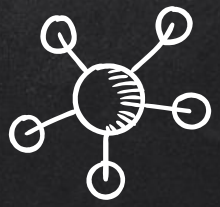


Git: découverte d'un outil de développement collaboratif et de gestion de versions

12,000,000+

utilisateurs git

38,000/AN

questions à propos de git sur Stack Overflow

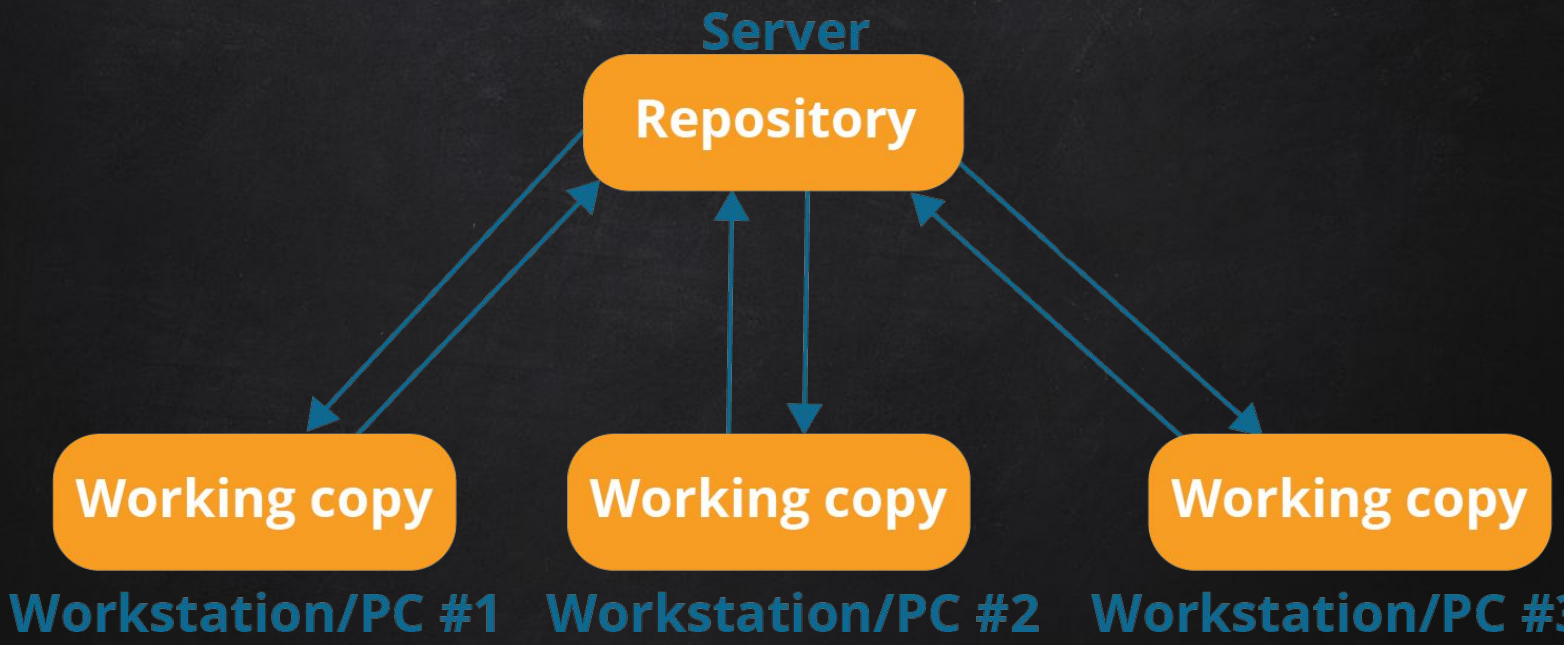
80%

des recherches sur la gestion de versions

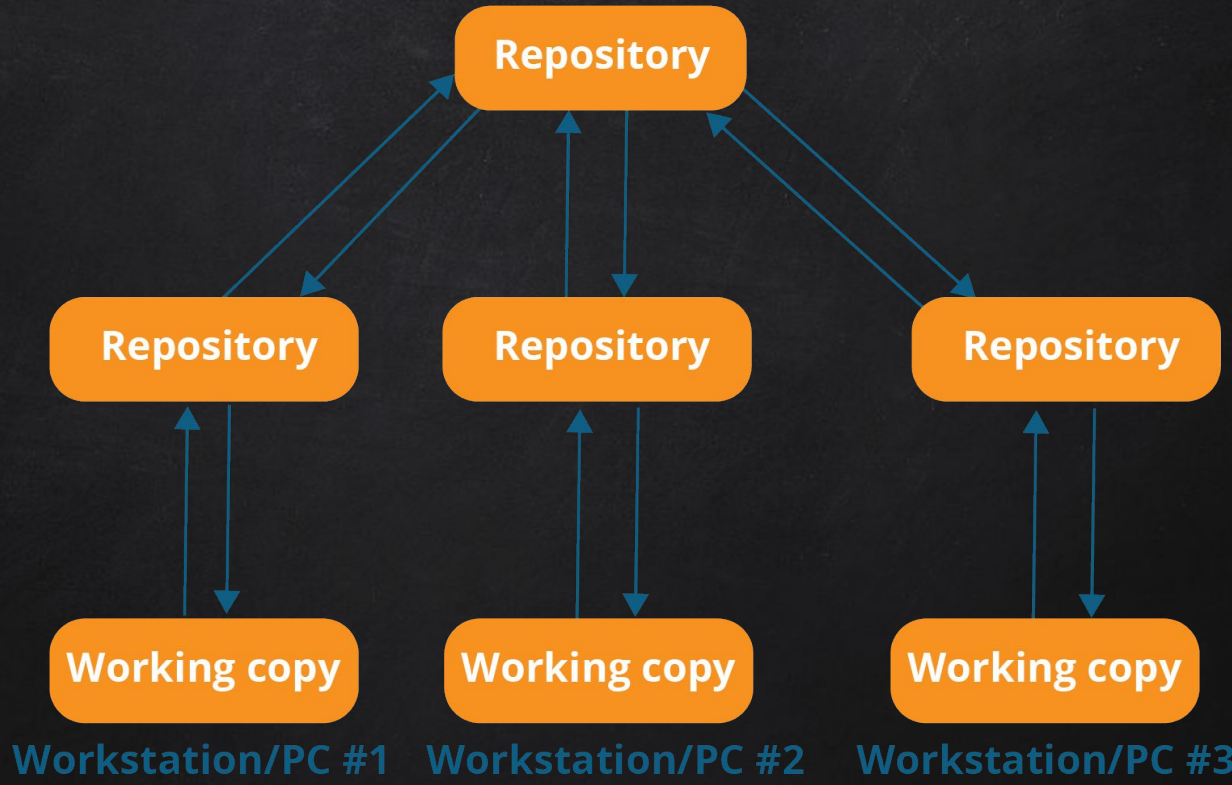
Pourquoi utiliser git ?

- Git conserve un historique version après version => facilité pour debugger
- Git permet de découper les modifications en commits ce qui apporte de la traçabilité du processus de développement
- Git permet l'utilisation de branches afin d'isoler chaque évolution du code
- Plusieurs développeurs peuvent travailler ensemble et en même temps
- Simple, performant et bien documenté
- Git ne nécessite aucun accès au réseau mis à part lors d'un push/pull ce qui permet de travailler offline

