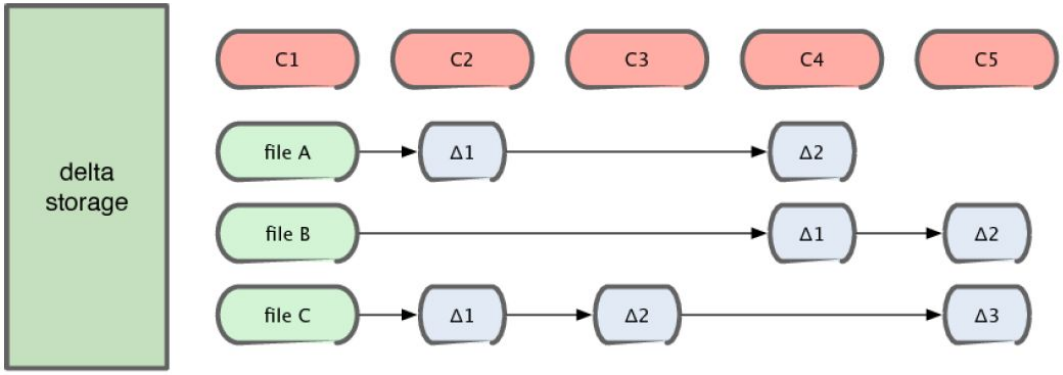
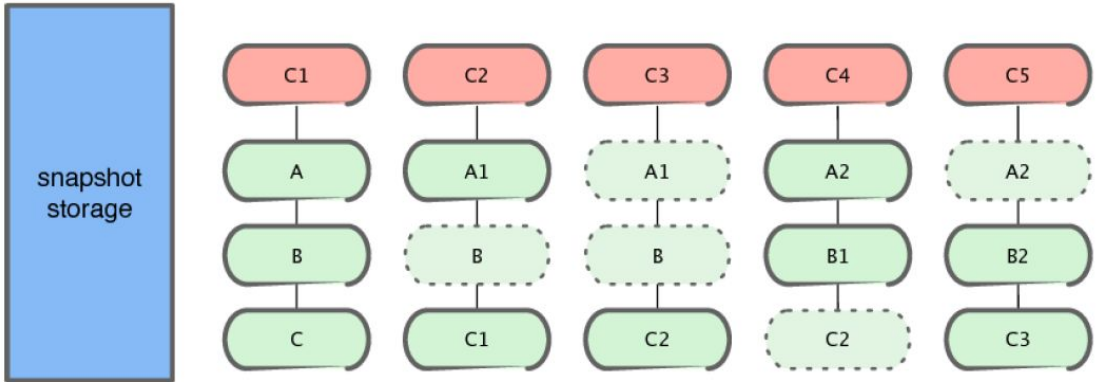
Git is choupi

Paris Web

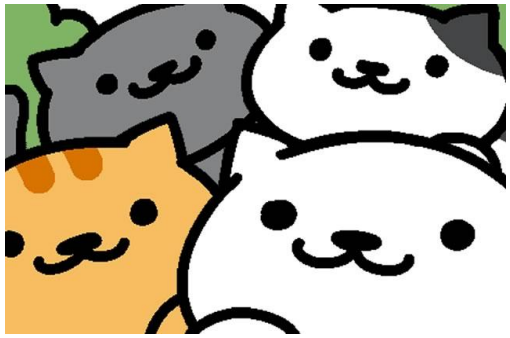
Dualité snapshot / changeset





Concept 1/3 : Notion d'objets / de SHA-1

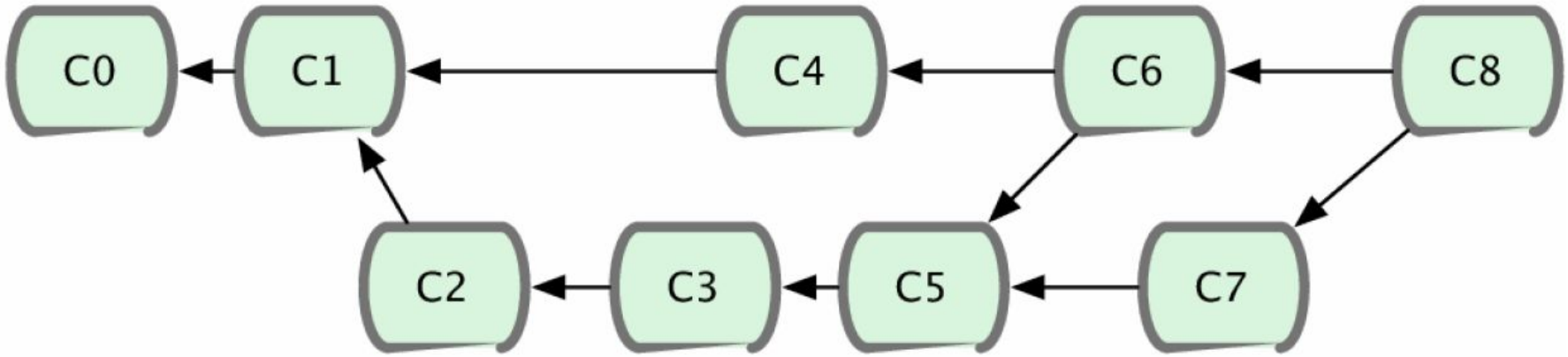
- Tout contenu est un objet.
- Tout objet a un unique SHA-1.



```
$ git log
bd6d984 Make getElementsByClass() work without given tag
1e542e4 Merge branch 'master' of github.com:splitbrain/dokuwiki
b704da0 Skipping plural form in plugin installed message
a1dee2b Fix some bugs and glitches in (mediamanager) tree
2d57e11 Merge pull request #50 from campino2k/master
```

Concept 2/3 : Notion de GOA

La relation entre les commits est un **graphe orienté acyclique**.

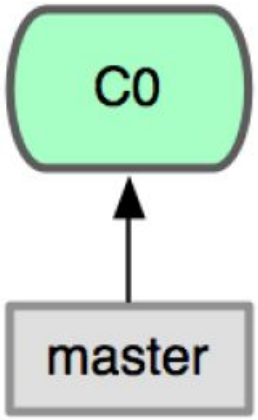


« Le graphe des commits »

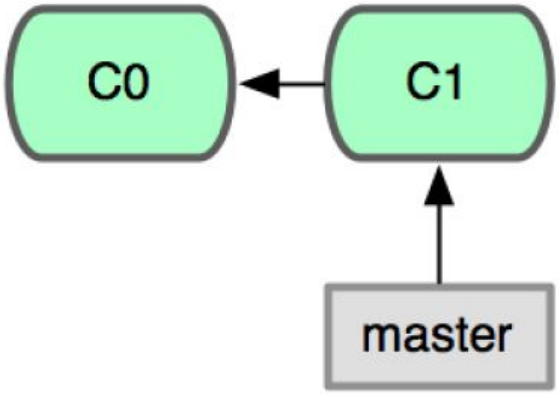
Exemple d'opérations sur le GOA

- **Commit** : Continuer le graphe
- **Merge** : Modélise la fusion
- **Rebase** : Simplifie le graphe
- **Squash** : Nettoie le graphe
- **Cherry-pick** : Déplace un noeud

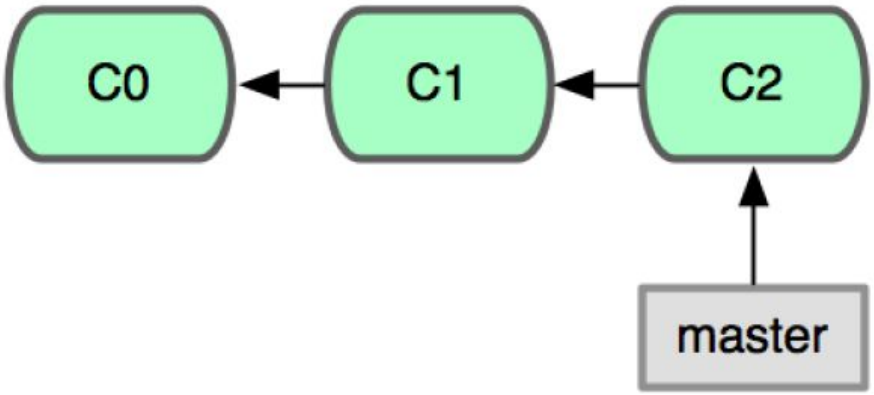
Commit : continuer le graphe



Commit : continuer le graphe



Commit : continuer le graphe



Manipulation

git init : créer un **dépôt** vide dans le dossier courant

Apparition d'un dossier .git

Créer un fichier puis **git add .** et **git commit**

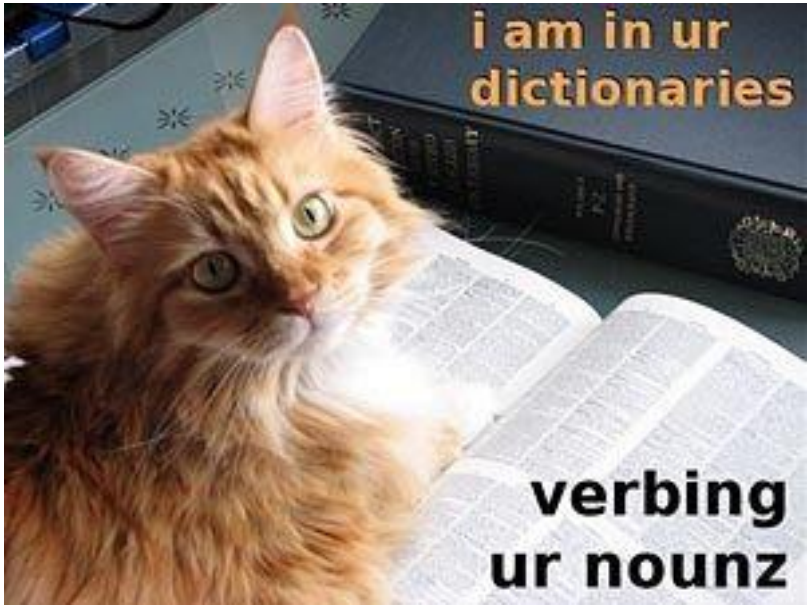
Regarder ce qu'il y a dans le .git

Vocabulaire « technique »

