# Formation Git Avancée

## IV - Réécrire l'histoire

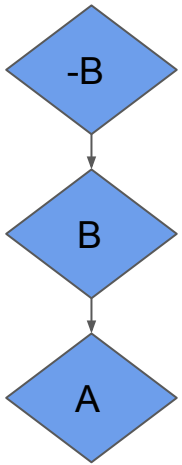
Codeur Pro

www.codeur-pro.fr

**Arnaud MERCIER**

arnaud.mercier.formation@gmail.com

# Annuler un commit




```
<body>
-   <!-- TITRE -->
-   <div>
+   <div id="top-banner">
+     Bienvenue dans cette formation GIT !
   </div>
</body>
```

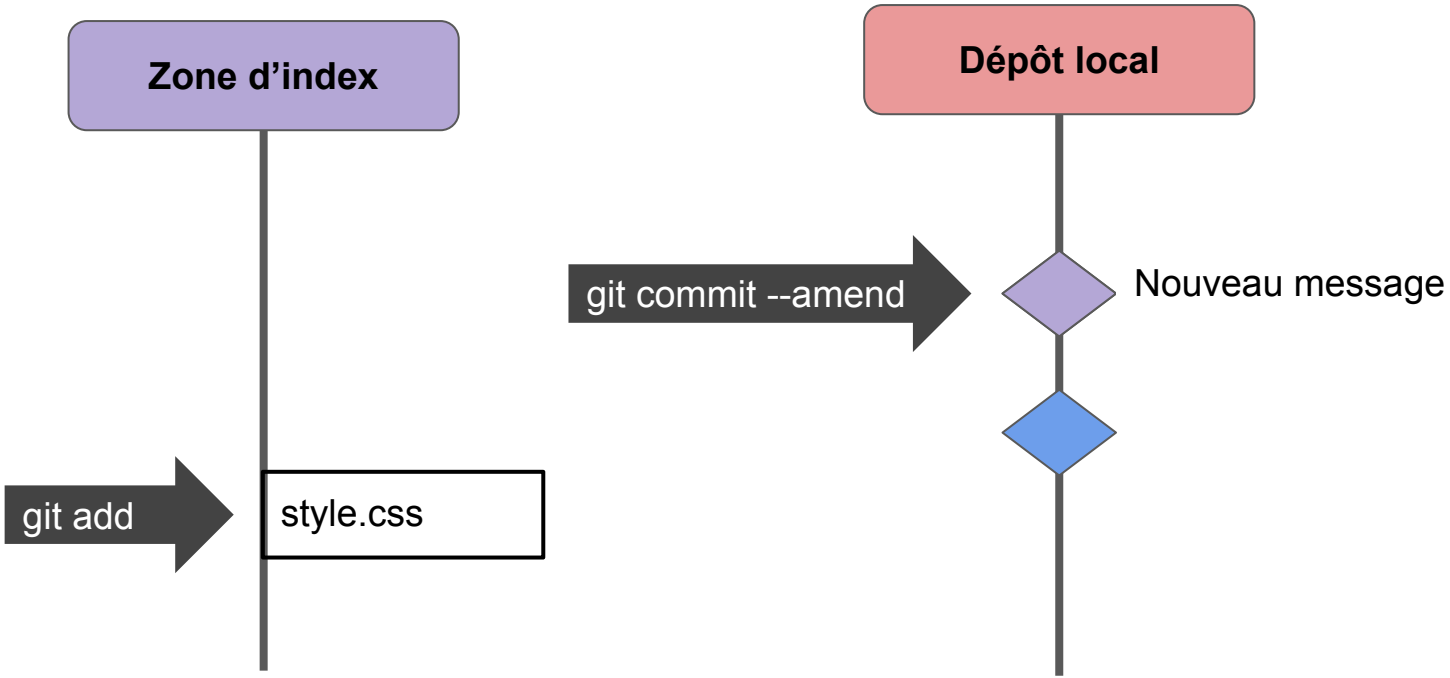
**commit B**

```
<body>
+   <!-- TITRE -->
+   <div>
-   <div id="top-banner">
-     Bienvenue dans cette formation GIT !
   </div>
</body>
```