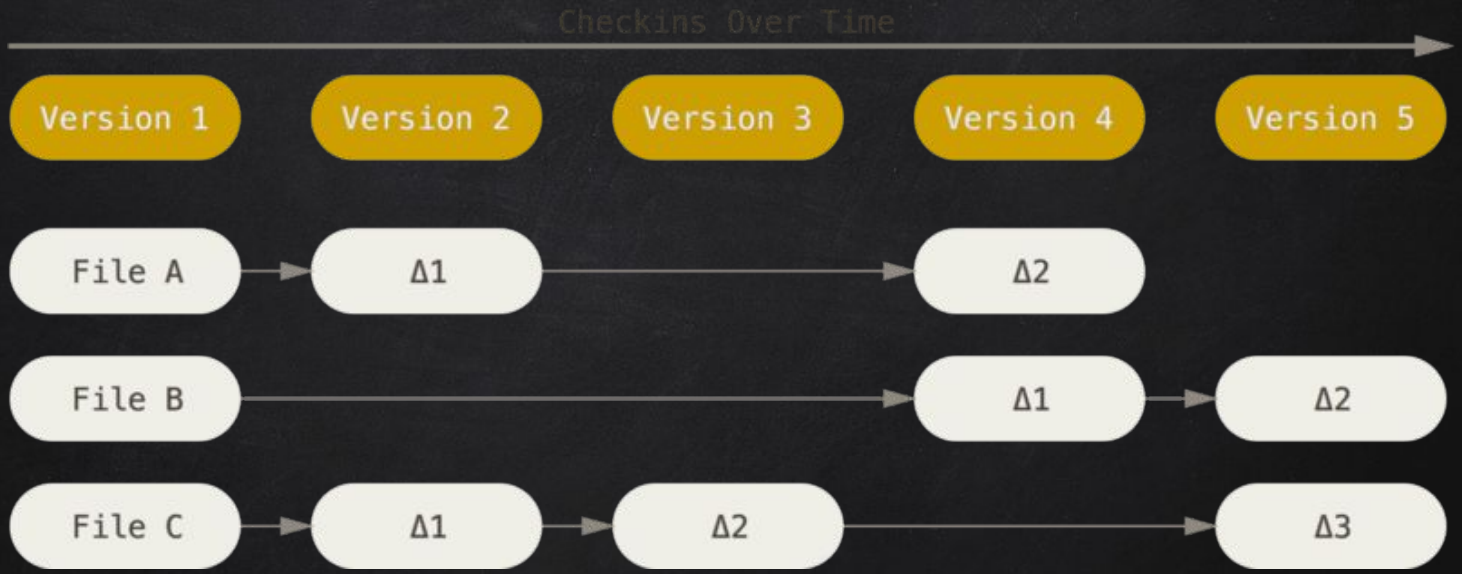
GESTION CENTRALISÉE



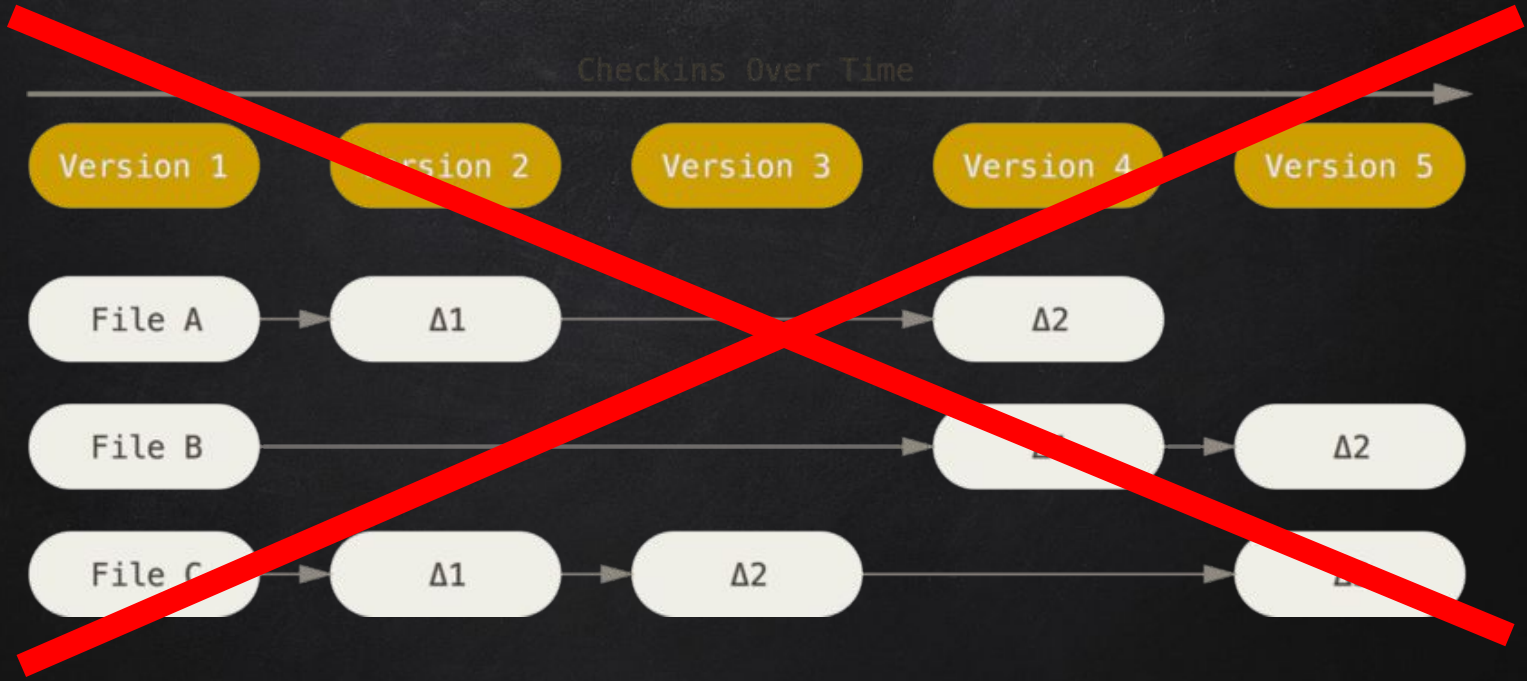
GESTION DISTRIBUÉE



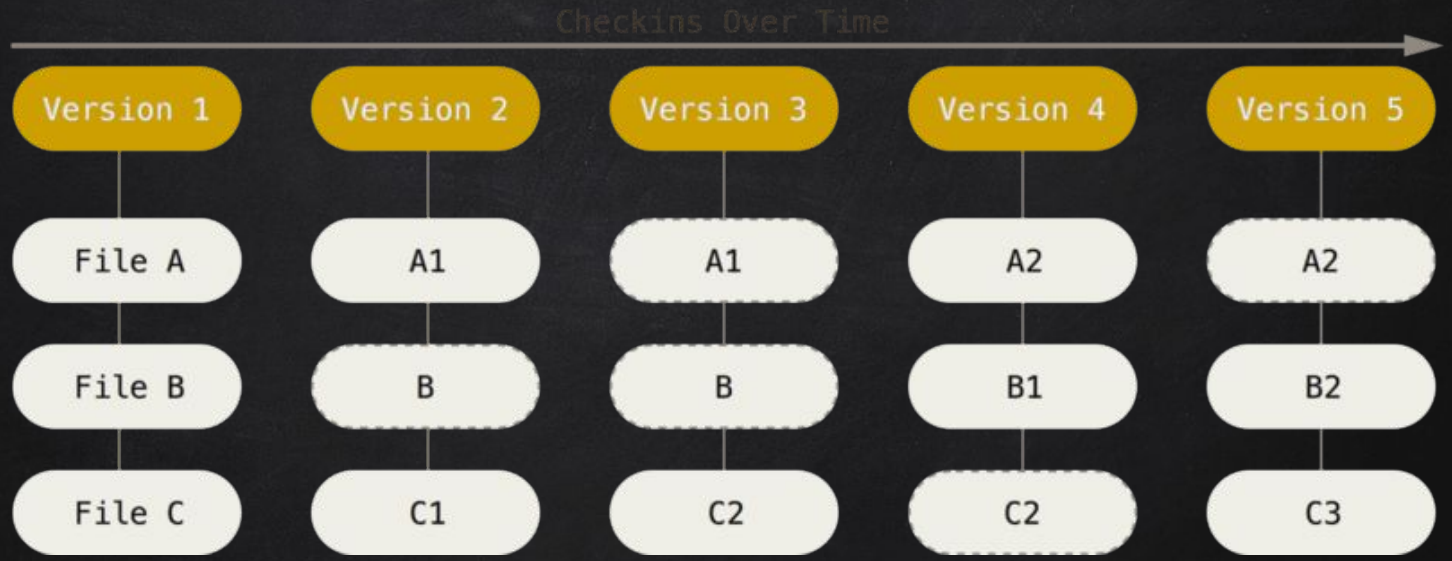
FONCTIONNEMENT



FONCTIONNEMENT



FONCTIONNEMENT



COMMITS

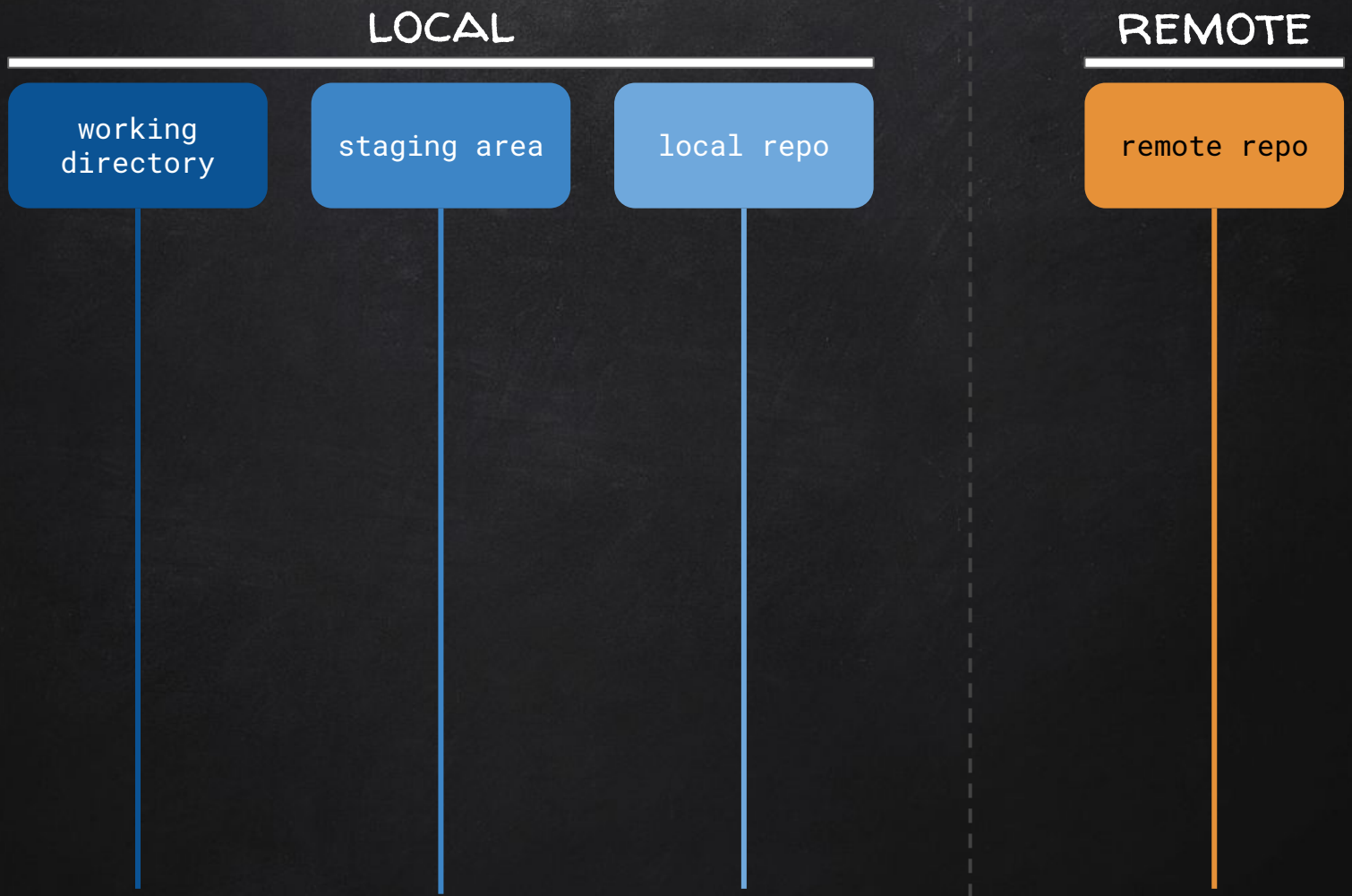
- HASH
- AUTEUR
- MESSAGE
- PARENTS
- FILS

Démarrage

```
$ git init
(initialiser un repo / dossier .git)

$ git clone git@github.com:akka/akka.git
(cloner un projet git existant)
```

Configuration

```
# Nom
git config --global user.name "meucaa"
# Email
git config --global user.email "lucas@gmail.com"
# Editeur
git config --global core.editor atom
```


LOCAL
REMOTE
working directory
staging area
local repo
remote repo

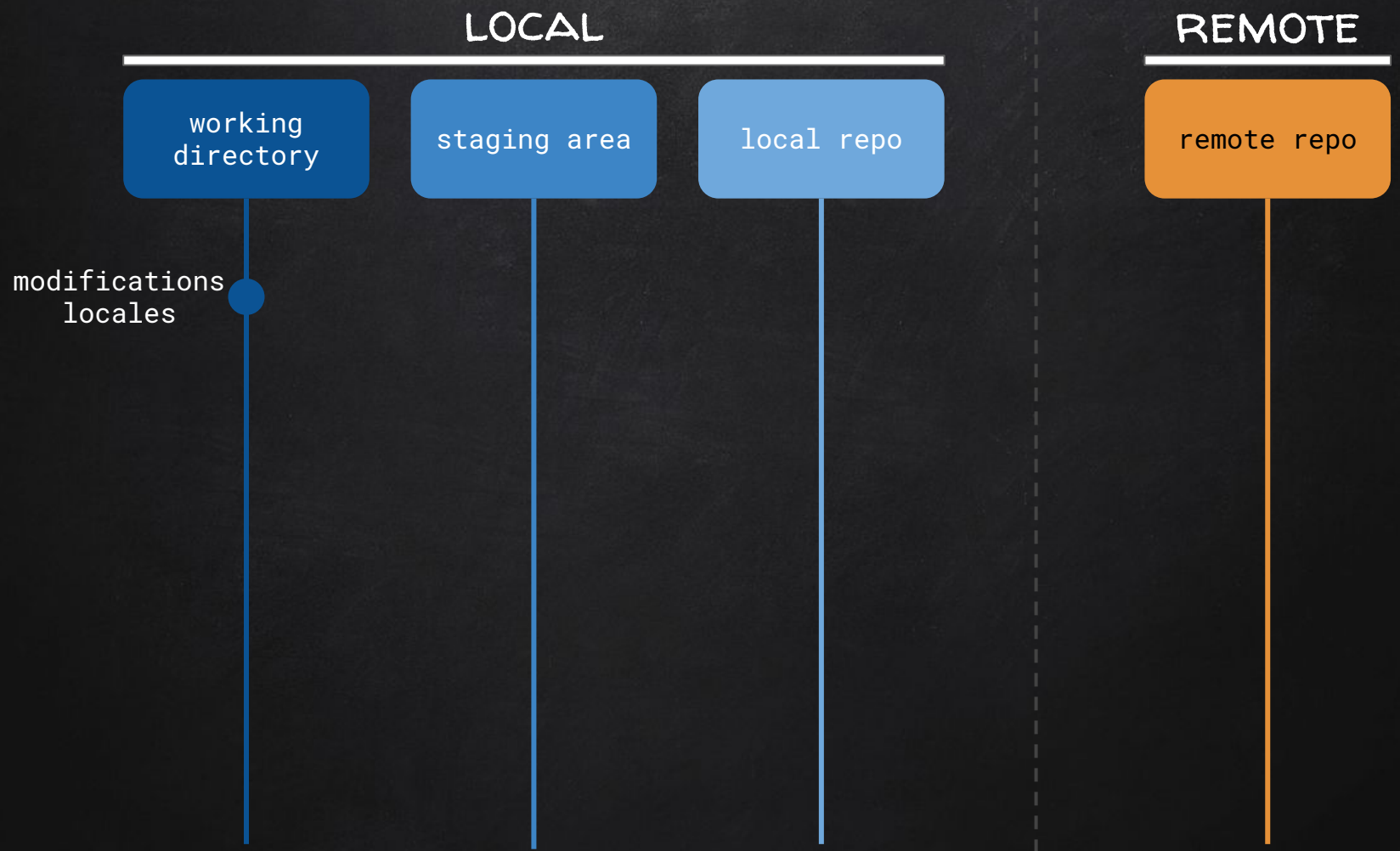
$ git status

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
```

Options utiles

-s, --short

-uall


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
modifications locales

$ git status

```
$ git status
On branch master.
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:

    modified:   Dockerfile
    modified:   README.md
```

```
$ git status -s
 M Dockerfile
 M README.md
```


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
modifications locales
git add

$ git add

```
$ git add -A
$ git add README.md
$ git add *.js
$ git add src/
```

Options utiles

-A -p/--patch

$ git status

```
$ git add Dockerfile
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:

    modified:   Dockerfile

Changes not staged for commit:

    modified:   README.md
```

```
$ git add Dockerfile
$ git status -s
 M README.md
M  Dockerfile
```

$ git add

```
$ git add --patch README.md
Texte existant ici
+BLABLABLA1
Texte existant aussi ici
+BLABLABLA2
Stage this hunk [y,n,q,a,d,/,s,e,?]? s
```

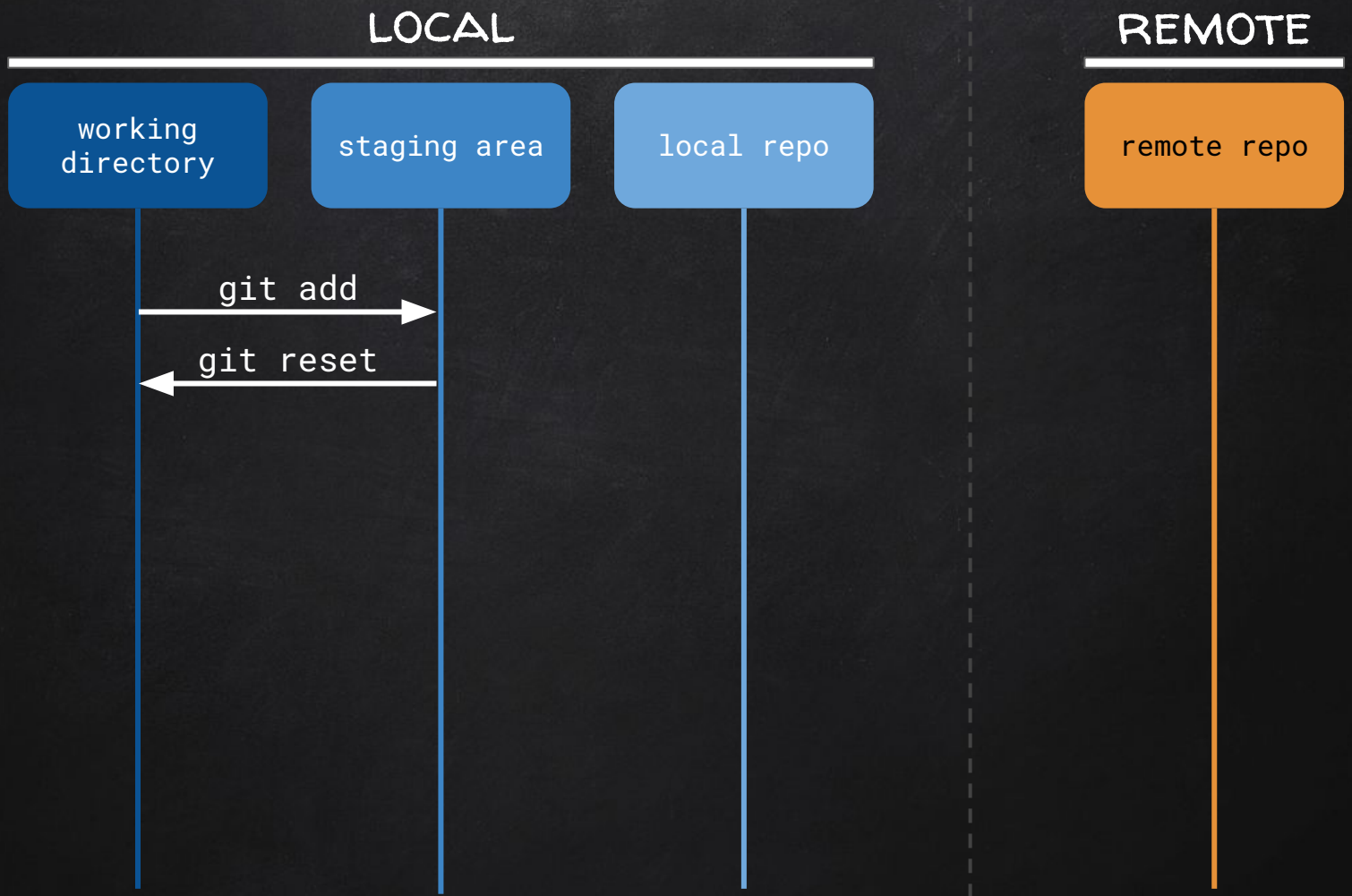
```
Texte existant ici
+BLABLABLA1
Texte existant aussi ici
Stage this hunk [y,n,q,a,d,/,s,e,?]? y
```

