

UN « STÔLZ » ?

?

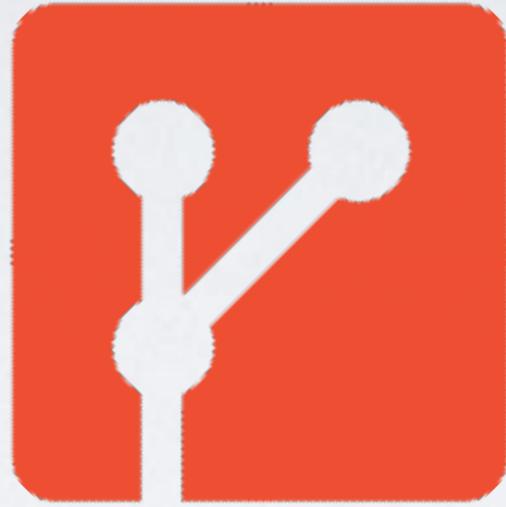
?



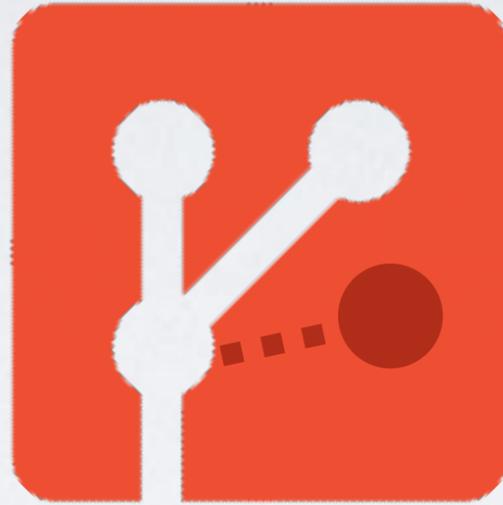
MAIS QU'EST-CE QUE J'FAIS LÀ ?

?

?

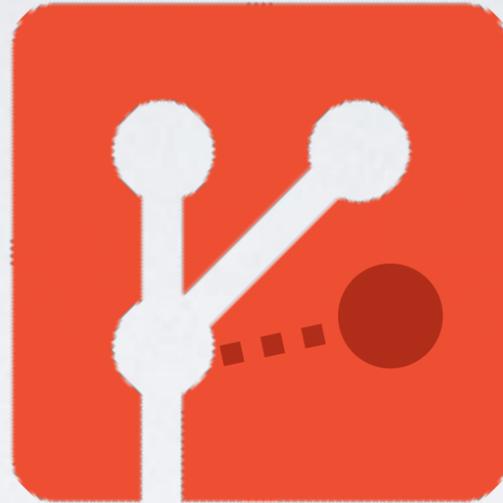


GIT



# GIT RESET

Rien ne se perd, tout se transforme



# GIT RESET

Rien ne se perd, tout se transforme

*(ou presque)*

COMMENÇONS AVEC  
QUELQUES RAPPELS

- **3 zones principales** : copie de travail, index, dépôt local
- **HEAD** : notre emplacement à tout instant
- **log** : historique final/actuel de nos commits
- **reflog** : cheminement et détours jusqu'à notre position actuelle
- **garbage collector** : on « dé-référence » mais ne supprime pas, le GC fera du nettoyage plus tard

# ÉTAPES D'UN COMMIT À TRAVERS LES ZONES

Copie de travail

Index

Dépôt local

# ÉTAPES D'UN COMMIT À TRAVERS LES ZONES

Copie de travail

Index

Dépôt local

Création/MàJ de fichiers

# ÉTAPES D'UN COMMIT À TRAVERS LES ZONES

Copie de travail

Index

Dépôt local

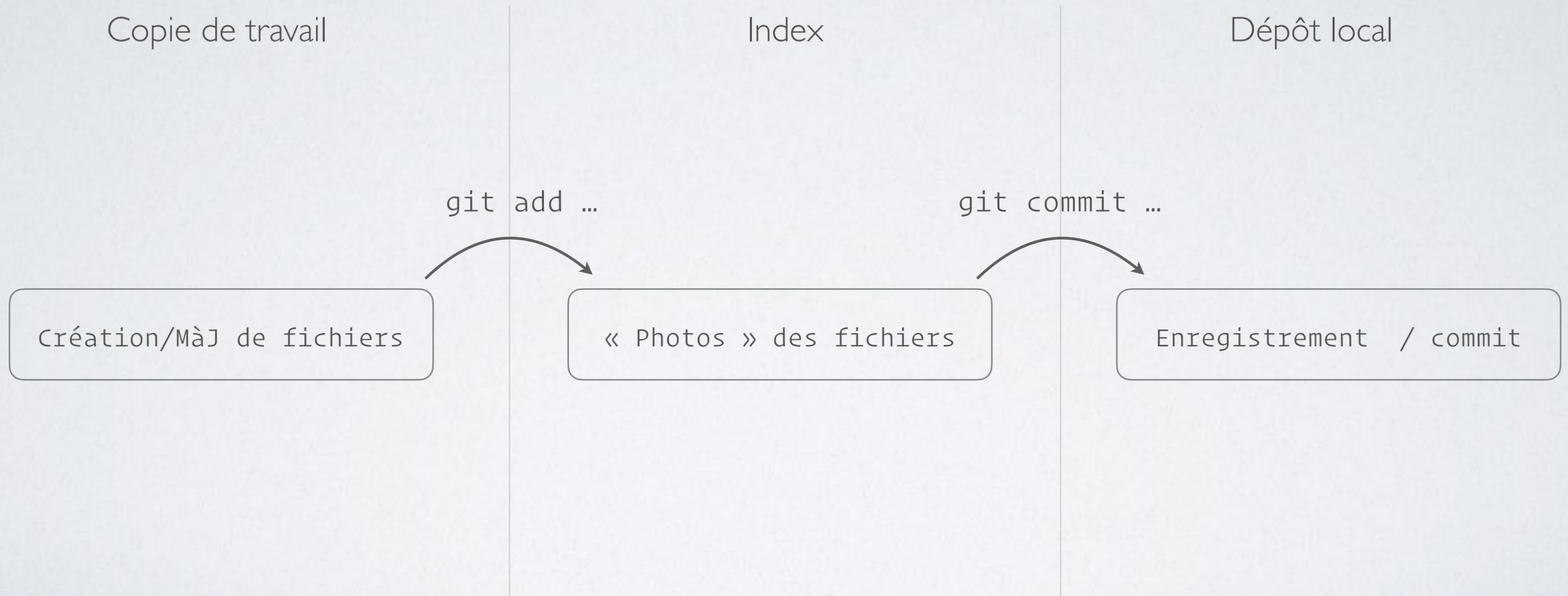
Création/MàJ de fichiers

`git add ...`

« Photos » des fichiers



# ÉTAPES D'UN COMMIT À TRAVERS LES ZONES



# ÉTAPES DE COMMITS ENTRE ZONES ET LOG

Copie de travail

Index

Dépôt local

Log

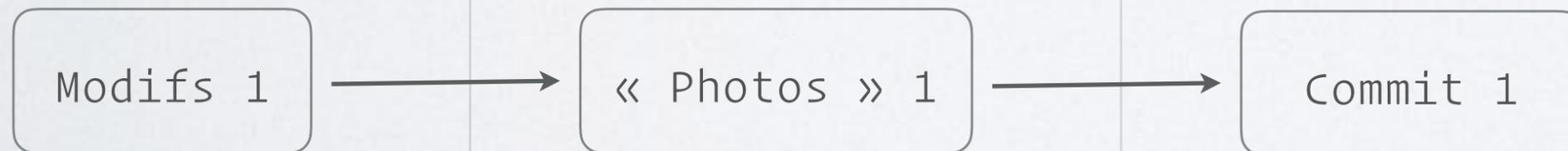
# ÉTAPES DE COMMITS ENTRE ZONES ET LOG

Copie de travail

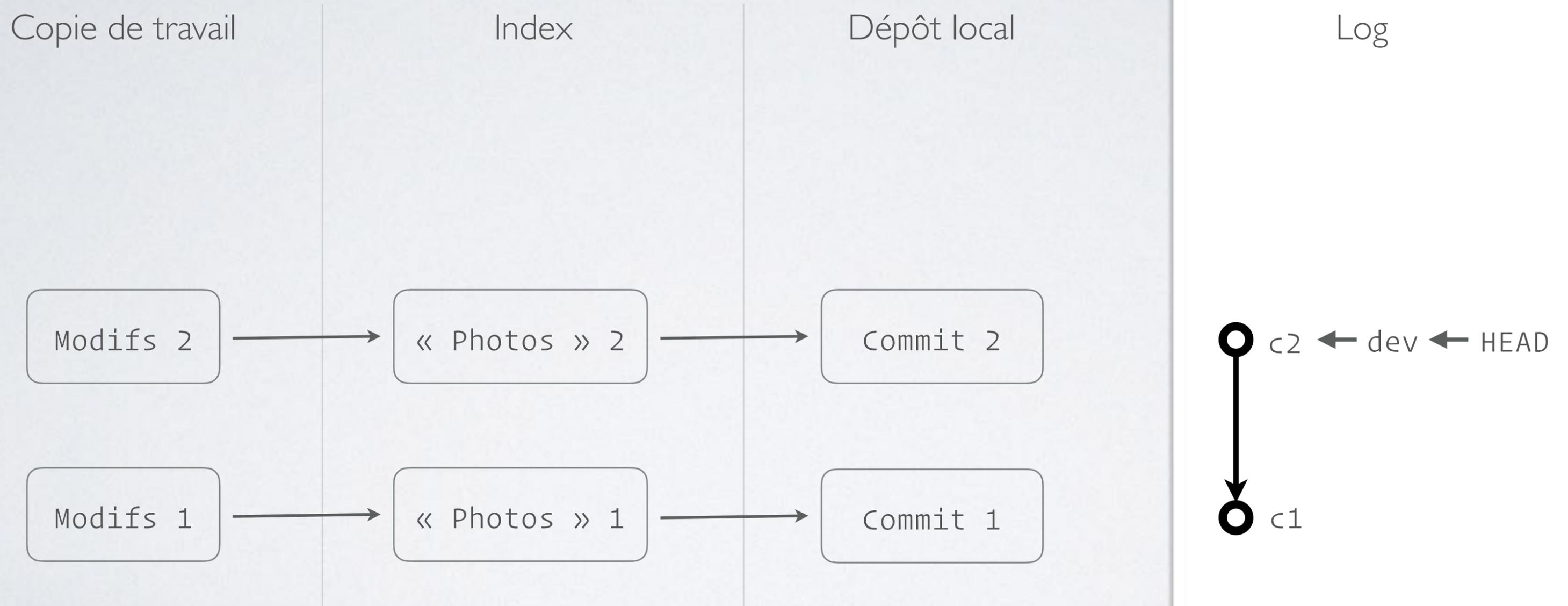
Index

Dépôt local

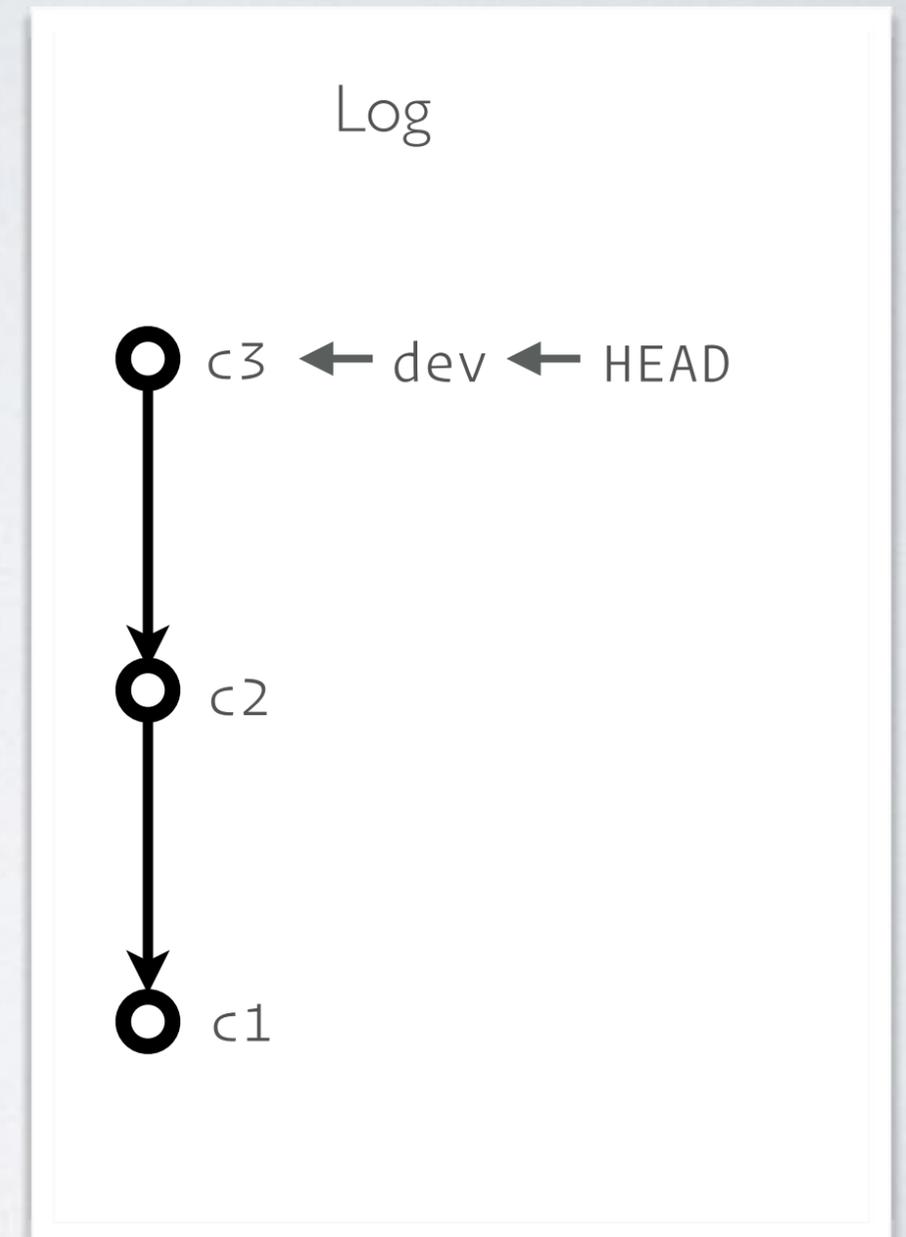
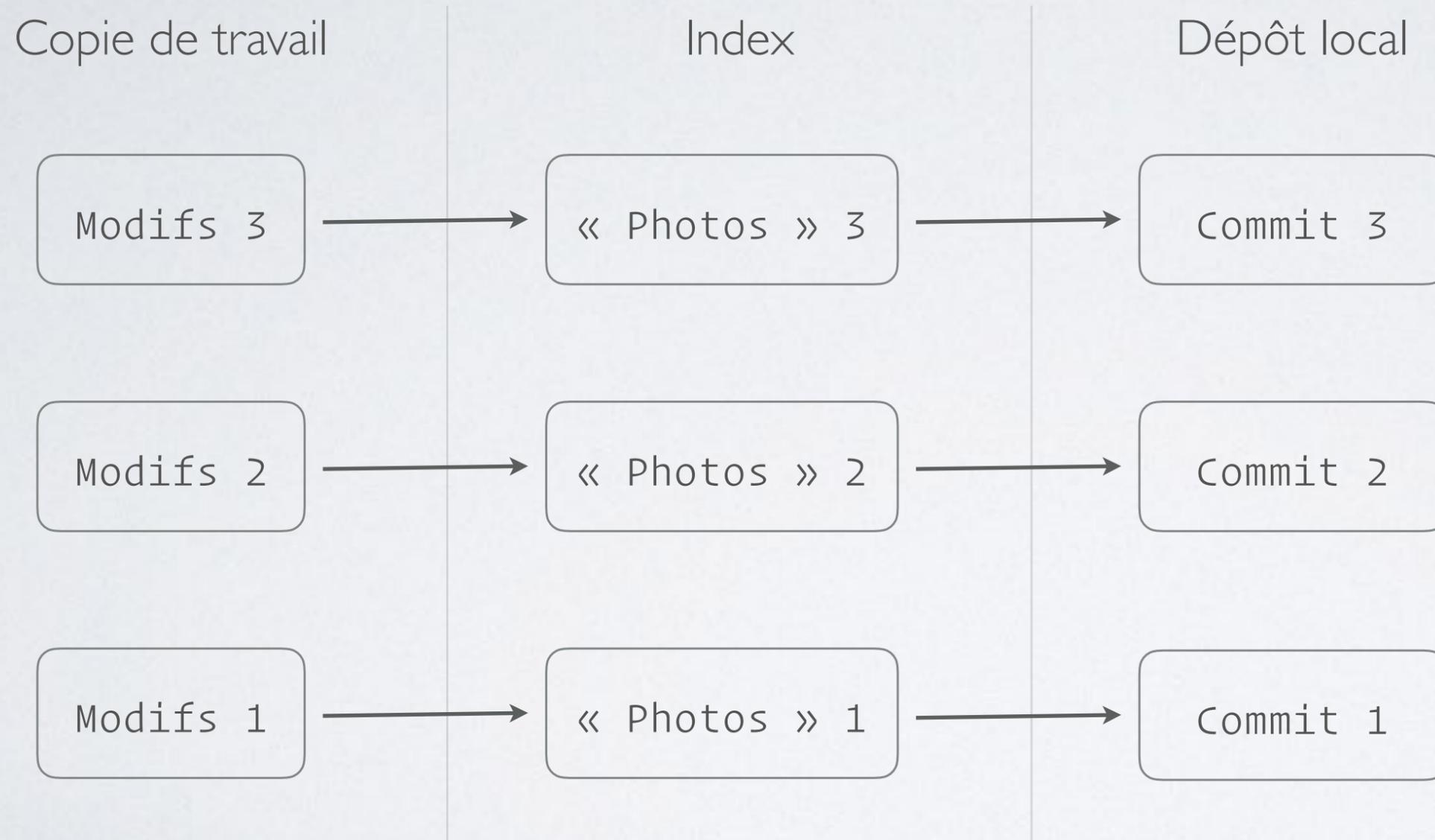
Log



