Git : Supprimer ou remplacer le dernier commit

Cas : Il n'a pas été envoyé sur un dépôt distant

En utilisant **git commit –amend**

Si le commit n'a pas été envoyé sur un dépôt distant, il est tout à fait possible de le modifier, sans devoir passer par un commit inverse. Il faut effectuer les modifications qui ont été oubliées, ou qui n'avaient pas lieu d'être puis commiter en utiliser l'argument **--amend** :

git commit –amend

Après cette commande, Git redemandera un nouveau message de commit. Rien n'empêche d'utiliser celui qui a déjà été utilisé.

Et en utilisant git reset

Si le commit est bon à mettre à la poubelle et que les modifications qu'il comporte sont totalement inutiles, il est possible d'utiliser la commande suivante :

git reset --hard HEAD^

Cette commande aura pour effet de remonter l'historique sur le commit parent de **HEAD** en mettant à jour le répertoire de travail et l'index (en d'autres termes, supprimer le dernier commit).

Le commit ne sera réellement supprimé que lorsque le garbage collector de Git le supprimera.
Il est aussi possible de supprimer le dernier commit, tout en conservant ses modifications dans le répertoire de travail, pour cela il faut utiliser la commande suivante :

git reset --mixed HEAD^

Cas : Il a été envoyé sur un dépôt distant (après un push)

Ce qu'il faut éviter

Pour annuler des commits, il existe la commande **git reset**.

git reset --hard HEAD~1
HEAD is now at 444b1cf

Celle-ci est pertinente tant que les commits n'ont pas été poussés. Git vous retiendra au **push** d'ailleurs :

git push
To /tmp/repo
 ! [rejected]   master -> master (non-fast-forward)
error: failed to push some refs to '/tmp/repo'

En effet, à partir du moment où un commit existe sur le serveur, il est potentiellement utilisé par des collaborateurs (mergé, à la base d'une branche, etc.).

On pourrait faire le sale et forcer le **push** :

git push -f
Total 0 (delta 0), reused 0 (delta 0)
To /tmp/repo
 + b67c343...444b1cf master -> master (forced update)

Mais il y a beaucoup mieux !

Ce qu'il faut faire

Annuler un **commit**, c'est finalement appliquer **l'inverse de son diff** !

On peut rediriger le **diff** des **commits** à annuler vers la commande **patch --reverse**

git diff HEAD^ | patch --reverse

Pour faire plus simple, il y a **git revert** !

Par exemple pour annuler les **trois derniers commits** :

git revert HEAD~3..HEAD

Ou pour annuler un **commit** en particulier :

git revert 444b1cff

Il suffit alors de pousser proprement le commit obtenu sur le serveur. Les éventuels collaborateurs qui avaient basé leur travail sur les commits annulés devront gérer les conflits au moment venu...