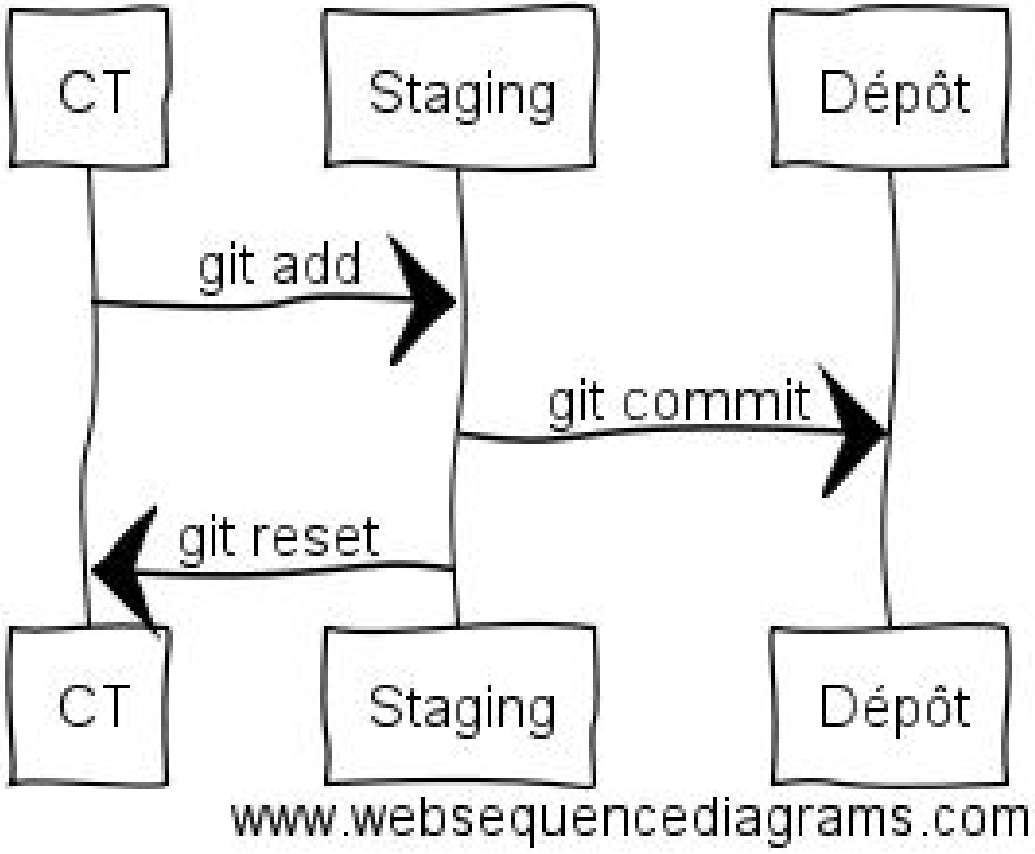
Dépôt : le dossier .git

Copie de travail : les fichiers que vous modifiez

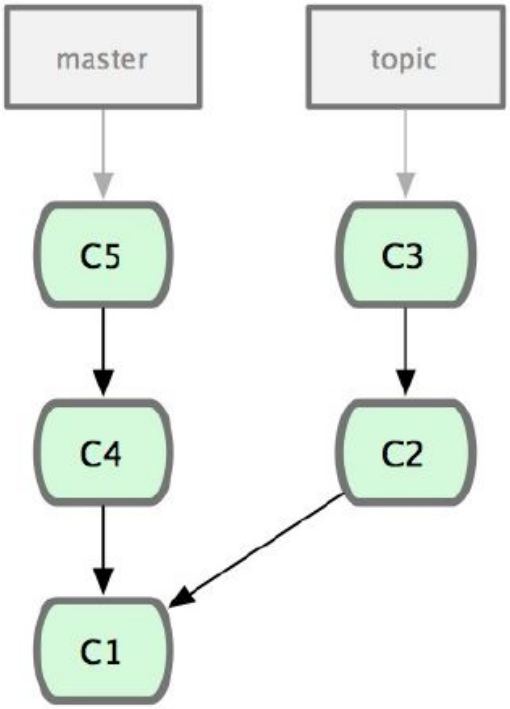
Staging / index : zone de transit entre les deux



