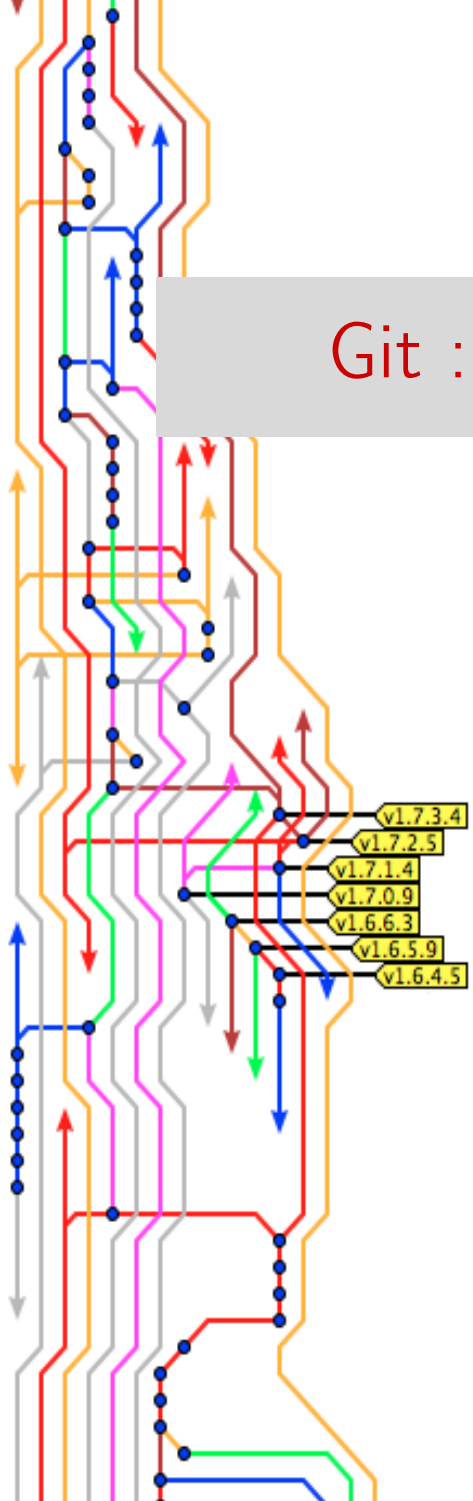


Git : un gestionnaire de versions intelligent

Benoit Daloze et Sébastien Wilmet

16 avril 2012

Les gestionnaires de versions

Commandes de base, créer un commit

Gérer plusieurs branches

Autres fonctionnalités

Les gestionnaires de versions

Commandes de base, créer un commit

Gérer plusieurs branches

Autres fonctionnalités

Moyens primitifs

Gérer un projet de programmation sans gestionnaire de versions :

- S'envoyer l'entièreté du code par mail ;
- Partage de fichiers sur un serveur (dropbox, . . .) ;
- S'envoyer des *patchs* :
  - commande `diff` : différence entre deux fichiers/dossiers ;
  - commande `patch` : appliquer le *patch* (la *diff*).
- etc.

Moyens primitifs

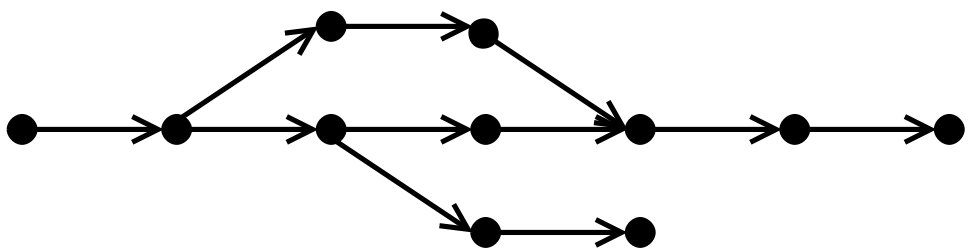
Gérer un projet de programmation sans gestionnaire de versions :

- S'envoyer l'entièreté du code par mail ;
- Partage de fichiers sur un serveur (dropbox, . . .) ;
- S'envoyer des *patchs* :
  - commande `diff` : différence entre deux fichiers/dossiers ;
  - commande `patch` : appliquer le *patch* (la *diff*).
- etc.

Pas très pratique !

Buts d'un gestionnaire de versions

- Faciliter la gestion d'un projet de programmation ;
- Garder l'historique de toutes les modifications (*commits*) ;
- Travailler en équipe ;
- Avoir des branches de développement :
  - pour développer une fonctionnalité séparément ;
  - pour une certaine version (2.4.0 $\rightarrow$ 2.4.1 $\rightarrow$ ...).



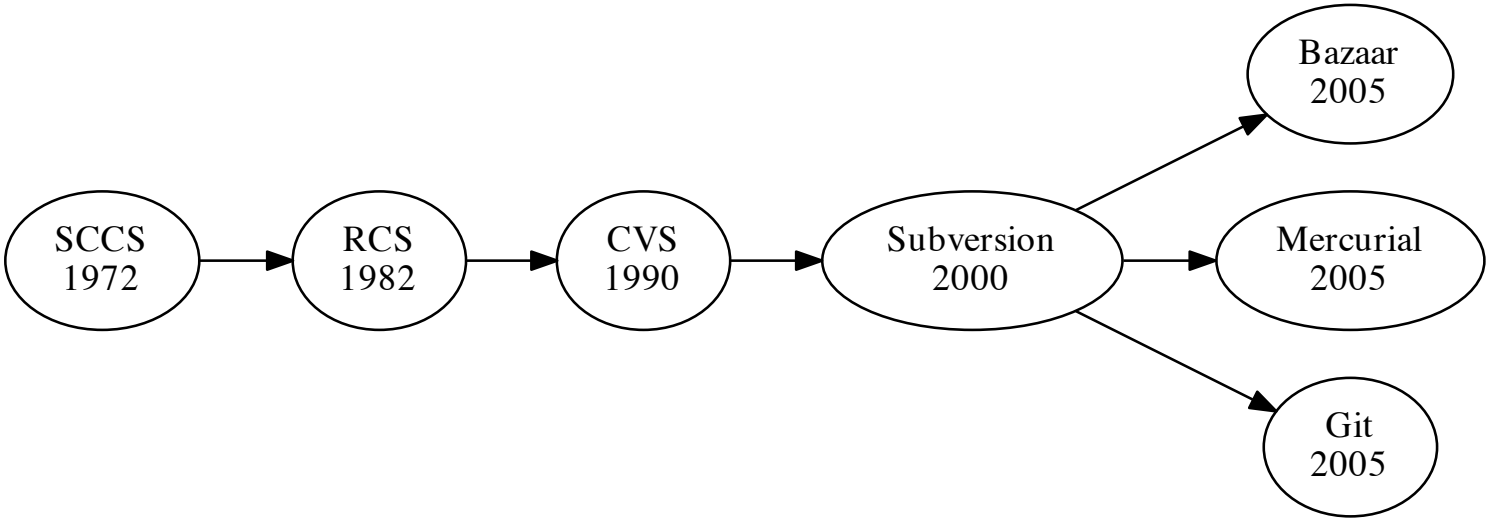
Micro-commits

Un *commit* = **une** modification bien particulière

Avantages :

- Plus facile à comprendre pour les autres
- Possibilité d'annuler un changement facilement (`git revert`)
- Trouver l'origine d'un bug (`git bisect`)
- ...

Historique des gestionnaires de versions



Subversion

Subversion (SVN) est **centralisé** :

- Un serveur central contient toutes les données ;
- Beaucoup de requêtes entre le client et le serveur (assez lent) ;
- Besoin d'une connexion internet pour travailler.

Git

Git est **décentralisé/distribué** :

- Toutes les données sont sur notre machine ;
- Les opérations sont très rapides ;
- Connexion internet seulement pour les *pull* et *push*.

Git

Git est **décentralisé/distribué** :

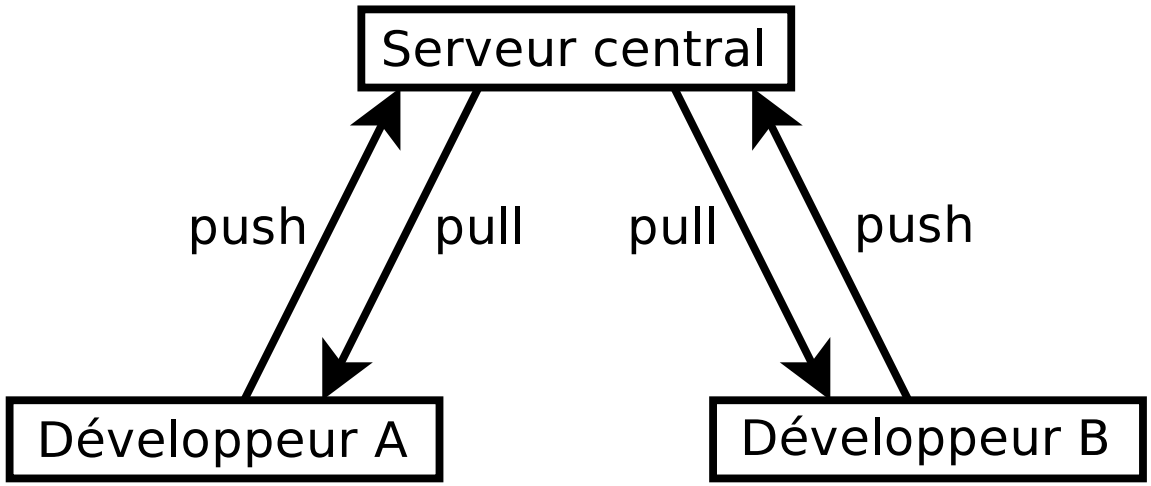
- Toutes les données sont sur notre machine ;
- Les opérations sont très rapides ;
- Connexion internet seulement pour les *pull* et *push*.

Git est aussi **plus puissant** et **plus flexible** :

- Pour la gestion des branches ;
- Possède de nombreuses fonctionnalités plus avancées.