```
Texte existant aussi ici
+BLABLABLA2
Stage this hunk [y,n,q,a,d,/,s,e,?]? n
```

```
$ git status -s
MM README.md
M  Dockerfile
```

$ git diff --cached

```
$ git diff --cached


diff --git a/about.html b/about.html
index d09ab79..0c20c33 100644
--- a/about.html
+++ b/about.html
@@ -19,7 +19,7 @@
   </div>

   <div id="headerContainer">
-    <h1>About&lt/h1>
+    <h1>About This Project&lt/h1>
   </div>

   <div id="contentContainer">

index 1932d95..d34d56a 100644
```

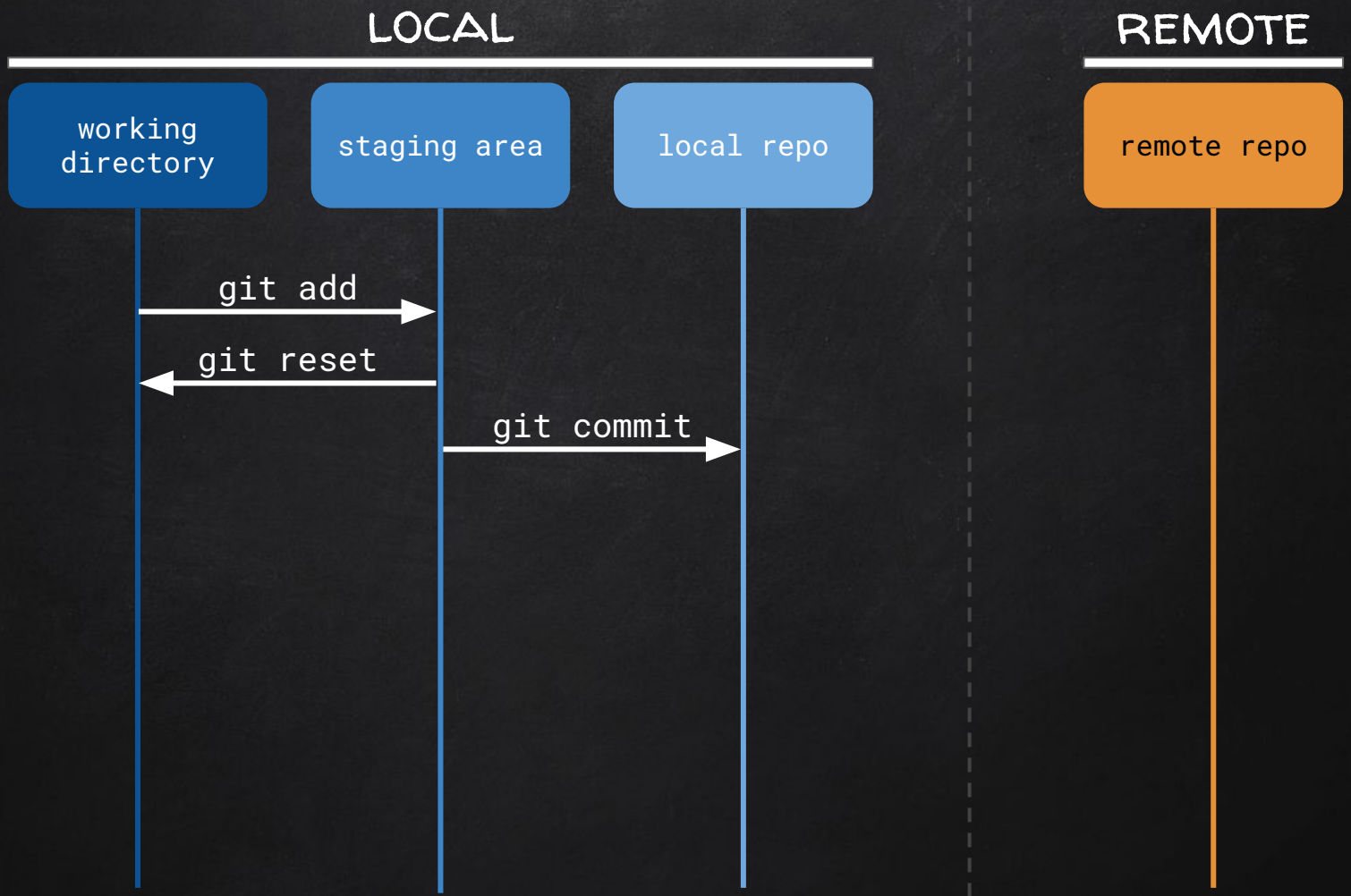

LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset

$ git reset

```
$ git status -s
MM README.md
M  Dockerfile

$ git reset Dockerfile
Unstaged changes after reset:
M    Dockerfile

$ git status -s
MM README.md
 M Dockerfile
```

Options utiles

-p/--patch


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset
git commit

$ git commit

```
$ git status -s
MM README.md
 M Dockerfile

$ git commit
(Ouvre l'éditeur configuré pour édition du message de commit)

$ git commit -m "Modification du README"
[master e389ac4] Modification du README.md
 1 file changed, 1 insertion(+)
```

$ GIT LOG

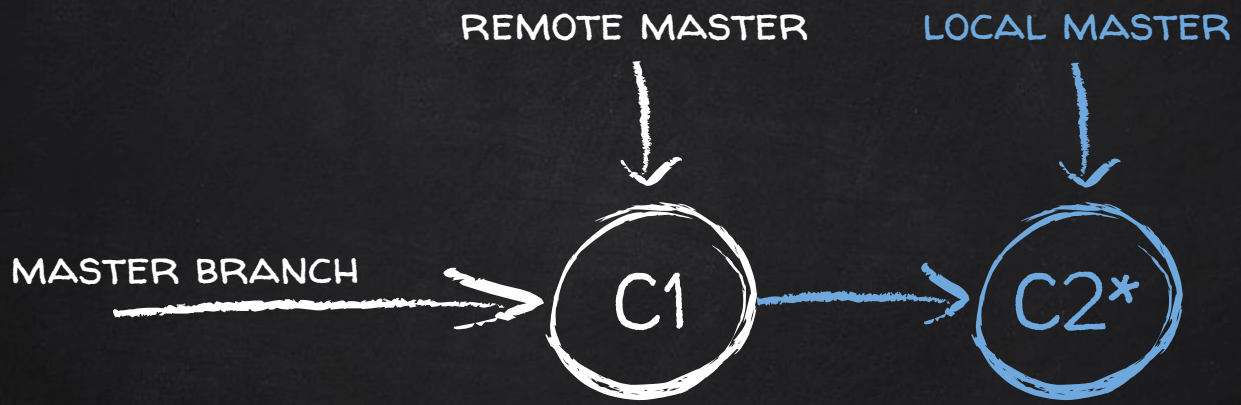
```
$ git log
commit e389ac4192abd2cb73e4c81000ac43 (HEAD -> master)
Author: meucaa <lucas.vuillier@gmail.com>
Date:   Sun May 13 19:42:11 2018 +0200

    Modification du README.md

commit ba01a2126a0f4ed097e88479829fcb (origin/master)
Author: meucaa <lucas.vuillier@gmail.com>
Date:   Sun Sep 25 01:20:03 2016 +0200

    Un message de commit précédent
. . . .
. . . .
```

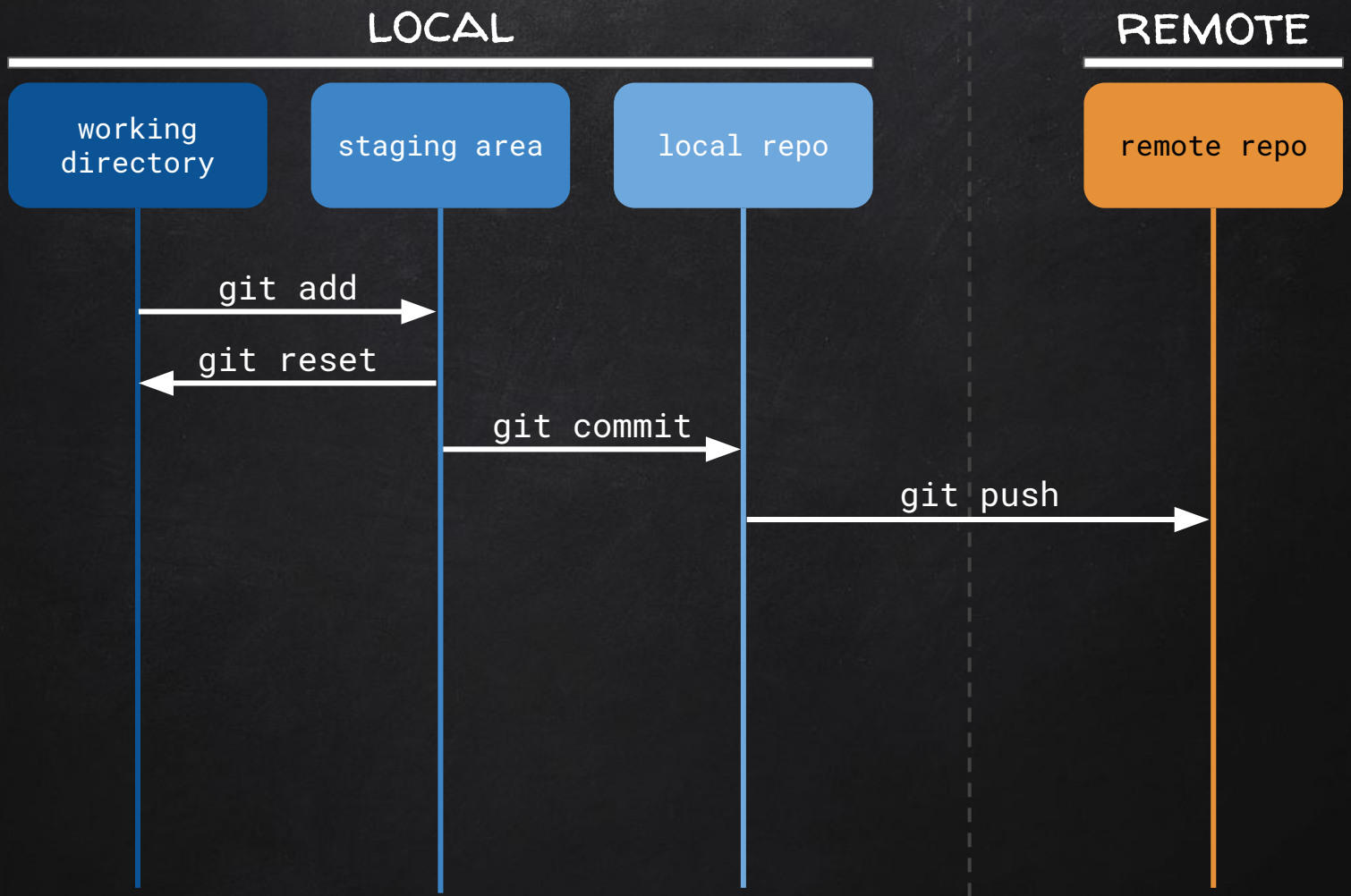
$ git log