**commit -B**

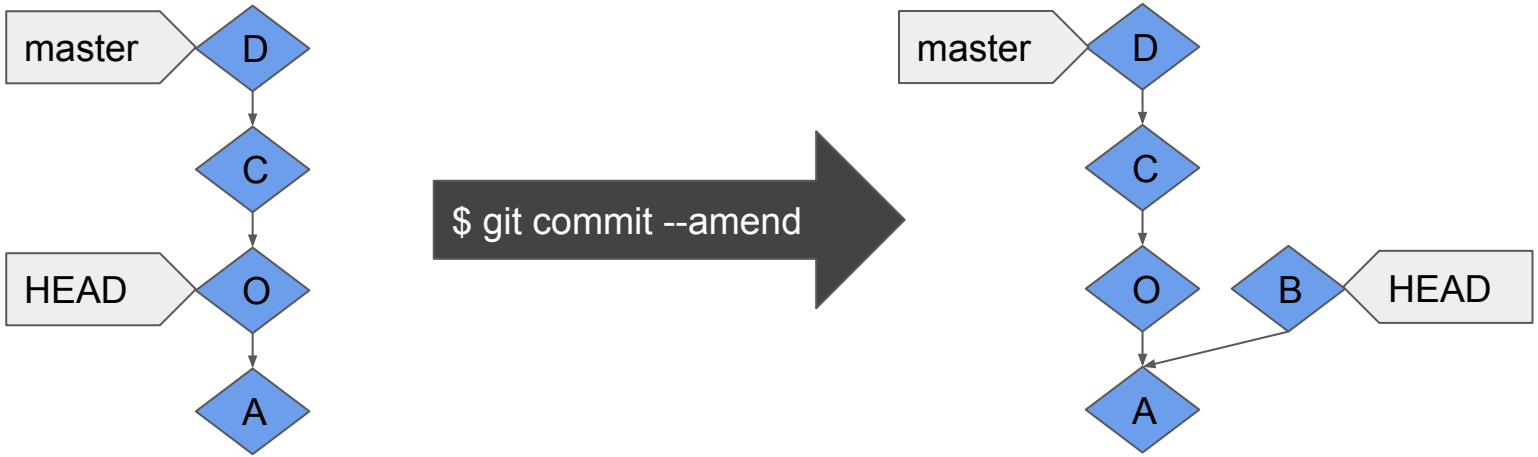
```
$ git revert <sha1>
```

# Modifier son dernier commit

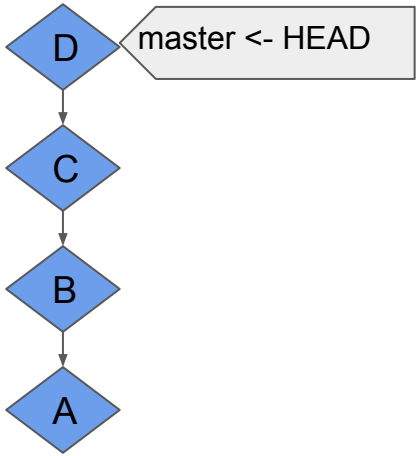


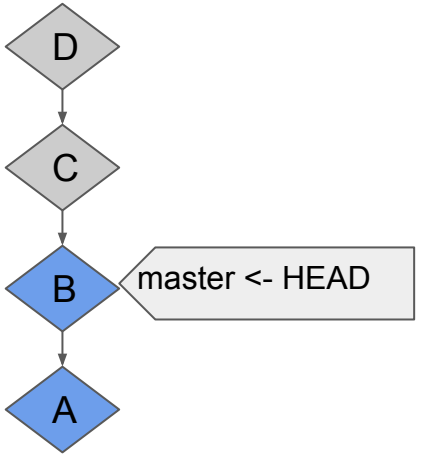

```
$ git add <fichiers oubliés>
$ git commit --amend -m “nouveau message”
```