# ÉTAPES DE COMMITS ENTRE ZONES ET LOG



# ÉTAPES DE COMMITS ENTRE ZONES ET LOG



# GIT « UNSTAGE »

Cas particulier de git reset

```
git reset <le-ou-les-fichiers-ou-chemins>
```

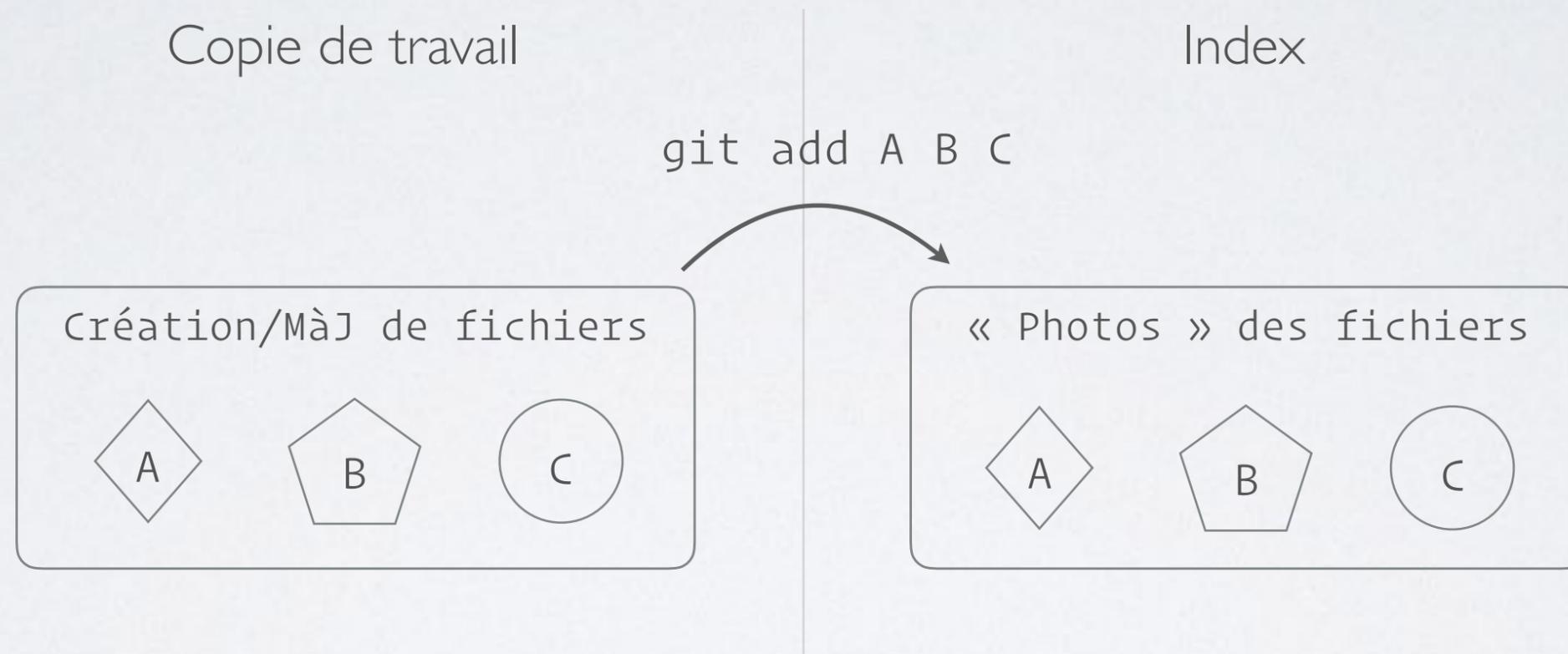
Après ajout de fichiers modifiés à l'index, pour « annuler » la photo.

Seul cas d'utilisation de `git reset` sur fichier et qui tend(ait) à nuire à la compréhension des autres options.

En cas d'ambiguïté entre références Git et chemins : `git reset -- <le-ou-les-fichiers-ou-chemins>`

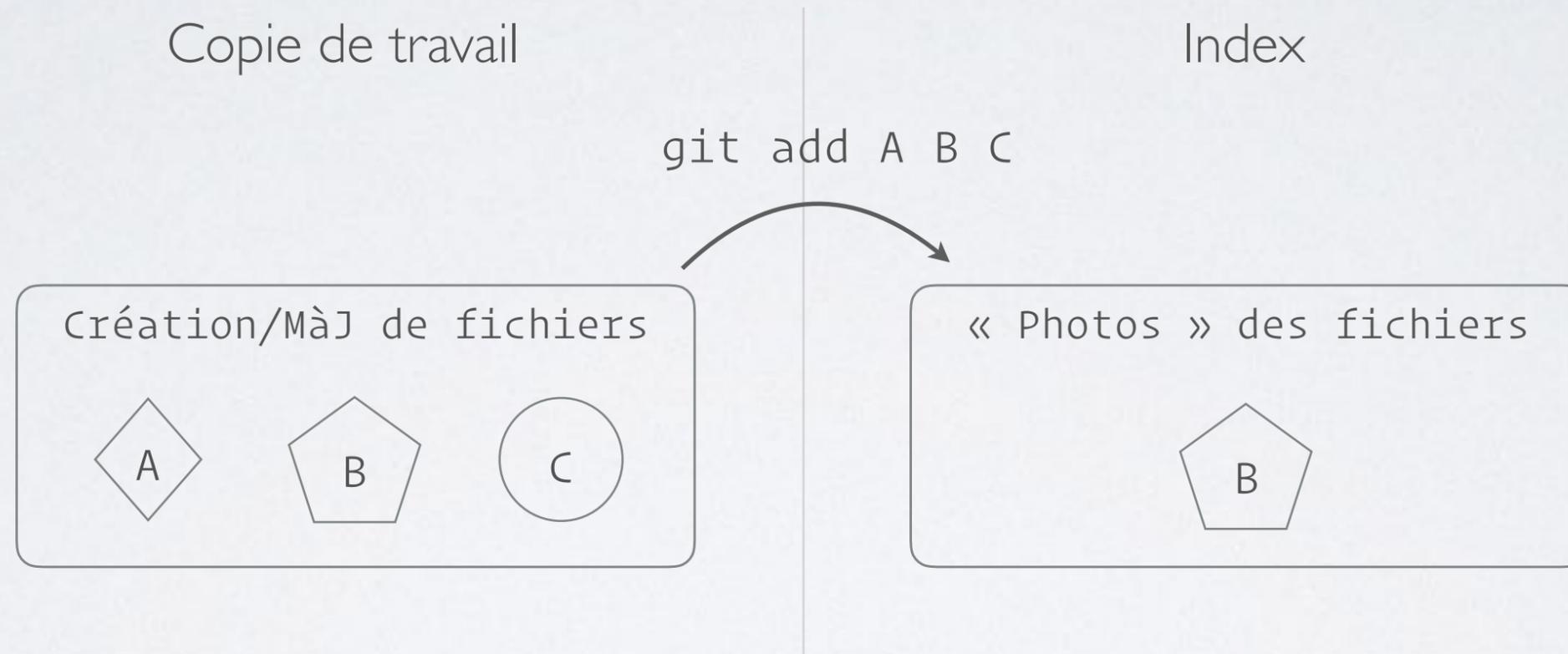
# GIT RESET SUR FICHIER

Annuler les photos mises en cache/index



# GIT RESET SUR FICHIER

Annuler les photos mises en cache/index



```
> git reset A C
```

# GIT RESTORE

Nouveauté GIT 2.23

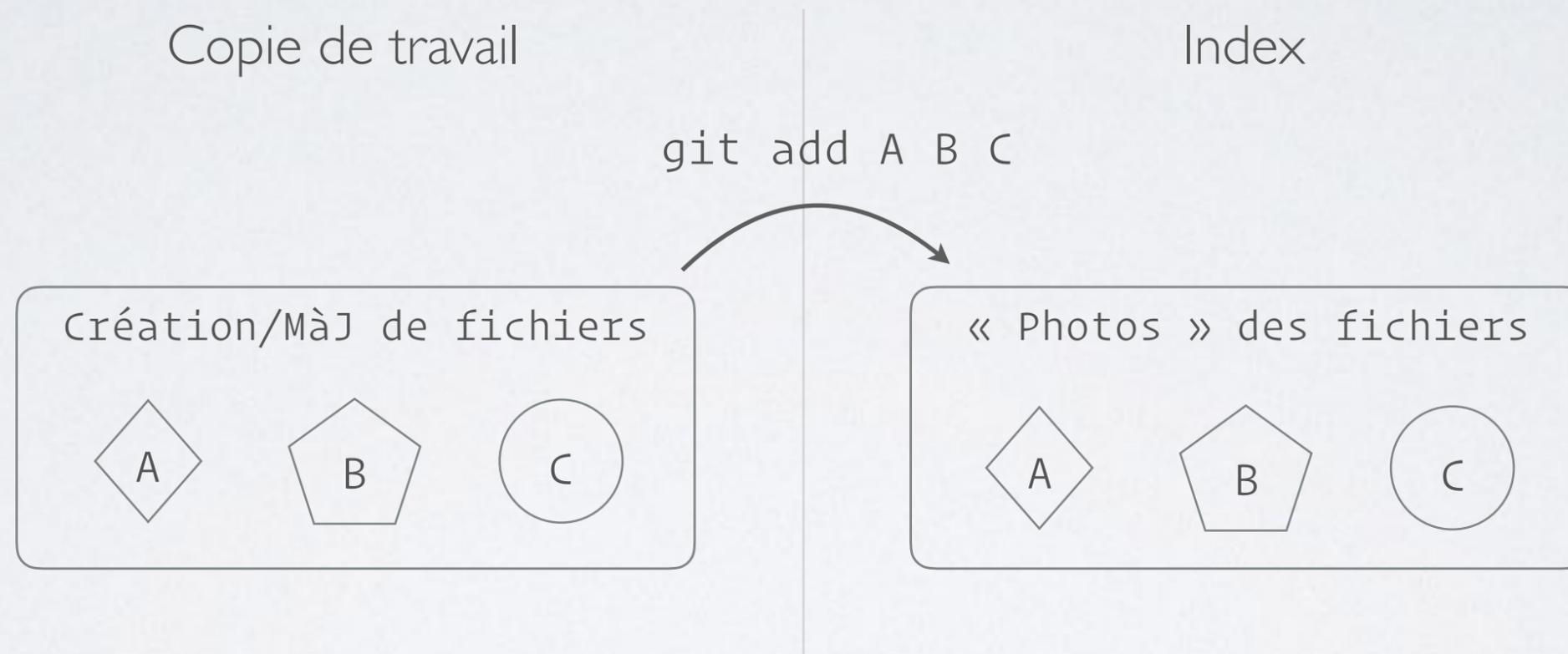
```
git restore <le-ou-les-fichiers-ou-chemins>
```

Terme plus clair, censé faciliter la compréhension des opérations.

« Restaure » la copie de travail (option `--worktree` par défaut),  
l'index avec l'option `--staged`.

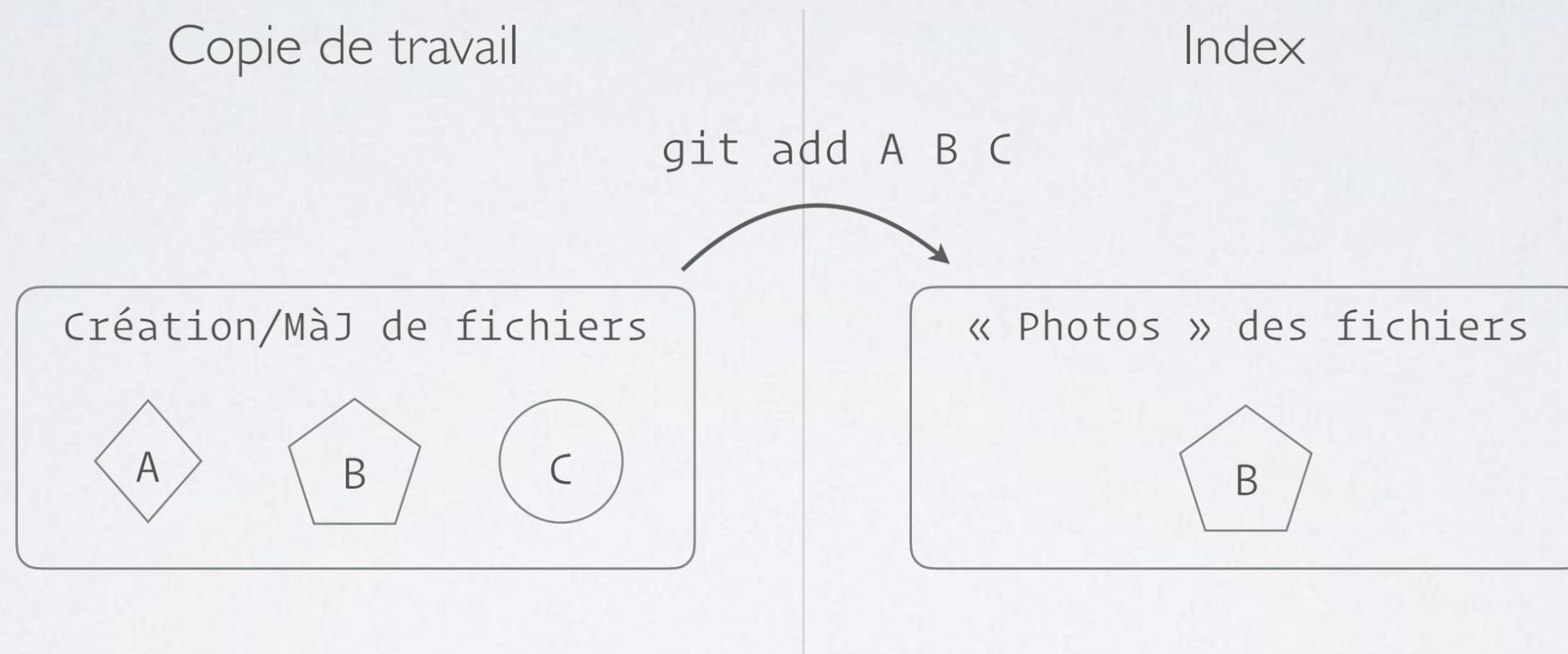
# GIT RESTORE --STAGED

Annuler les photos mises en cache/index



# GIT RESTORE --STAGED

Annuler les photos mises en cache/index



```
> git restore --staged A C
```

# ET POUR LES AUTRES CAS ?

Reset, mais pas sur des fichiers

```
git reset --<mode> <revision>
```

Déplace HEAD à l'emplacement voulu.

Le mode définit les opérations à effectuer sur l'index et la copie de travail.