```
$ git log --oneline -2
e389ac4 (HEAD -> master) Modification du README.md
ba01a21 (origin/master) Un message de commit précédent
```

Options utiles

--oneline
-10
--grep=<regexp>
--author=<pattern>


REMOTE MASTER
LOCAL MASTER
MASTER BRANCH
C1
C2*


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset
git commit
git push

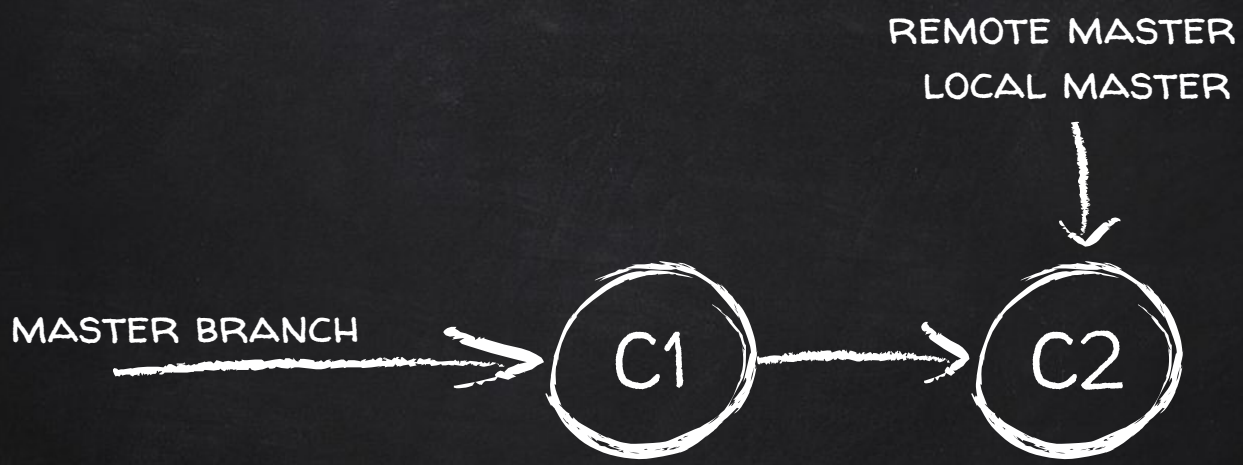
$ git push

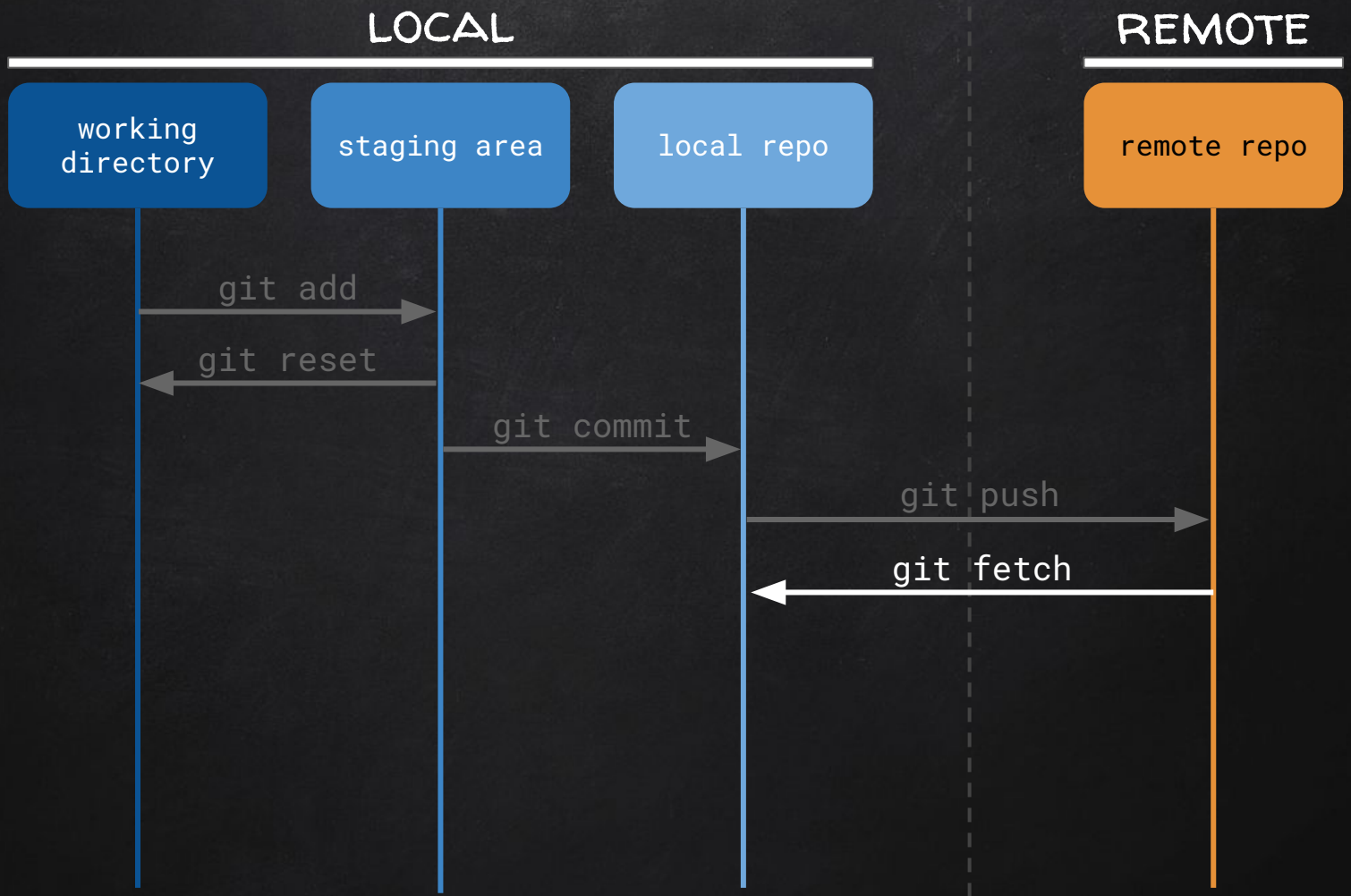
```
$ git push origin master
To github.com:meucaa/nom-du-projet.git
   ba01a21..e389ac4  master -> master
```

Options utiles

--force

-u/--set-upstream


REMOTE MASTER
LOCAL MASTER
MASTER BRANCH
C1
C2

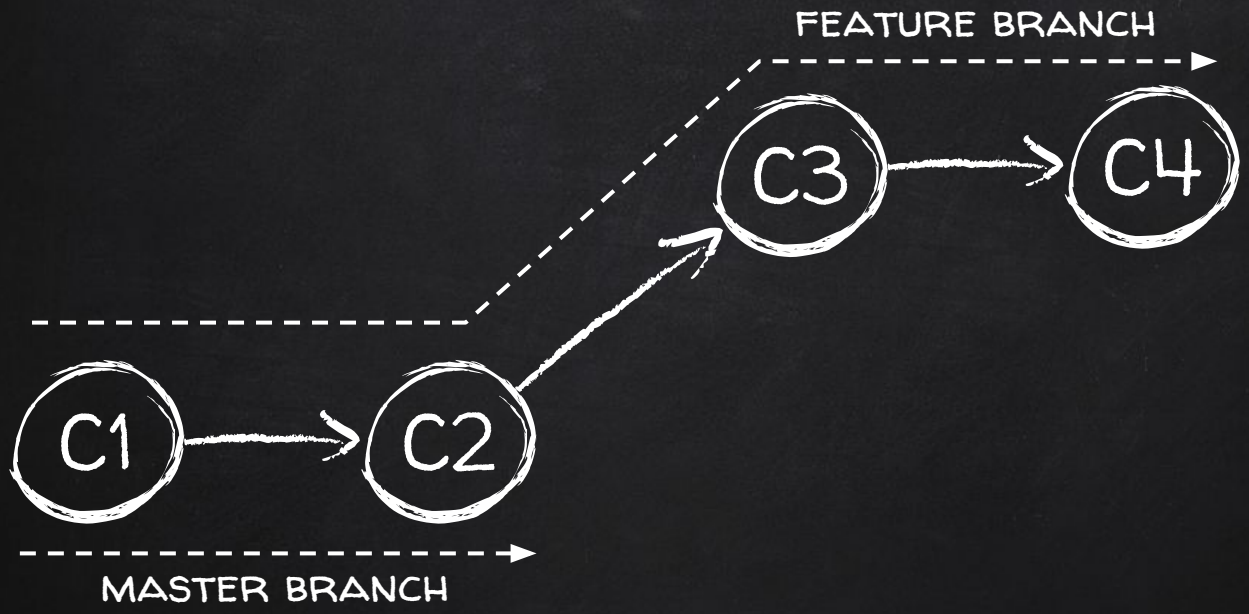

LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset
git commit
git push
git fetch

BRANCHES

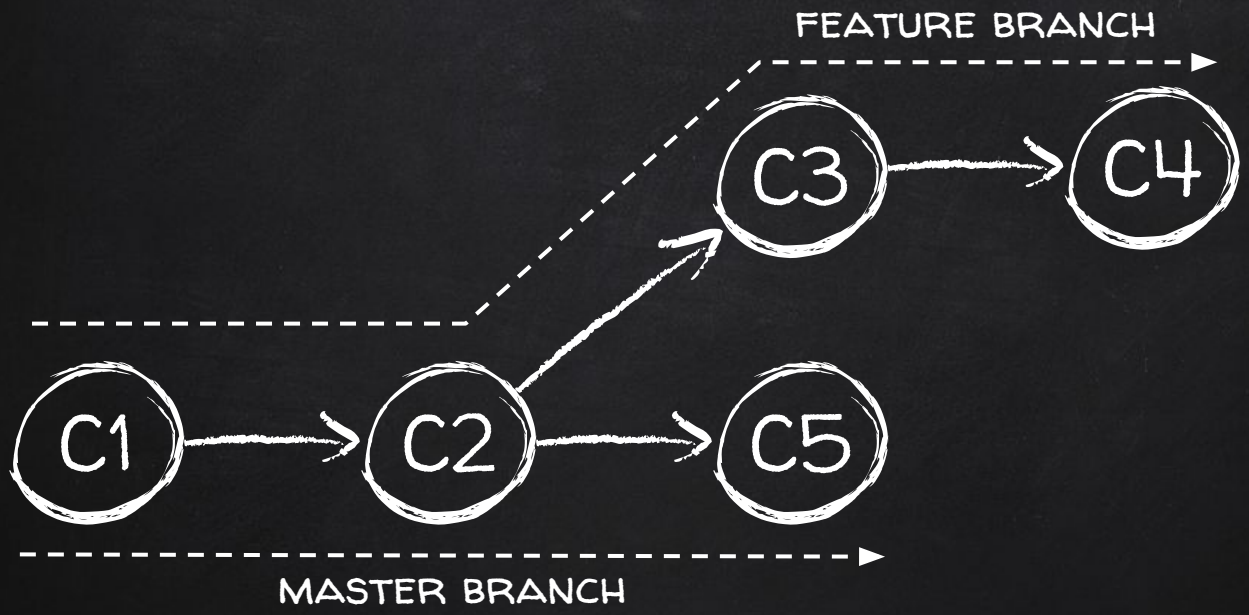
Créer une branche

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.


$ git checkout -b feature
Switched to a new branch 'feature'
```