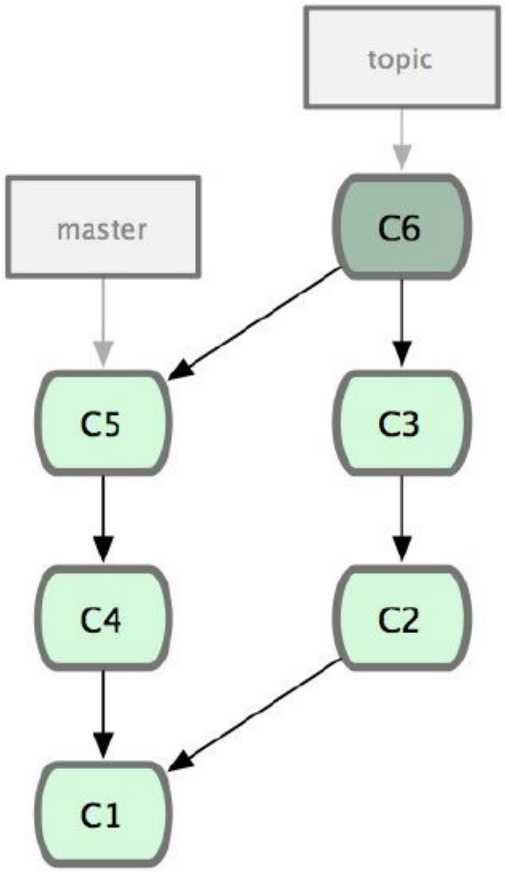
Passer de l'un à l'autre



Merge : fusion

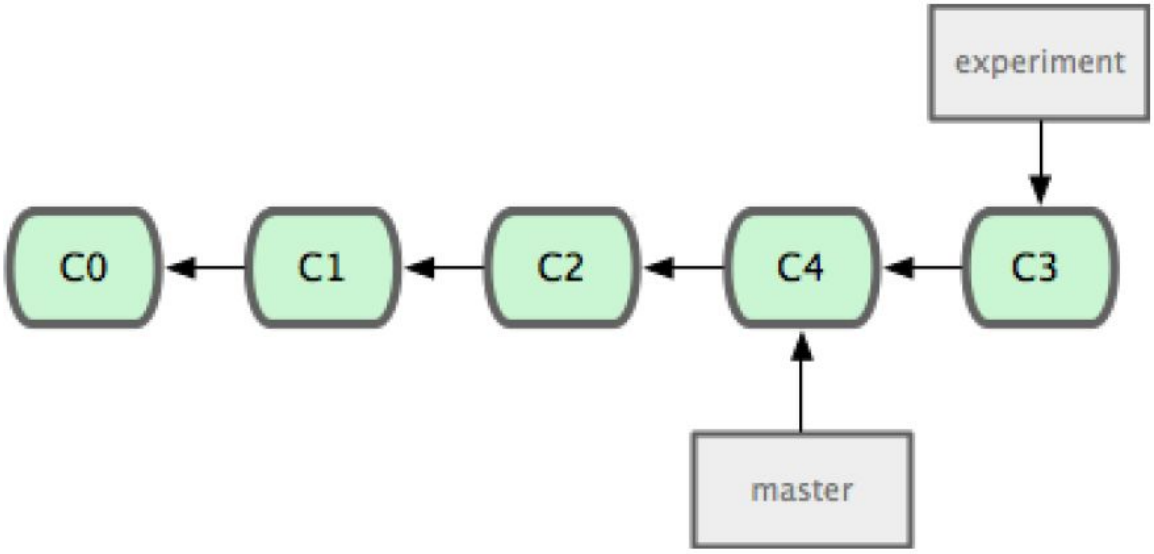


Merge : fusion

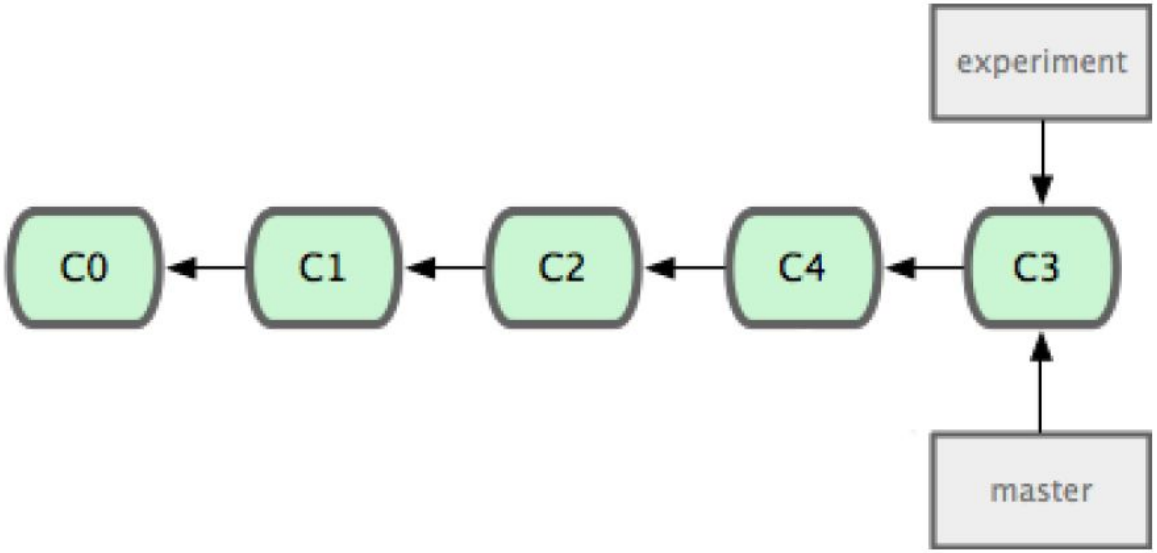


```
git merge master
```

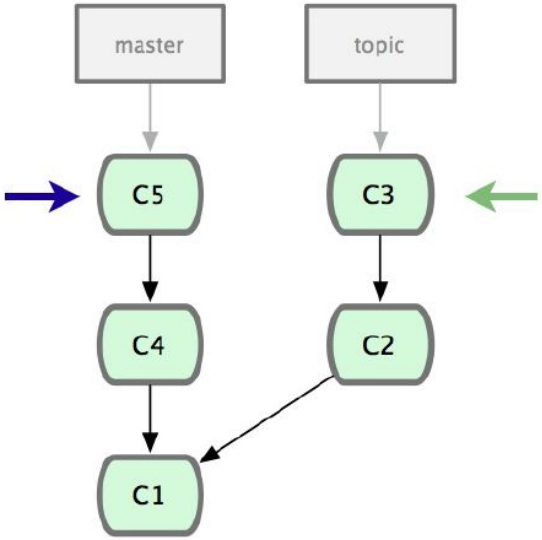
Merge : le cas du fast forward



Merge : le cas du fast forward

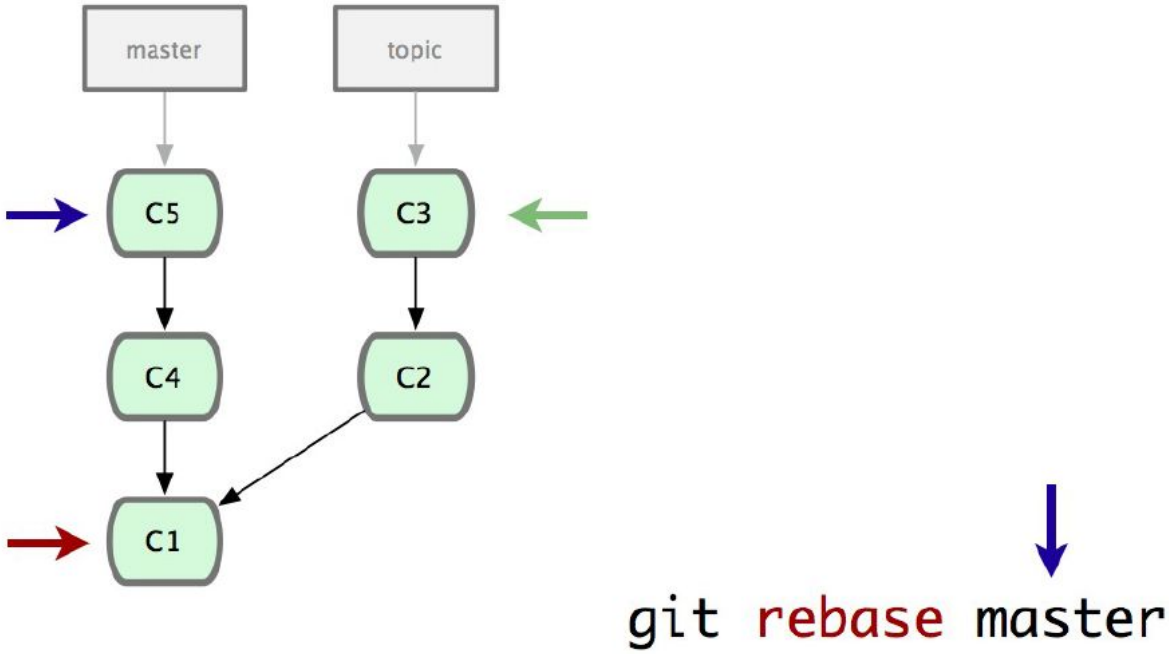


Rebase: simplifie

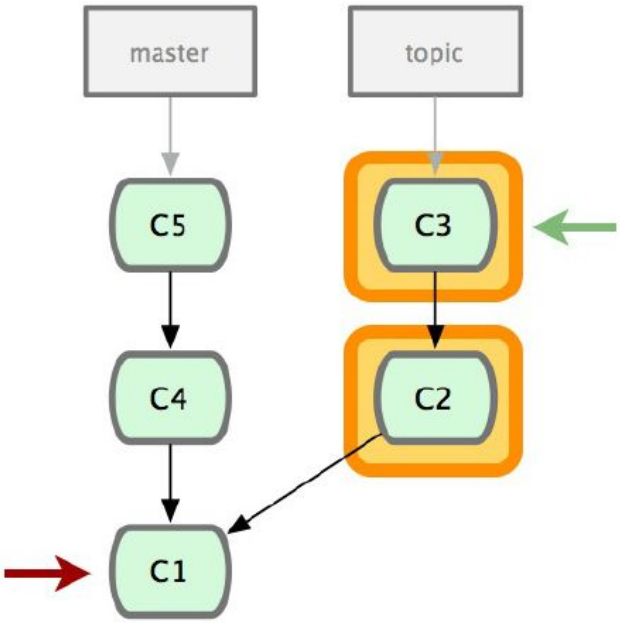



```
git rebase master
```

Rebase : simplifie

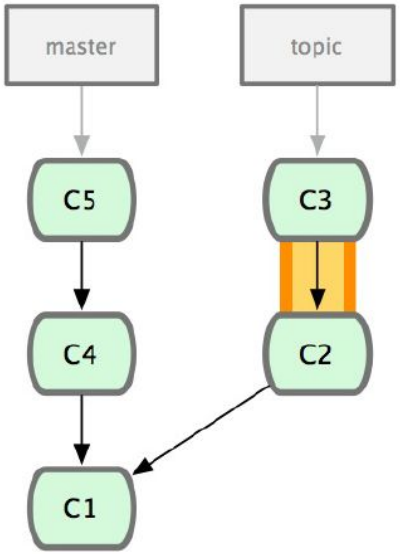


Rebase : simplifie




```
git rebase master
```

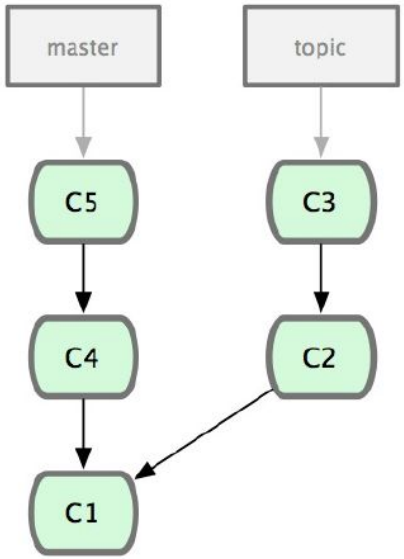
Rebase : simplifie

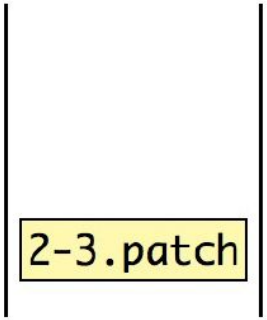



```