# EFFETS DES MODES SUR LES ZONES

	Déplace HEAD	Défait l'index	Défait le(s) commit(s) dans la copie de travail
--soft	✓		
--mixed	✓	✓	
--keep --merge --hard	✓	✓	✓

# EFFETS DES MODES SUR LES ZONES

	Déplace HEAD	Défait l'index	Défait le(s) commit(s) dans la copie de travail
--soft	✓		
--mixed	✓	✓	
--keep --merge --hard	✓	✓	✓

**Wait... what?**

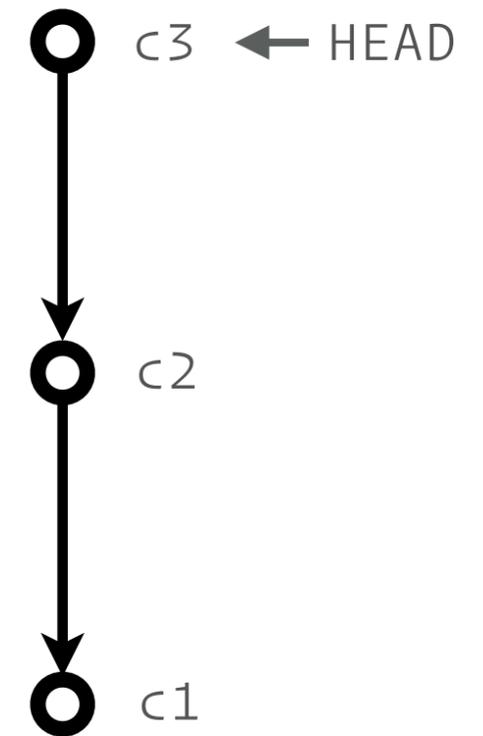
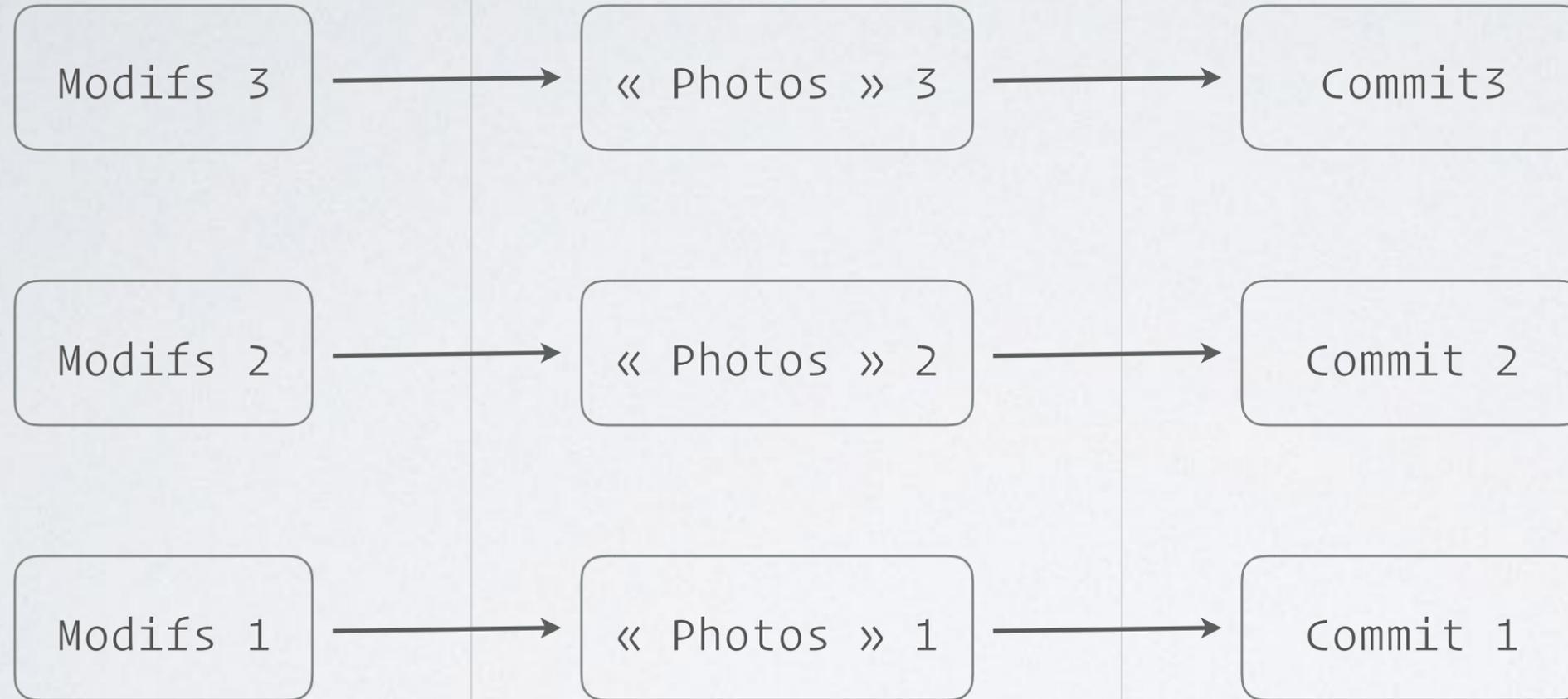
# GIT RESET --SOFT

Copie de travail

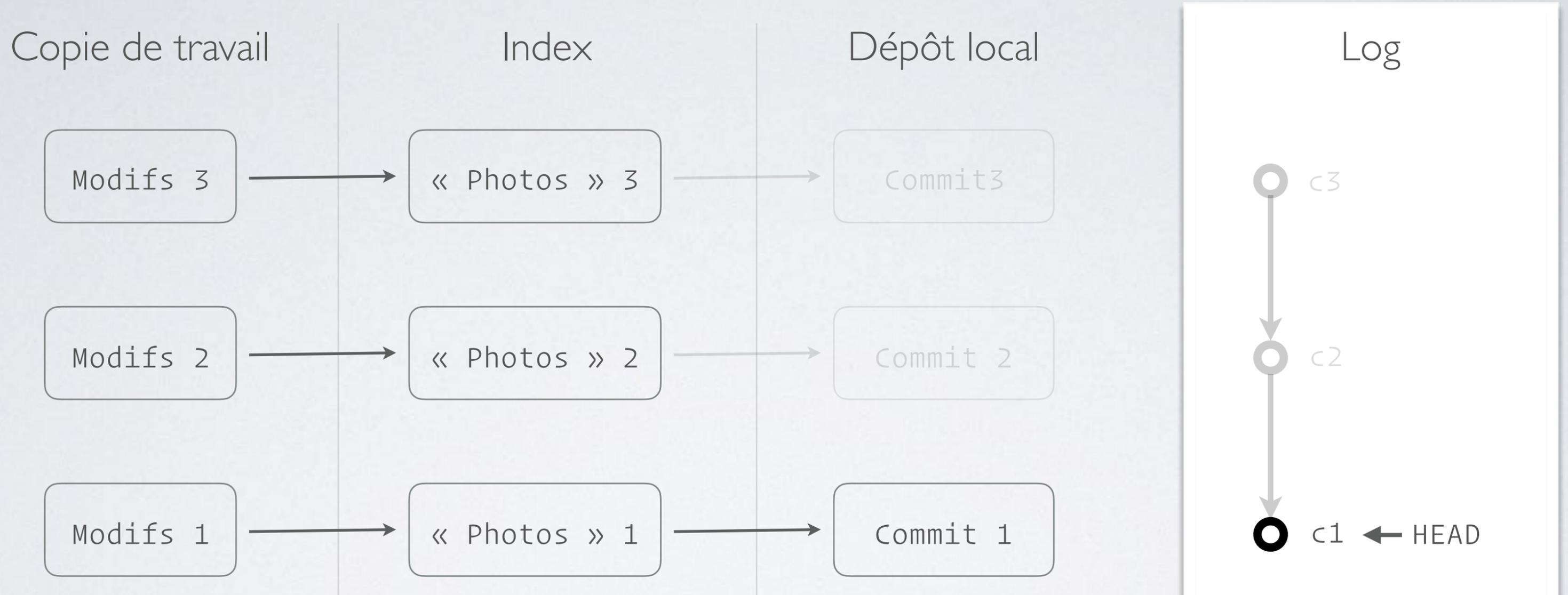
Index

Dépôt local

Log

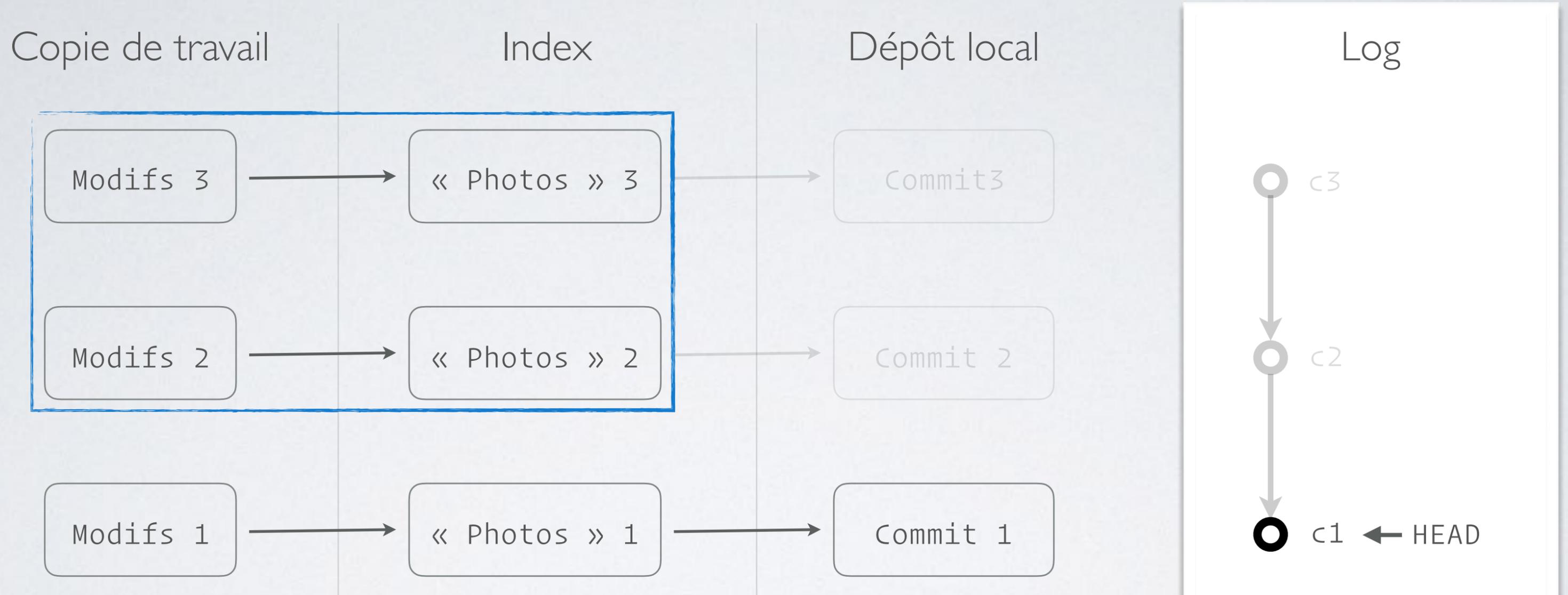


# GIT RESET --SOFT



```
> git reset --soft c1
```

# GIT RESET --SOFT



```
> git reset --soft c1
```

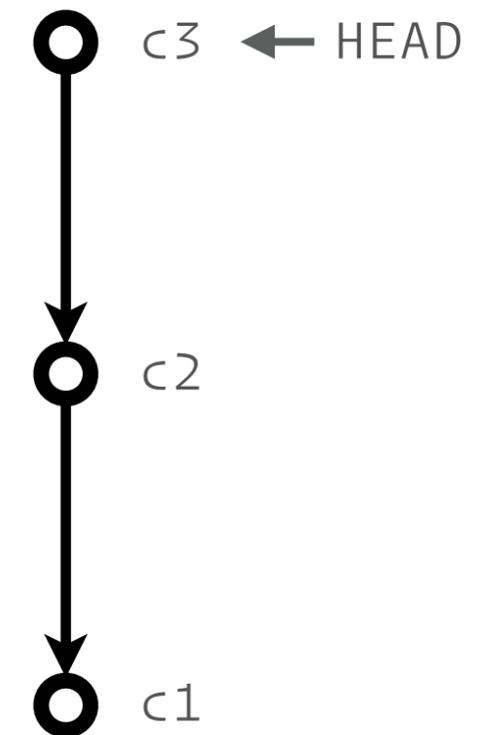
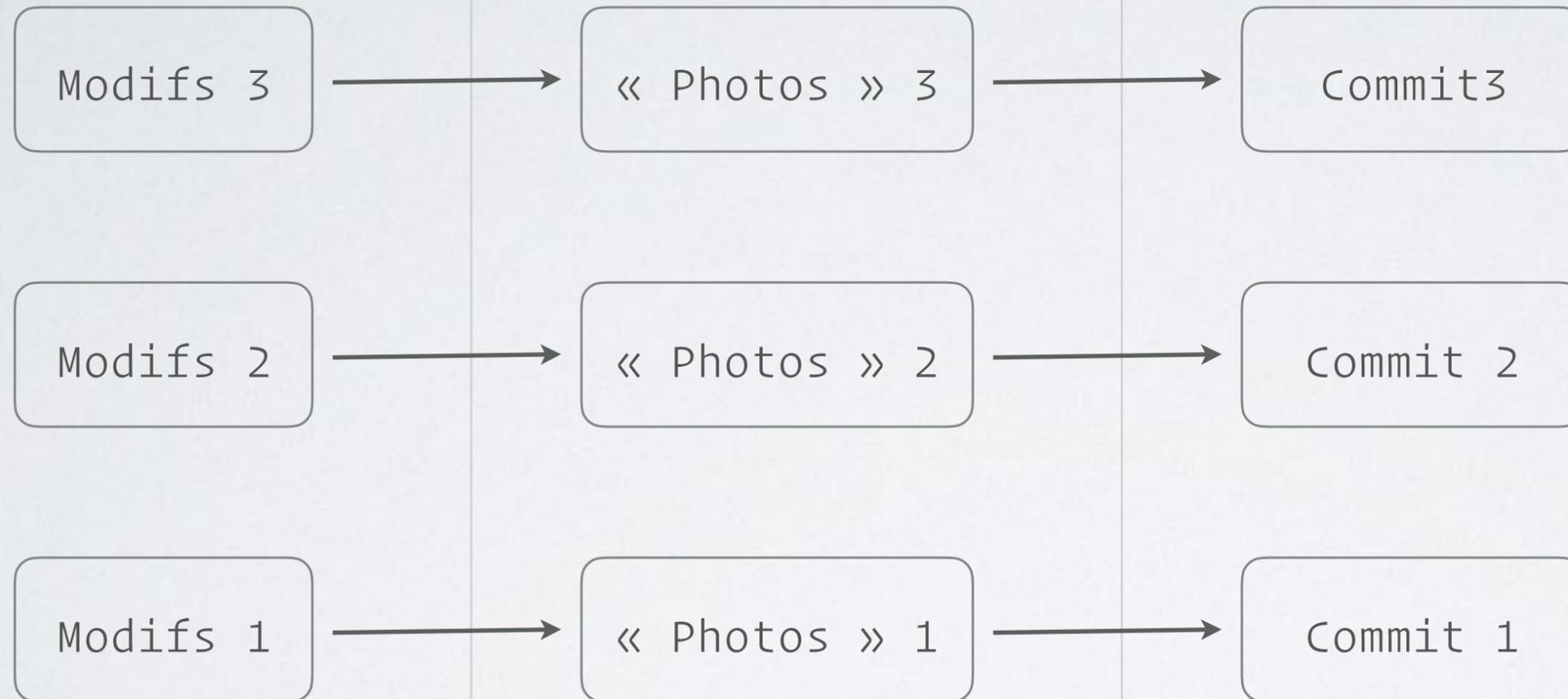
# GIT RESET --MIXED