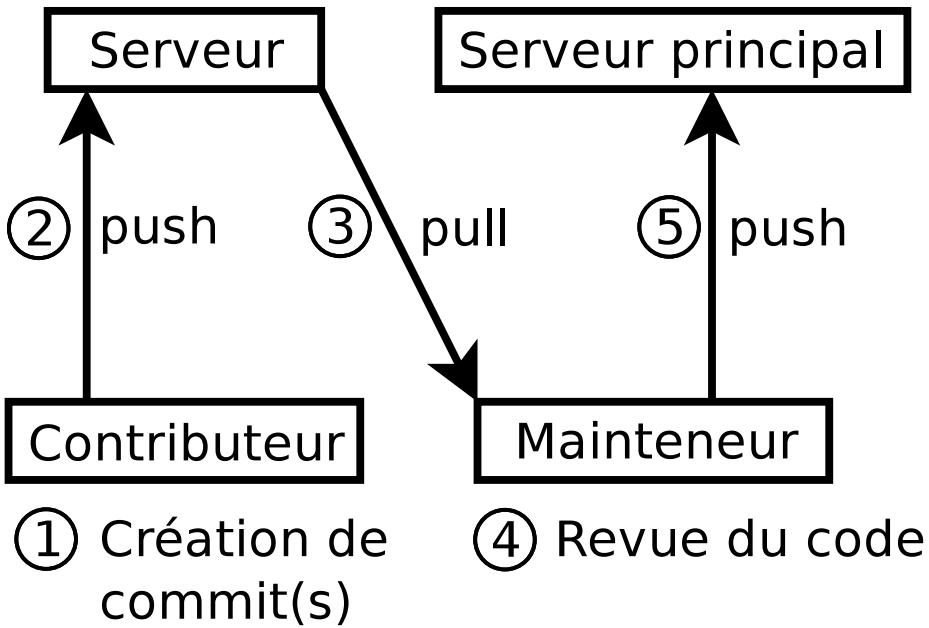
Serveur central avec Git

- Une manière simple de travailler en équipe ;
- Accès en écriture pour tous les développeurs.



Git est décentralisé

- Seul les mainteneurs ont accès en écriture ;
- Les contributeurs font des *pull requests*.



Les gestionnaires de versions

Commandes de base, créer un commit

Gérer plusieurs branches

Autres fonctionnalités

Créer le dépôt Git

Pour un nouveau projet :

```
$ mkdir project
$ cd project/
$ git init
```

Créer le dépôt Git

Pour un nouveau projet :

```
$ mkdir project
$ cd project/
$ git init
```

Pour un projet existant :

```
$ git clone git://example.net/project
$ cd project/
```

Créer le dépôt Git

Pour un nouveau projet :

```
$ mkdir project
$ cd project/
$ git init
```

Pour un projet existant :

```
$ git clone git://example.net/project
$ cd project/
```

Répertoire caché `.git/` (unique) :

```
$ ls .git/
config  description  HEAD  hooks/  info/  objects/  refs/
```

États d'un fichier

Untracked

- non pris en compte par Git

États d'un fichier

Untracked

- non pris en compte par Git

Unmodified/Committed

- aucune modification

États d'un fichier

Untracked

- non pris en compte par Git

Unmodified/Committed

- aucune modification

Modified

- fichier modifié
- pas pris en compte pour le prochain commit

États d'un fichier

Untracked

- non pris en compte par Git

Unmodified/Committed

- aucune modification

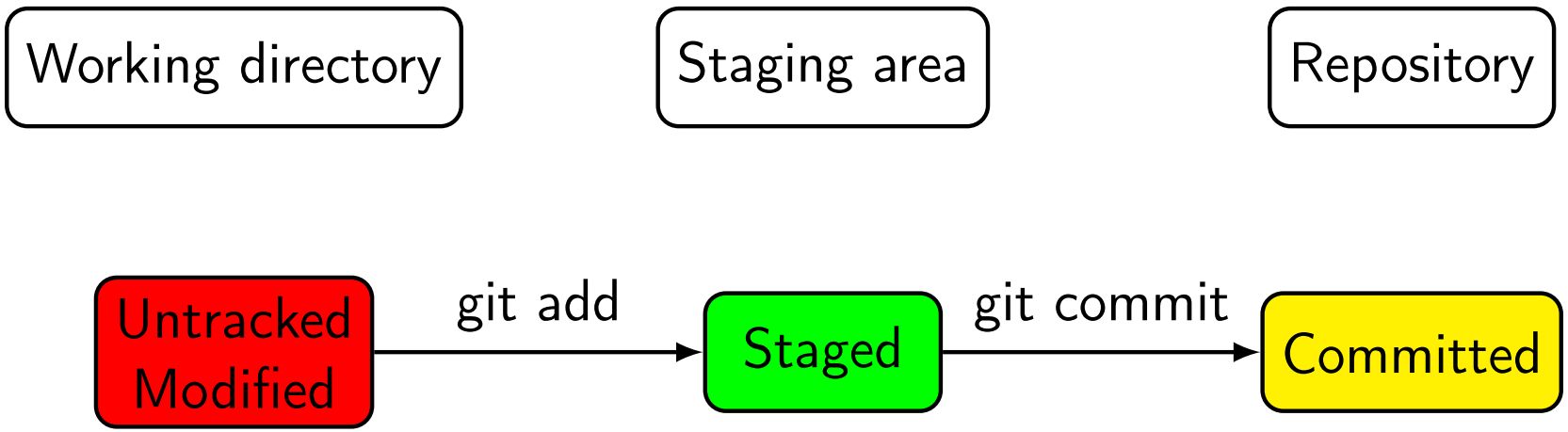
Modified

- fichier modifié
- pas pris en compte pour le prochain commit

Staged

- fichier ajouté, modifié, supprimé ou déplacé
- pris en compte pour le prochain commit

États d'un fichier



Créer un nouveau fichier

```
$ echo hello > README
```

État du fichier : ***untracked***

Créer un nouveau fichier

```
$ echo hello > README
```

État du fichier : ***untracked***

```
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       README
nothing added to commit but untracked files present
```

Créer un nouveau fichier

```
$ git add README
```

État du fichier : ***untracked*** $\longrightarrow$ ***staged***

Créer un nouveau fichier

```
$ git add README
```

État du fichier : ***untracked*** $\longrightarrow$ ***staged***

```
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README
#
```

Créer le commit

```
$ git commit
[écrire le message du commit]
```

Créer le commit

```
$ git commit
[écrire le message du commit]

$ git log
commit c3aab8bb6cca644162a4fa82df283682717da3d4
Author: Your Name <foo@bar.be>
Date:   Wed Feb 1 15:19:03 2012 +0100

    Titre du commit (pas trop long)

    Plus longue description.
    Ligne vide après le titre.
```

Modifier un fichier

État du fichier README : ***unmodified***

Modifier un fichier

État du fichier README : ***unmodified***

```
$ echo world >> README
```

État : ***modified***

Modifier un fichier

État du fichier README : ***unmodified***

```
$ echo world >> README
```

État : ***modified***

```
$ git add README
```

État : ***staged***

Modifier un fichier

État du fichier README : ***unmodified***

```
$ echo world >> README
```

État : ***modified***

```
$ git add README
```

État : ***staged***

```
$ git commit
```

État : ***committed***

Diff

Voir les modifications avant de créer un commit.

```
$ echo new-text > README

$ git diff
diff --git a/README b/README
index 2e85c45..4320c6f 100644
--- a/README
+++ b/README
@@ -1,2 +1 @@
-hello
-world
+new-text
```

Aide

La liste des commandes :

```
$ git help
```

Page de manuel d'une commande :

```
$ git help <cmd>
$ git help <cmd> --web
$ git <cmd> --help
```

Résumé des commandes

- `git init`
- `git clone`
- `git status`
- `git add <file>`
- `git rm/mv <file>`
- `git commit`
- `git log`
- `git diff`
- `git help`

Les gestionnaires de versions

Commandes de base, créer un commit

Gérer plusieurs branches

Autres fonctionnalités

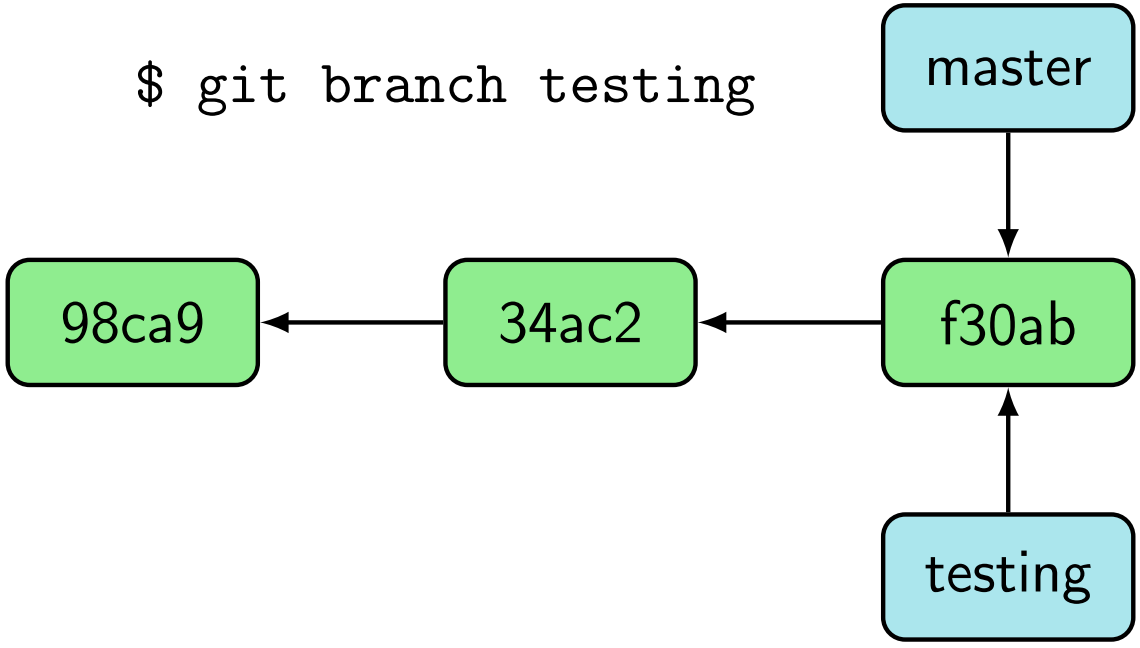
Créer une branche

Une branche :

- Pointe vers un commit ;
- Le pointeur « avance » quand on crée un commit.

Un commit pointe vers son ou ses commit(s) parent(s).