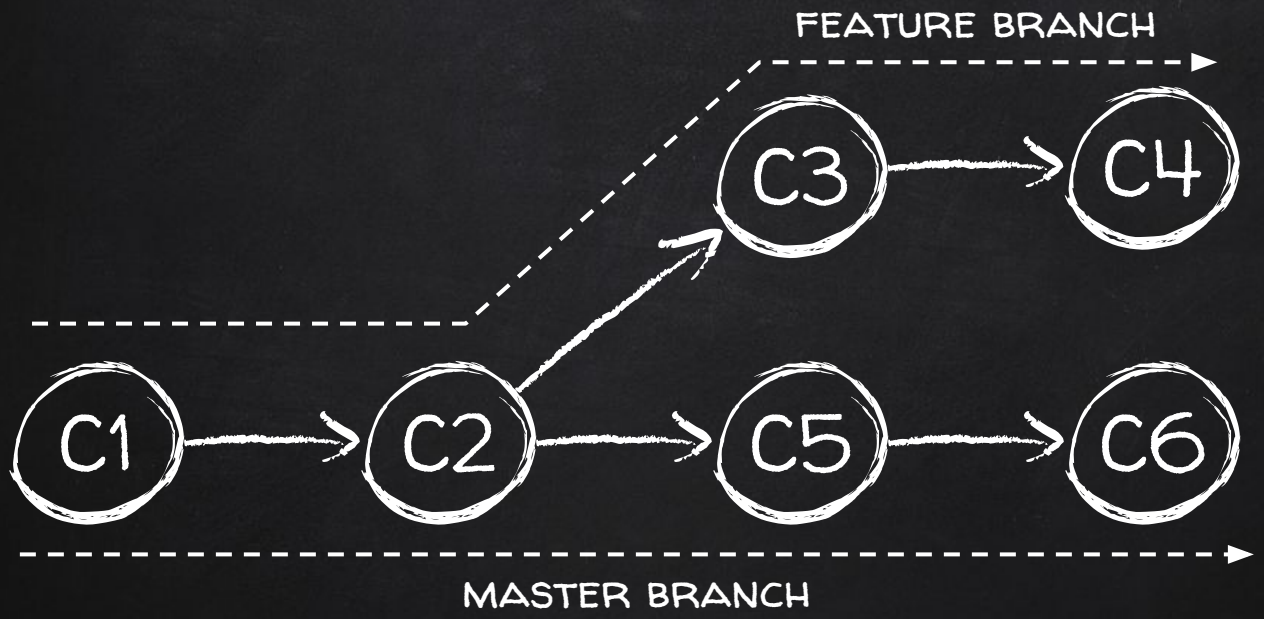
BRANCHES



BRANCHES

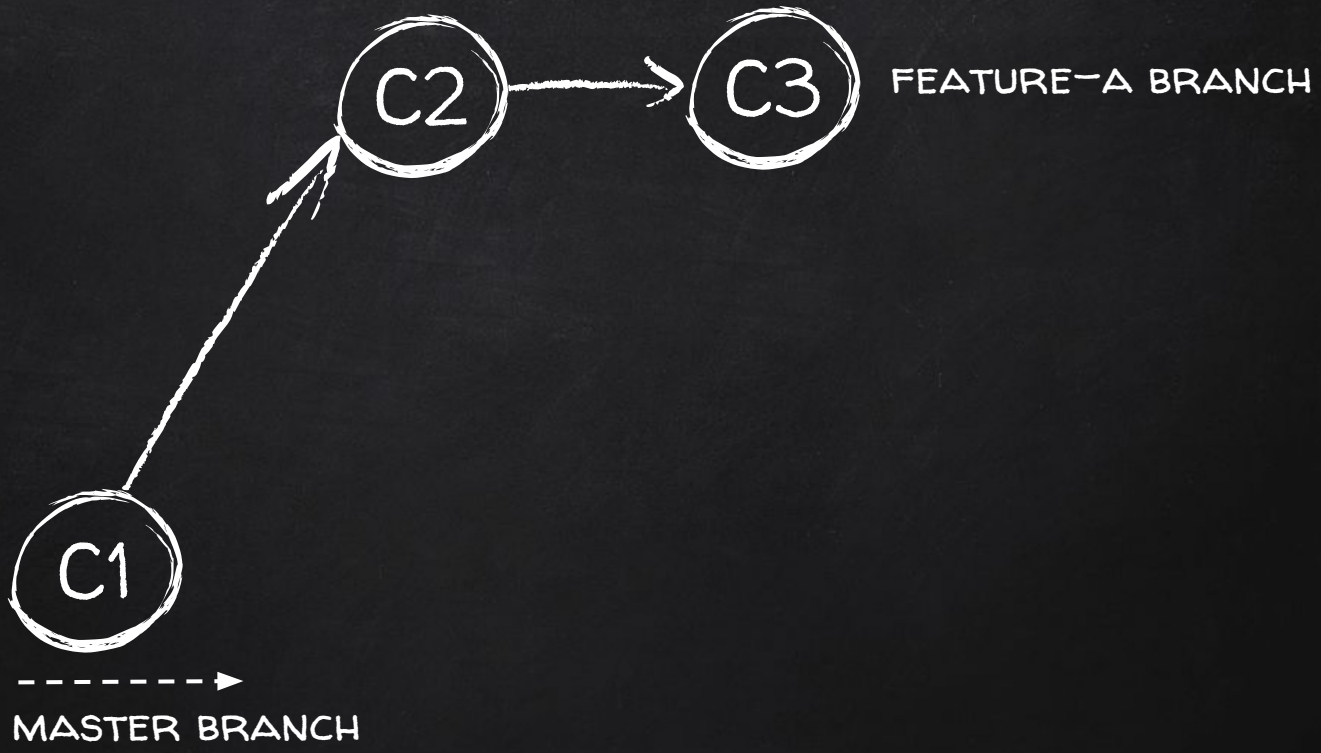


BRANCHES

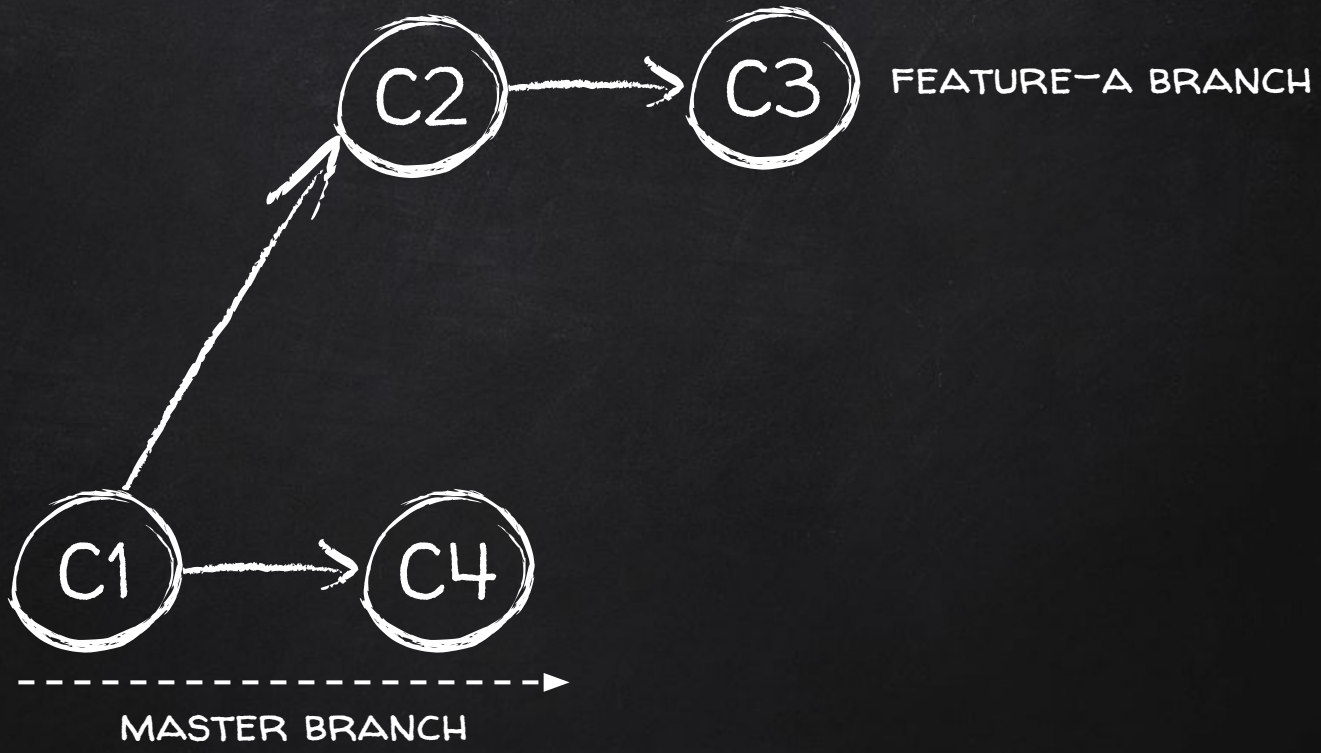


BRANCHES

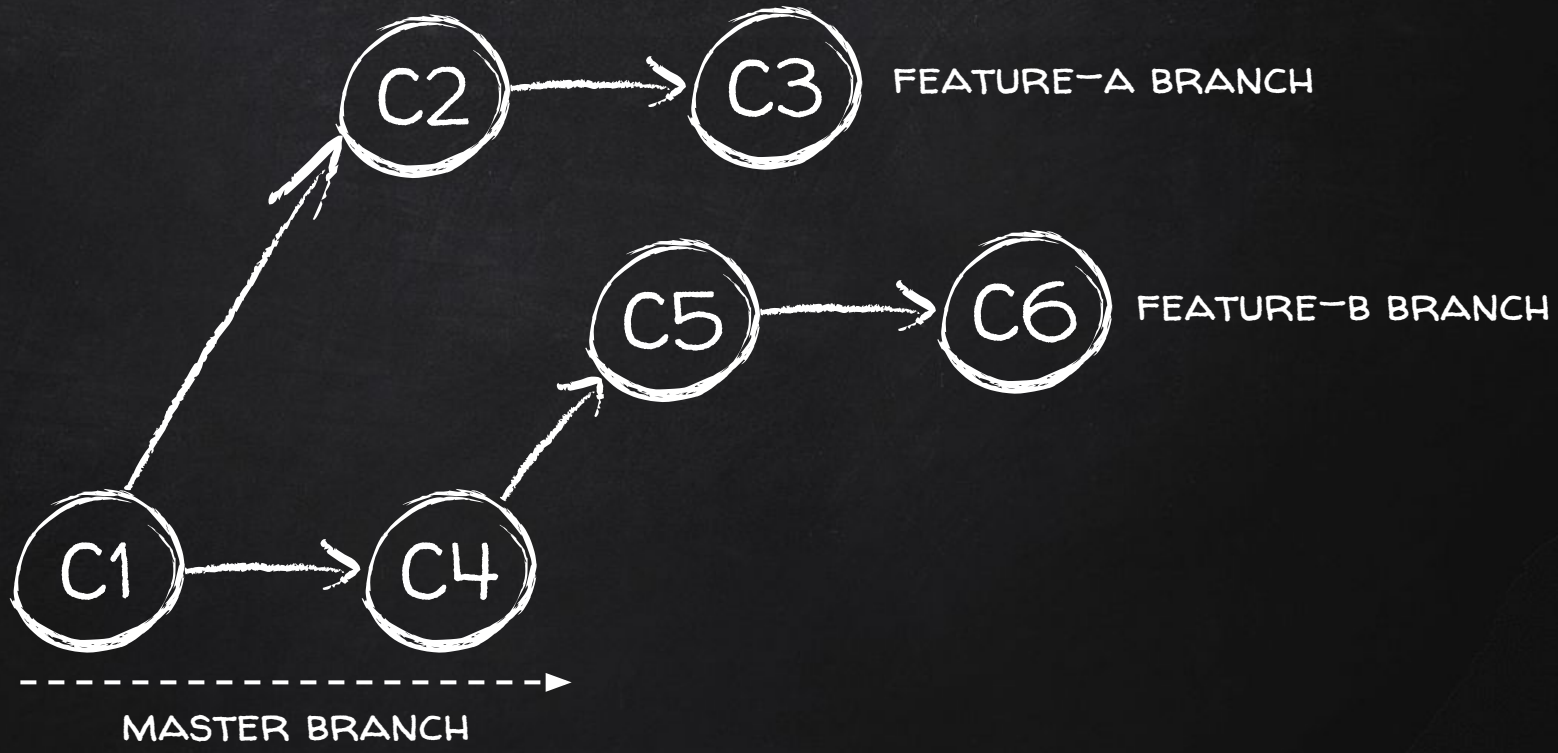
BRANCHES



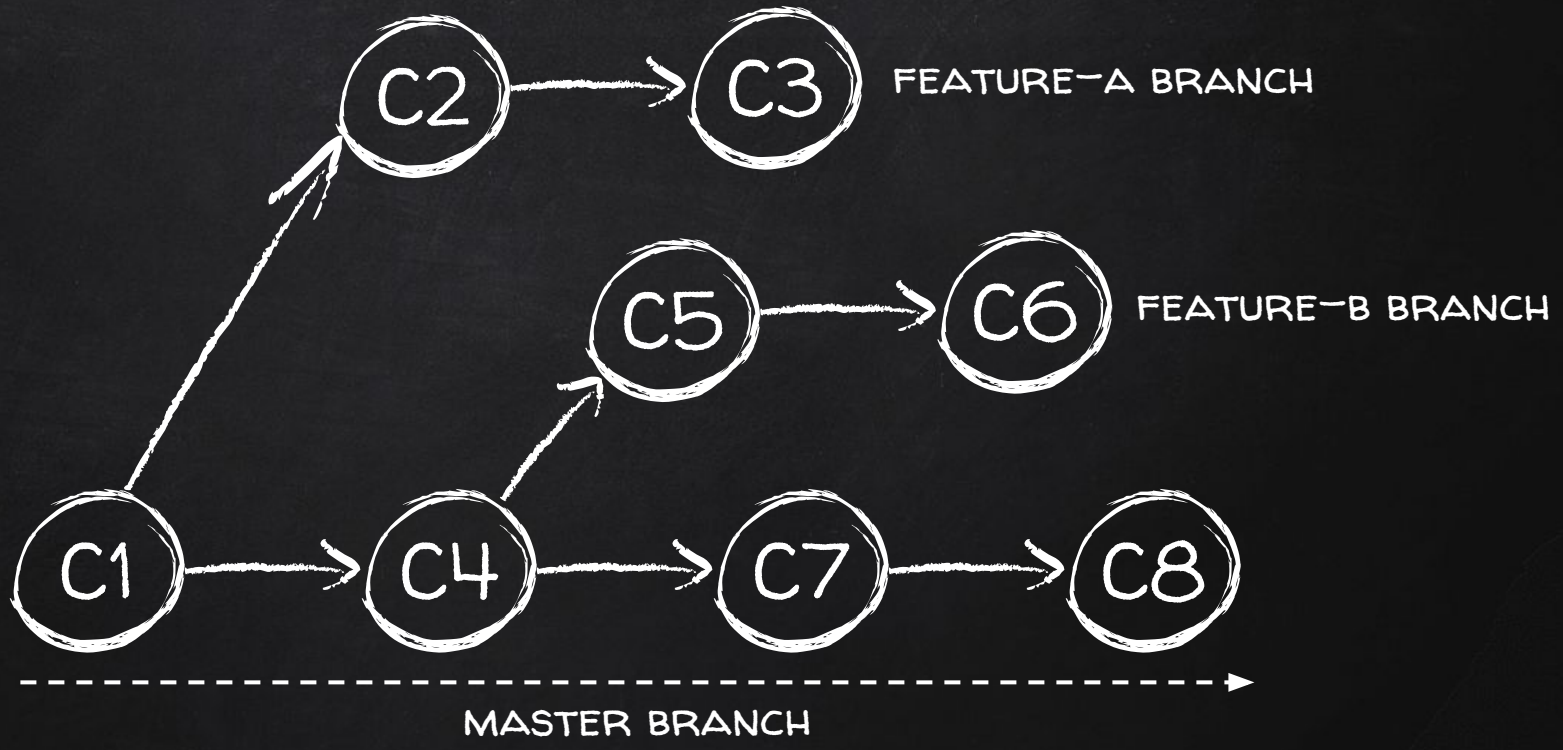
BRANCHES



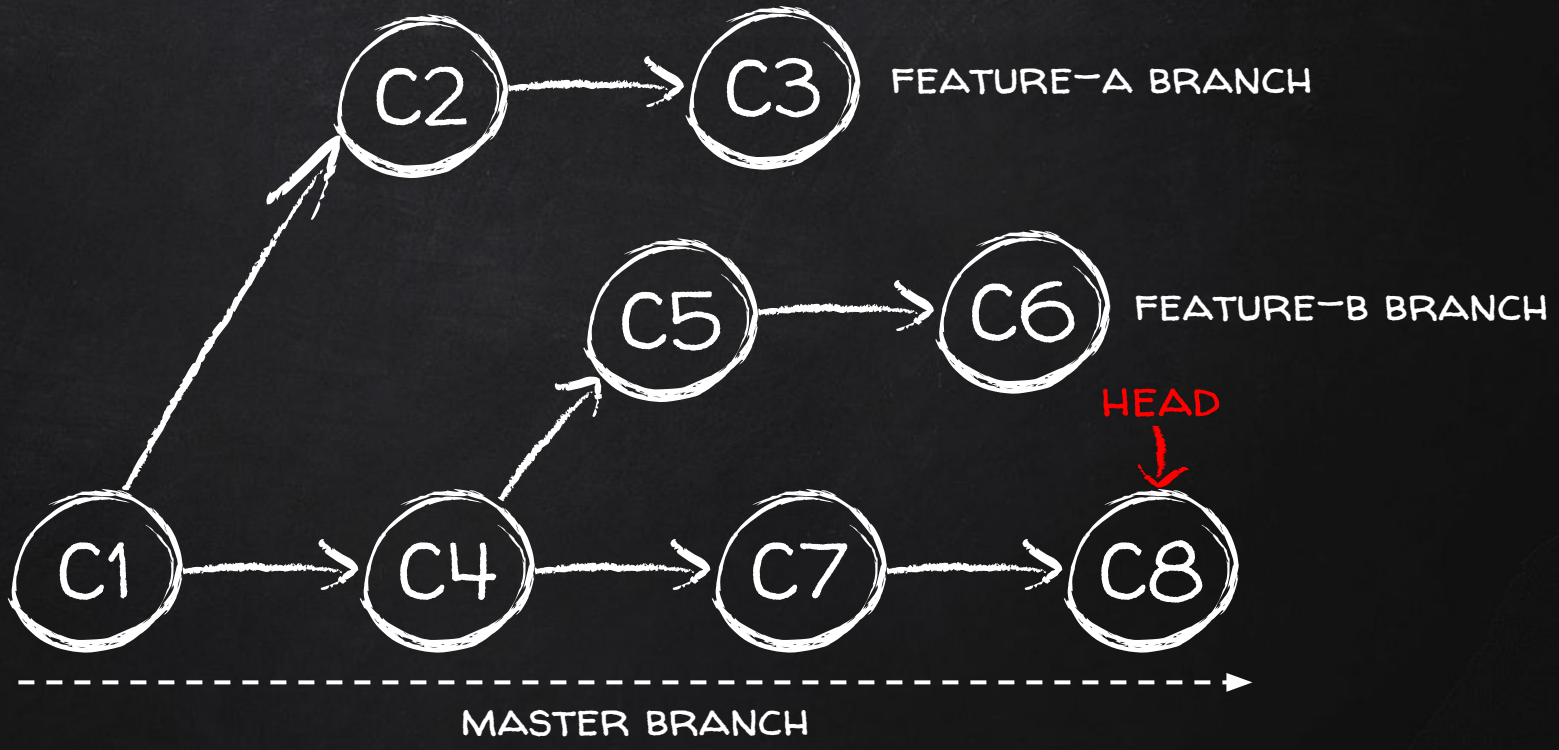
BRANCHES


C2
C3
FEATURE-A BRANCH
C5
C6
FEATURE-B BRANCH
C1
C4
MASTER BRANCH

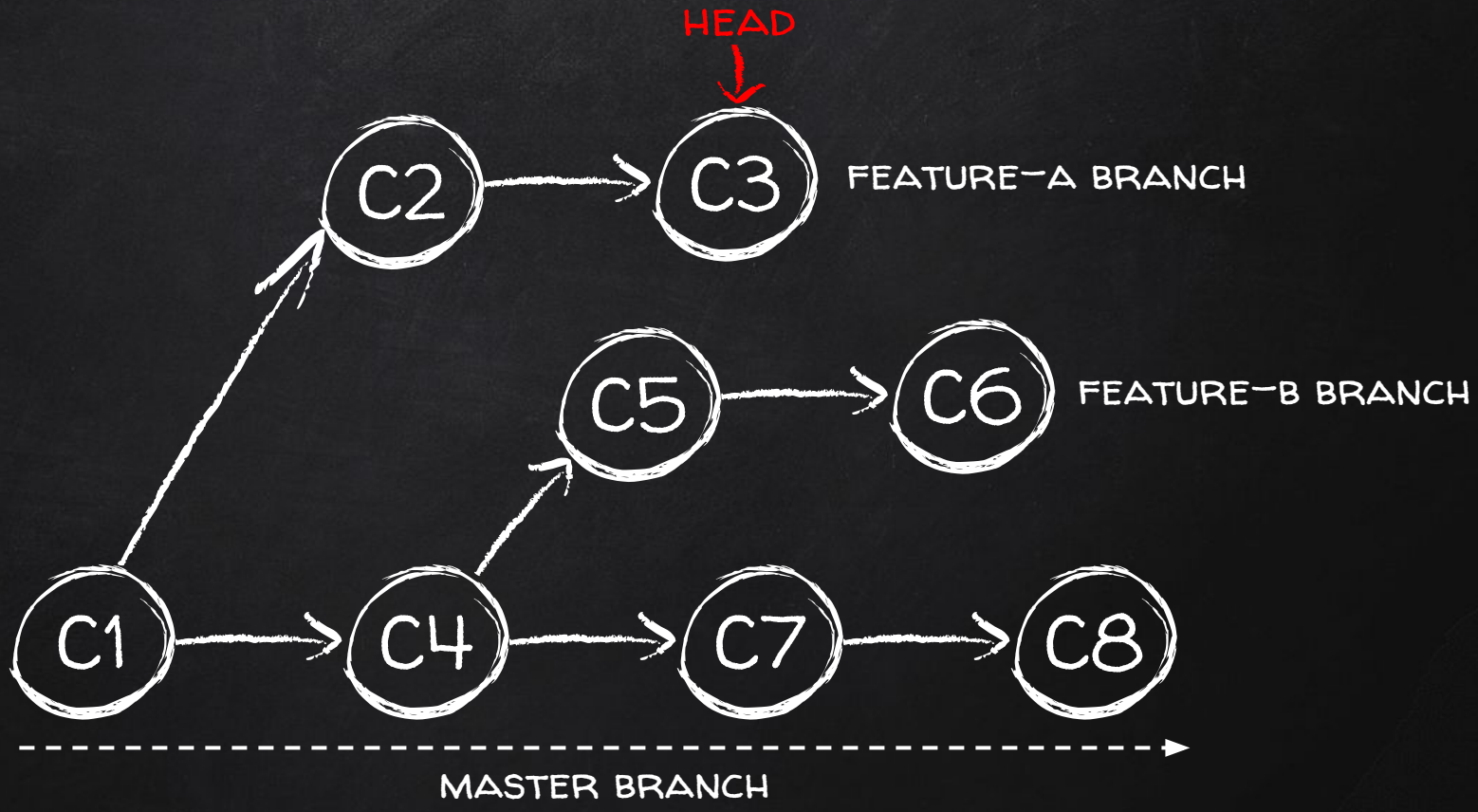