git diff c2 c3 > 2-3.patch
```

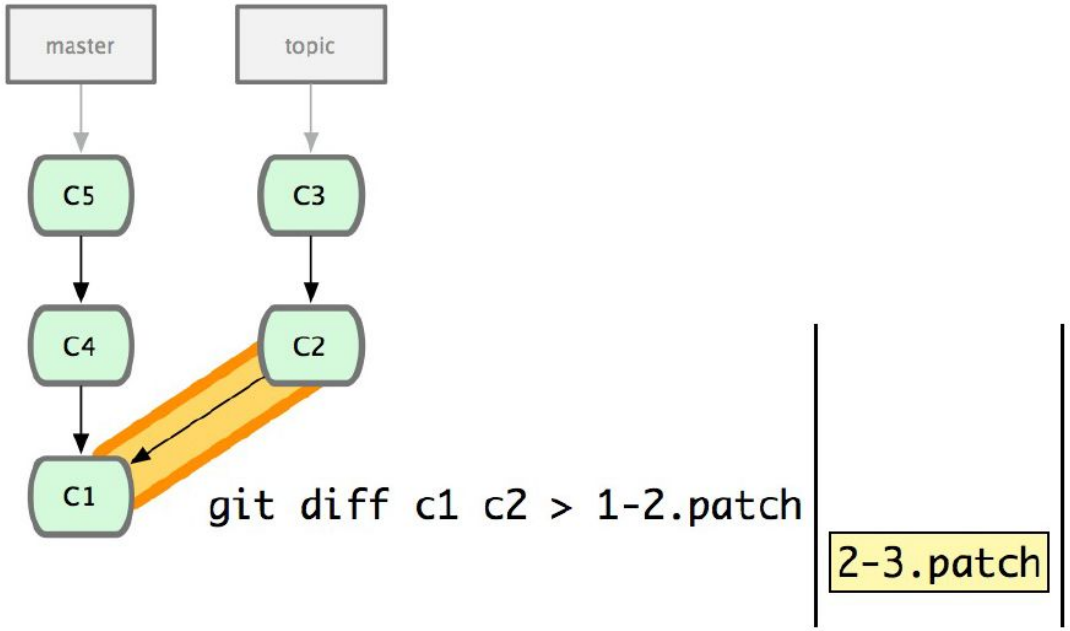
```
git rebase master
```

Rebase : simplifie

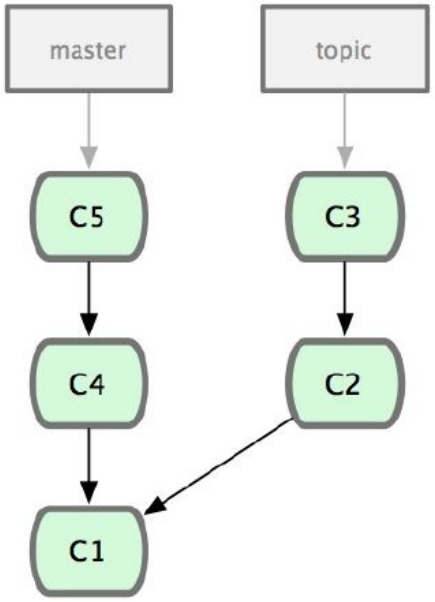


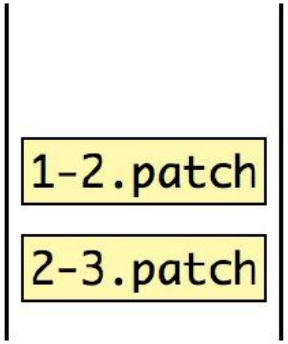


Rebase : simplifie

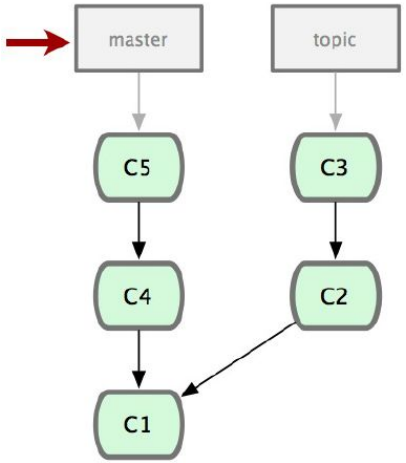


Rebase : simplifie

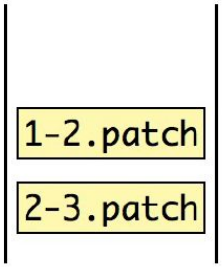




Rebase : simplifie

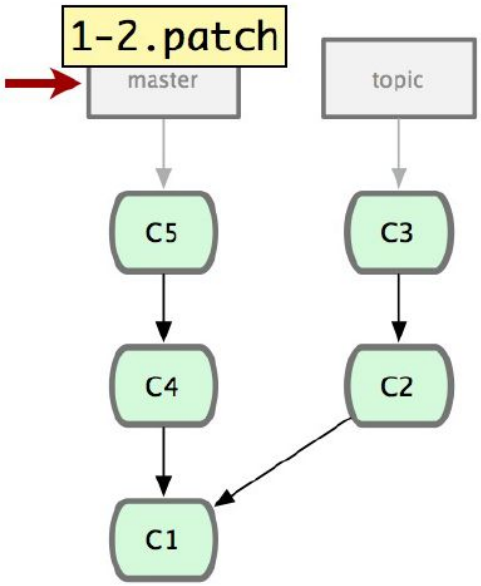


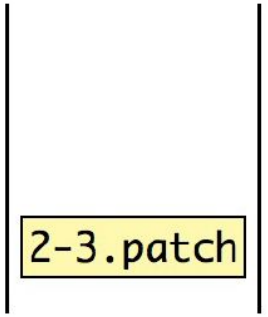

```
git rebase master
```



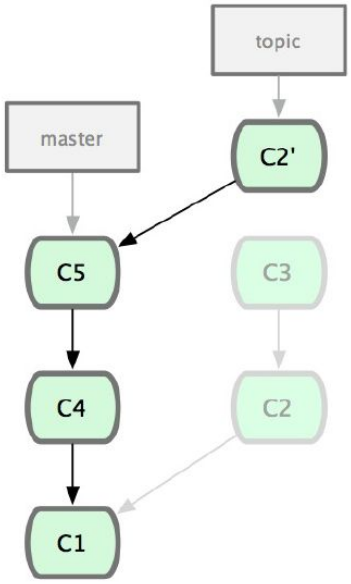
Rebase : simplifie

```
git rebase master
```

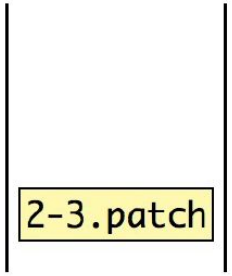




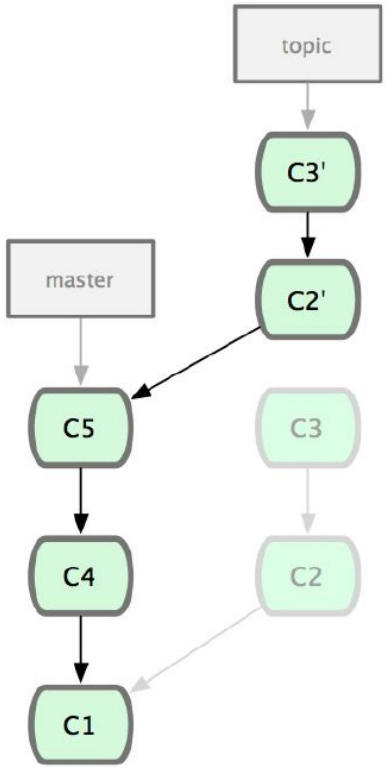
Rebase : simplifie




```
git rebase master
```