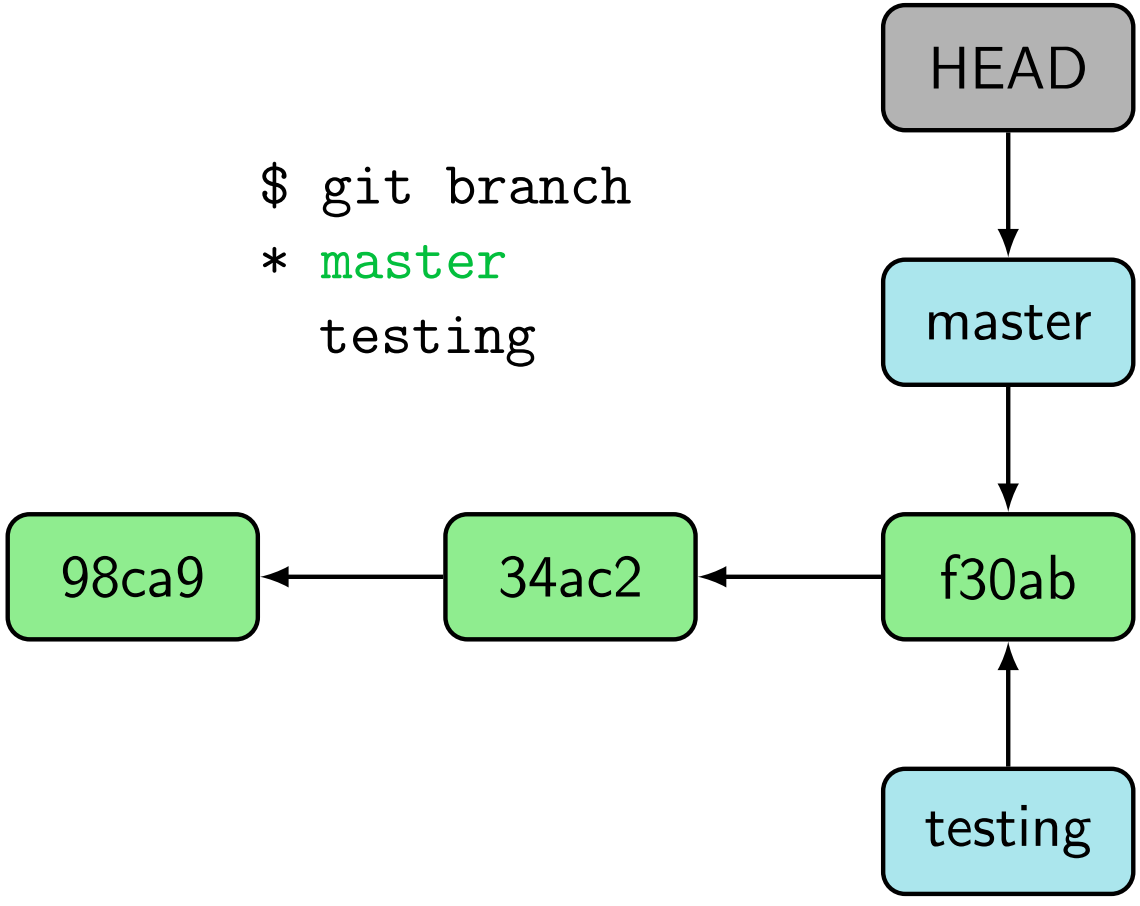
Créons une branche :



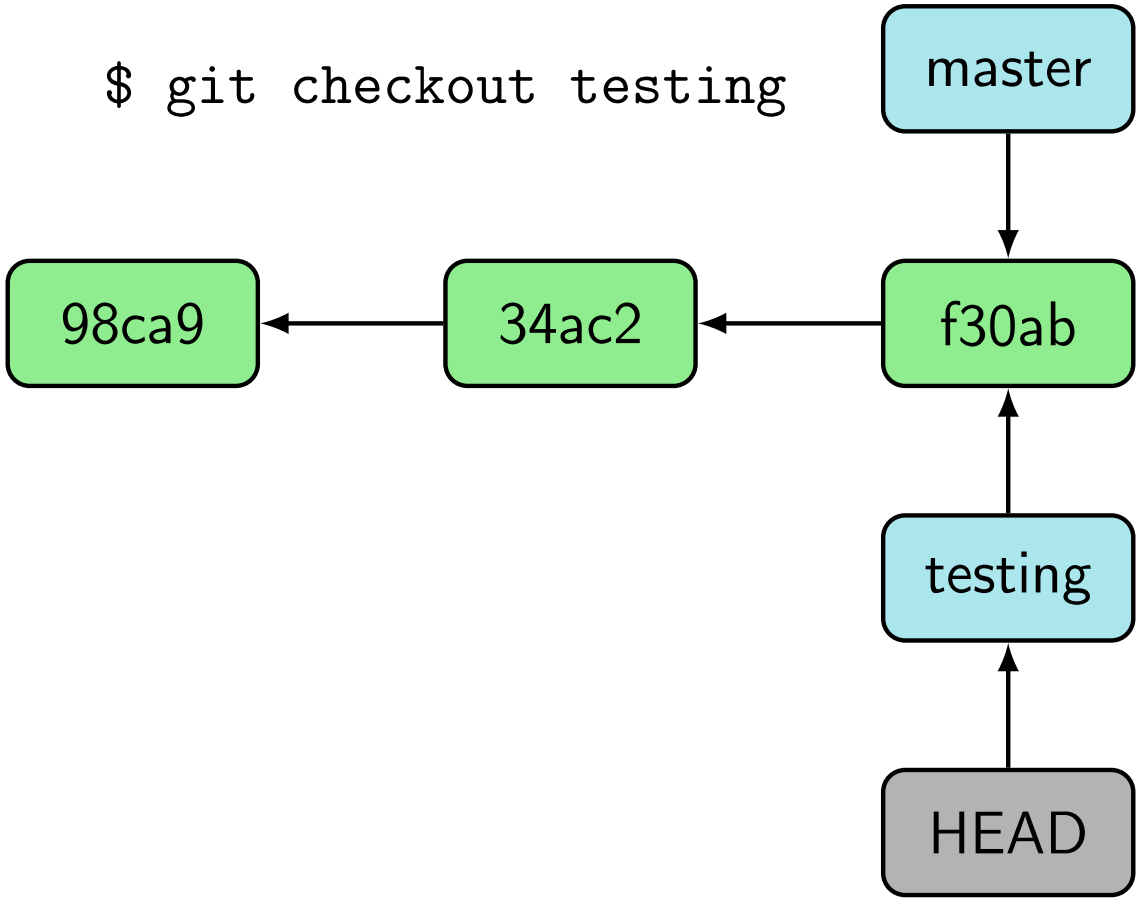
HEAD

Le pointeur `HEAD` pointe vers la branche courante.