# Rien de se perd, rien ne se transforme, tout se crée !



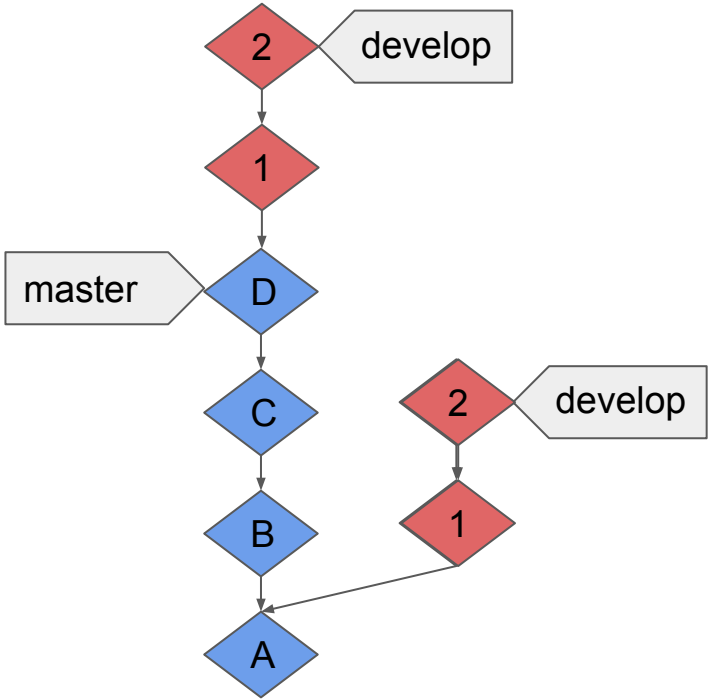
# Git reset



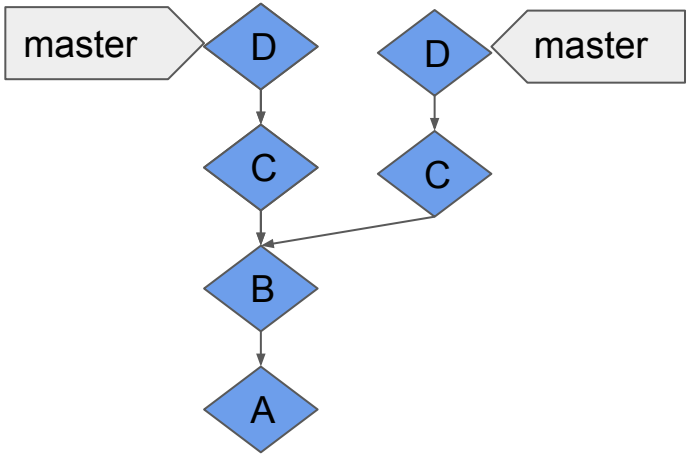




**$ git reset [mode] [ref/sha1]**

# 50 nuances de rebases




$ rebase master




$ rebase HEAD~2

# Rebase interactif

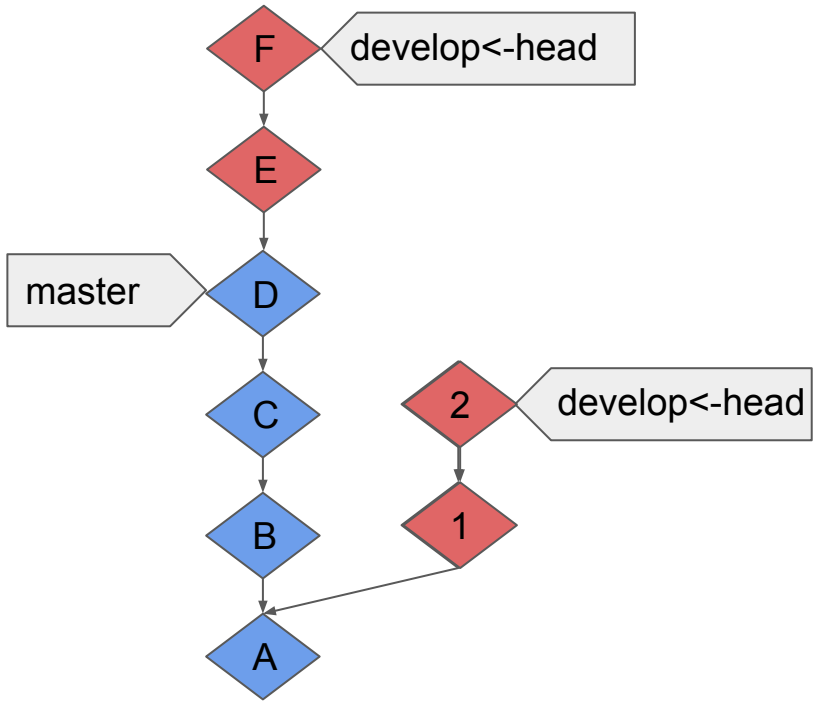
1. $ git rebase -i REF
2. Editer le script du rebase
3. Exécution du script

```
$ git rebase -i HEAD~3
pick commitB message commit B
pick commitC message commit C
pick commitD message commit D

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

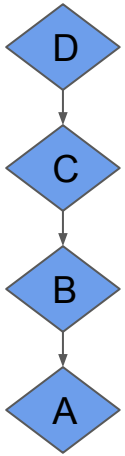
**Ordre d'affichage du rebase interactif est l'inverse du git log**

# Rebase interactif

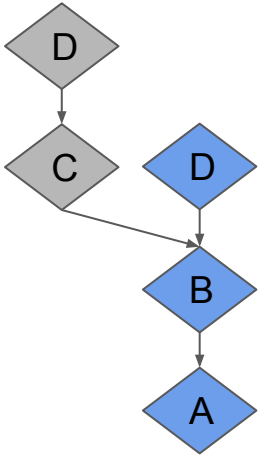