BRANCHES


C2
C3
FEATURE-A BRANCH
C5
C6
FEATURE-B BRANCH
C1
C4
C7
C8
MASTER BRANCH

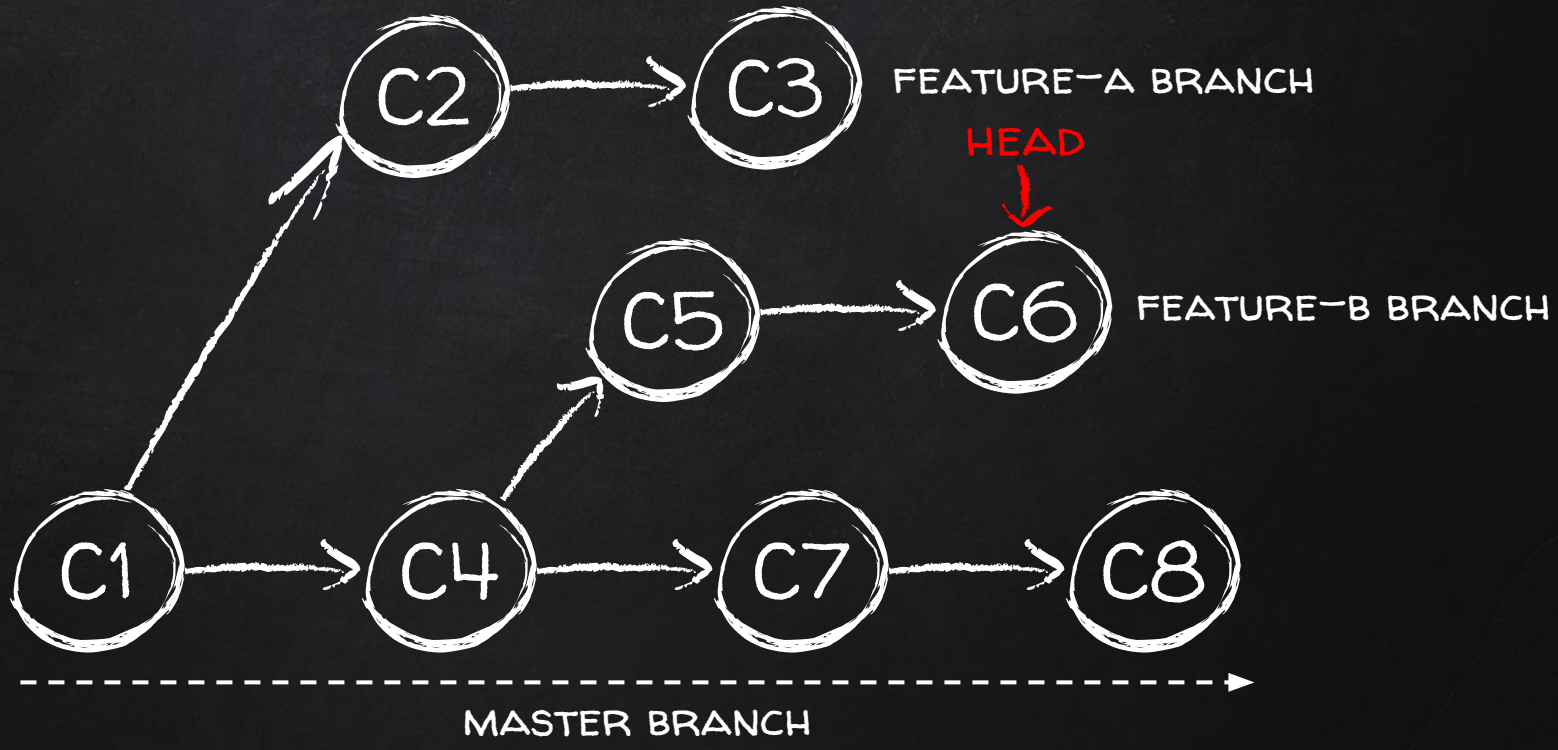
$ GIT CHECKOUT



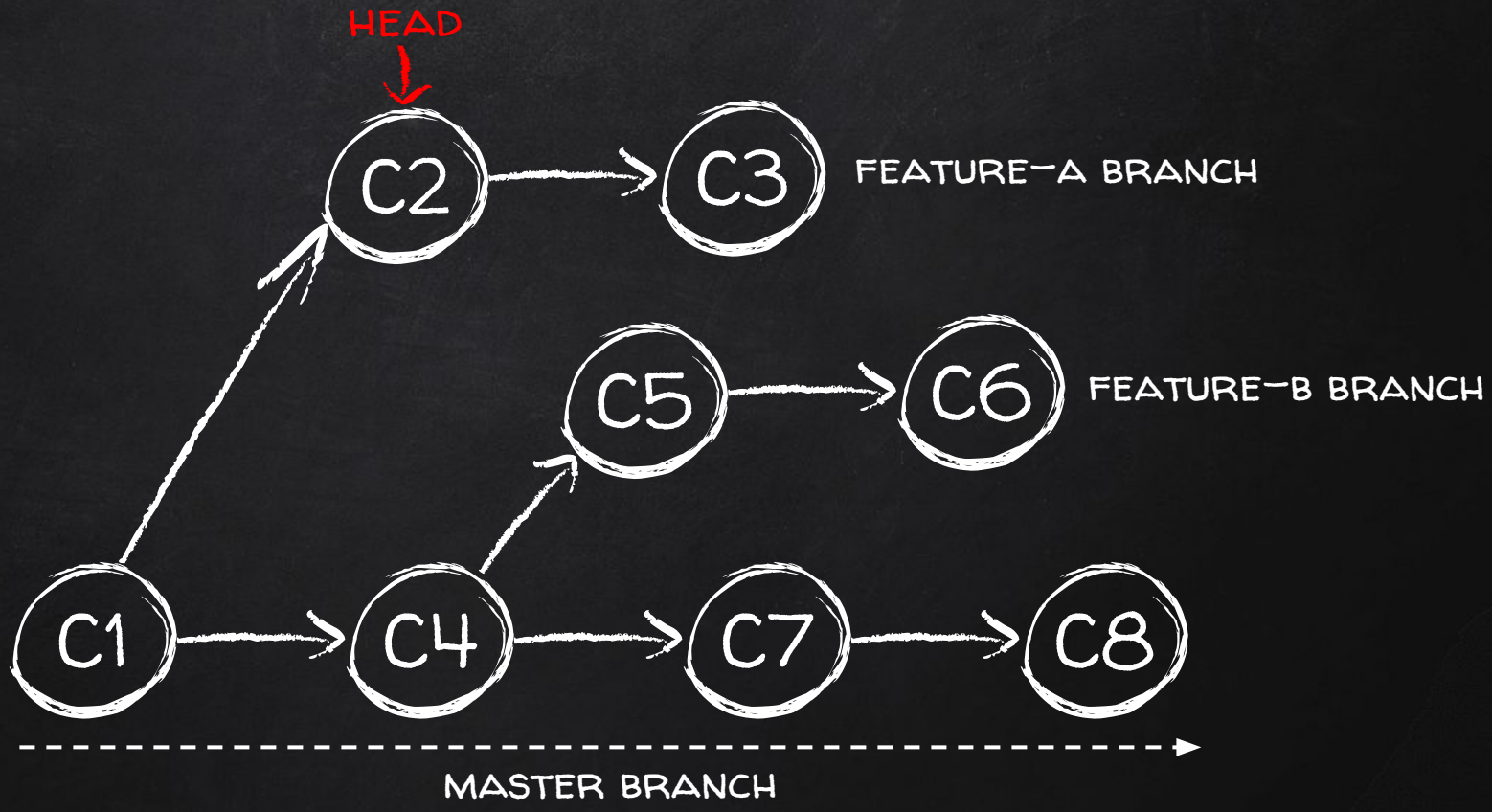
$ GIT CHECKOUT FEATURE-A



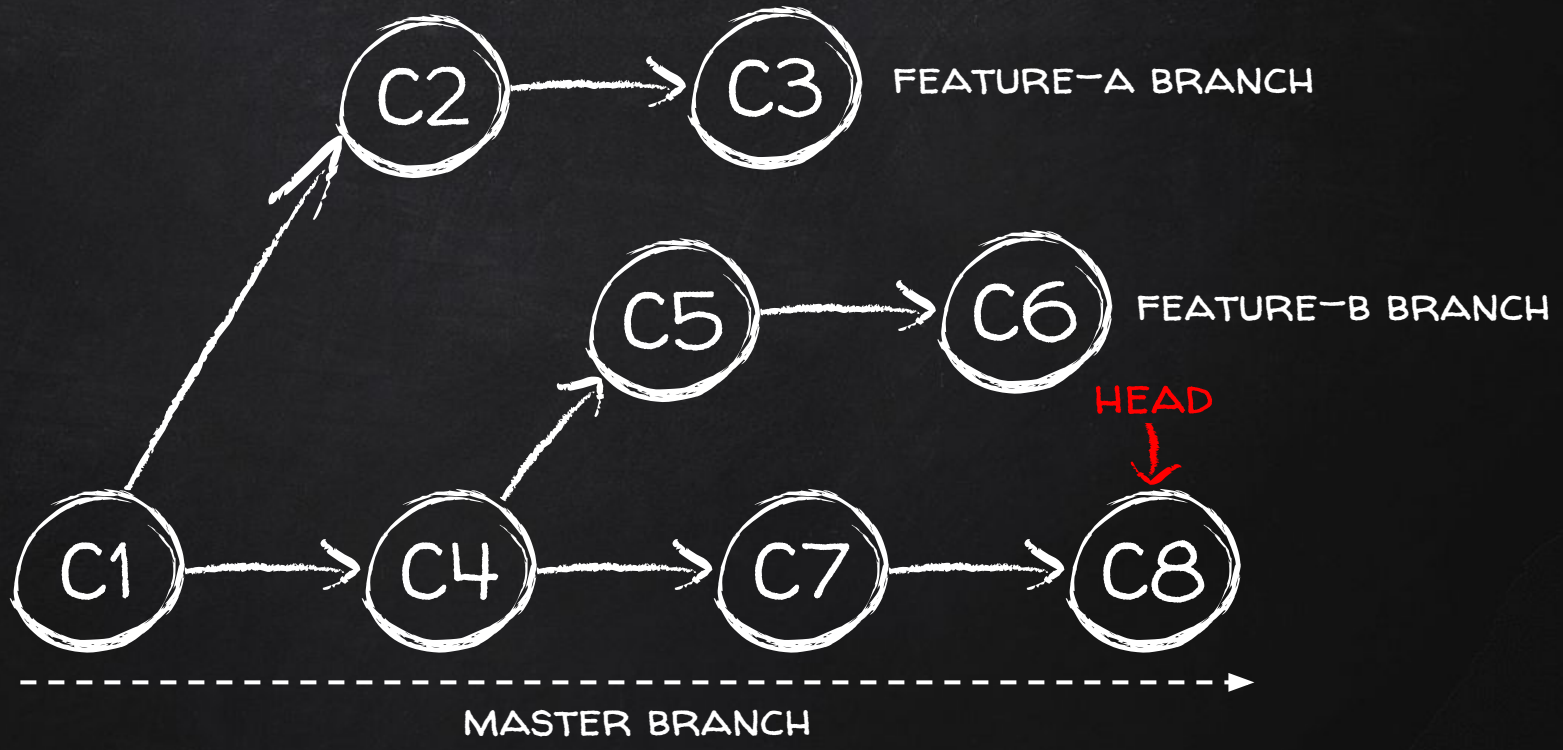
$ GIT CHECKOUT FEATURE-B



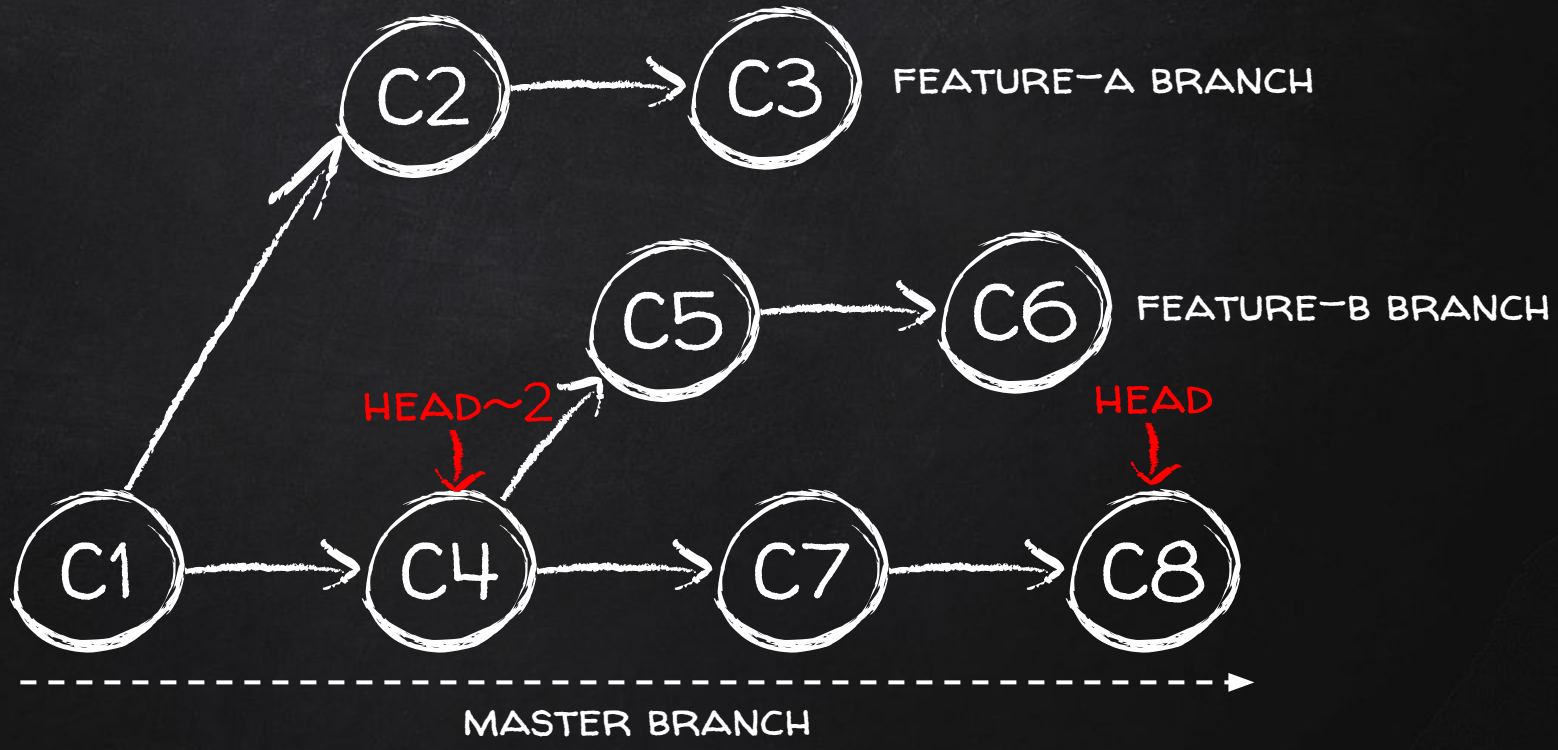
$ GIT CHECKOUT C2



$ GIT CHECKOUT C8



$ GIT CHECKOUT HEAD~2



$ git branch

```
$ git branch
  feature-a
  feature-b