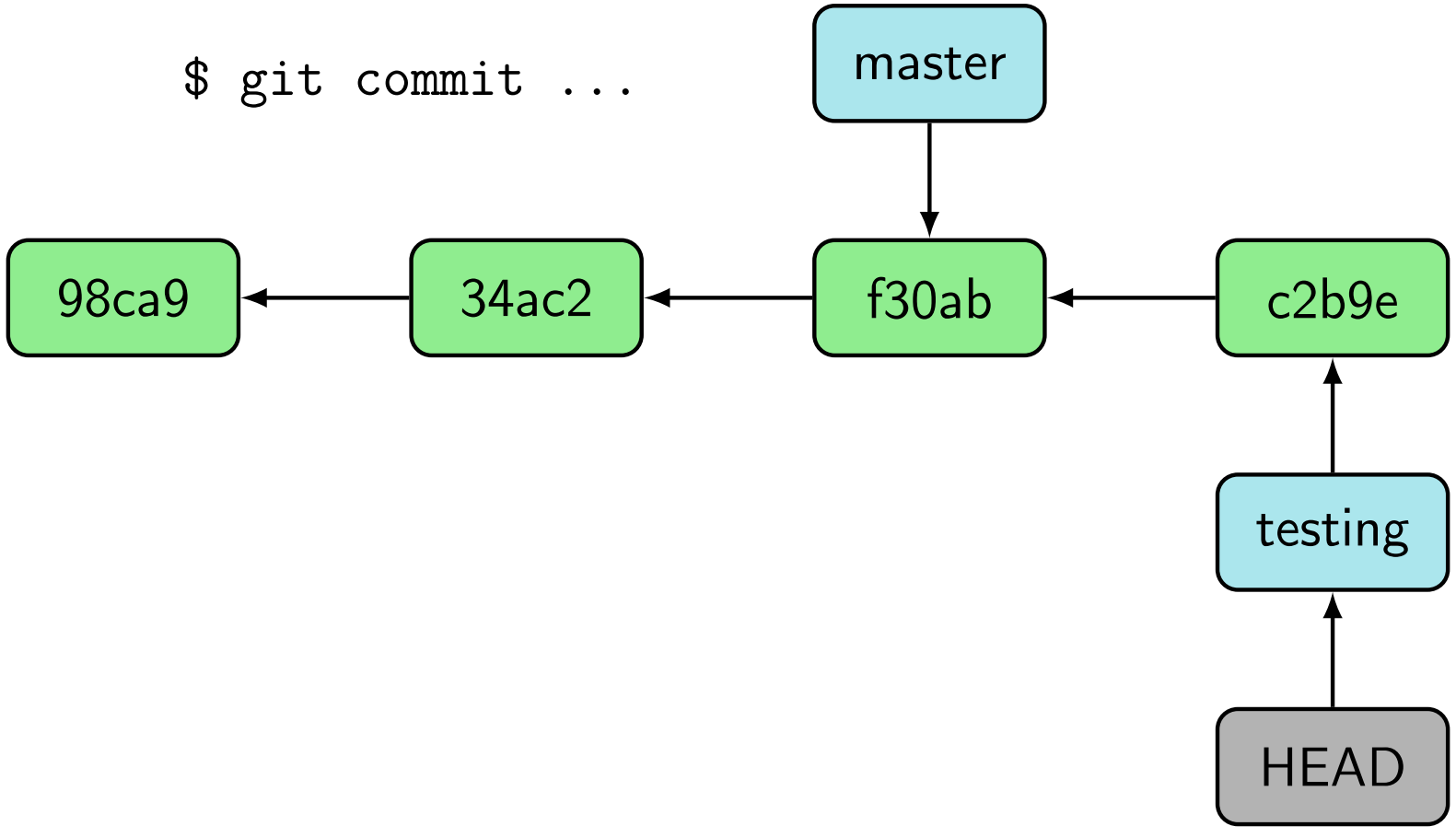
```
$ git branch
* master
  testing
```



Changer de branche

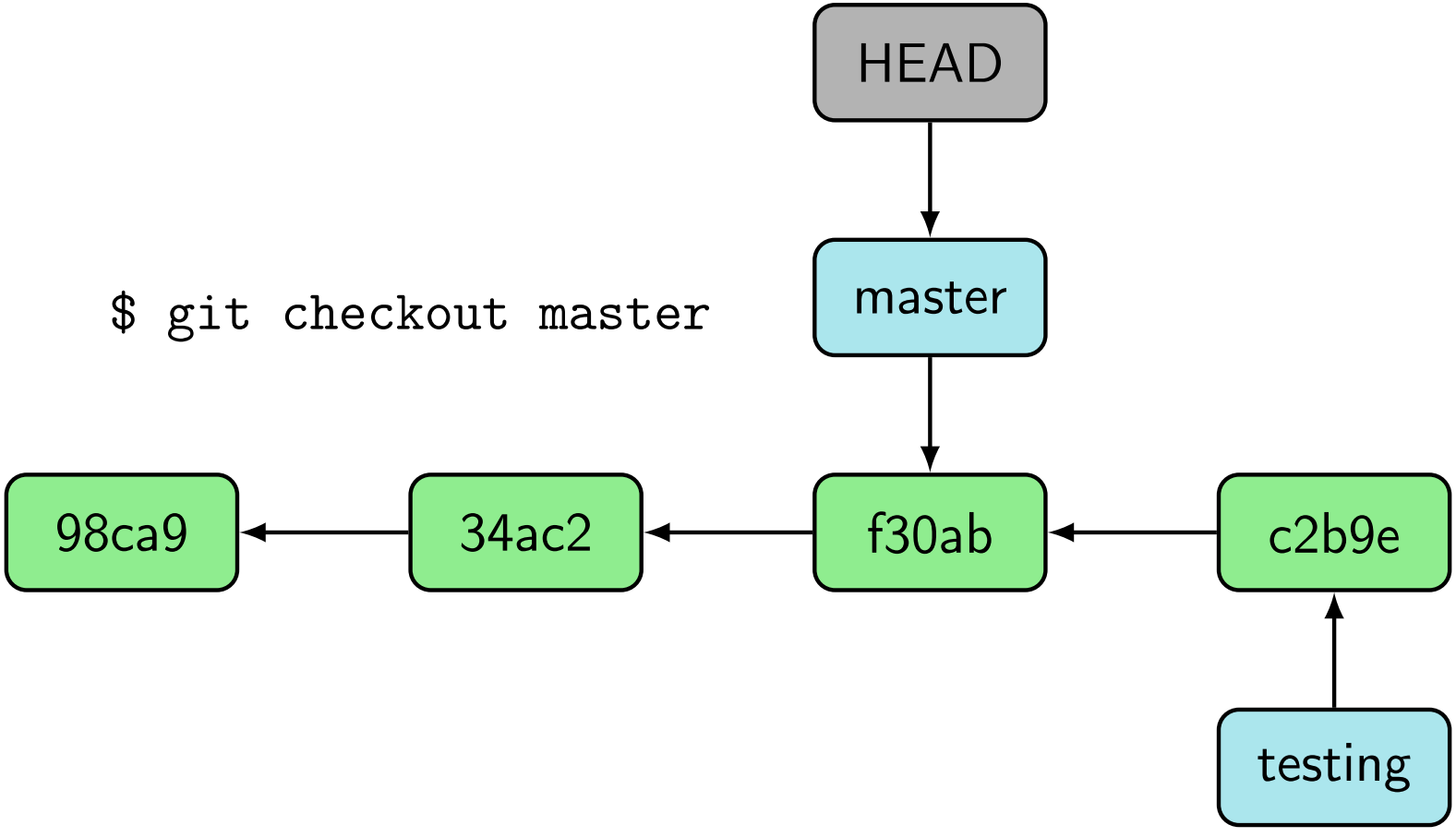


Créer un commit sur testing



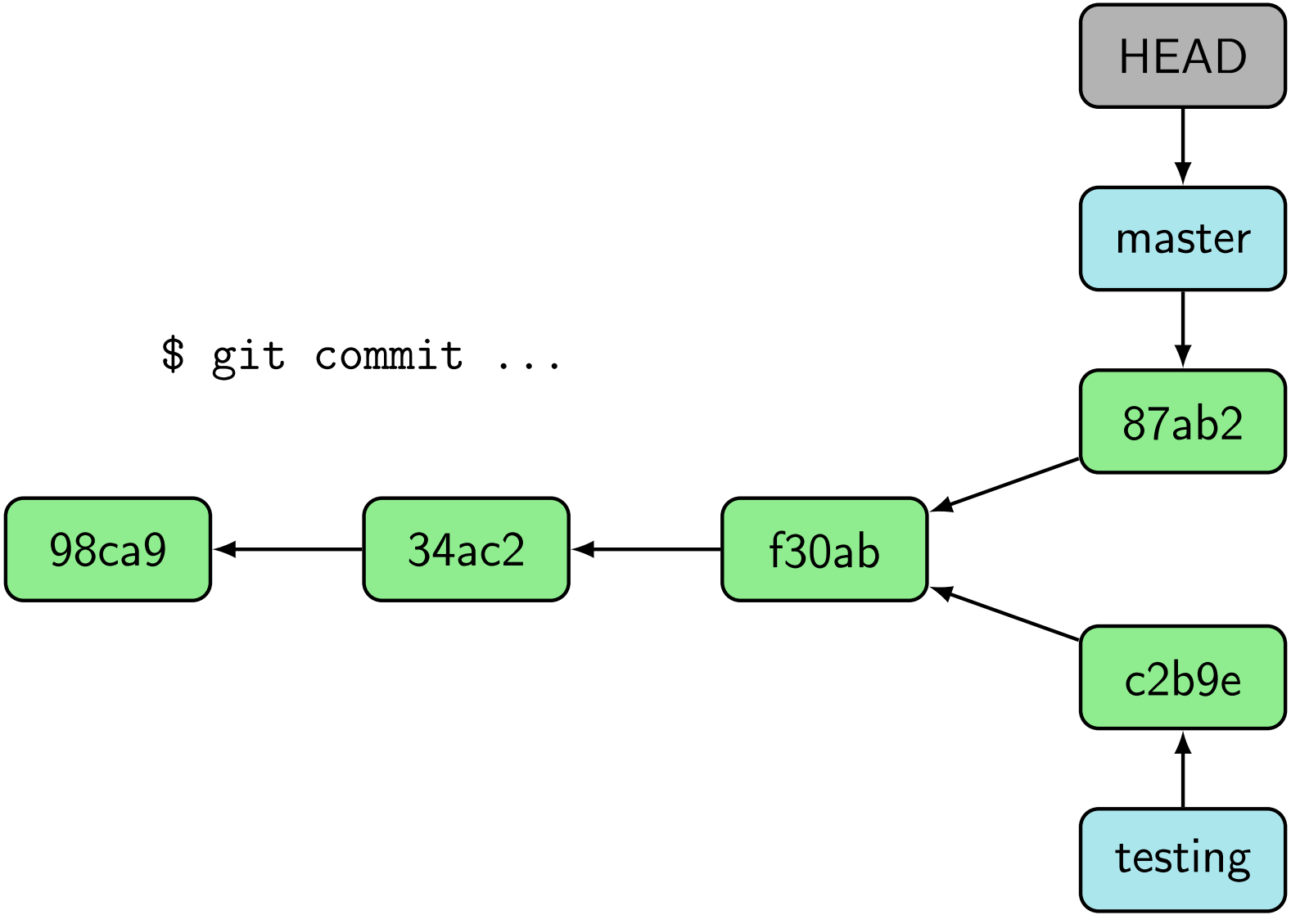
Revenir sur master

```
$ git checkout master
```