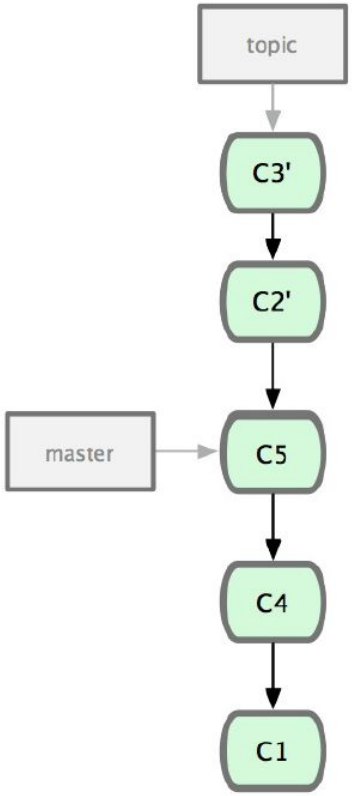


Rebase : simplifie



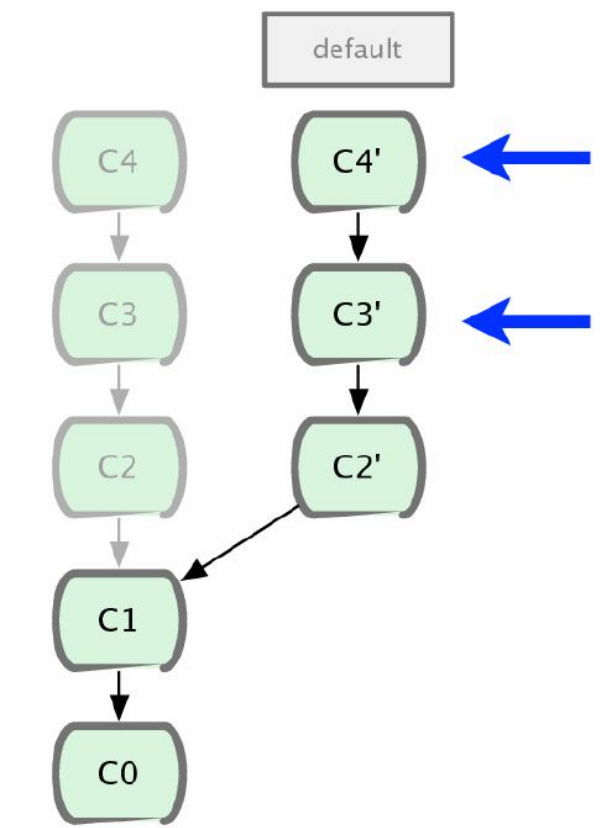

```
git rebase master
```

Rebase : simplifie

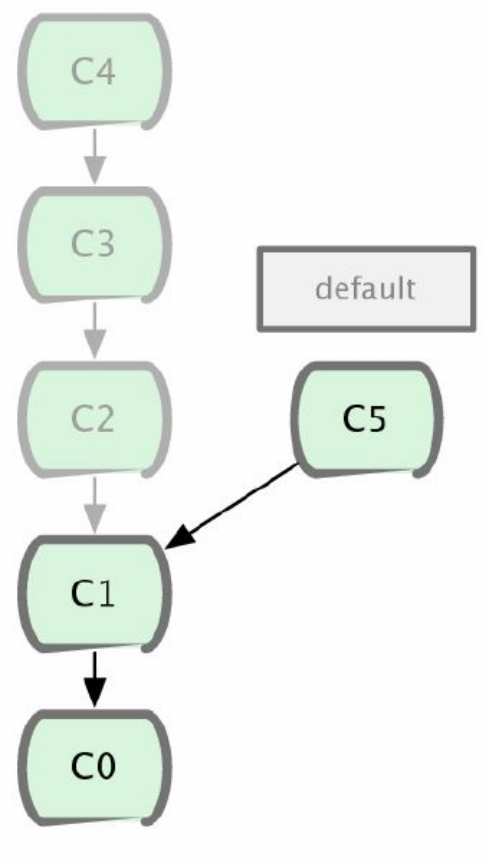



```
git rebase master
```

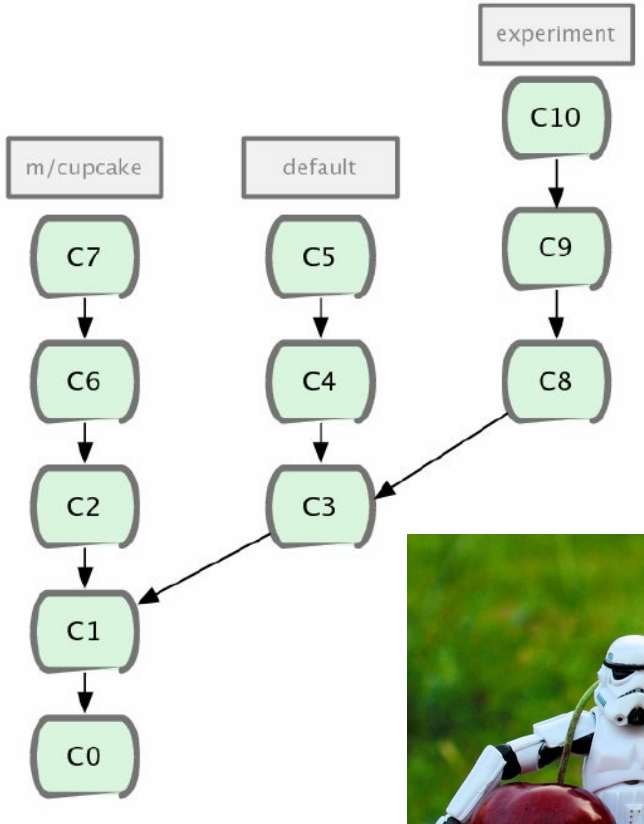
Squash : nettoie



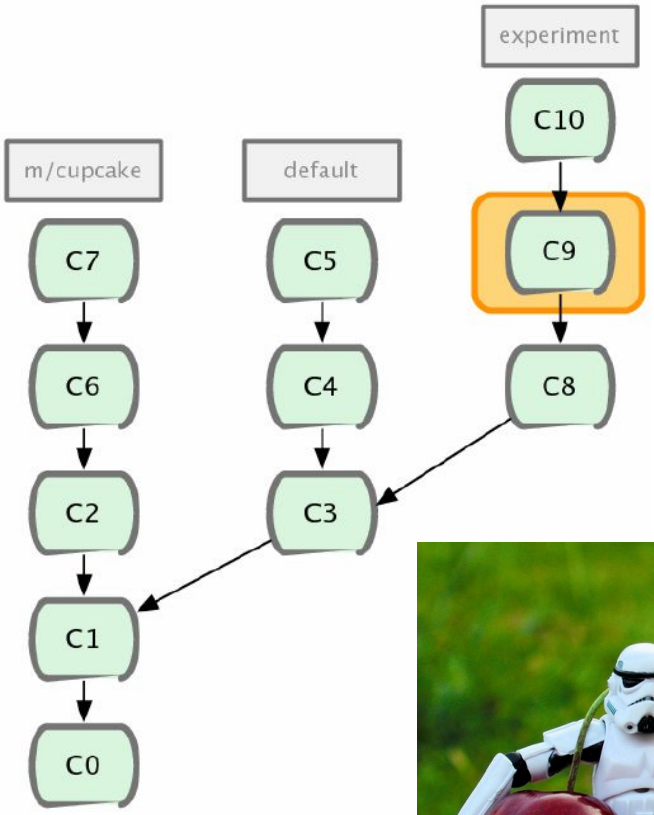
Squash : nettoie



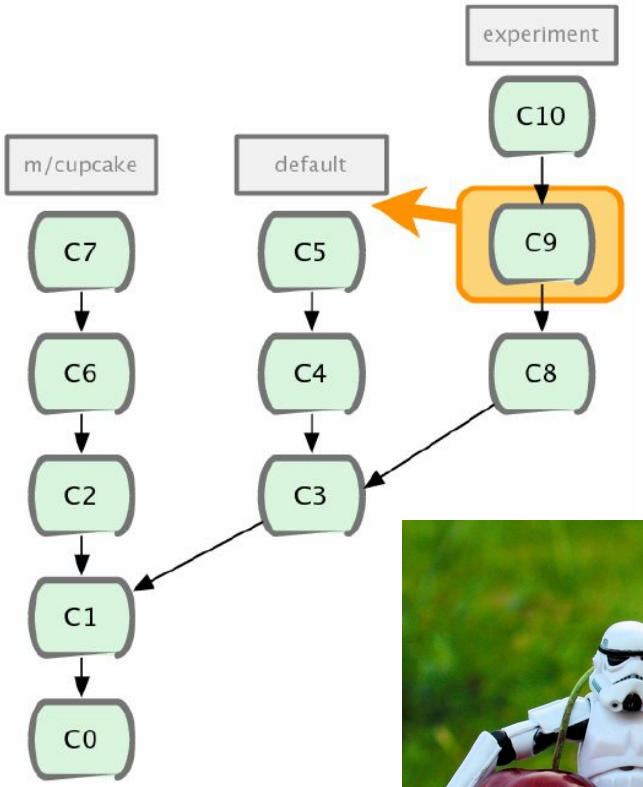
Cherry-pick : déplace un noeud



Cherry-pick : déplace un noeud

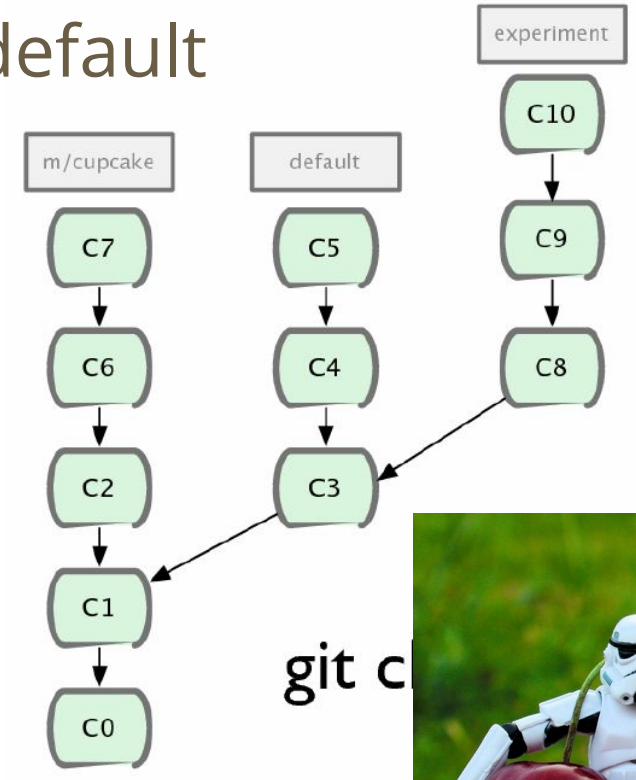


Cherry-pick : déplace un noeud



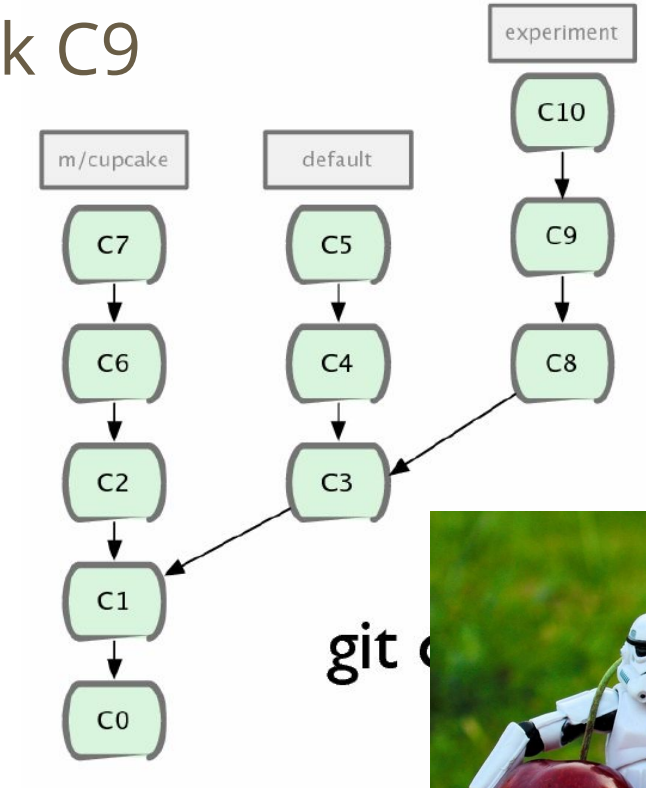
Cherry-pick : déplace un noeud

git checkout default

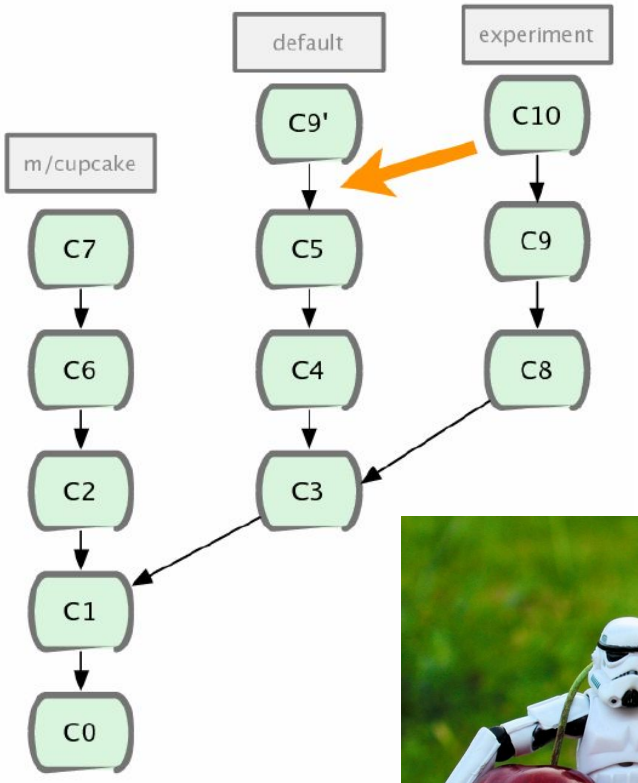


Cherry-pick : déplace un noeud

git cherry-pick C9