```
$ rebase -i master
```

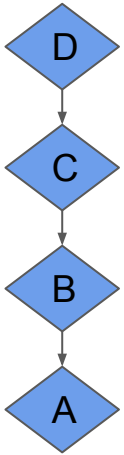
# Rebase interactif: Supprimer



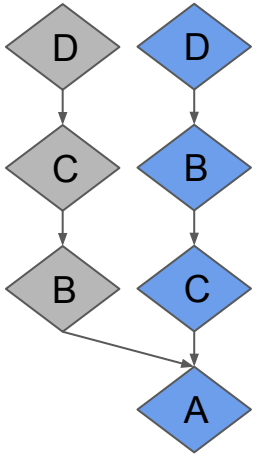

```
$ git rebase -i HEAD~2
drop commit C
pick commit D
```



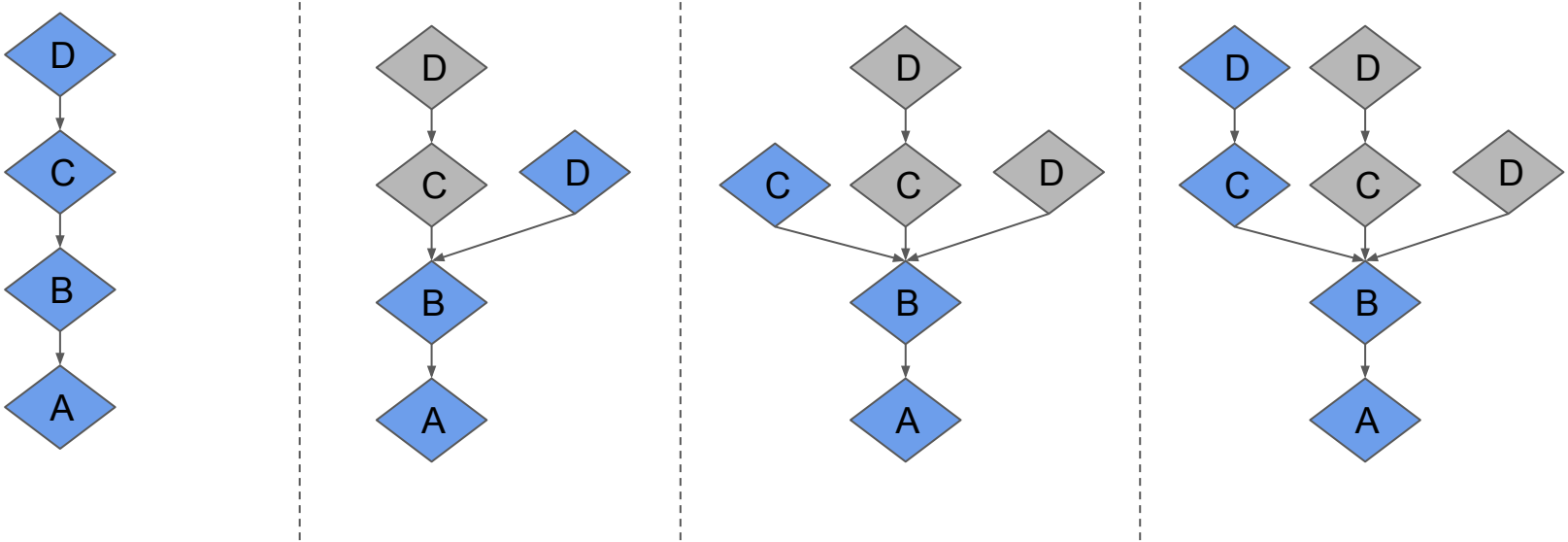
# Rebase interactif: Réordonner