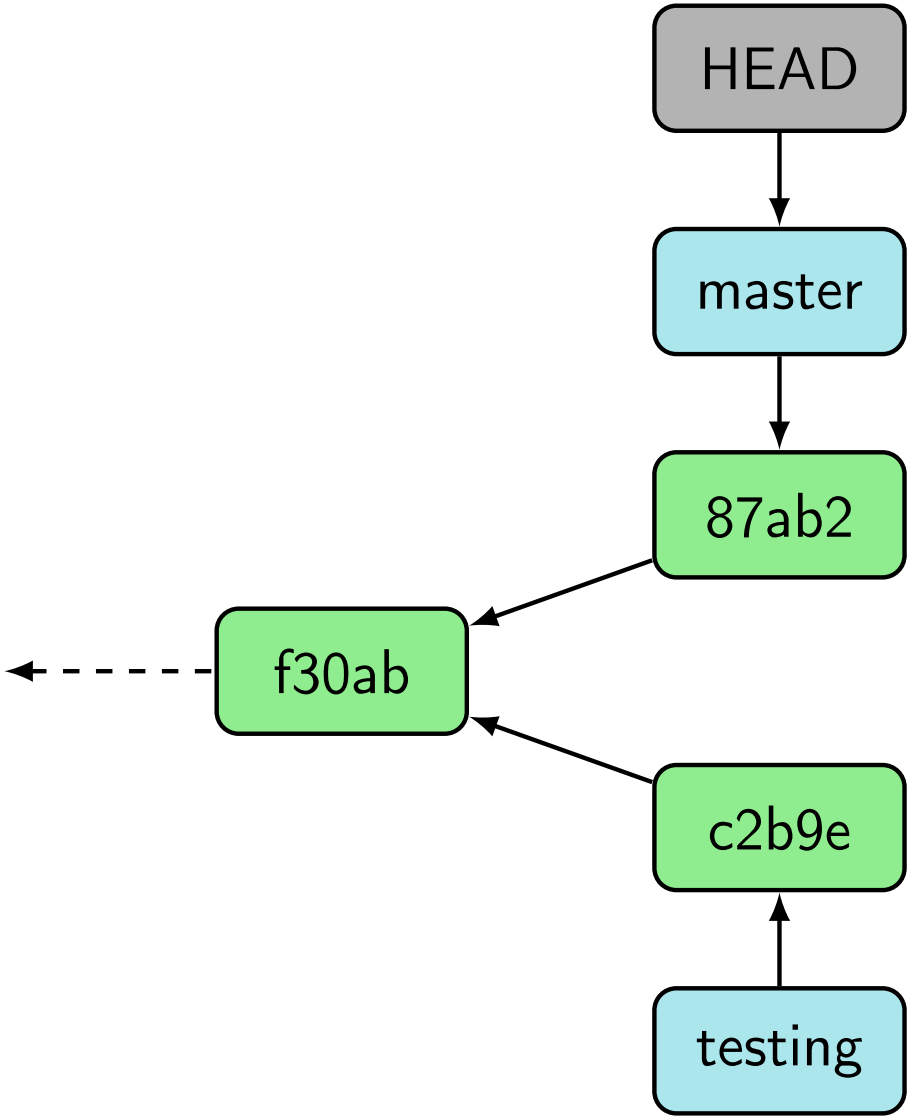


Créer un commit sur master

```
$ git commit ...
```

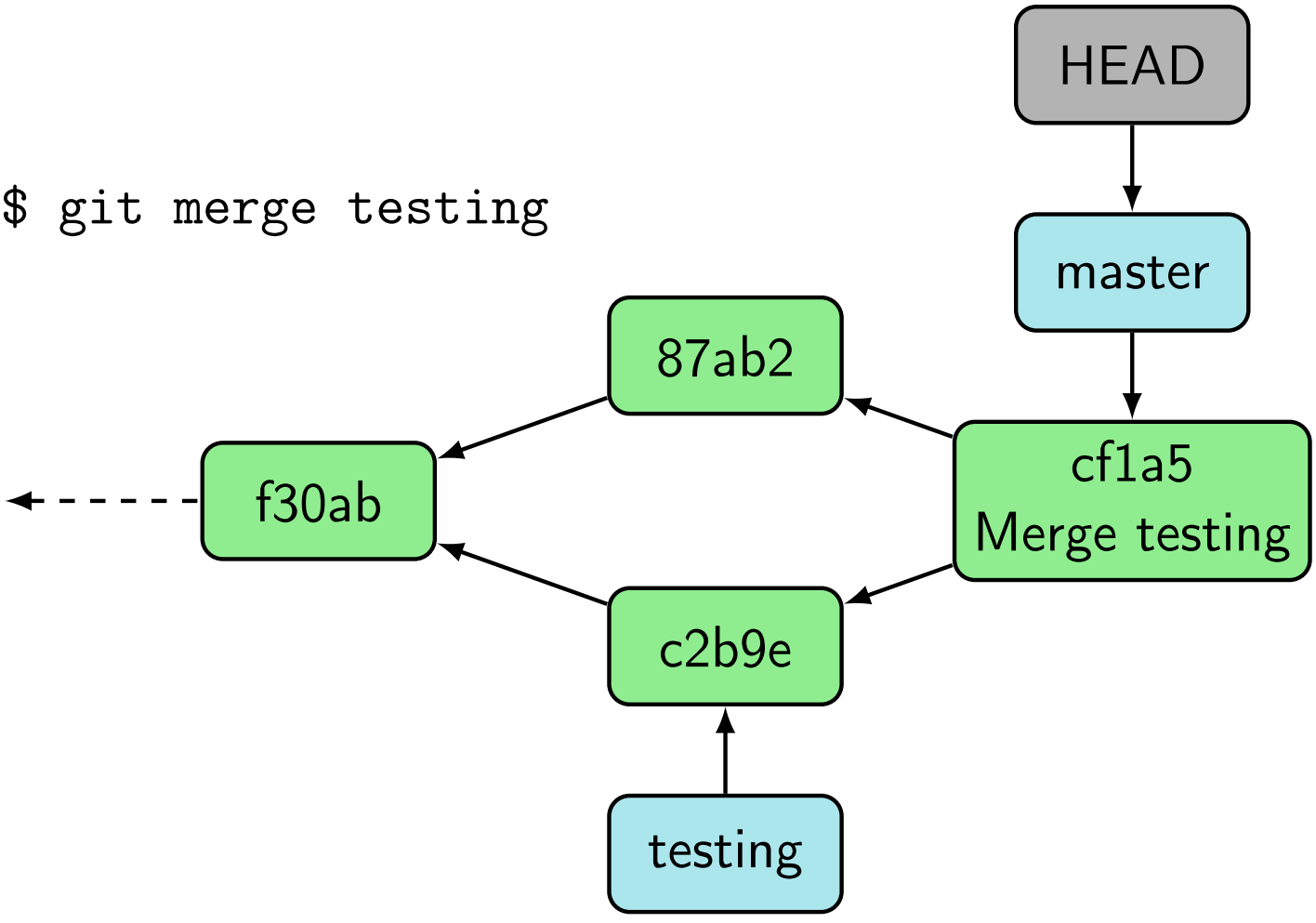


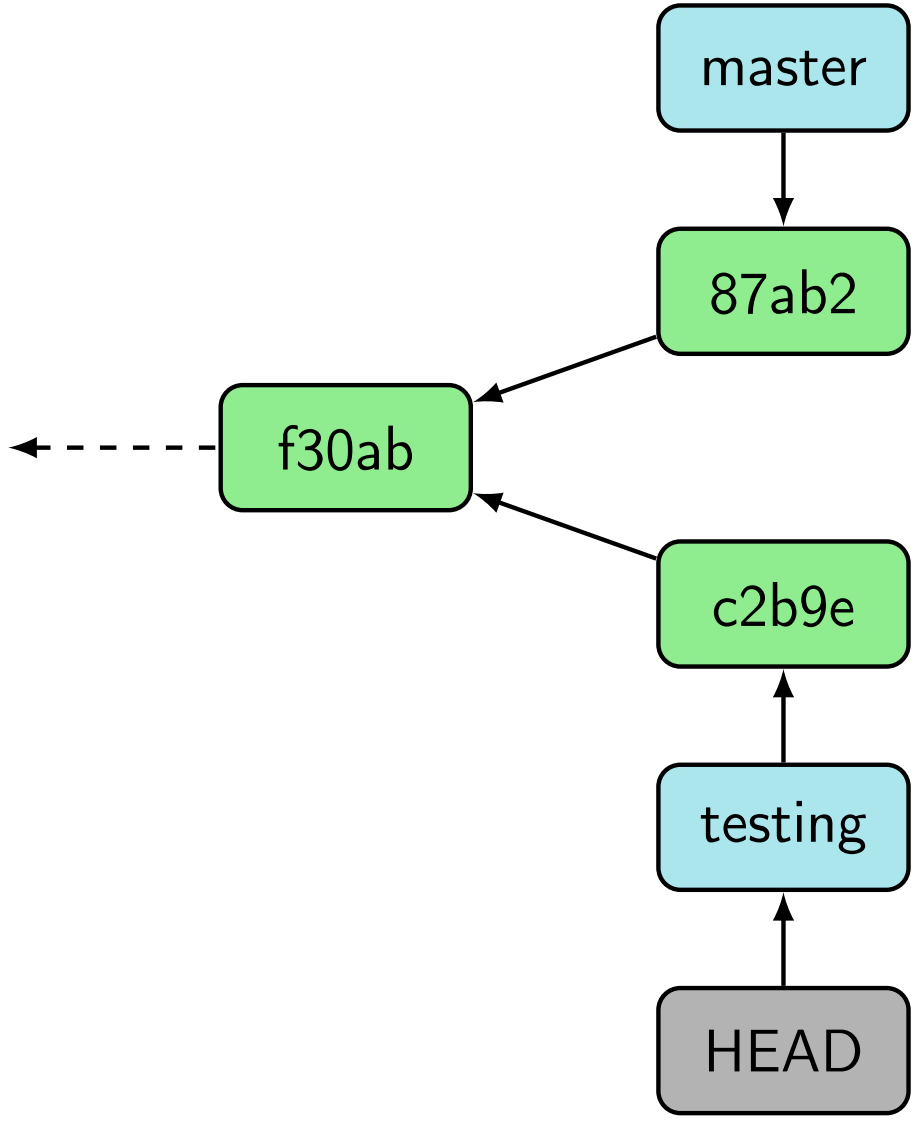
Divergence



Merge

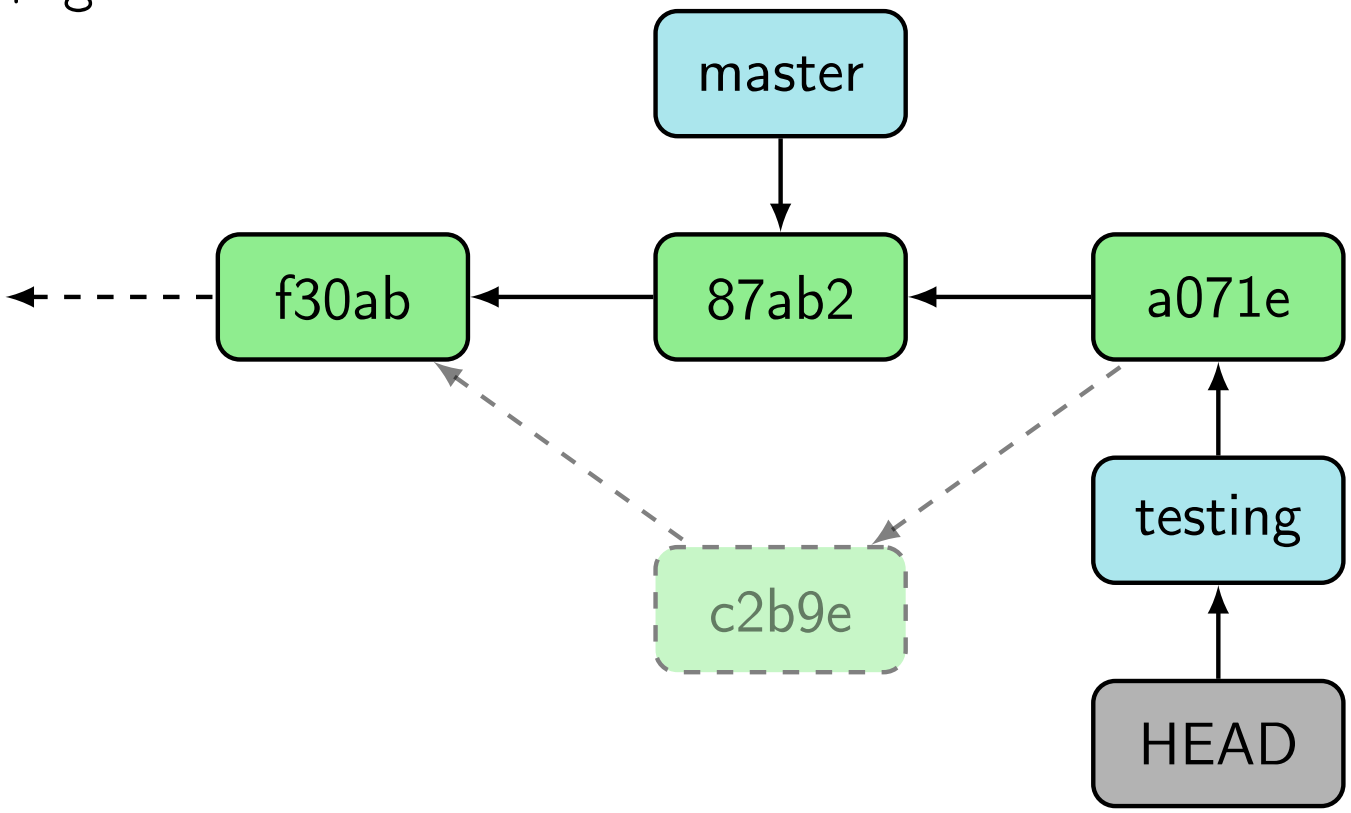
```
$ git merge testing
```




master
87ab2
f30ab
c2b9e
testing
HEAD

Rebase

```
$ git rebase master
```



Remotes

Remote : référence vers un dépôt distant.