Cherry-pick : déplace un noeud



Synthèse des 2 premiers épisodes

- Les commits sont reliés dans un GOA
- On peut manipuler ce GOA
- Les objets sont immutables
- Les objets ne sont pas effacés

Se déplacer dans le GOA

Exemple de SHA-1

de154d0635c8bc4e09a2d76fdda345ba75fd3ecb

Se déplacer dans le GOA

Exemple de SHA-1

de154d0635c8bc4e09a2d76fdda345ba75fd3ecb

Se déplacer dans le GOA

Exemple de SHA-1

?

Concept 3/3 : notion de références

Comparable au DNS:

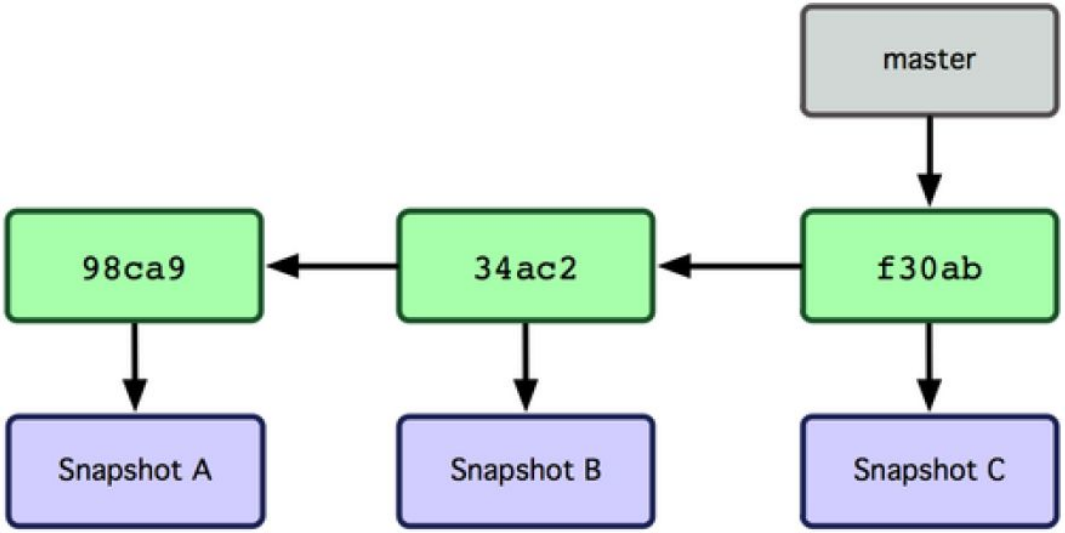
$ ping paris-web.fr

Envoi d'une requête 'ping' sur paris-web.fr [92.243.17.115]

Deux types de références

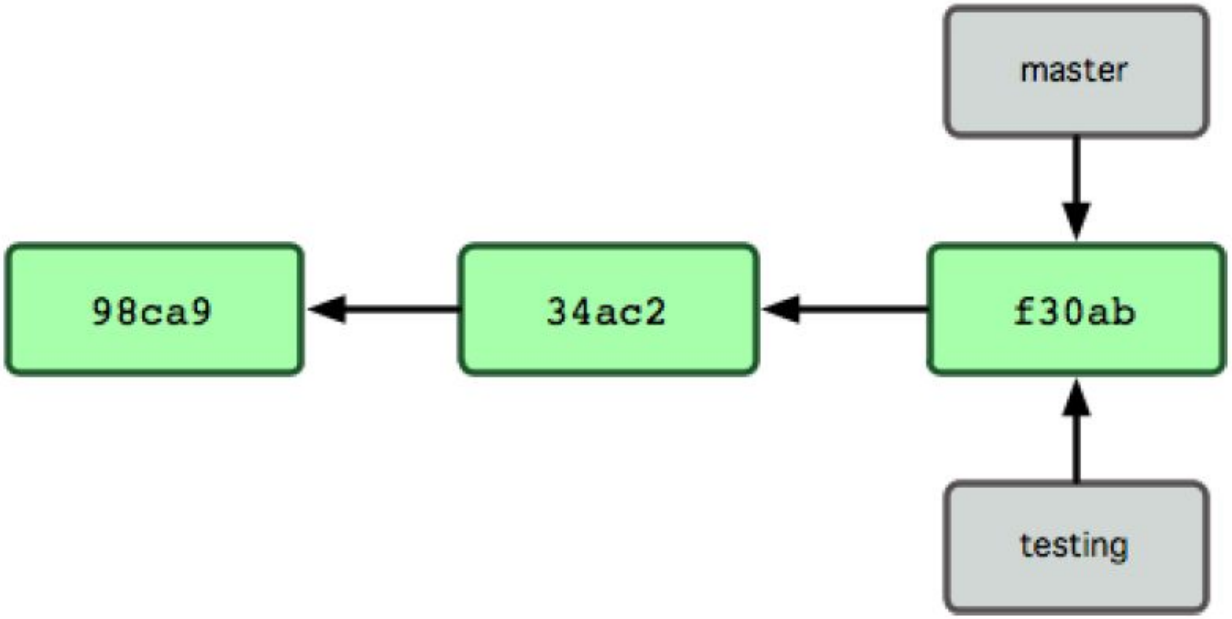
- Les branches (déplacés par Git)
- Les tags (posés par l’utilisateur)