```
$ git rebase -i HEAD~3
pick commit C
pick commit B
pick commit D
```

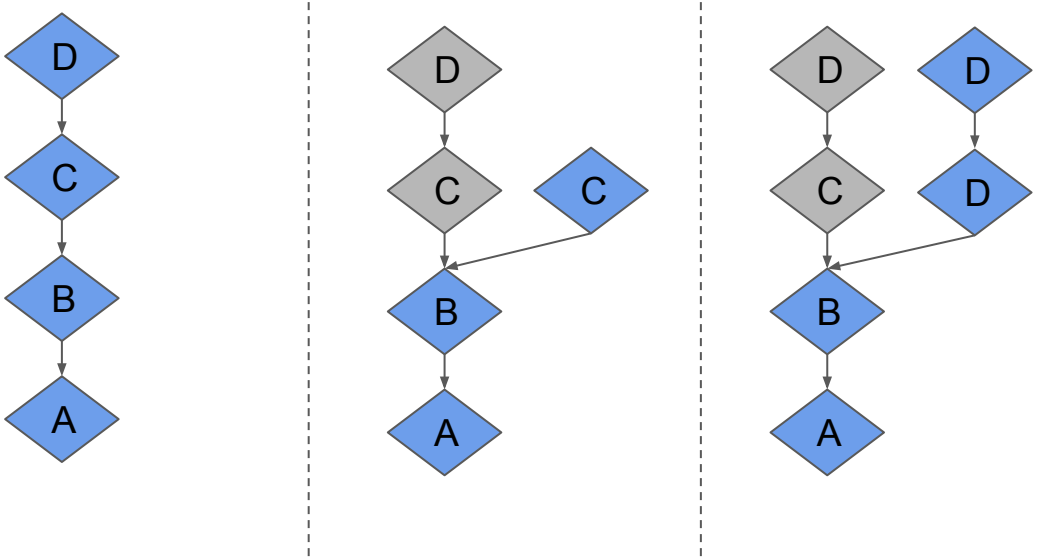


# Rebase ordre git log



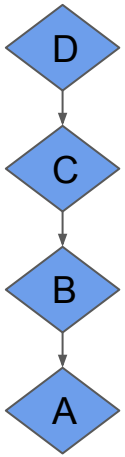

```
$ rebase HEAD~2
```

# Rebase ordre inverse git log

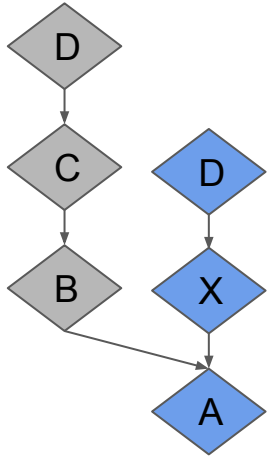



```
$ rebase HEAD~2
```

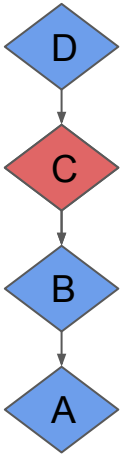
# Rebase interactif: Fusionner




```
$ git rebase -i HEAD~3
pick commit B
squash commit C
pick commit D
```



# Rebase interactif: Editer




1

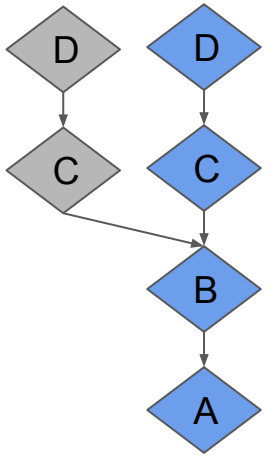
```
$ git rebase -i HEAD~3
pick commit B
edit commit C
pick commit D
```

2

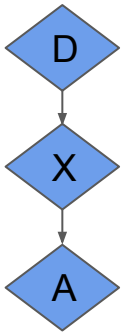
```
$ git commit --amend
```

3

```
$ git rebase --continue
```



# Rebase interactif: Découper




1

```
$ git rebase -i HEAD~2
edit commit X
pick commit D
```

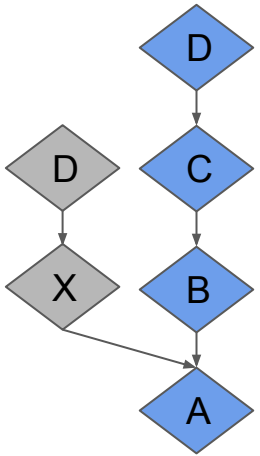
2

```
$ git add <files>
$ git commit -m”B”
```

```
$ git add <files>
$ git commit -m”C”
```

3

```
$ git rebase --continue
```



# Bilan

1. $ git rebase -i HEAD~[N]
2. Editer le script du rebase
3. Exécution du script

```
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

**Ordre d’affichage du rebase interactif est l’inverse du git log**