* master


$ git branch -v
  feature-a a109fa6 Message du commit C3
  feature-b e389ac4 Message du commit C6
* master    b43aaf5 Message du commit C8
```

Options utiles

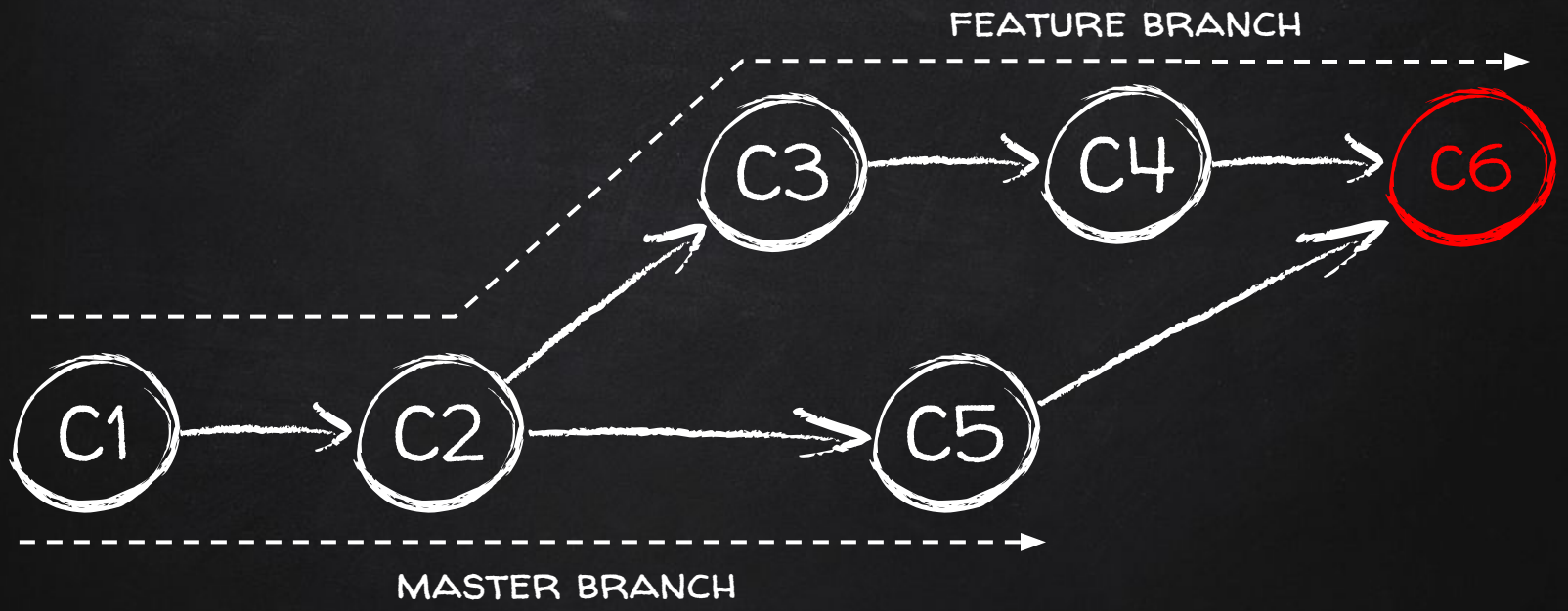
-a -d/-D -v

Merge or Rebase

Quand merger ?

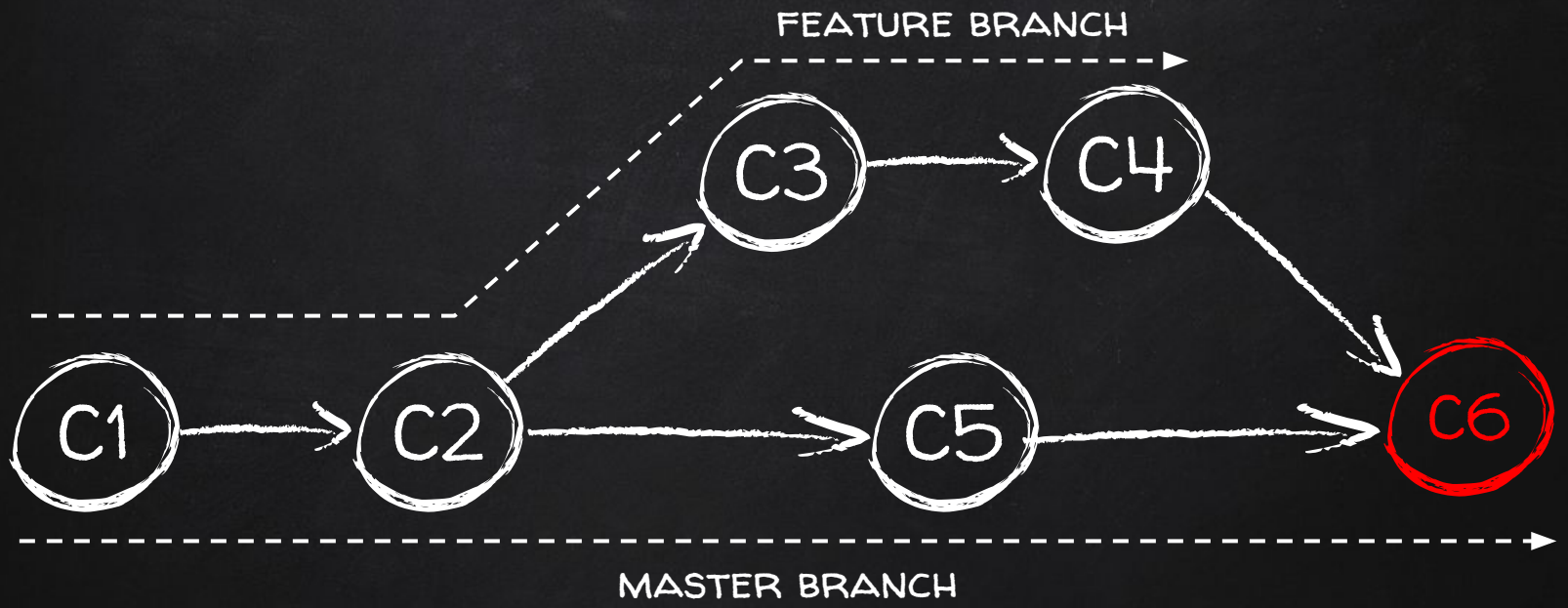
- Rappatrier des changements d'une branche sur laquelle on est en retard
- Intégrer un ensemble de modifications sur la branche source