Copie de travail

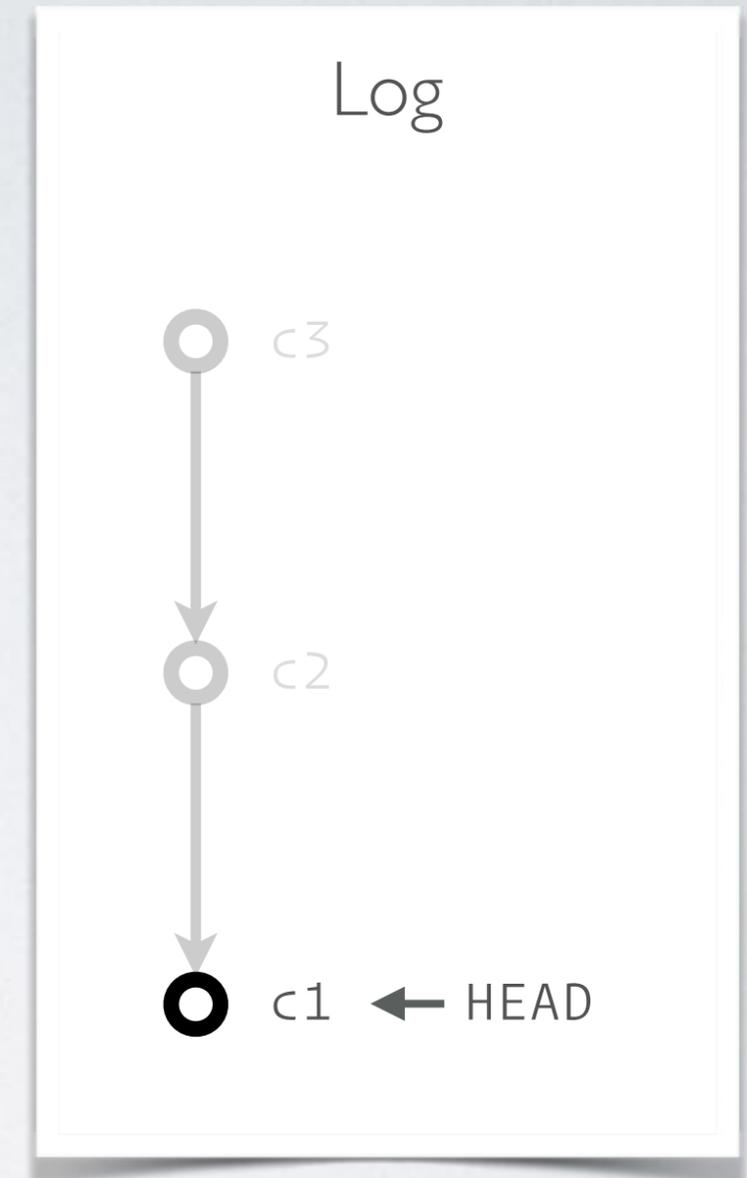
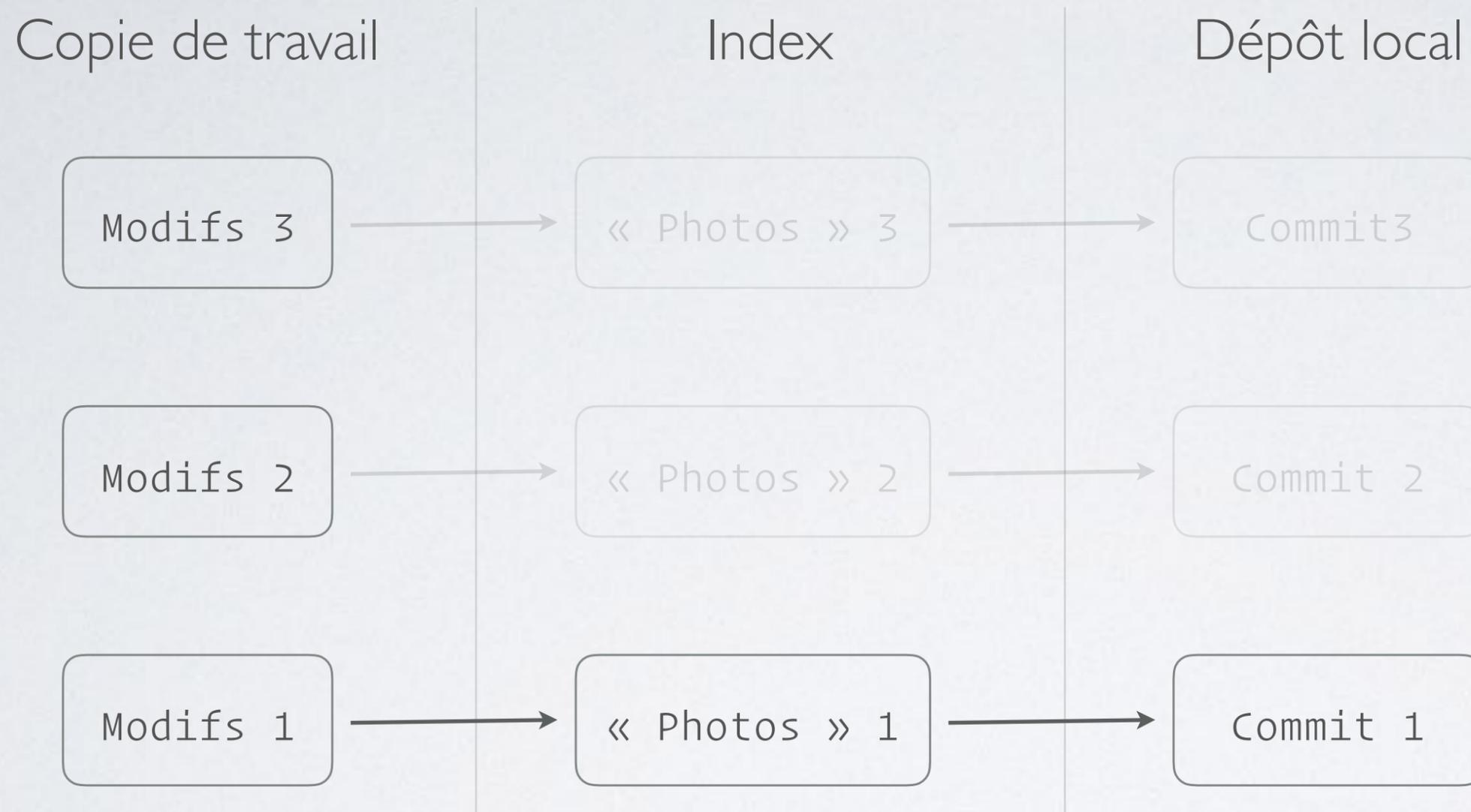
Index

Dépôt local

Log



# GIT RESET --MIXED



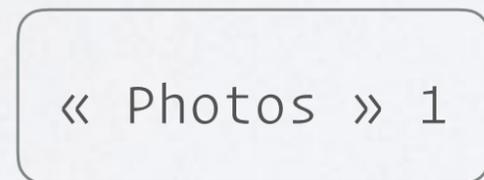
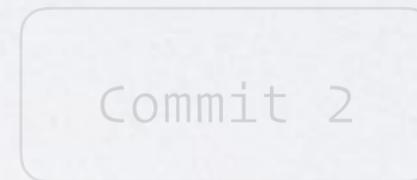
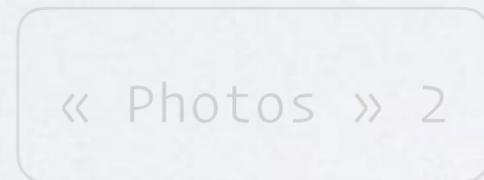
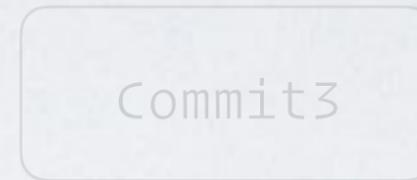
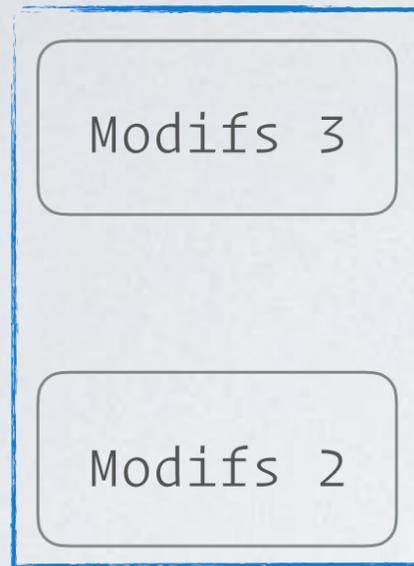
```
> git reset --mixed c1
```

# GIT RESET --MIXED

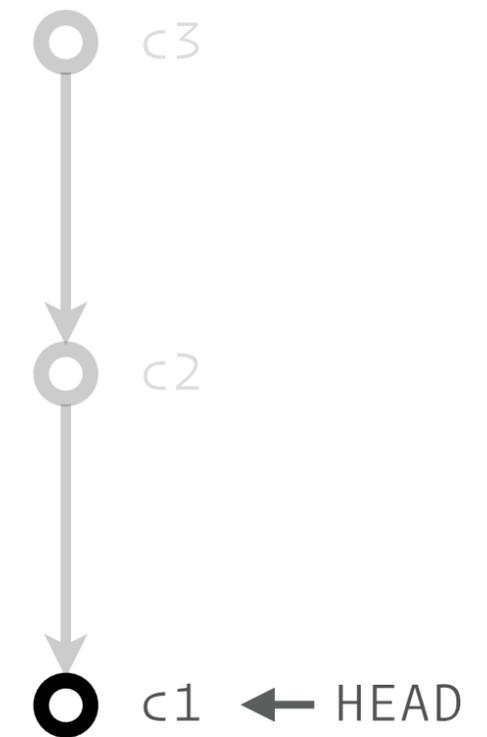
Copie de travail

Index

Dépôt local



Log



```
> git reset --mixed c1
```

# DIFFÉRENCES ENTRE KEEP / MERGE / HARD

	Déplace HEAD	Défait l'index	Défait le(s) commit(s) dans la copie de travail
--keep	✓	✓	✓
--merge	✓	✓	✓
--hard	✓	✓	✓

# DIFFÉRENCES ENTRE KEEP / MERGE / HARD

	Déplace HEAD	Défait l'index	Défait le(s) commit(s) dans la copie de travail
--keep	✓	✓ *	✓ *
--merge	✓	✓	✓ *
--hard	✓	✓	✓

\* Préserve les modifications en cours / non commitées

# DIFFÉRENCES ENTRE KEEP / MERGE / HARD

	Déplace HEAD	Défait l'index	Défait le(s) commit(s) dans la copie de travail
--keep	✓	✓ *	✓ *
--merge	✓	✓	✓ *
--hard	✓ ☠	YOLO!	☠ ✓

