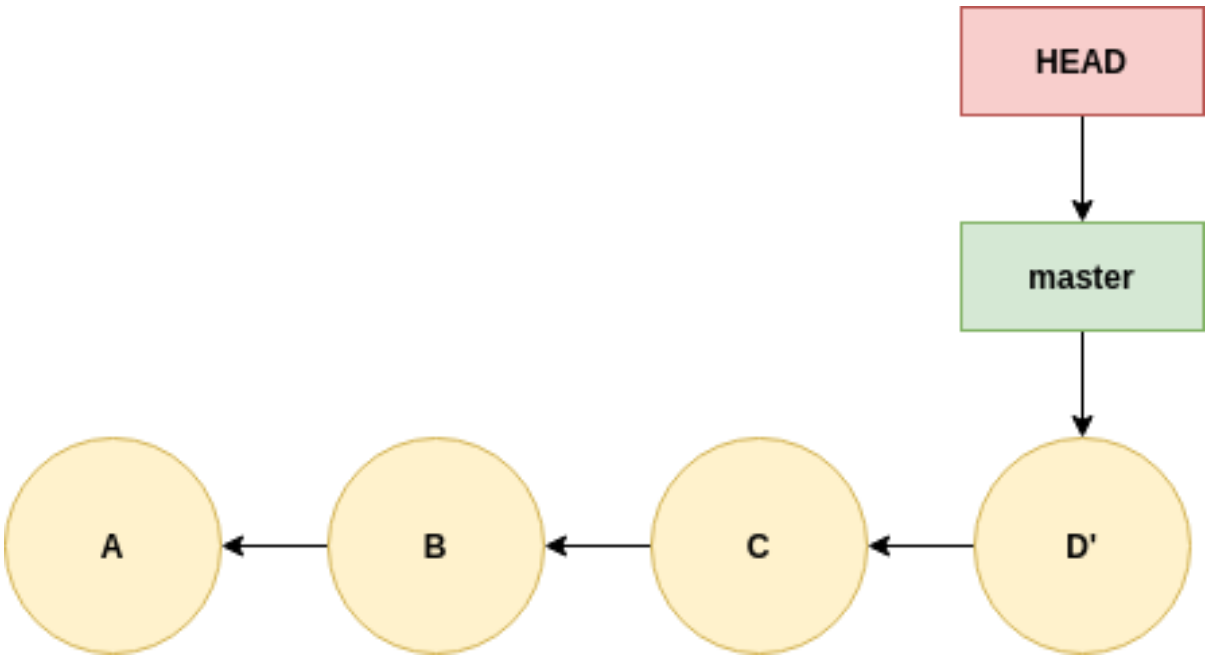
Où `D` est le `hash` du `commit D`.

Un nouveau `commit` contenant les changements du `commit D`, que nous appellerons `D'` est créé sur `main`.

Nous pouvons ensuite abandonner la branche :

```
git branch -D branche
```

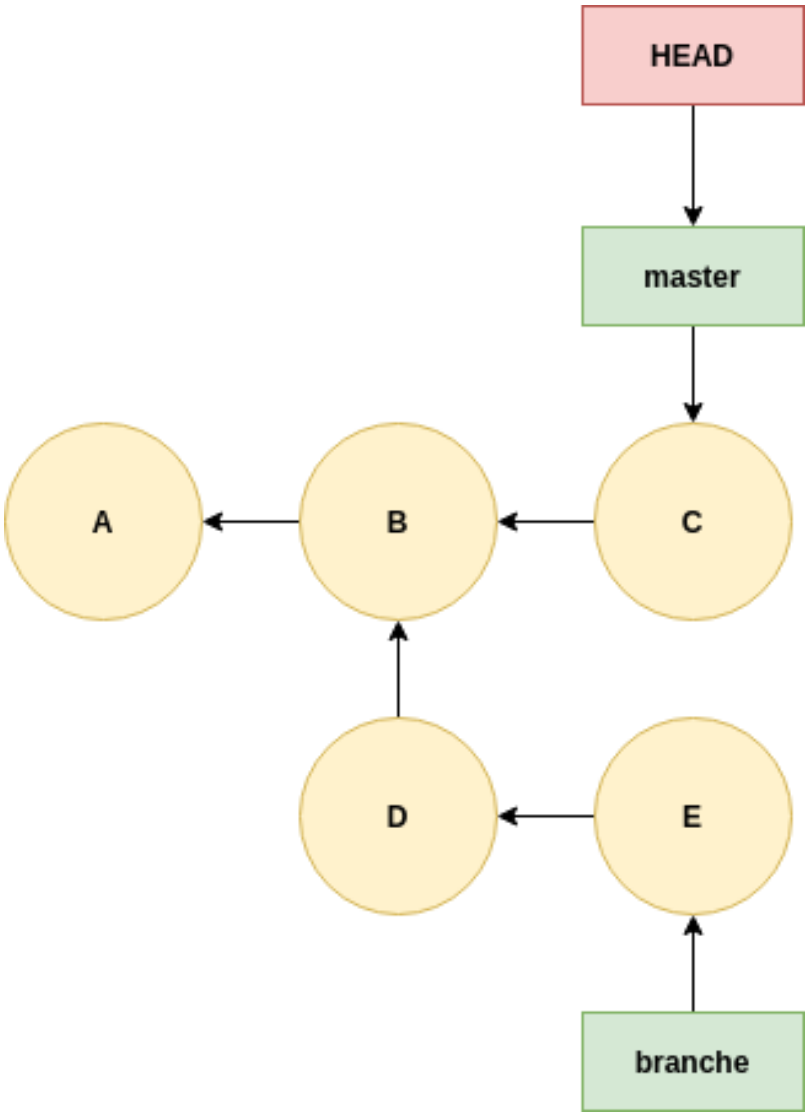
Nous avons enfin :



`git cherry-pick` peut donc être utilisé lorsque vous voulez effectuer un rebasage d'un `commit` unique.