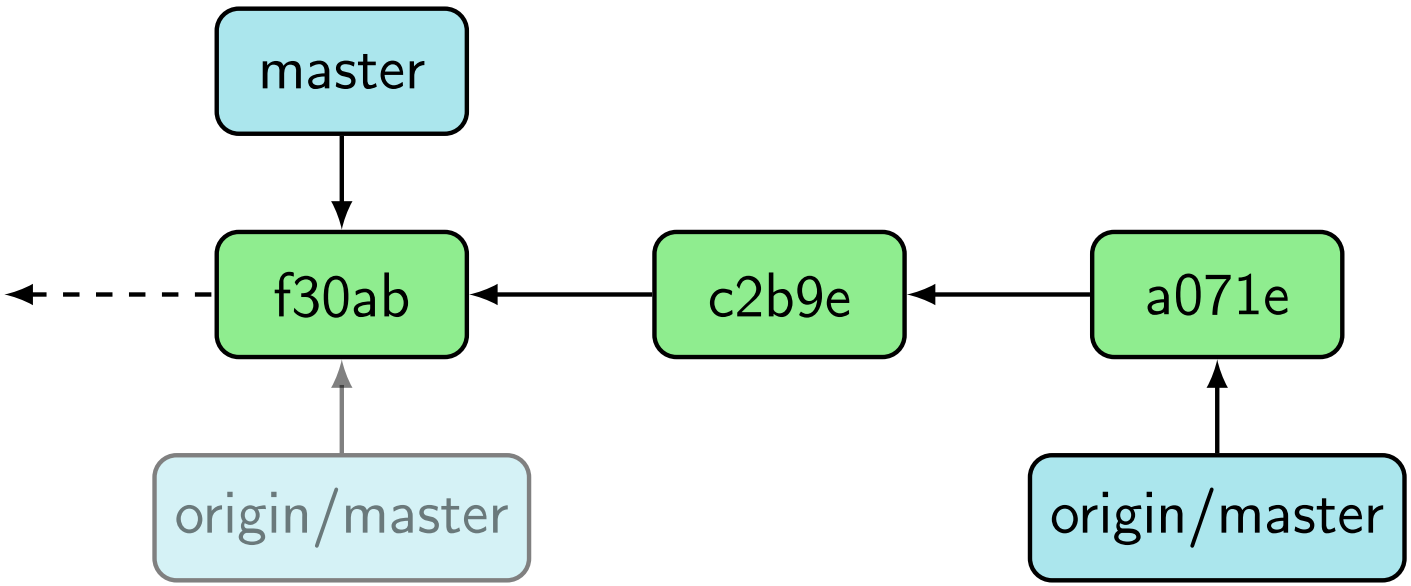
```
$ git remote -v
origin git://git.kernel.org/pub/scm/git/git.git (fetch)
origin git://git.kernel.org/pub/scm/git/git.git (push)

$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/man
  remotes/origin/master
  remotes/origin/next
```

git pull [remote] [branch]

`git pull origin master`

- git fetch origin
- git merge origin/master

```
$ git fetch origin
```

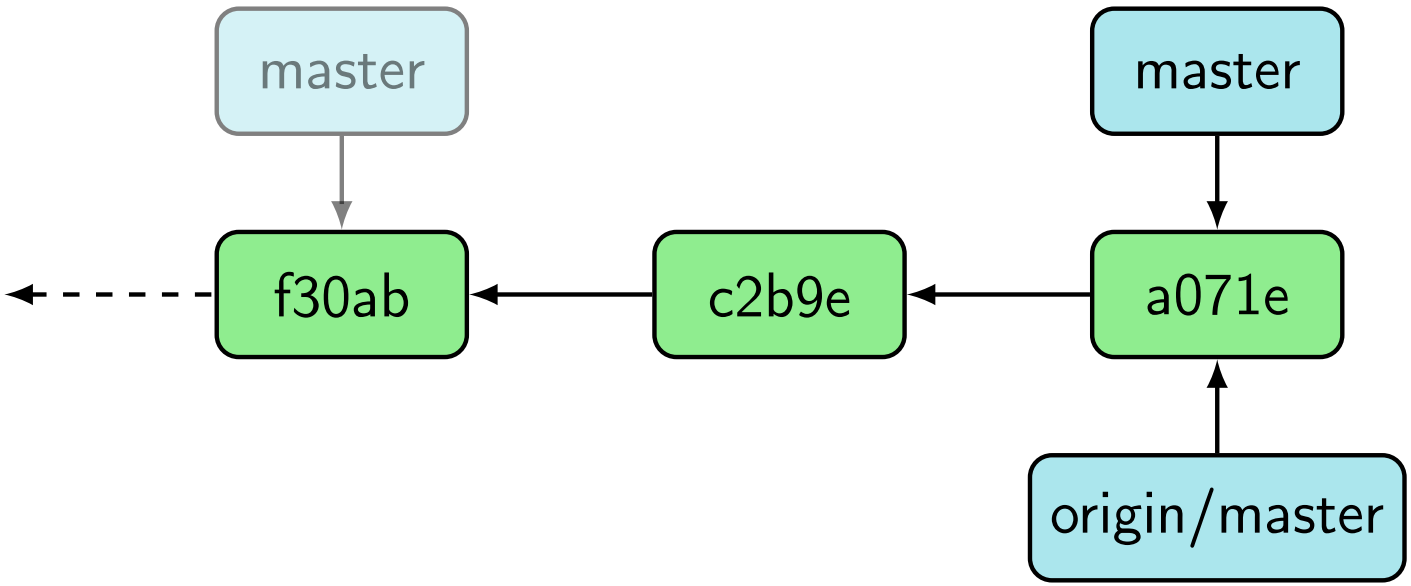


git pull [remote] [branch]

`git pull origin master`

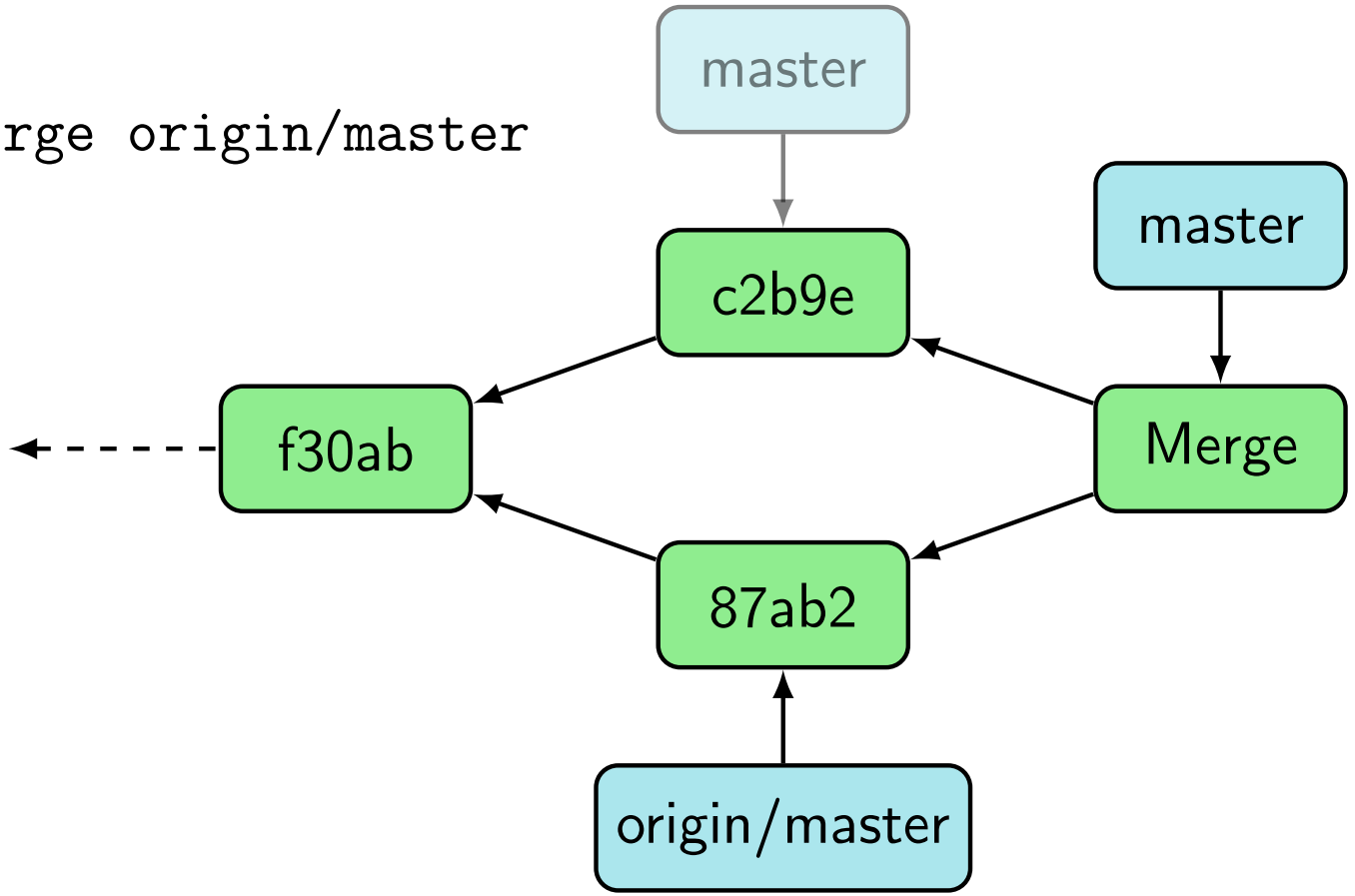
- git fetch origin
- git merge origin/master

```
$ git merge origin/master
```