\* Préserve les modifications en cours / non commitées

# GIT RESET --KEEP

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

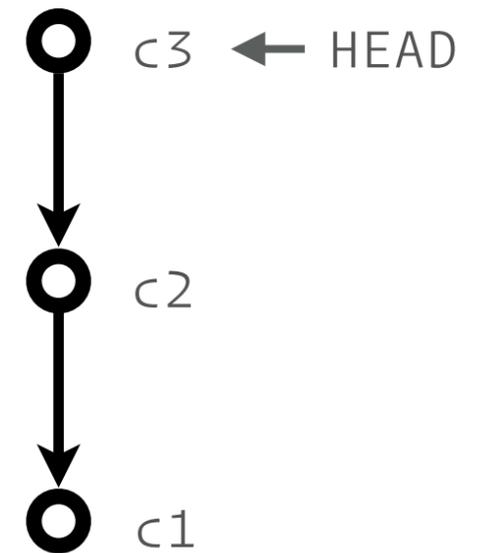
Dépôt local

Commit3

Commit 2

Commit 1

Log



# GIT RESET --KEEP

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

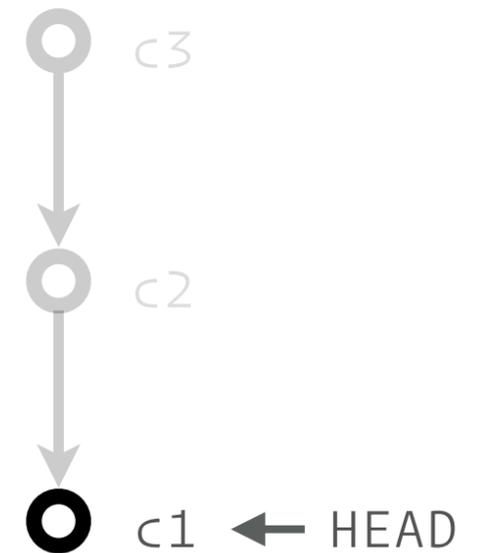
Dépôt local

Commit3

Commit 2

Commit 1

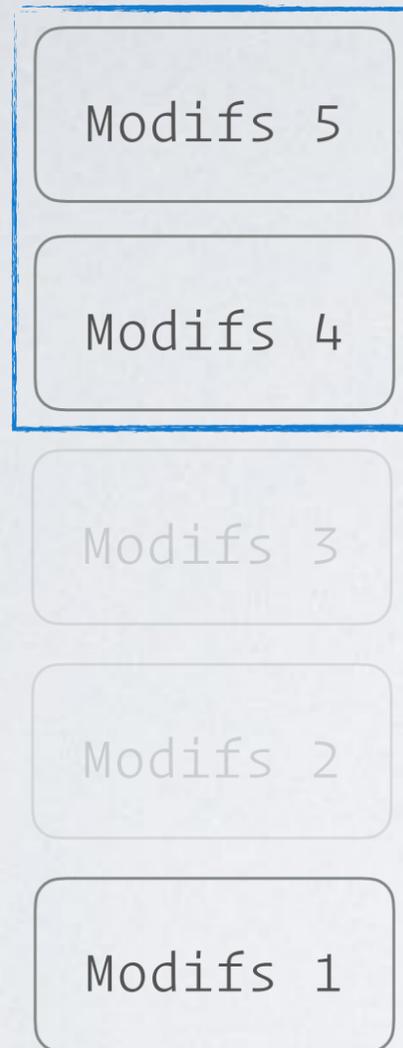
Log



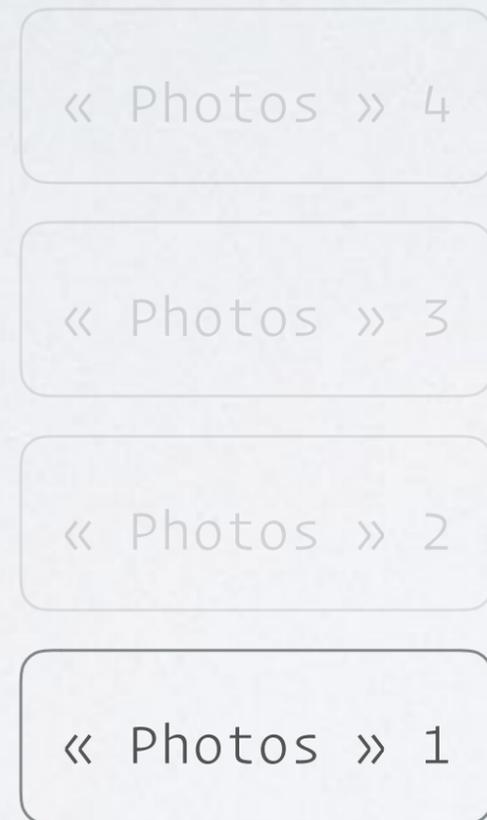
```
> git reset --keep c1
```

# GIT RESET --KEEP

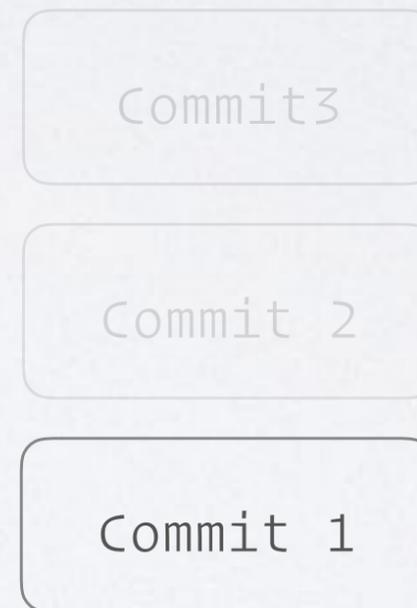
Copie de travail



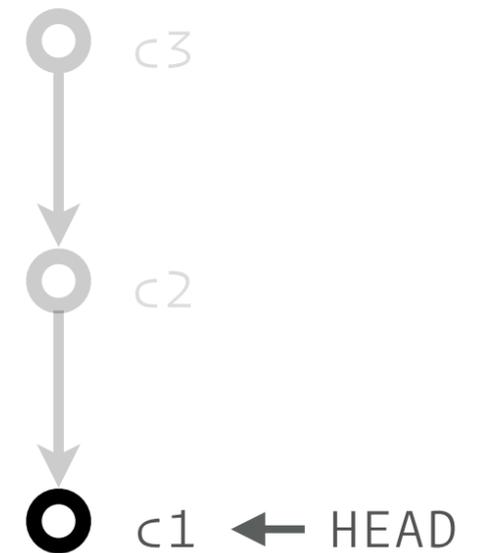
Index



Dépôt local



Log



```
> git reset --keep c1
```

# GIT RESET --MERGE

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

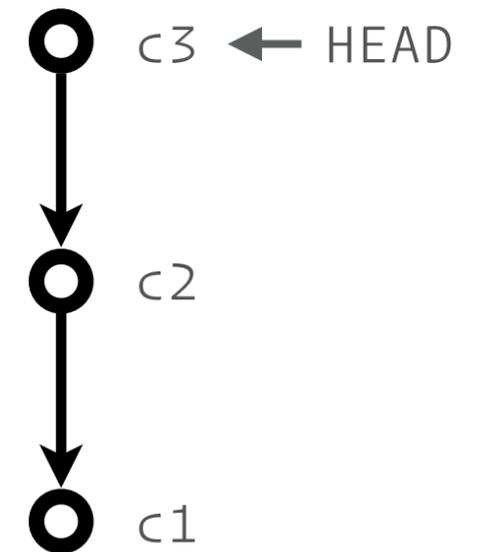
Dépôt local

Commit3

Commit 2

Commit 1

Log



# GIT RESET --MERGE

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

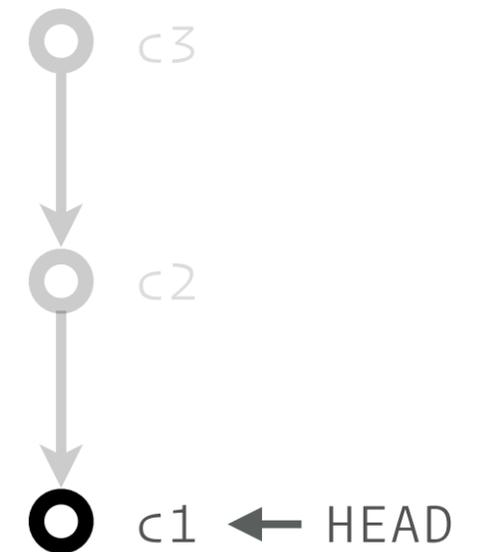
Dépôt local

Commit3

Commit 2

Commit 1

Log



```
> git reset --merge c1
```

# GIT RESET --MERGE

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

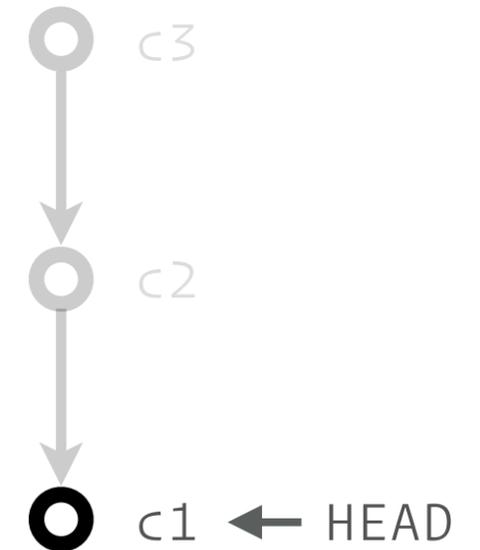
Dépôt local

Commit3

Commit 2

Commit 1

Log



```
> git reset --merge c1
```

# GIT RESET --HARD

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

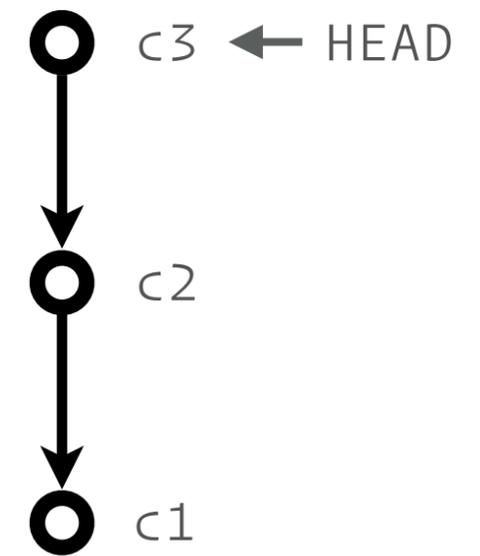
Dépôt local

Commit3

Commit 2

Commit 1

Log



# GIT RESET --HARD

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

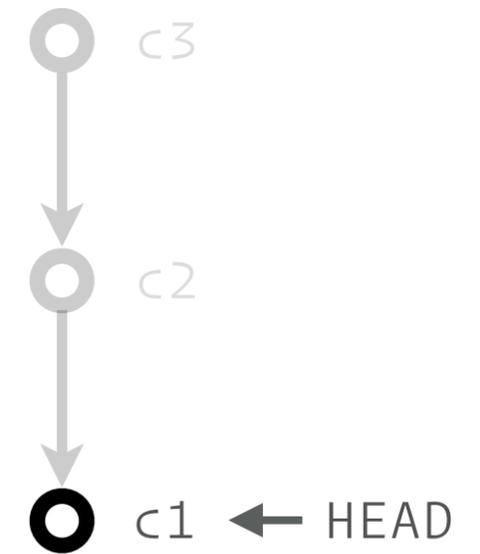
Dépôt local

Commit3

Commit 2

Commit 1

Log



```
> git reset --hard c1
```

C'EST BIEN BEAU TOUT ÇA,  
MAIS QUELS SONT LES CAS PRATIQUES ?

RESET ME PERMET DE...

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :
  - pour les supprimer

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :
  - pour les supprimer
  - pour les regrouper

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :
  - pour les supprimer
  - pour les regrouper
- Faire une branche dédiée pour mes X derniers commits

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :
  - pour les supprimer
  - pour les regrouper
- Faire une branche dédiée pour mes X derniers commits
- Faire table rase des modifs en cours

# RESET ME PERMET DE...

- Corriger/compléter mon dernier commit
- Défaire les X derniers commits :
  - pour les supprimer
  - pour les regrouper
- Faire une branche dédiée pour mes X derniers commits
- Faire table rase des modifs en cours
- Resetter un reset 

# J'AI COMMITÉ TROP VITE

J'ai besoin de modifier/compléter mon dernier commit

```
git reset --soft HEAD~1
```

ou

```
git reset --mixed HEAD~1
```

Dans les deux cas on défait le commit en gardant les modifs  
(soit dans le stage et la copie de travail, soit uniquement dans la copie de travail)

# MODIFIER MON DERNIER COMMIT

Copie de travail

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 3

« Photos » 2

« Photos » 1

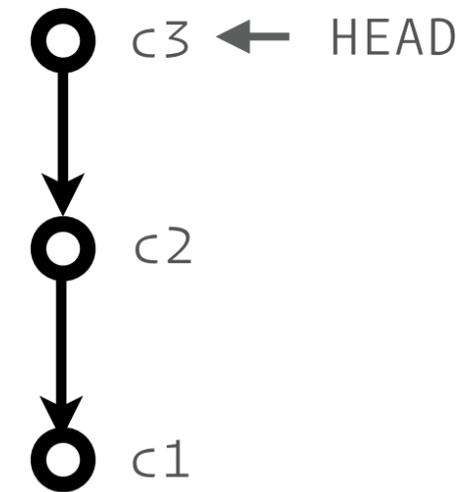
Dépôt local

Commit3

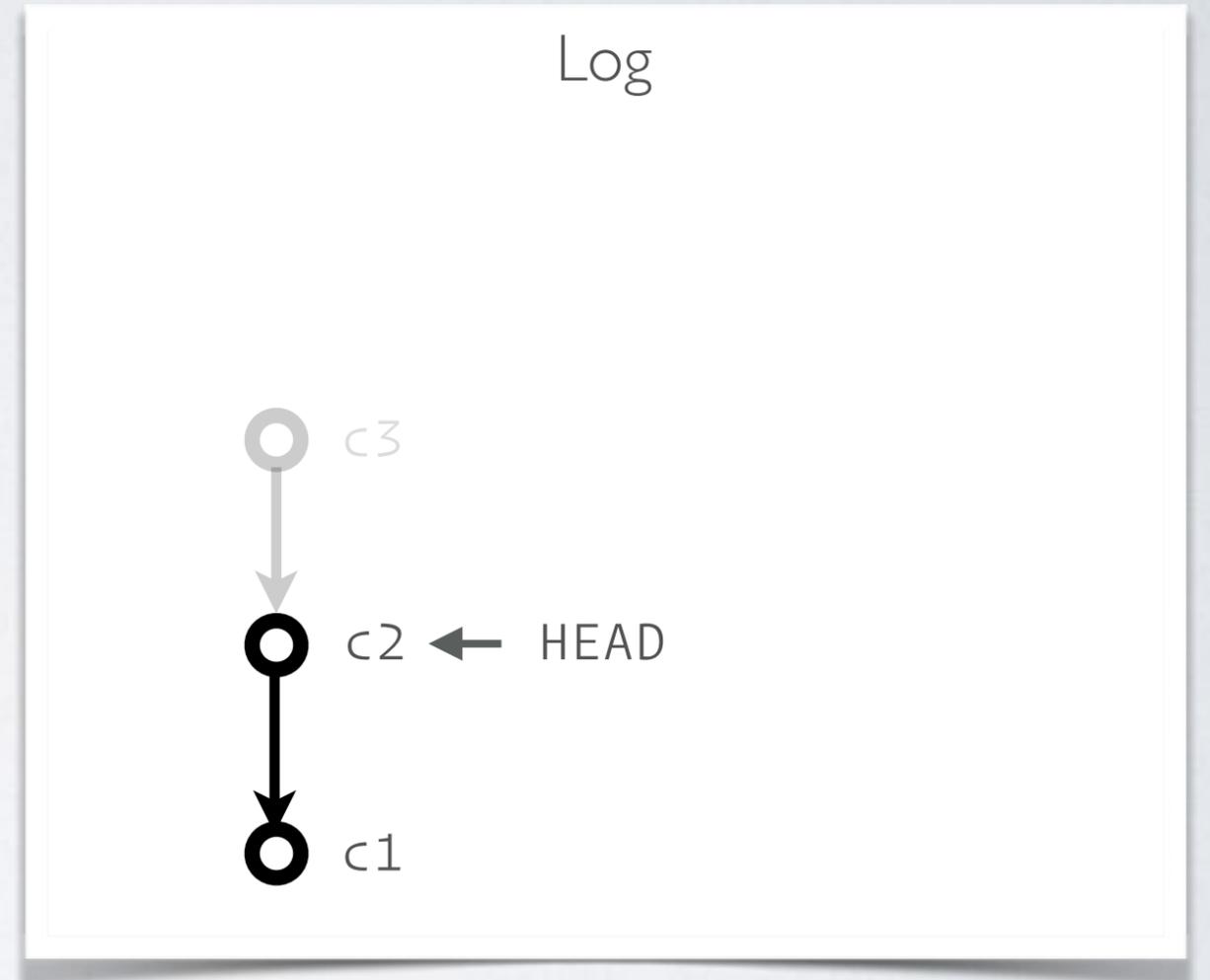
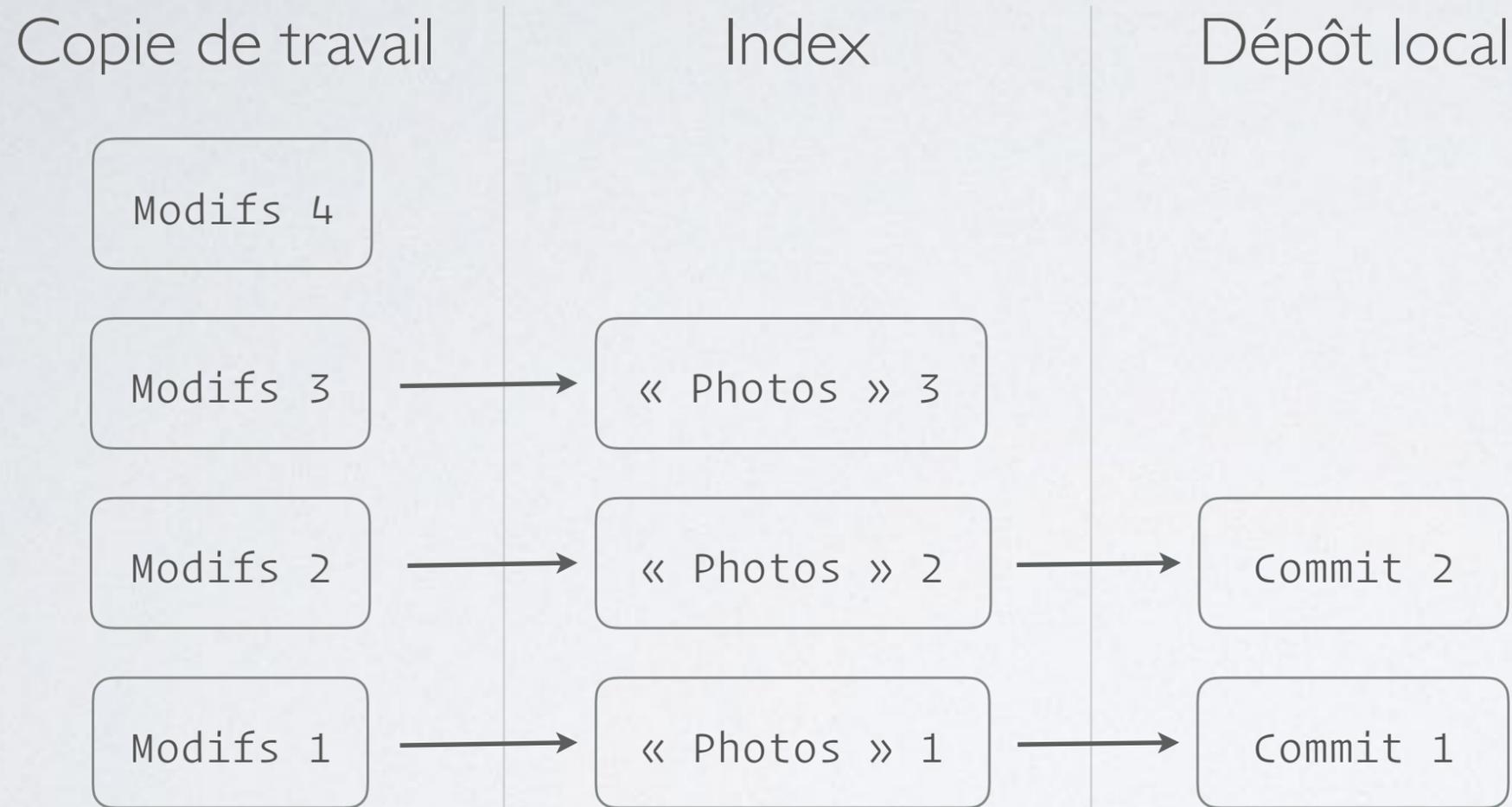
Commit 2

Commit 1

Log



# MODIFIER MON DERNIER COMMIT



```
> git reset --soft c2
```

# MODIFIER MON DERNIER COMMIT

Copie de travail

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 3'

« Photos » 2

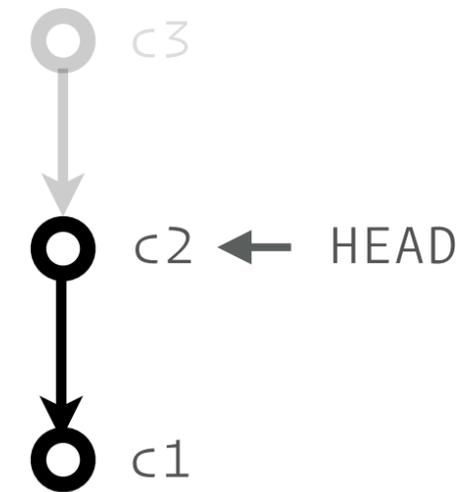
« Photos » 1

Dépôt local

Commit 2

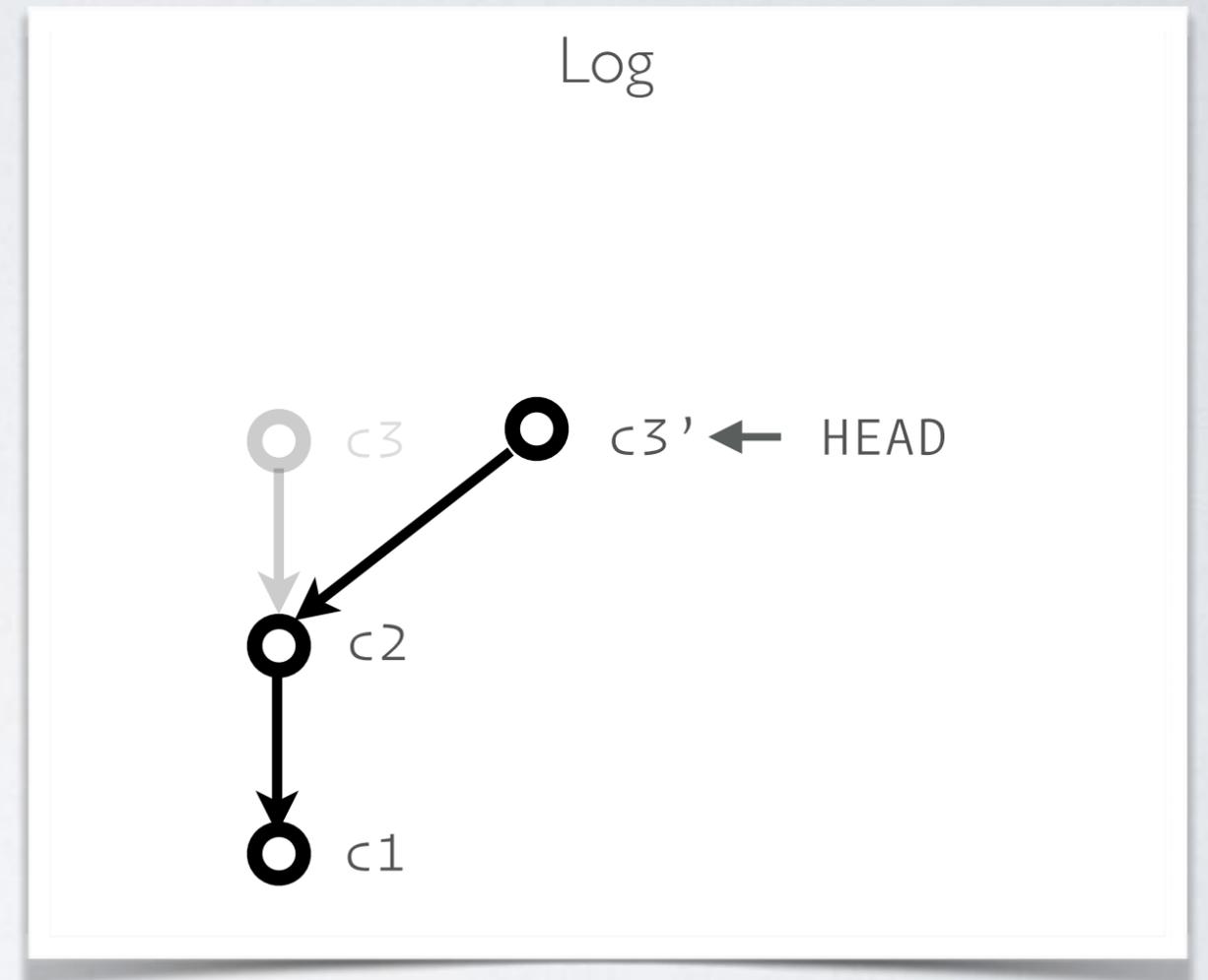
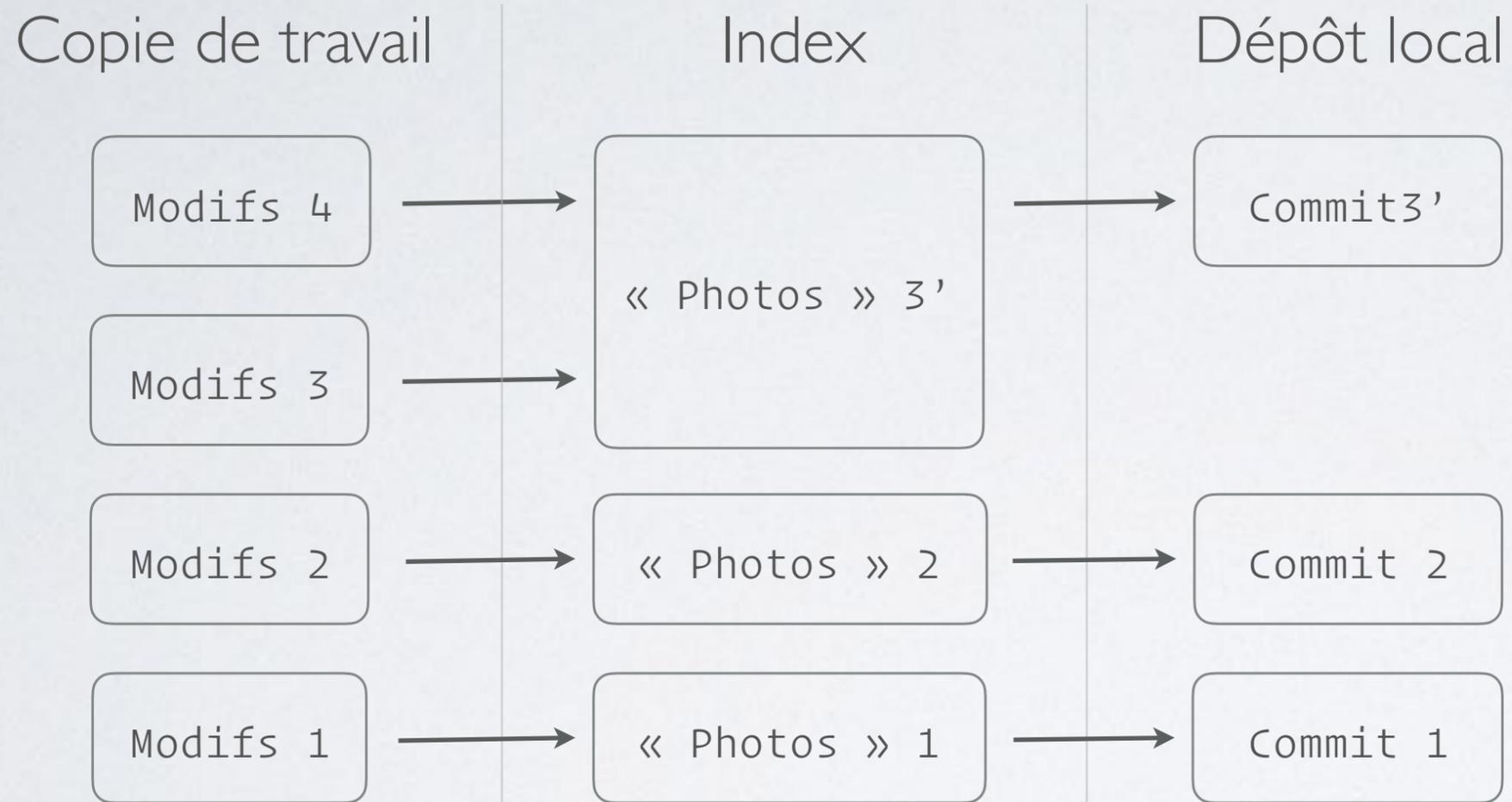
Commit 1