Les branches par l'exemple



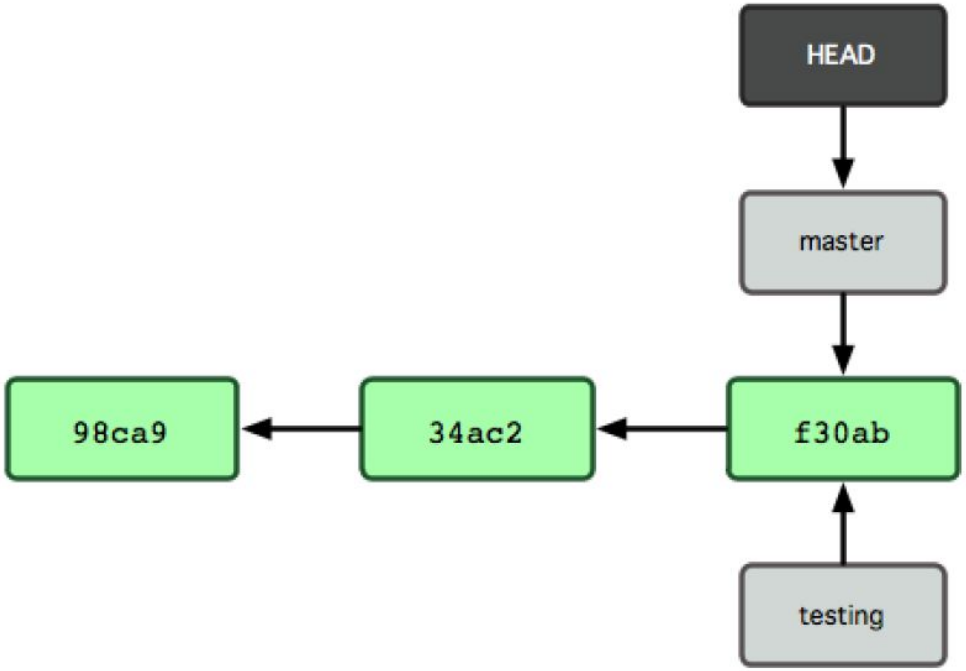
Les branches par l'exemple

Git branch testing



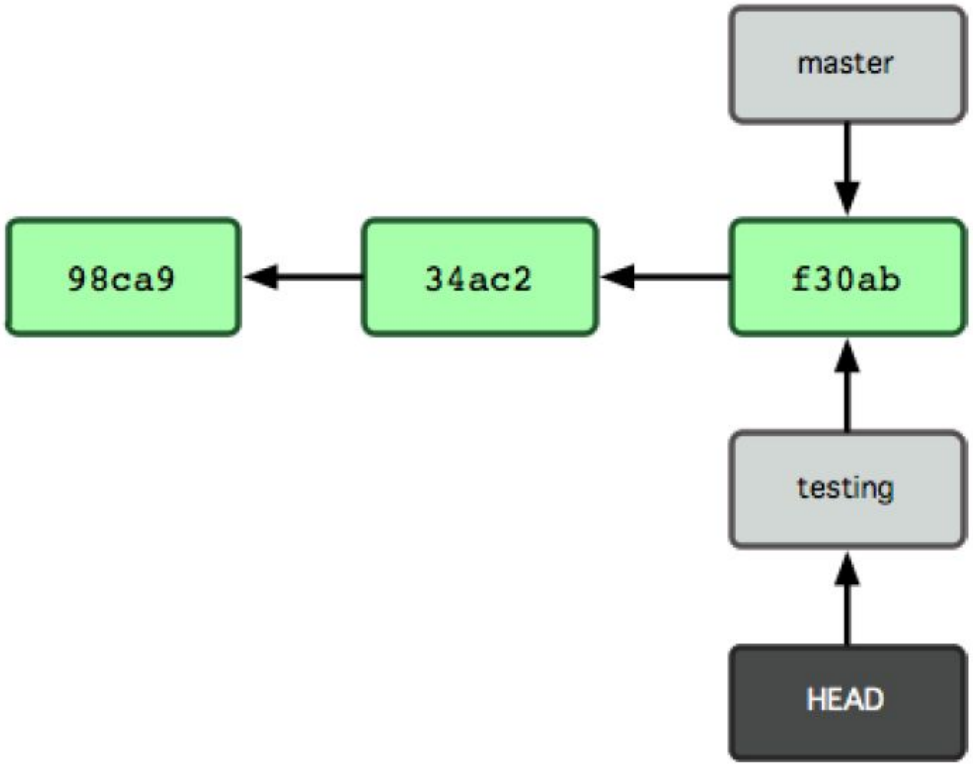
Les branches par l'exemple

Git checkout master



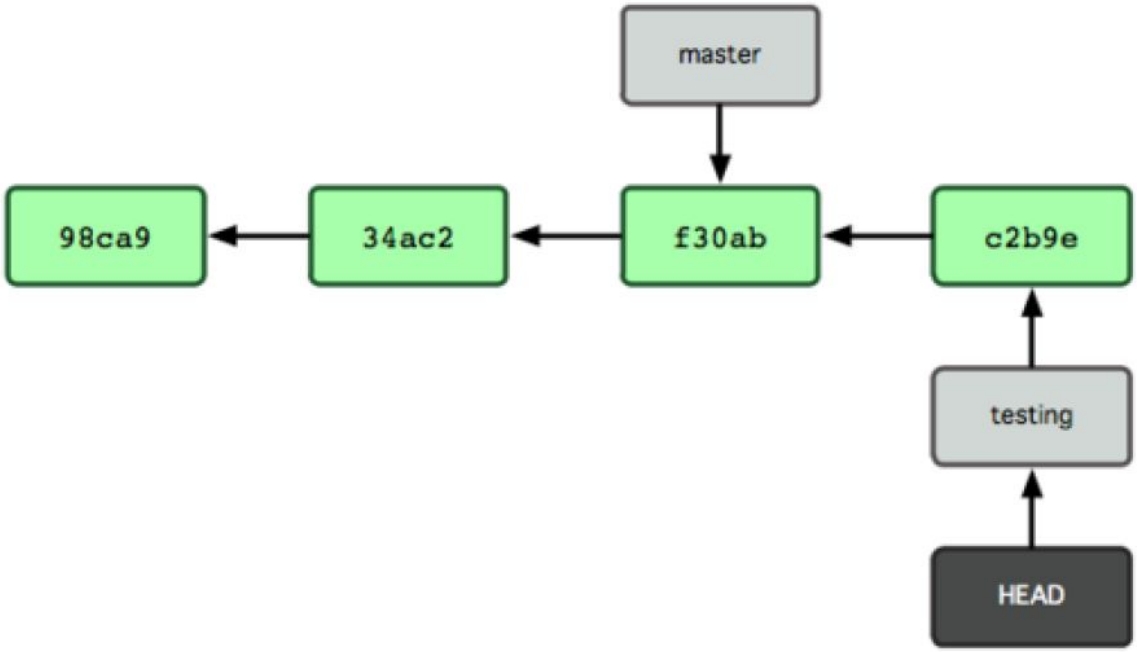
Les branches par l'exemple

Git checkout testing



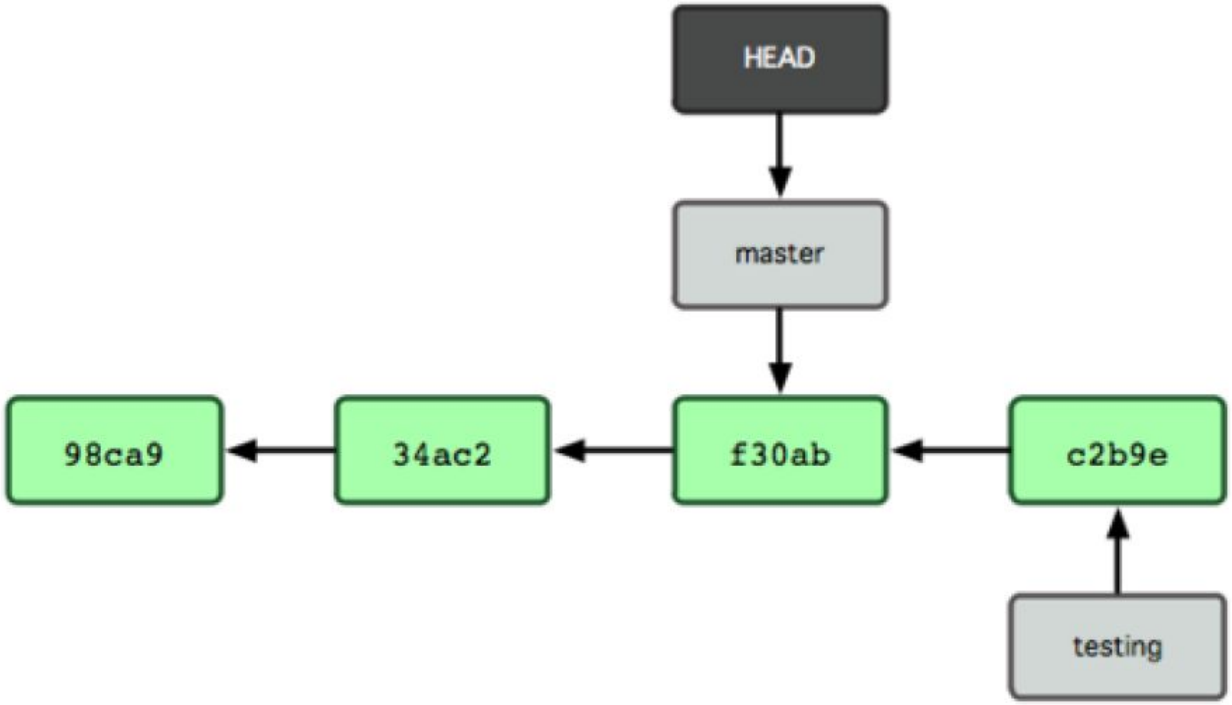
Les branches par l'exemple

Git commit



Les branches par l'exemple

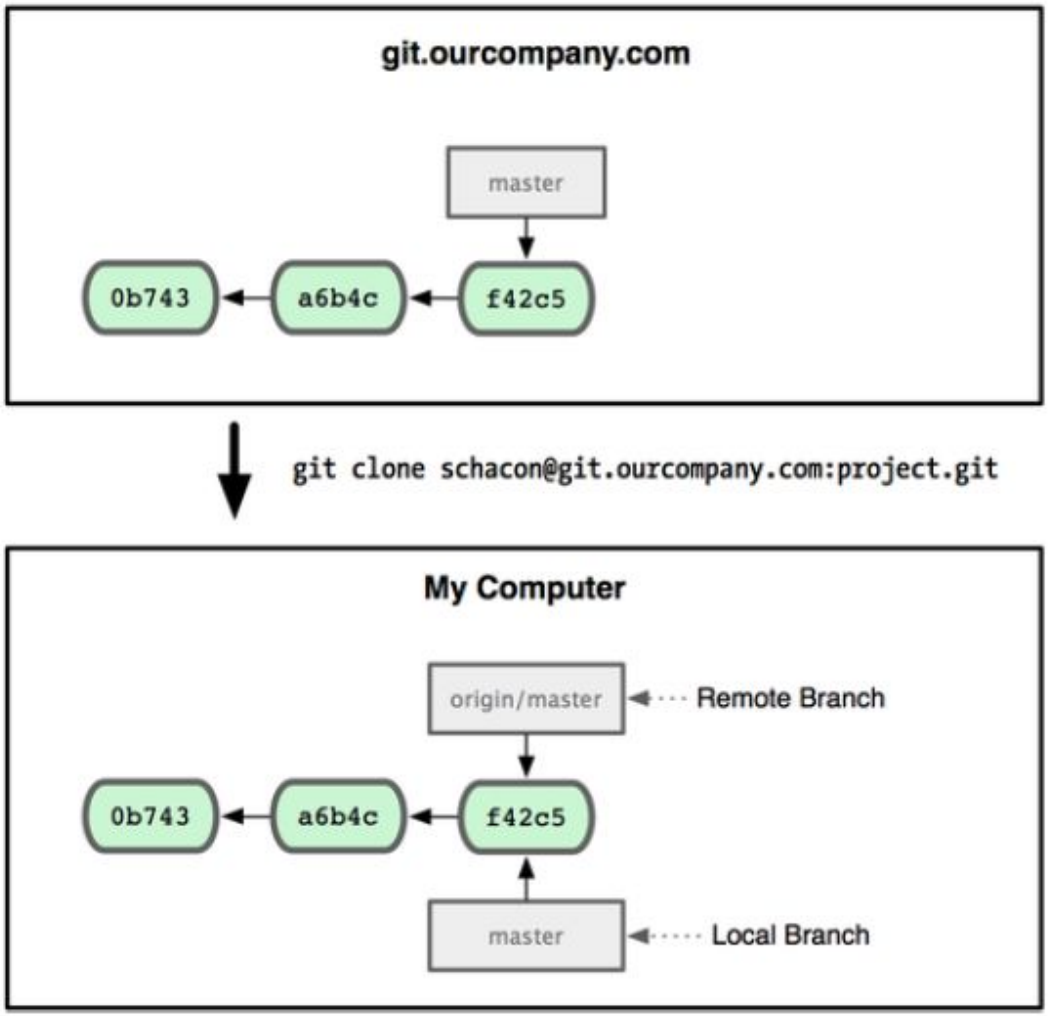
Git checkout master



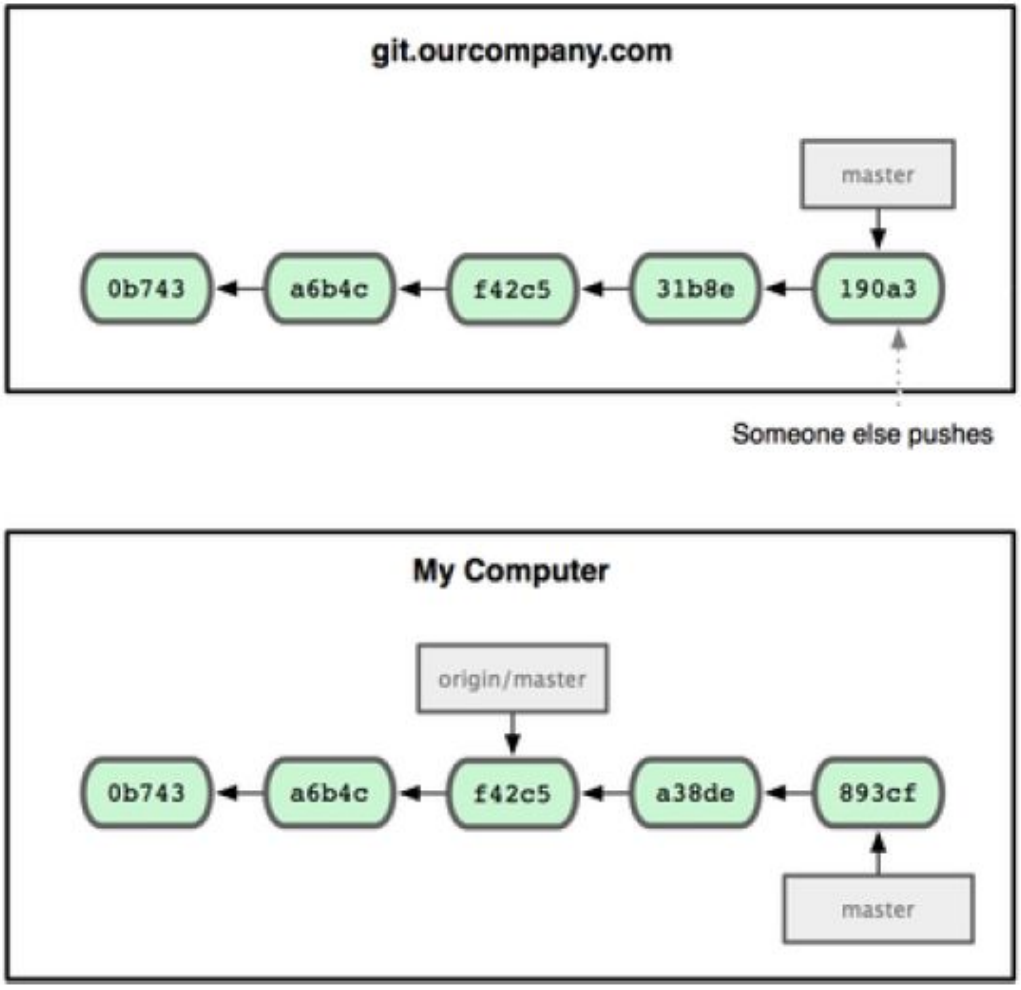
Désigner un commit

- Sha1
- Branche
- Tag
- commit^^
- commit~4

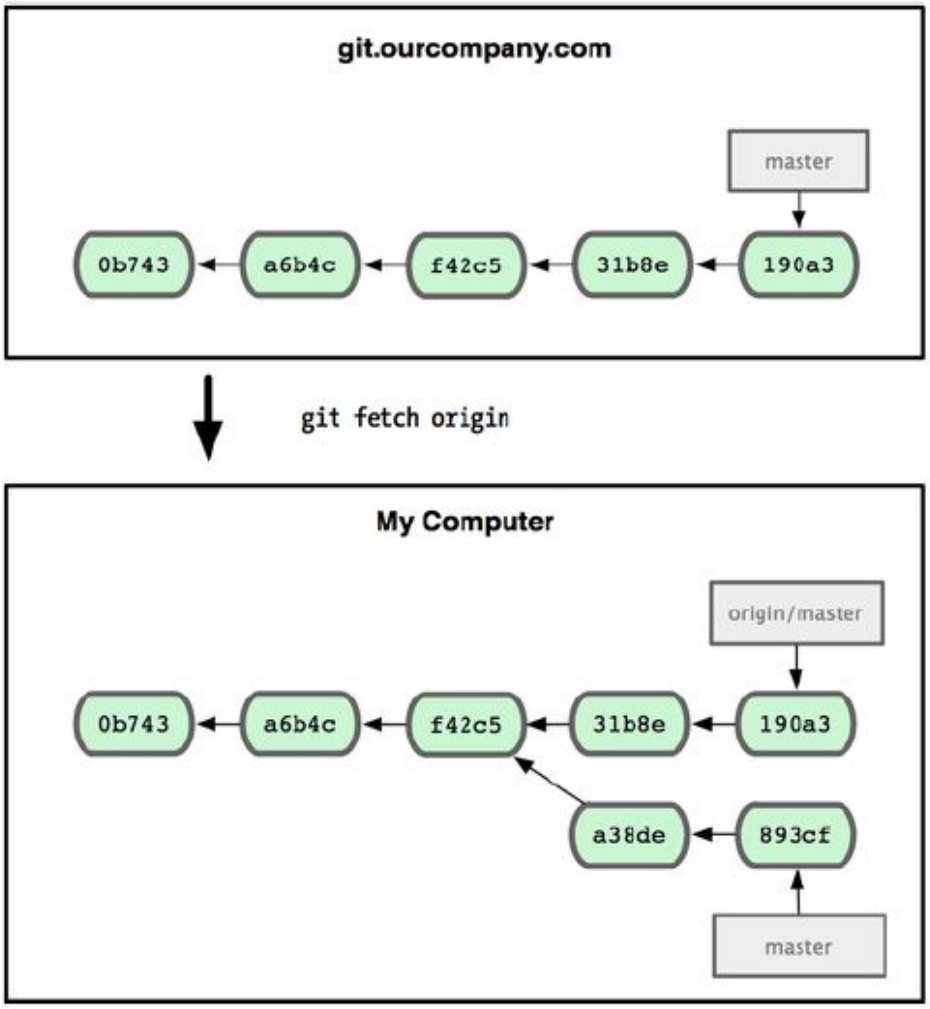
Remote


git.ourcompany.com
master
0b743
a6b4c
f42c5
git clone schacon@git.ourcompany.com:project.git
My Computer
origin/master
Remote Branch
0b743
a6b4c
f42c5
master
Local Branch

Remote


git.ourcompany.com
master
0b743
a6b4c
f42c5
31b8e
190a3
Someone else pushes
My Computer
origin/master
0b743
a6b4c
f42c5
a38de
893cf
master

Remote


git.ourcompany.com
master
0b743
a6b4c
f42c5
31b8e
190a3
git fetch origin
My Computer
origin/master
0b743
a6b4c
f42c5
31b8e
190a3
a38de
893cf
master

Workflows

Chez TEA

- 60 dépôts, ~30 actifs
- Un sujet = Une branche = Une PR
- Fusion vers master… ou vers une grosse branche de feature