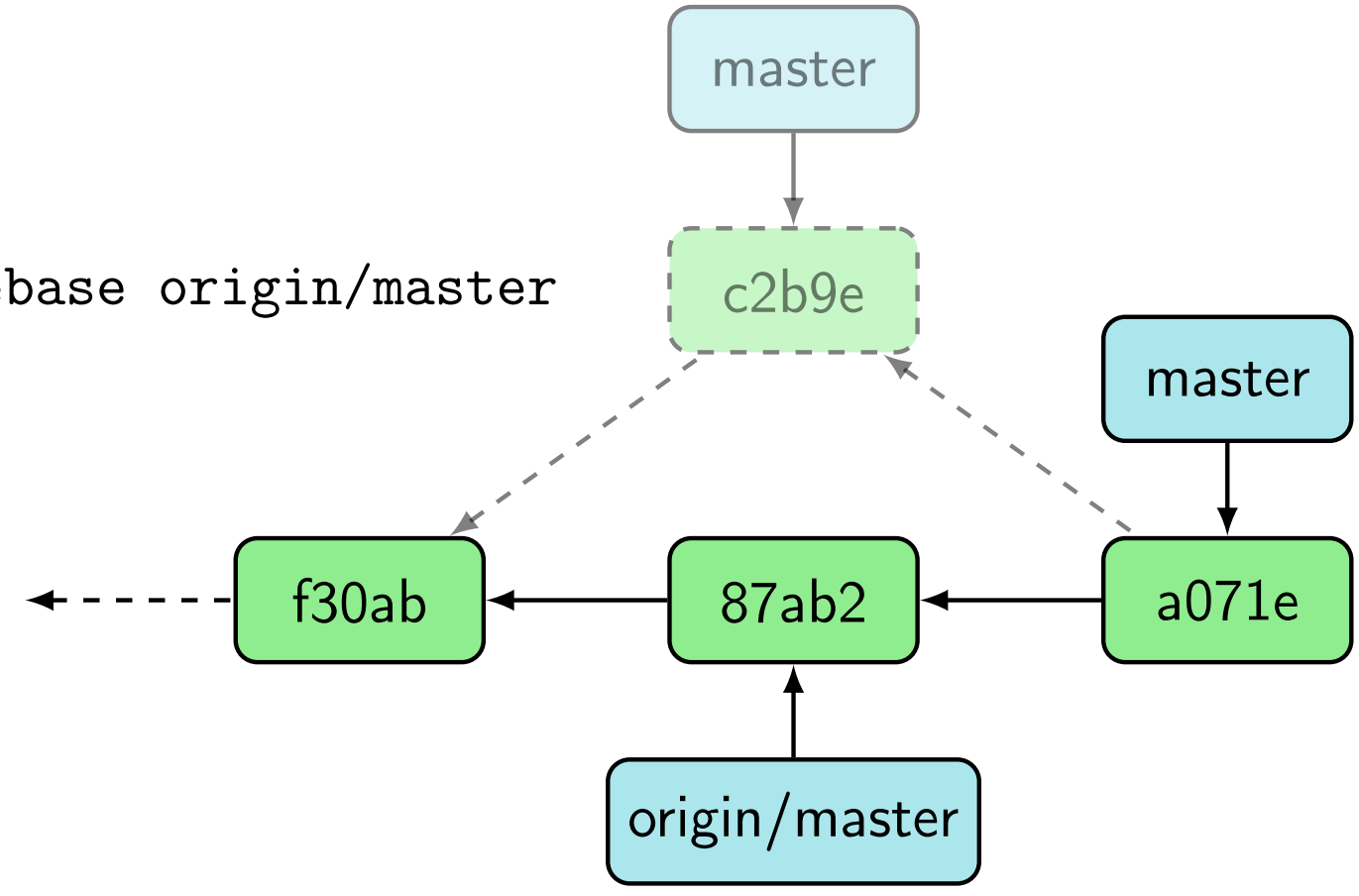
git pull : merge

```
$ git merge origin/master
```



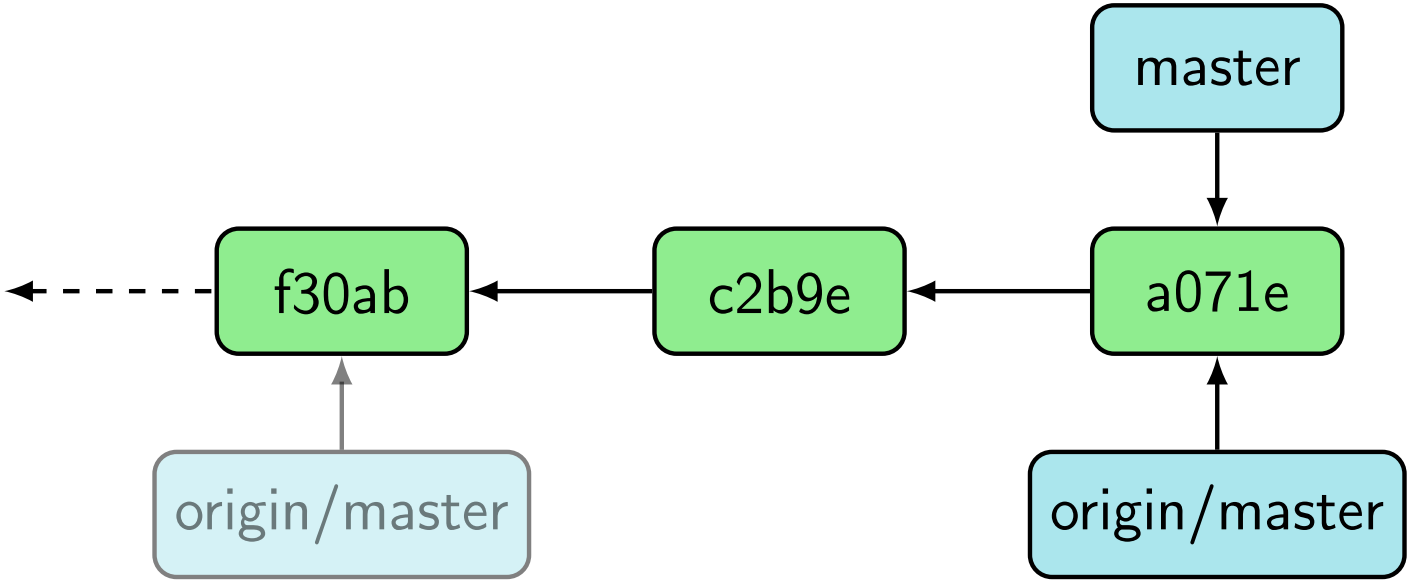
git pull --rebase

```
$ git rebase origin/master
```



git push [remote] [branch]

```
$ git push origin master
```



Commandes principales

browse

- `git status`
- `git log`
- `git diff`

commit

- `git add`
- `git commit`

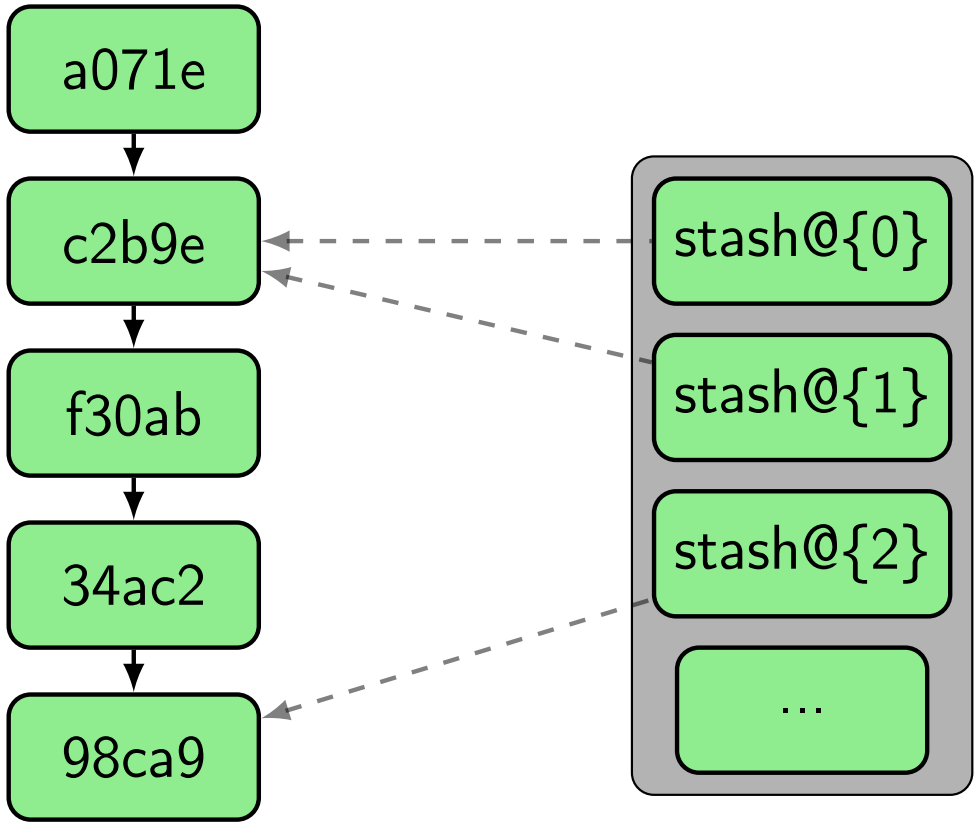
branch

- `git branch / checkout`
- `git merge / rebase`

remote

- `git pull`
- `git push`

git stash



git stash

git stash

- save [message]

git stash

git stash

- save [message]
- list
- show [stash]

git stash

git stash

- save [message]
- list
- show [stash]
- apply [stash]
- pop [stash]

git stash save

```
$ git status
# On branch master
# Changes not staged for commit:
#
#       modified:   main.c
#
no changes added to commit (use "git add"/"git commit -a")

$ git stash save
Saved working directory and index state WIP on master: 8540fea
HEAD is now at 8540fea message

$ git status
# On branch master
nothing to commit (working directory clean)
```

git stash list, show

```
$ git stash list
stash@{0}: WIP on master: 8540fea message

$ git stash show -p
diff -git a/main.c b/main.c
index f2ad6c7..2f773ae 100644
--- a/main.c
+++ b/main.c
@@ -1,3 +1,5 @@
+#include <stdio.h>
+
 int main() {
    printf("Hello, world!");
    return 0;
 }
```