Log



```
> git add ...
```

# MODIFIER MON DERNIER COMMIT



```
> git commit ...
```

# POURQUOI PAS « AMEND » ?

```
git commit --amend ...
```

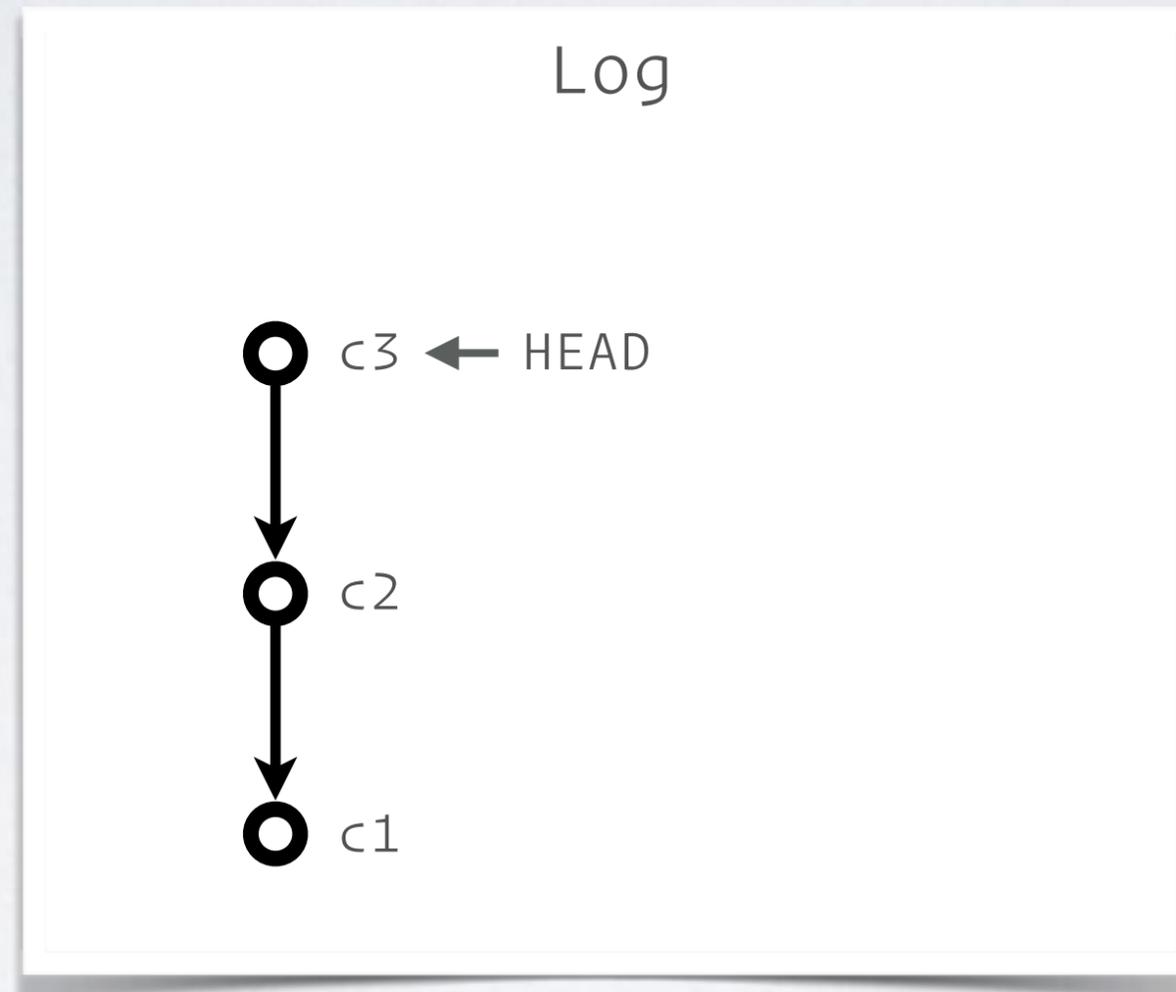
souvent plus rapide pour ajouter des choses au dernier commit  
ou changer certaines méta-données (message, auteur...).

Implicitement ça fait du

```
git reset --soft HEAD~1 && git commit ...
```

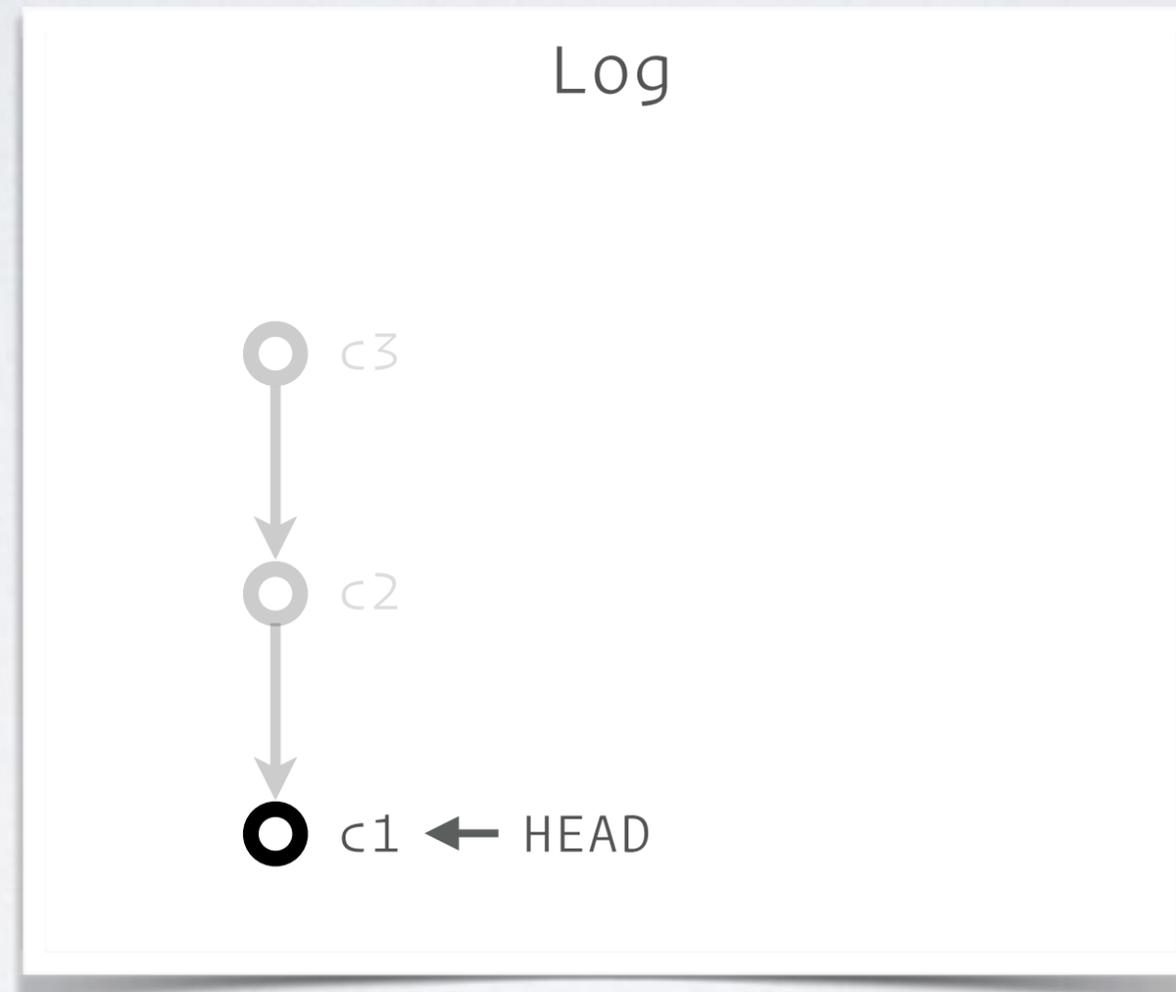
# JE VEUX REGROUPER DES COMMITS

Je m'aperçois que mes 2 derniers commits traitent d'un sujet unique



# JE VEUX REGROUPEZ DES COMMITS

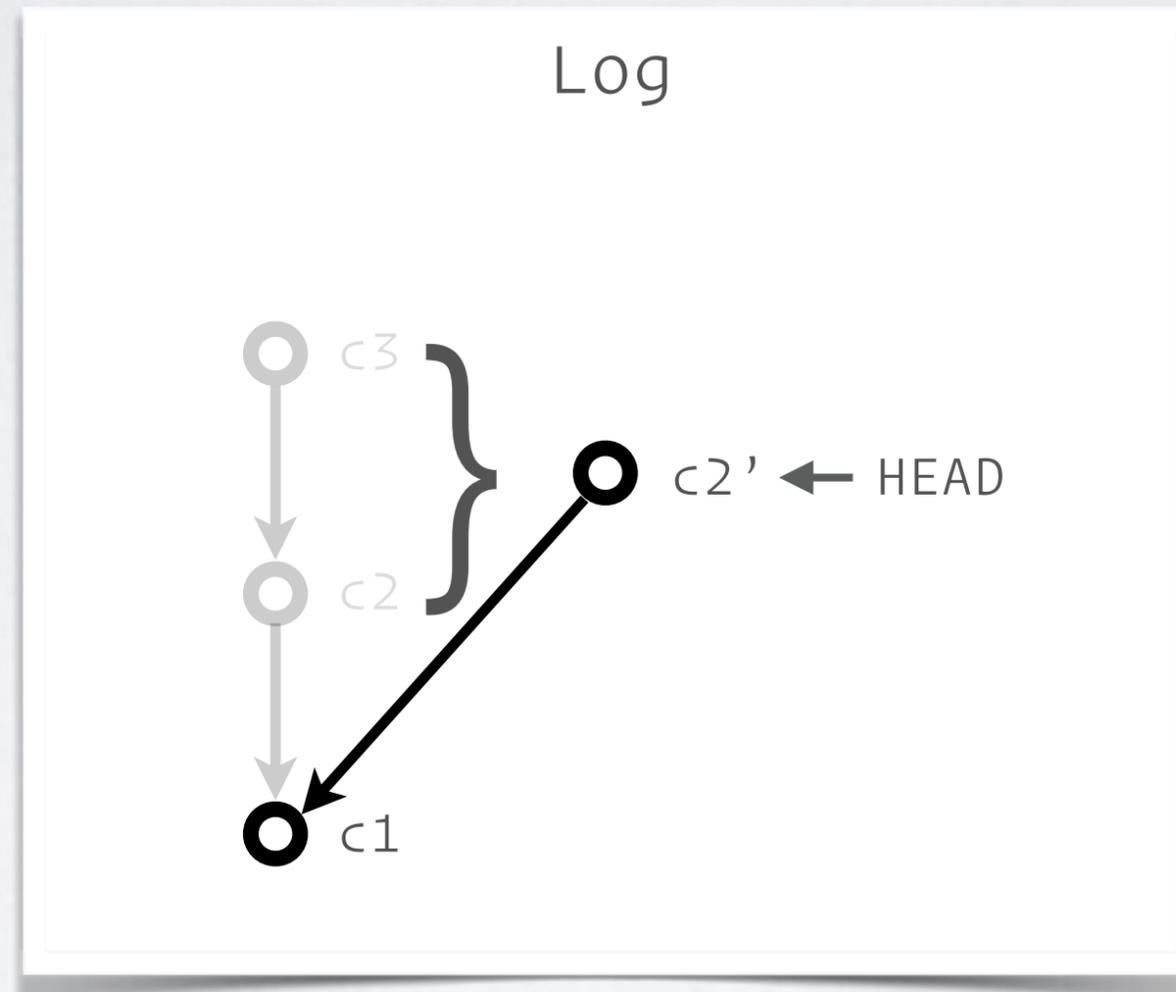
Je m'aperçois que mes 2 derniers commits traitent d'un sujet unique



```
> git reset --soft c1
```