- Déploiement ~ à chaque merge sur master

POULE

REQUEST

Criteo - Les premières années

- 2010:
  - Un seul repo, une seule branche master
  - Pas de notion de permissions
  - Far west total
  - Ça marche car il n’y a que quelques dizaines de devs

Criteo - Les premières années

- 2010:
  - Un seul repo, une seule branche master
  - Pas de notion de permissions
  - Far west total
  - Ça marche car il n’y a que quelques dizaines de devs
- 2011:
  - Un seul repo, 7 branches “master par équipe”
  - Gentleman agreement pour les permissions
  - Un release manager merge une fois par semaine

Criteo - Plusieurs repos

- 2012
  - 33 repos : un par composant.
  - Seule l’équipe owner peut pusher
  - Chaque équipe publie sa librairie,

Criteo - Plusieurs repos

- 2012
  - 33 repos : un par composant.
  - Seule l’équipe owner peut pusher
  - Chaque équipe publie sa librairie,
- 2013
  - 160+ repos
  - On souffre...

Criteo - Workflow actuel

- Toujours un repo par composant…
- … mais l’usine logicielle build toutes les masters, plusieurs fois par heure…
- …et on n’a qu’une branche par repo
- Merge avant qu’une feature soit finie, pour détecter les conflits au plus tôt
- Git pull --rebase
- Git rerere ?