Commandes avancées

git grep

- grep(1) sur les fichiers pris en compte par git

git tag

- crée un *tag*, une référence fixe vers un commit

git revert

- crée un *commit* qui annule un autre

git blame

montre qui a modifié le fichier, ligne par ligne

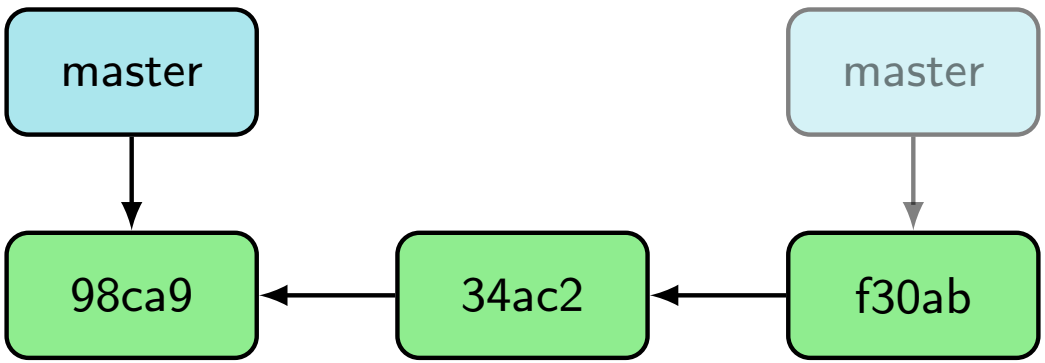
```
$ git blame git.c
85023 (Junio C Hamano      2006-12-19  1) #include "builtin.h"
2b11e (Johannes Schindelin 2006-06-05  2) #include "cache.h"
fd5c3 (Thiago Farina       2010-08-31  3) #include "exec_cmd.h"
fd5c3 (Thiago Farina       2010-08-31  4) #include "help.h"
575ba (Matthias Lederhofer 2006-06-25  5) #include "quote.h"
d8e96 (Jeff King           2009-01-28  6) #include "run-command.h"
8e49d (Andreas Ericsson    2005-11-16  7)
822a7 (Ramsay Allan Jones  2006-07-30  8) const char git_usage_string[]
f2dd8 (Jon Seymour         2011-05-01  9)        "git [--version] [--ex
```

git reset

Change le pointeur de la branche courante

```
$ git reset HEAD^ # recule la branche d'un commit
Unstaged changes after reset:
M       README

$ git status
# On branch master
# Changes not staged for commit:
#
#       modified:   README
```

```
$ git reset --hard 98ca9
```



`git commit --amend`

Pour modifier le dernier *commit* :

- ajouter une modification ;
- modifier le message du *commit*.

```
$ edit some_file
$ git add some_file
$ git commit --amend
```

git rebase --interactive

Permet de modifier l'historique

```
$ git rebase -i HEAD~3

pick 4eeebe5 bulk-checkin: allow the same data to be multiply hashed
pick 127b177 bulk-checkin: support chunked-object encoding
pick 973951a chunked-object: fallback checkout codepaths

# Rebase 617312b..973951a onto 617312b
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git reflog

Log des opérations sur les *commits*

```
$ git reflog
b0d66 HEAD@{0}: commit (amend): add a setting to require a filter to be
97395 HEAD@{1}: checkout: moving from master to man
b0d66 HEAD@{2}: rebase: aborting
9cda8 HEAD@{3}: rebase -i (pick): grep: drop grep_buffer's "name" param
b9ef9 HEAD@{4}: rebase -i (pick): convert git-grep to use grep_source i
837de HEAD@{5}: rebase -i (pick): grep: make locking flag global
84f3d HEAD@{6}: checkout: moving from master to 84f3d
b0d66 HEAD@{7}: pull: Fast-forward
f6b50 HEAD@{8}: cherry-pick: add a TODO
98c05 HEAD@{9}: reset: moving to HEAD^
e11ee HEAD@{10}: checkout: moving from master to pu
77eac HEAD@{11}: commit: add a TODO
75f43 HEAD@{12}: commit: use the correct format identifier for unsigned
f88cc HEAD@{13}: pull --rebase: git grep
07873 HEAD@{14}: pull : Fast-forward
f70f7 HEAD@{15}: clone: from git://git.kernel.org/pub/scm/git/git.git
```

git add --patch

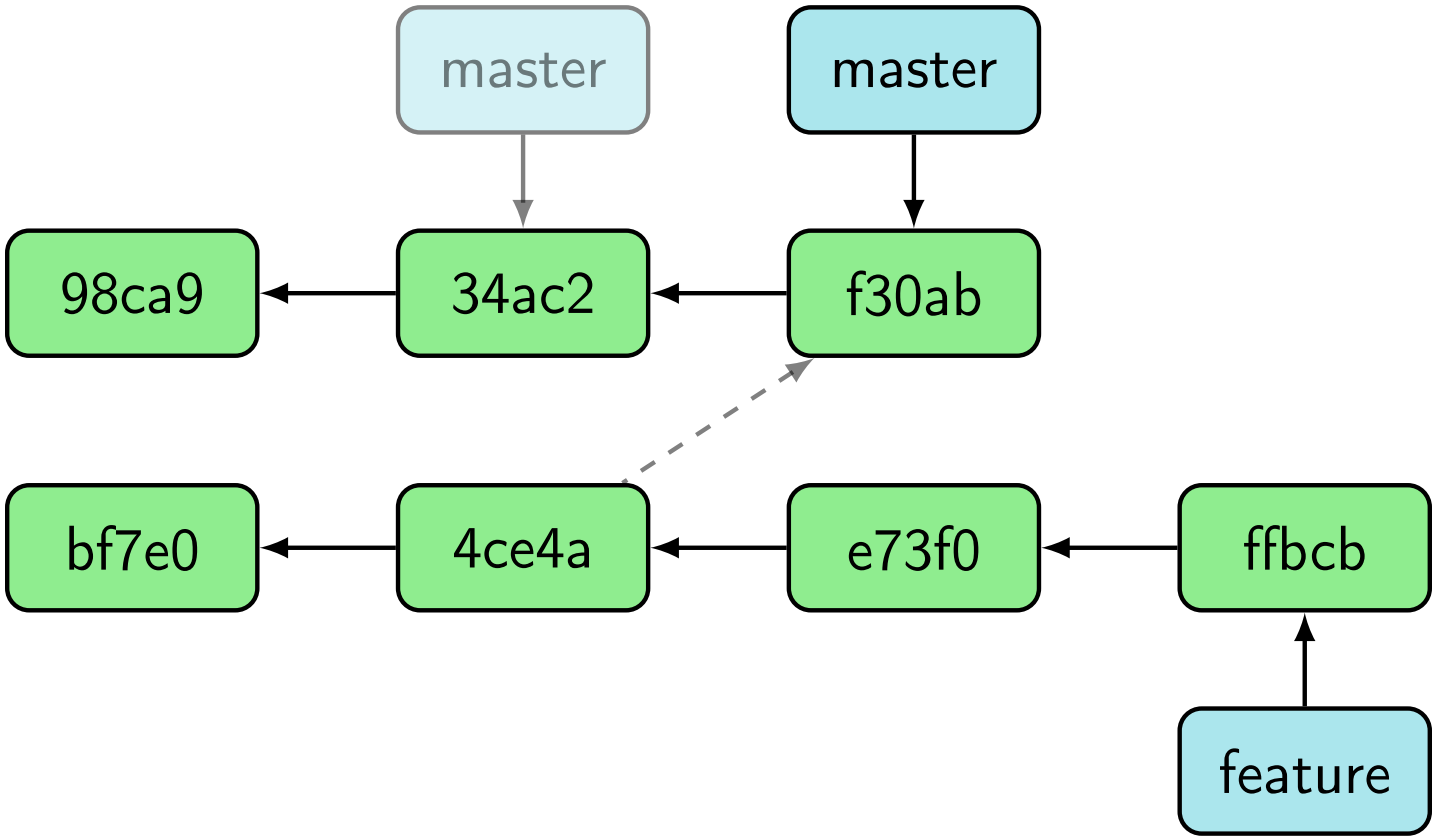
Permet d'ajouter une partie des modifications d'un fichier

```
$ git add -p
diff --git a/README b/README
@@ -1,5 +1,7 @@
 1
+2
 3
+4
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
Split into 2 hunks.
@@ -1,2 +1,3 @@
 1
+2
 3
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
@@ -2,4 +3,5 @@
 3
+4
Stage this hunk [y,n,q,a,d,/,K,g,e,?]? y
```

git cherry-pick

Applique un *commit* sur la branche courante.

```
$ git cherry-pick 4ce4a
```



git bisect

Permet de trouver l'origine d'un bug

```
$ git bisect start bad_commit good_commit
Bisecting: 20 revisions left to test after this (roughly 4 steps)
[e4f3edc] Sync with maint

$ git bisect run ./mytests
Bisecting: 9 revisions left to test after this (roughly 3 steps)
[42226d] pack-objects: remove bogus comment
...
Bisecting: 0 revisions left to test after this (roughly 1 step)
[9a4749] checkout -m: no need to insist on having all 3 stages

8280f2e0a0f4776c4b3008c9172fc0a3e7361534 is the first bad commit
commit 8280f2e0a0f4776c4b3008c9172fc0a3e7361534
Author: eregon <mail@me>
Date:   Sat Feb 18 17:36:47 2012 +0100

    Noooo! You found my hidden bug!
```

Services d'hébergement

- GitHub
- Gitorious
- Bitbucket, assembla (dépôts privés gratuits)
- En INGI (voir le wiki)
  `http://wiki.student.info.ucl.ac.be/index.php/Git`

Liens

- Les manpages : `git help [--web] <cmd>`
- `http://git-scm.com/book` : Pro Git book
- `http://louvilug.tuxfamily.org/` : Slides, aide-mémoire, exercices