# JE VEUX REGROUPER DES COMMITS

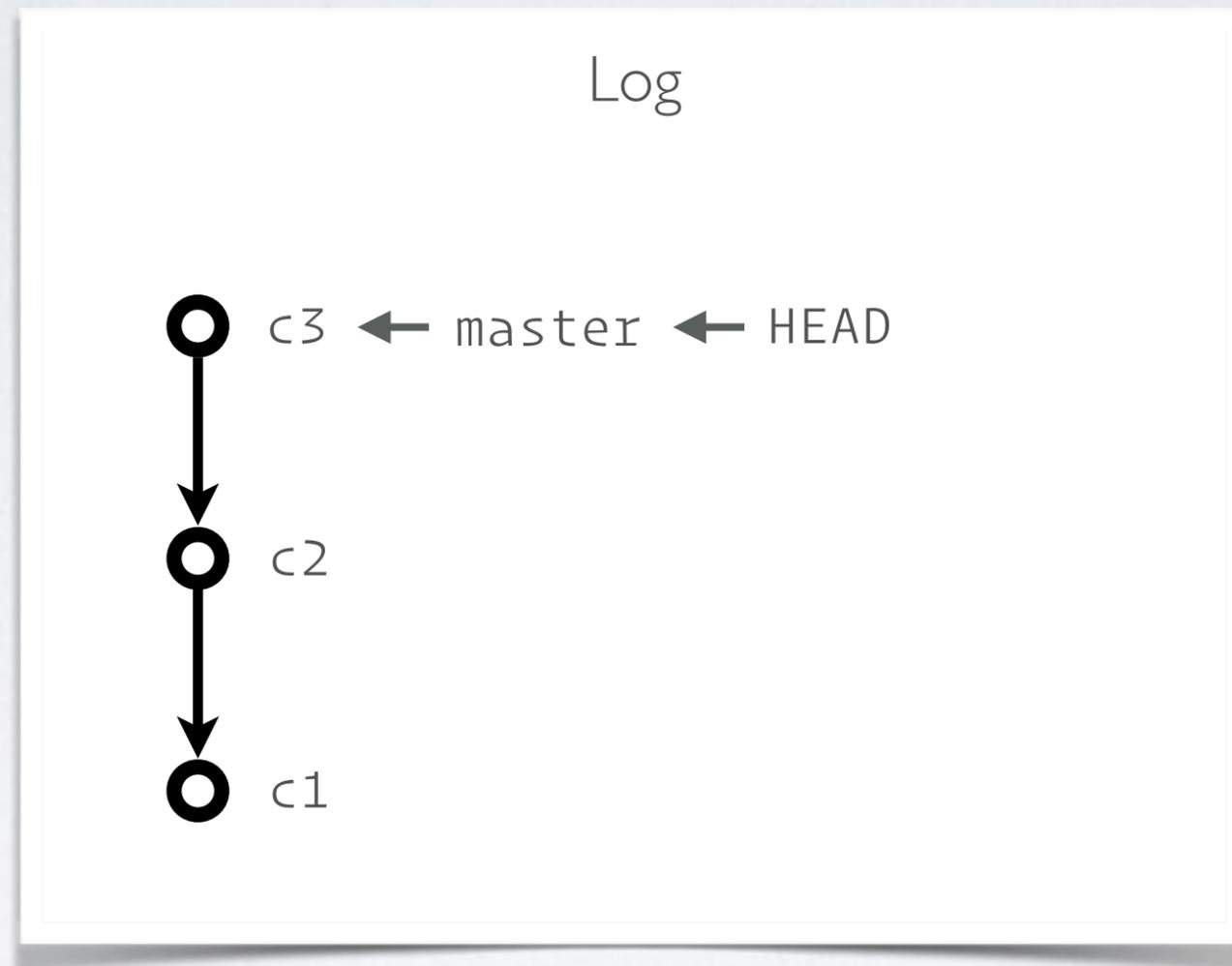
Je m'aperçois que mes 2 derniers commits traitent d'un sujet unique



```
> git commit ...
```

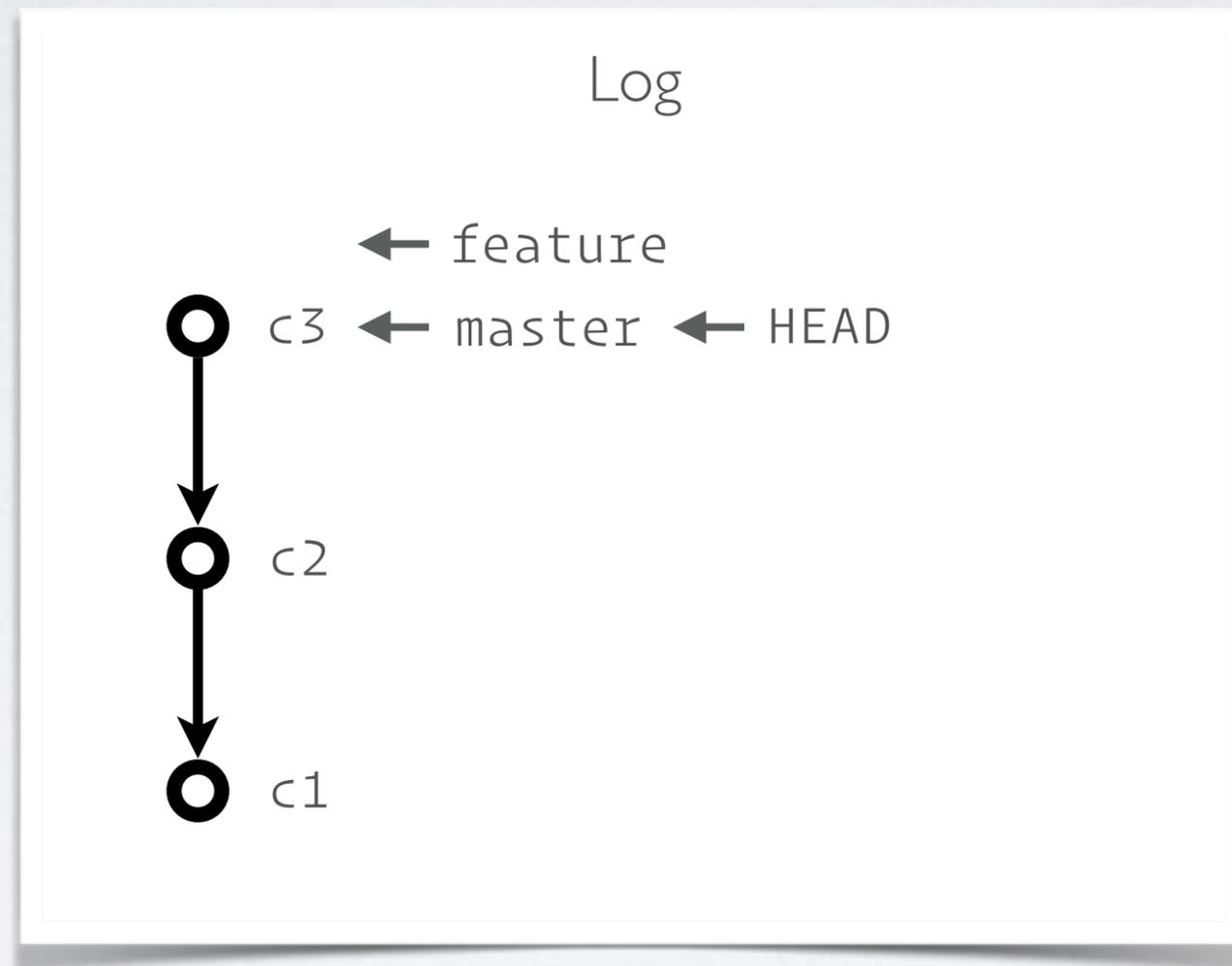
# J'AURAIS DÛ FAIRE UNE BRANCHE PLUS TÔT

Je m'aperçois que mes 2 derniers commits devraient être sur une branche dédiée



# J'AURAIS DÛ FAIRE UNE BRANCHE PLUS TÔT

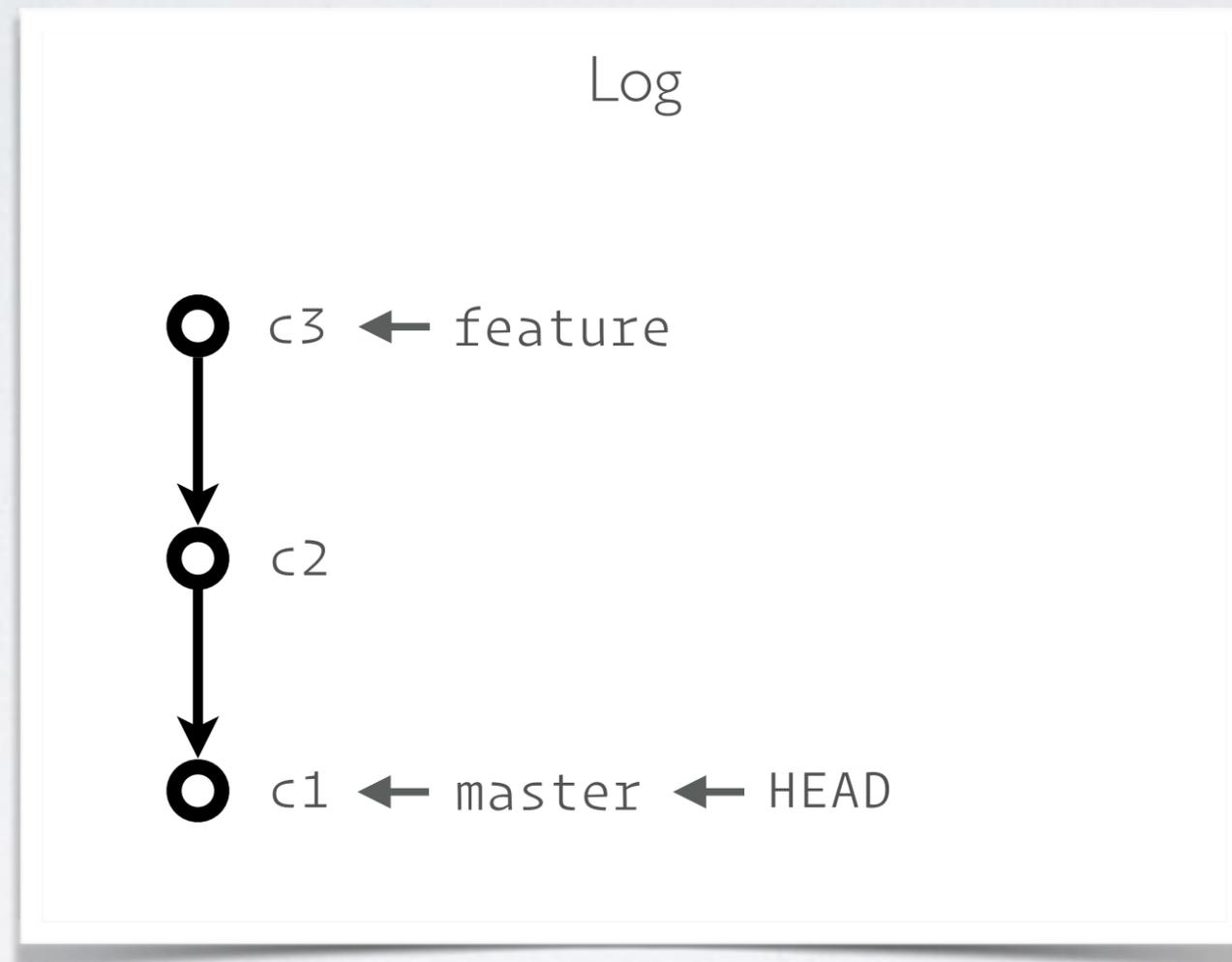
Je m'aperçois que mes 2 derniers commits devraient être sur une branche dédiée



```
> git branch feature
```

# J'AURAIS DÛ FAIRE UNE BRANCHE PLUS TÔT

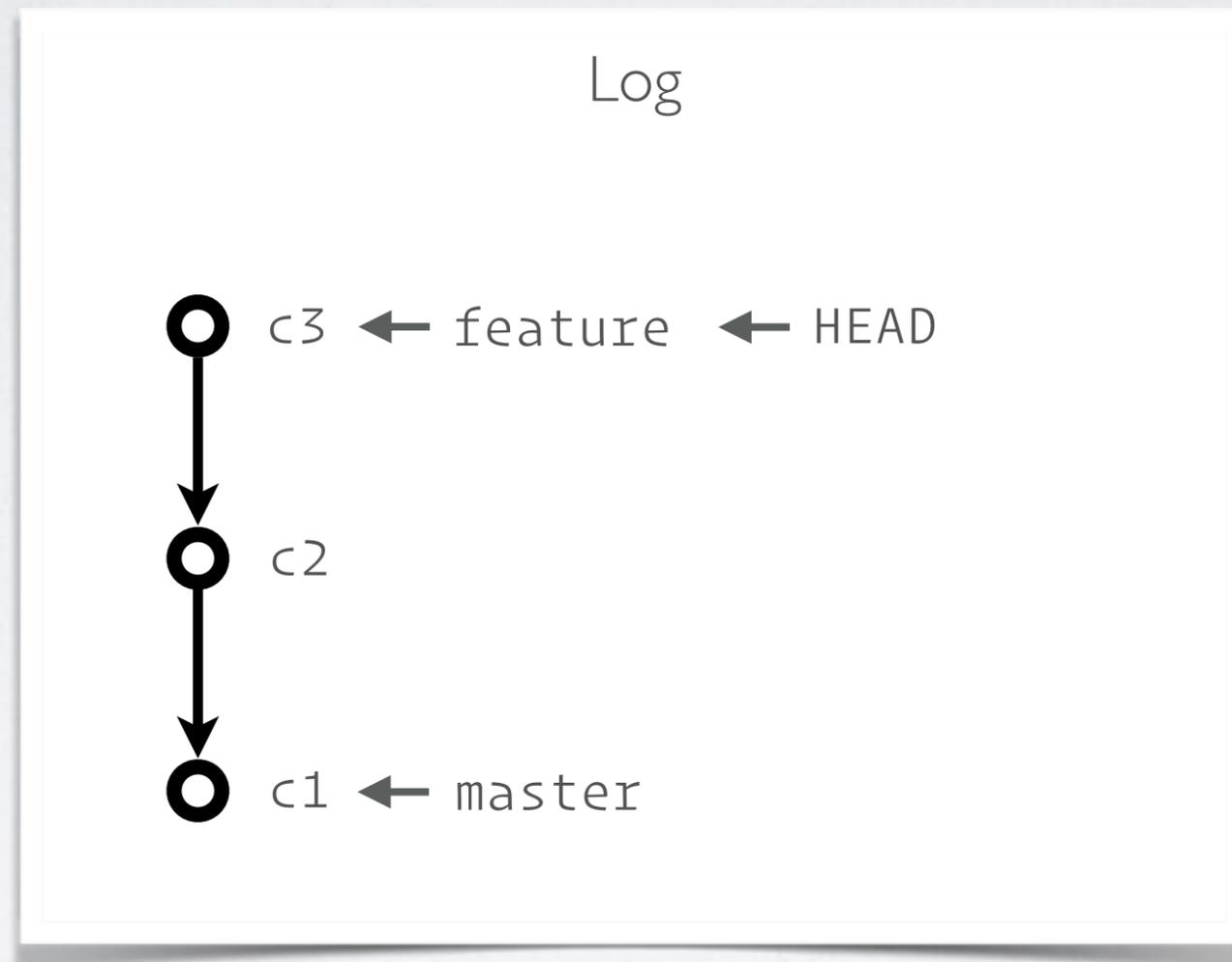
Je m'aperçois que mes 2 derniers commits devraient être sur une branche dédiée



```
> git reset --soft c1
```

# J'AURAIS DÛ FAIRE UNE BRANCHE PLUS TÔT

Je m'aperçois que mes 2 derniers commits devraient être sur une branche dédiée



```
> git checkout feature
```

# PURGER TOUTES LES MODIFS EN COURS

```
git reset --hard [HEAD]
```

défait le stage et la copie de travail pour remettre  
les chemins versionnés à l'état de HEAD

(donc tout sauf les fichiers non trackés)

# GIT RESET --HARD

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

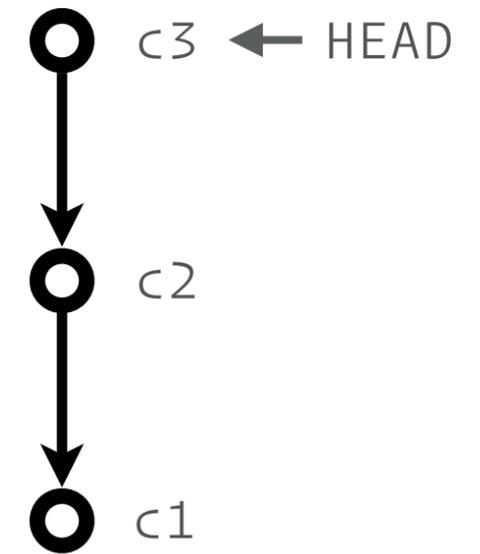
Dépôt local

Commit3

Commit 2

Commit 1

Log



# GIT RESET --HARD

Copie de travail

Modifs 5

Modifs 4

Modifs 3

Modifs 2

Modifs 1

Index

« Photos » 4

« Photos » 3

« Photos » 2

« Photos » 1

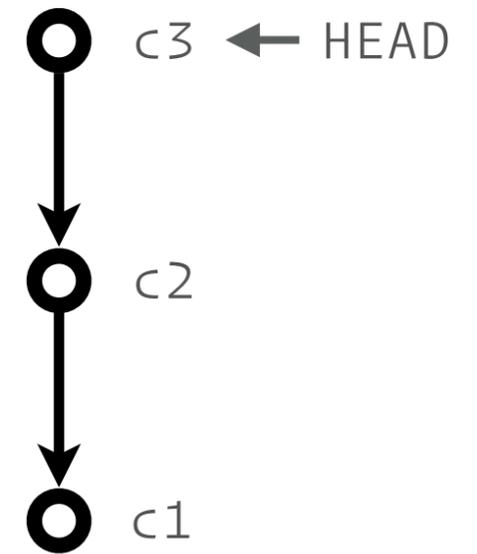
Dépôt local

Commit3

Commit 2

Commit 1

Log



```
> git reset --hard HEAD
```

DÉFAIRE C'EST BIEN, MAIS PEUT-ON  
REFAIRE SI ON S'EST TROMPÉ ?

# CTRL-Y

```
git reflog + git reset
```

Le reflog liste tous les emplacements parcourus par HEAD et les étiquettes de branches.

Si on fait X pas en arrière, on peut les refaire en avant.

Strictement local, purgé par le garbage collector.

# LOG + REFLOG

Comment notre reflog est-il construit **depuis HEAD** ?