Autre cas d'utilisation de `cherry-pick`

Prenons la situation suivante :



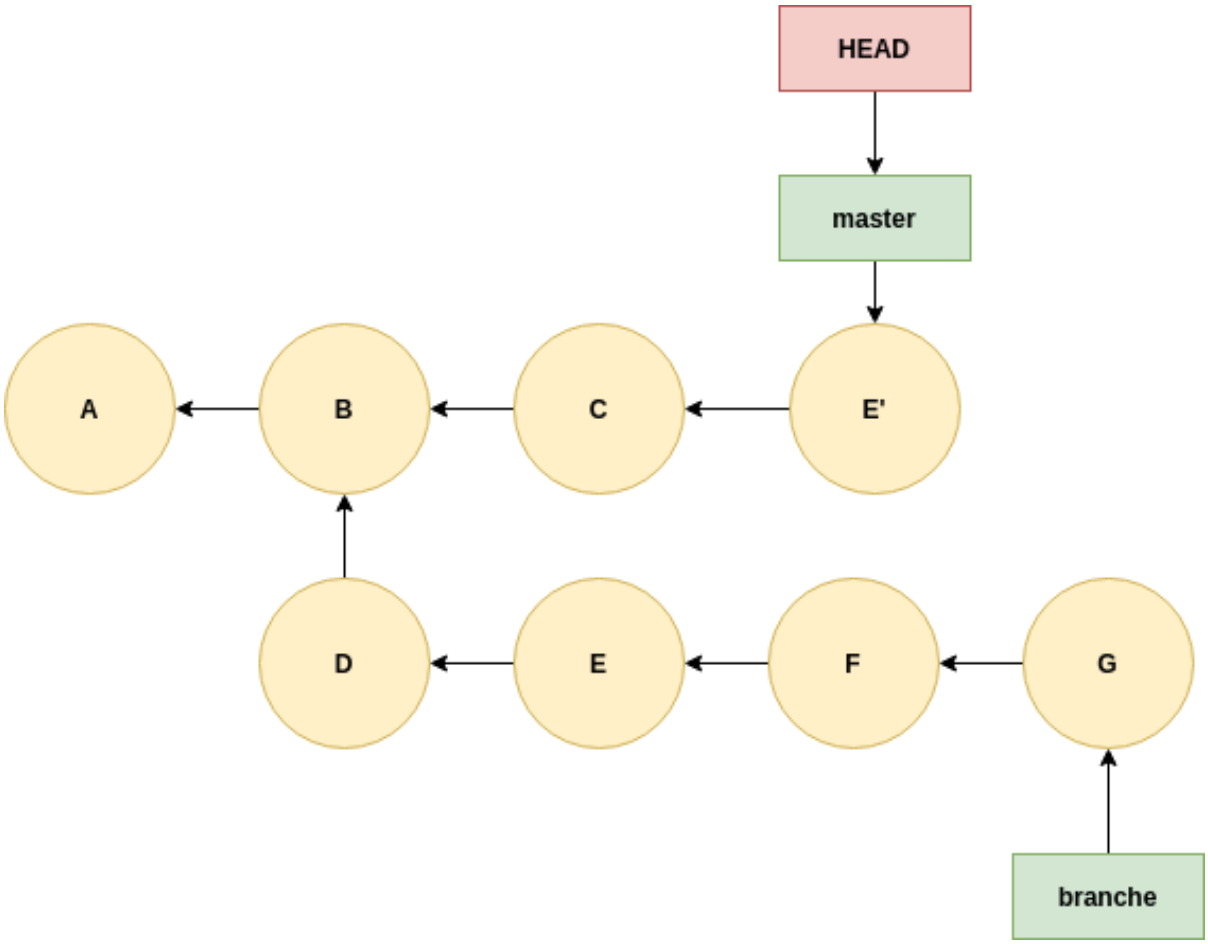
Vous ne souhaitez pas immédiatement fusionner la branche dans `main` car la fonctionnalité est loin d'être terminée, mais vous avez besoin absolument de modifications apportées par le `commit E`.

Ce n'est pas idéal, mais pour vous en sortir, vous pouvez faire :

```
git cherry-pick E
```

`E` étant bien sûr le `hash` entier ou raccourci du `commit E`.

Ce qui vous donne :



Le `commit E'` contient toutes les modifications correspondant à la version du `commit E` mais il n'a pas le même `hash` car la date de `commit` est différente, un autre objet `commit` est donc créé.

A noter que vous pouvez `cherry-pick` plusieurs `commits` d'un coup :

```
git cherry-pick hash1 hash2
```