Log

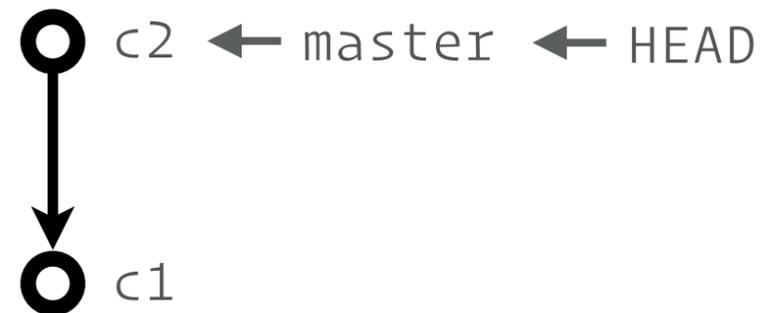
Reflog



# LOG + REFLOG

Comment notre reflog est-il construit **depuis HEAD** ?

Log



Reflog

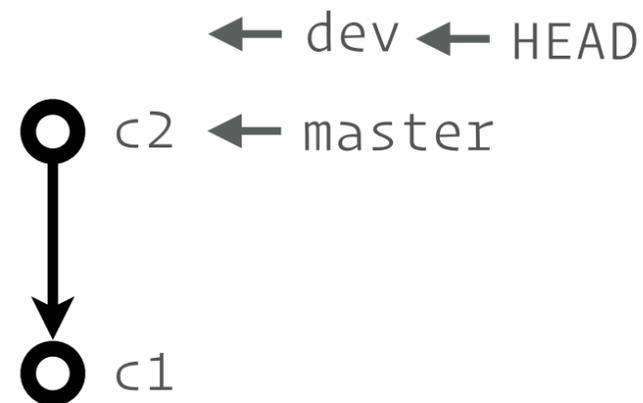
```
c2 HEAD@{0} commit: message de c2
c1 HEAD@{1} commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis HEAD** ?

Log



Reflog

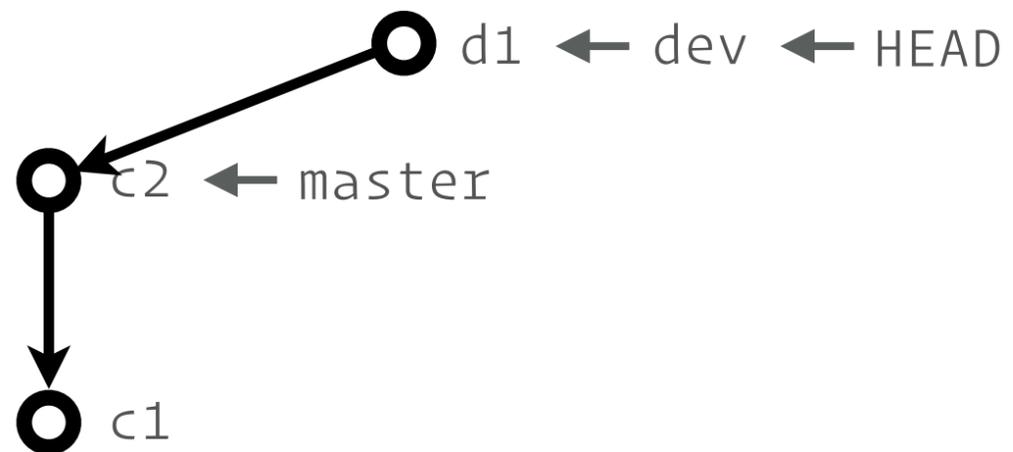
```
c2 HEAD@{0} checkout: moving...to dev
c2 HEAD@{1} commit: message de c2
c1 HEAD@{2} commit: message de c1
```

```
> git checkout -b dev
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis HEAD** ?

Log



Reflog

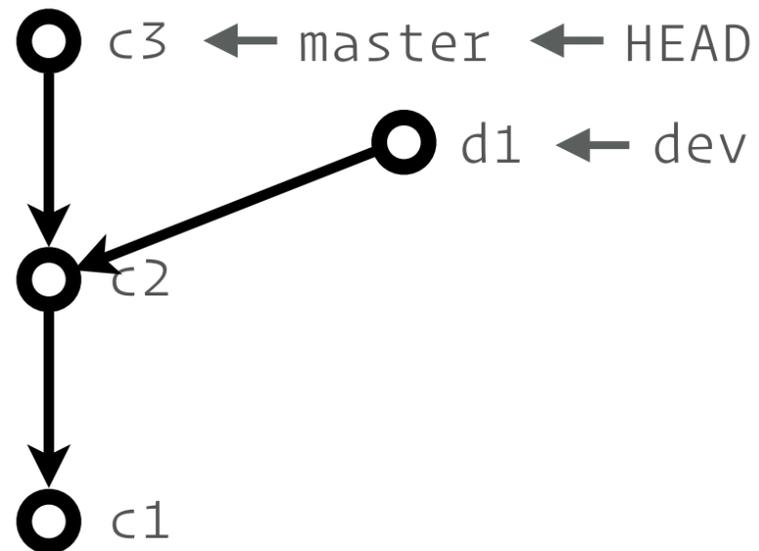
```
d1 HEAD@{0} commit: message de d1
c2 HEAD@{1} checkout: moving...to dev
c2 HEAD@{2} commit: message de c2
c1 HEAD@{3} commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis HEAD** ?

Log



Reflog

```
c3 HEAD@{0} commit: message de c3
c2 HEAD@{1} checkout: moving...to master
d1 HEAD@{2} commit: message de d1
c2 HEAD@{3} checkout: moving...to dev
c2 HEAD@{4} commit: message de c2
c1 HEAD@{5} commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log

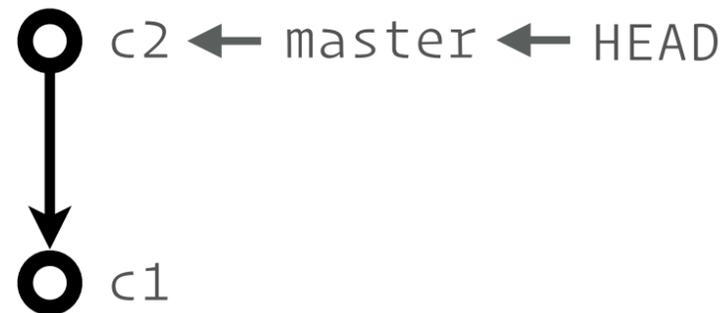
Reflog --branches



# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log



Reflog --branches

```
c2 master@{0} commit: message de c2
c1 master@{1} commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log



Reflog --branches

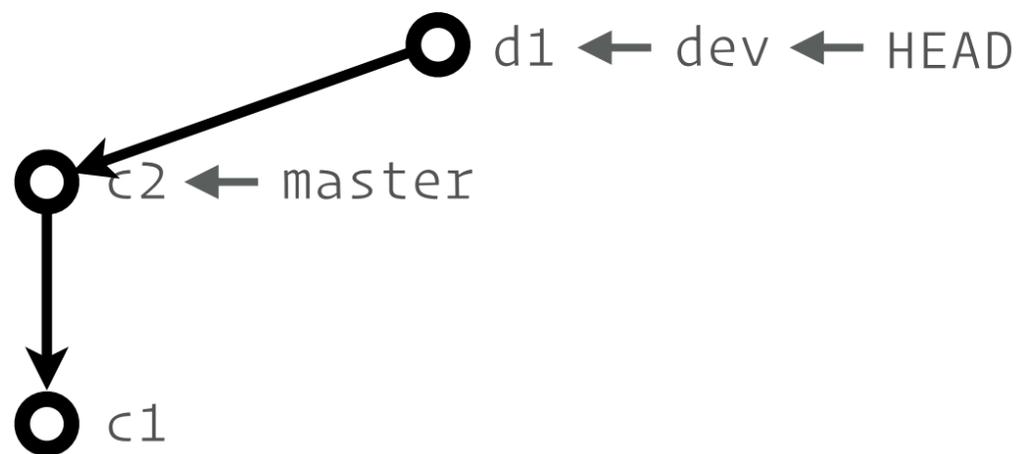
```
c2      dev@{0}  branch: created from HEAD
c2  master@{0}  commit: message de c2
c1  master@{1}  commit: message de c1
```

```
> git checkout -b dev
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log



Reflog --branches

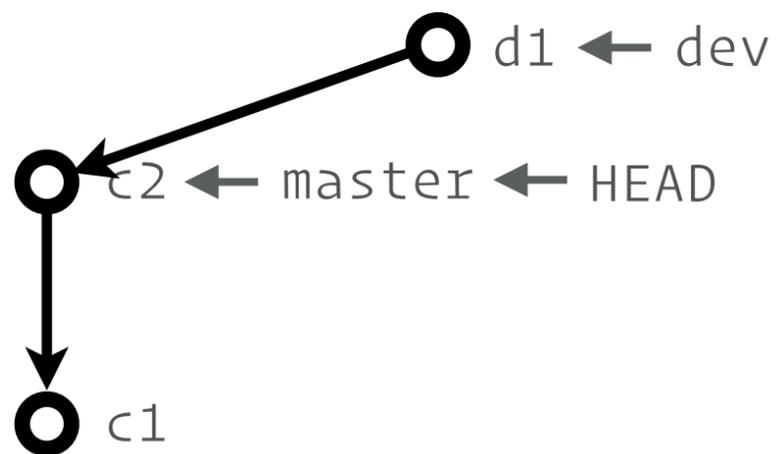
```
d1      dev@{0}  commit: message de d1
c2      dev@{1}  branch: created from HEAD
c2      master@{0}  commit: message de c2
c1      master@{1}  commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log



Reflog --branches

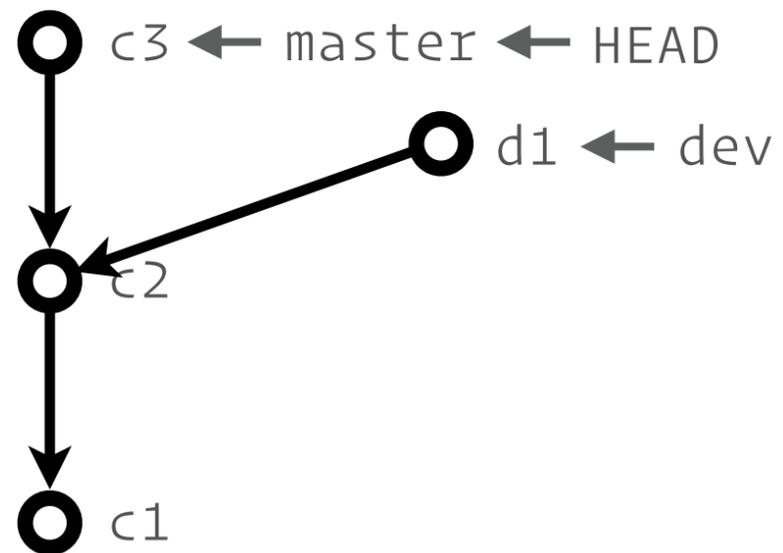
```
d1 dev@{0} commit: message de d1
c2 dev@{1} branch: created from HEAD
c2 master@{0} commit: message de c2
c1 master@{1} commit: message de c1
```

```
> git checkout master
```

# LOG + REFLOG

Comment notre reflog est-il construit **depuis les branches** ?

Log



Reflog --branches

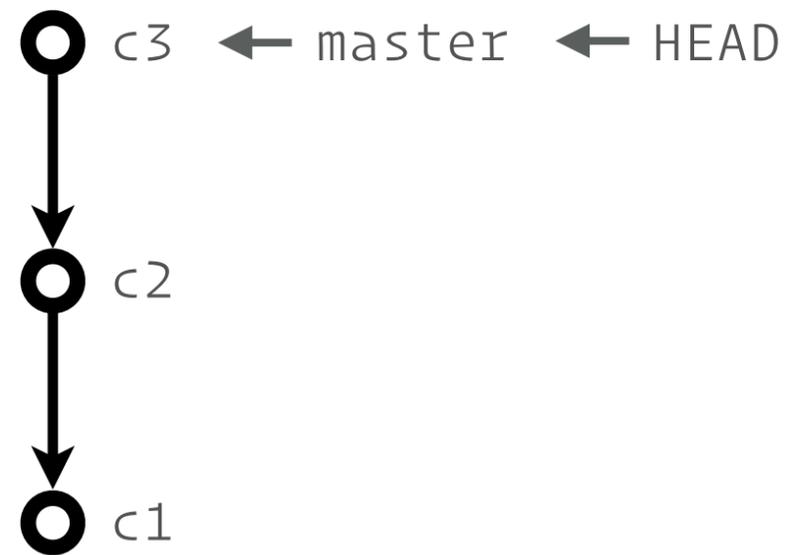
```
c3 master@{0} commit: message de c3
d1 dev@{0} commit: message de d1
c2 dev@{1} branch: created from HEAD
c2 master@{1} commit: message de c2
c1 master@{2} commit: message de c1
```

```
> git commit ...
```

# LOG + REFLOG

Retour vers le futur

Log



Reflog --branches

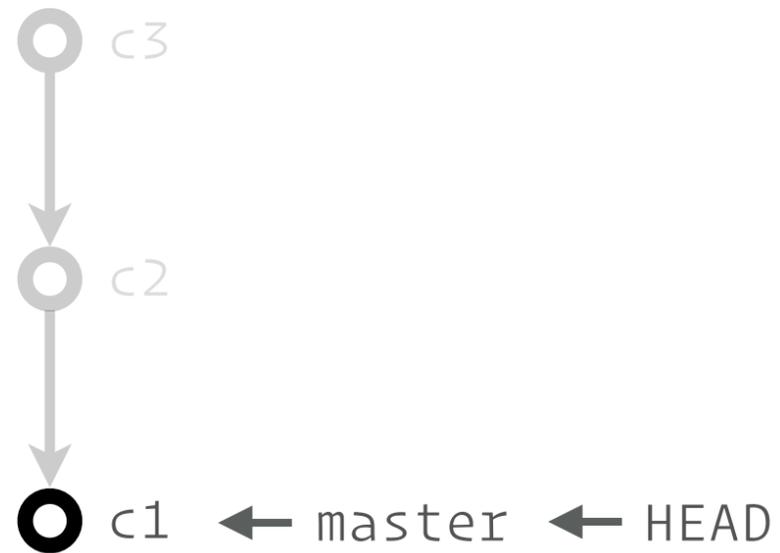
```
c3  master@{0}  commit: message de c3
c2  master@{1}  commit: message de c2
c1  master@{2}  commit: message de c1
```



# LOG + REFLOG

Retour vers le futur

Log



Reflog --branches

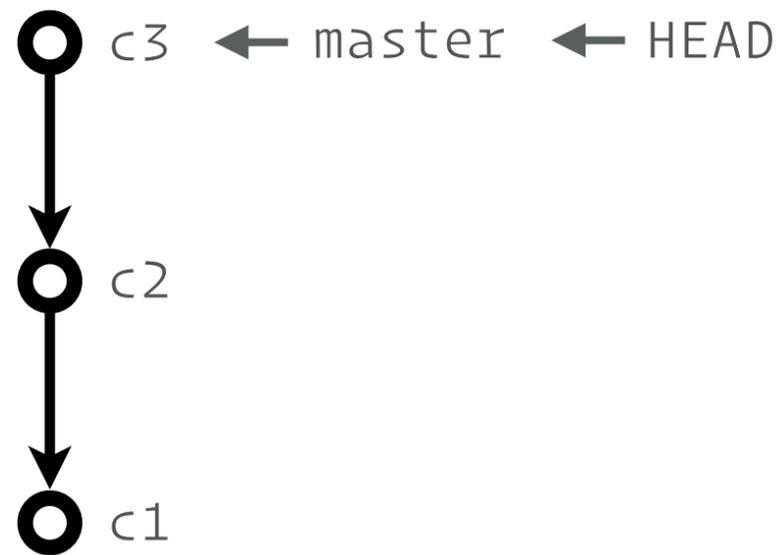
```
c1  master@{0}  reset: moving to c1
c3  master@{1}  commit: message de c3
c2  master@{2}  commit: message de c2
c1  master@{3}  commit: message de c1
```

```
> git reset --keep c1
```

# LOG + REFLOG

Retour vers le futur

Log



Reflog --branches

```
c3  master@{0}  reset: moving to c3
c1  master@{1}  reset: moving to c1
c3  master@{2}  commit: message de c3
c2  master@{3}  commit: message de c2
c1  master@{4}  commit: message de c1
```

```
> git reset --keep master@{1}
```

# ÇA MARCHE AVEC TOUT...

On peut défaire/refaire de actions type

- ★ commit,
- ★ reset,
- ★ merge,
- ★ rebase
- ★ ...

# ÇA MARCHE AVEC TOUT...

On peut défaire/refaire de actions type

- ★ commit,
- ★ reset,
- ★ merge,
- ★ rebase
- ★ ...



# POUR ALLER PLUS LOIN...



<https://bit.ly/pw19-article-reset>



<https://bit.ly/pw19-video-reset>



<http://bit.ly/PW19-reset-slides>

# MERCI

Avec un peu de chance on a du temps pour des questions 🐱

🐦 @mbrehin 🐦