Mettre a jour sa branche

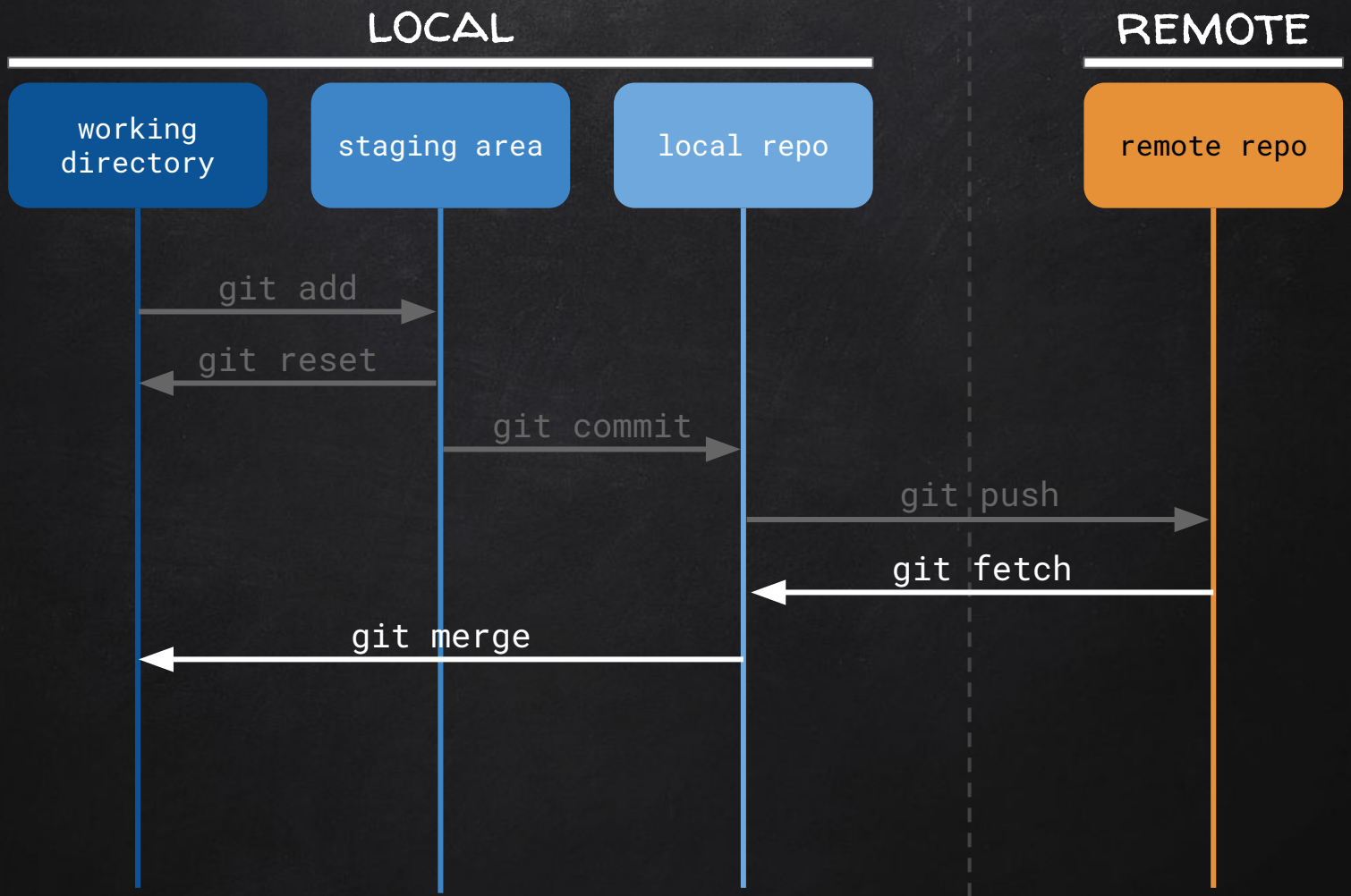


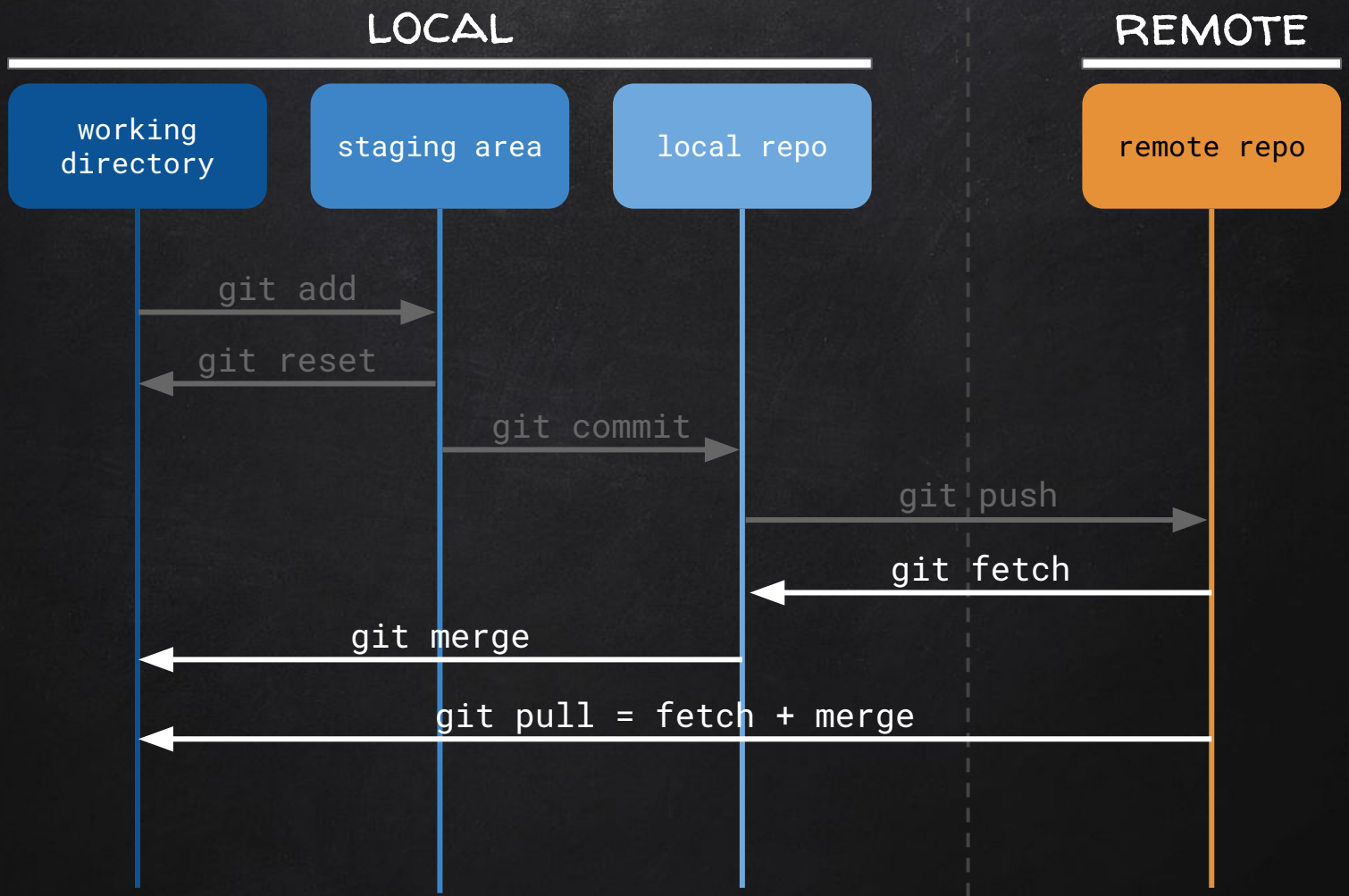
MERGE COMMIT

INTÉGRER SUR LA SOURCE

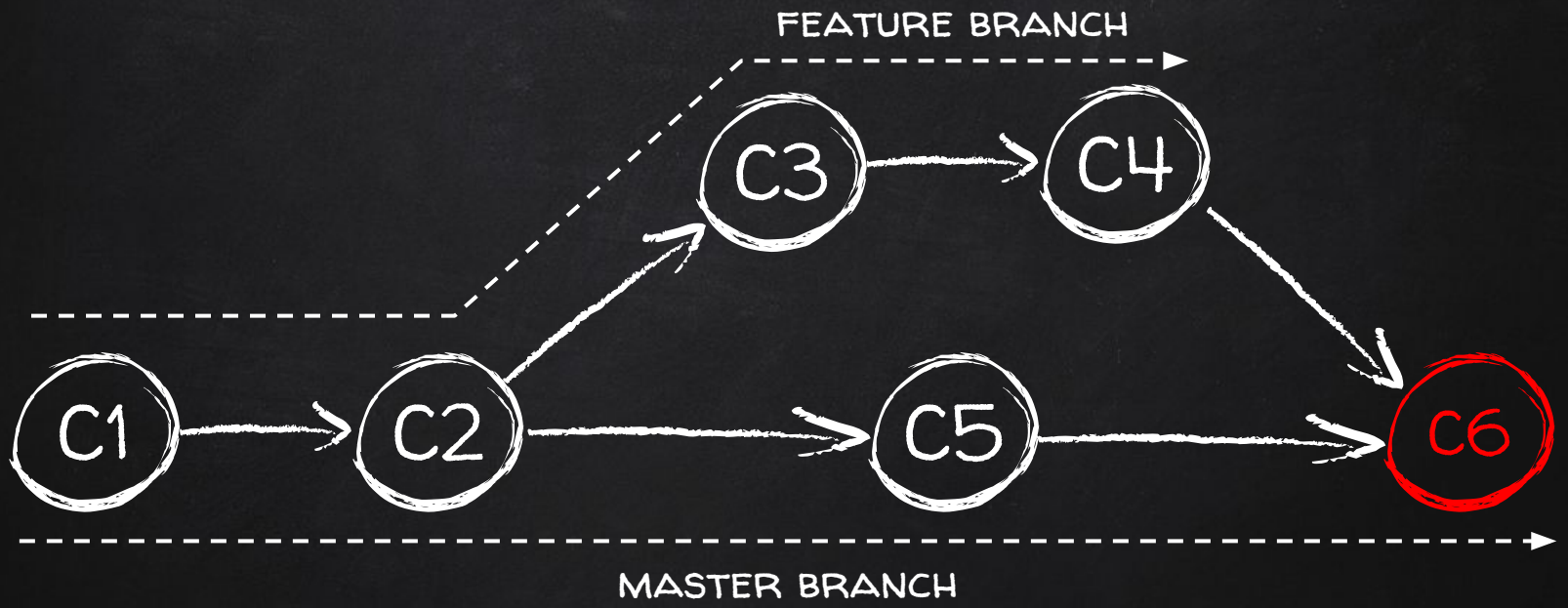


MERGE COMMIT


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset
git commit
git push
git fetch
git merge


LOCAL
REMOTE
working directory
staging area
local repo
remote repo
git add
git reset
git commit
git push
git fetch
git merge
git pull = fetch + merge

CONFLITS LORS D'UN MERGE



MERGE COMMIT

CONFLITS LORS D'UN MERGE

```
$ git checkout master
$ git merge feature
$ git status
# On branch master
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:      README.md
#
no changes added to commit (use "git add" and/or "git commit
-a")
```

CONFLITS LORS D'UN MERGE

```
//autre code
<<<<<<< HEAD
modification issue de la branche master
=======
modification issue de la branche feature
>>>>>>> feature
//autre code
```

CONFLITS LORS D'UN MERGE

```
//autre code
modification issue de la branche master
//autre code
```

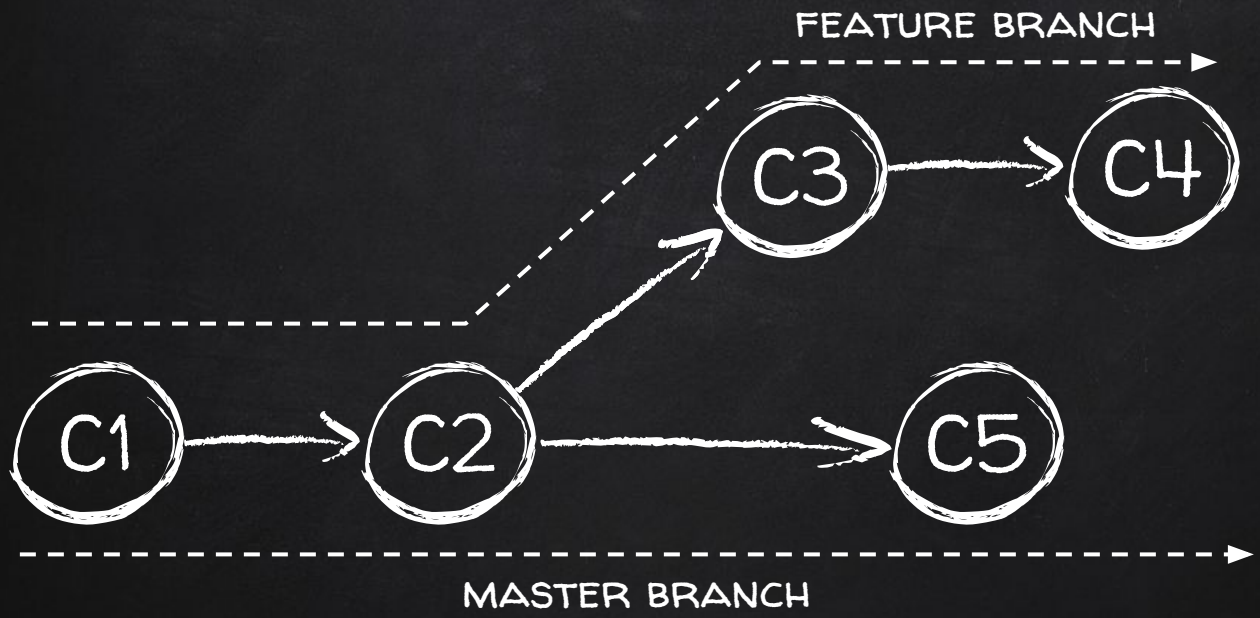
CONFLITS LORS D'UN MERGE

```
$ git add README.md
$ git commit
```

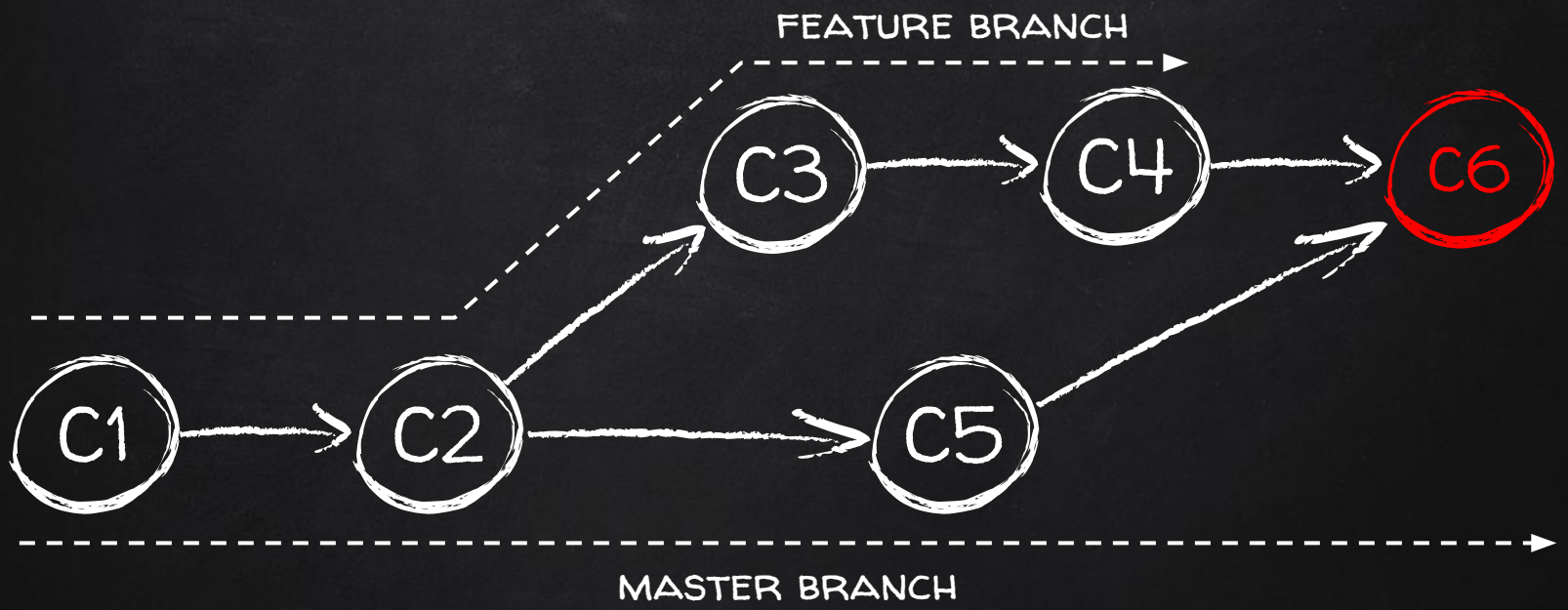
```
$ git push origin master
```

Mettre à jour sa branche

AVANT MERGE

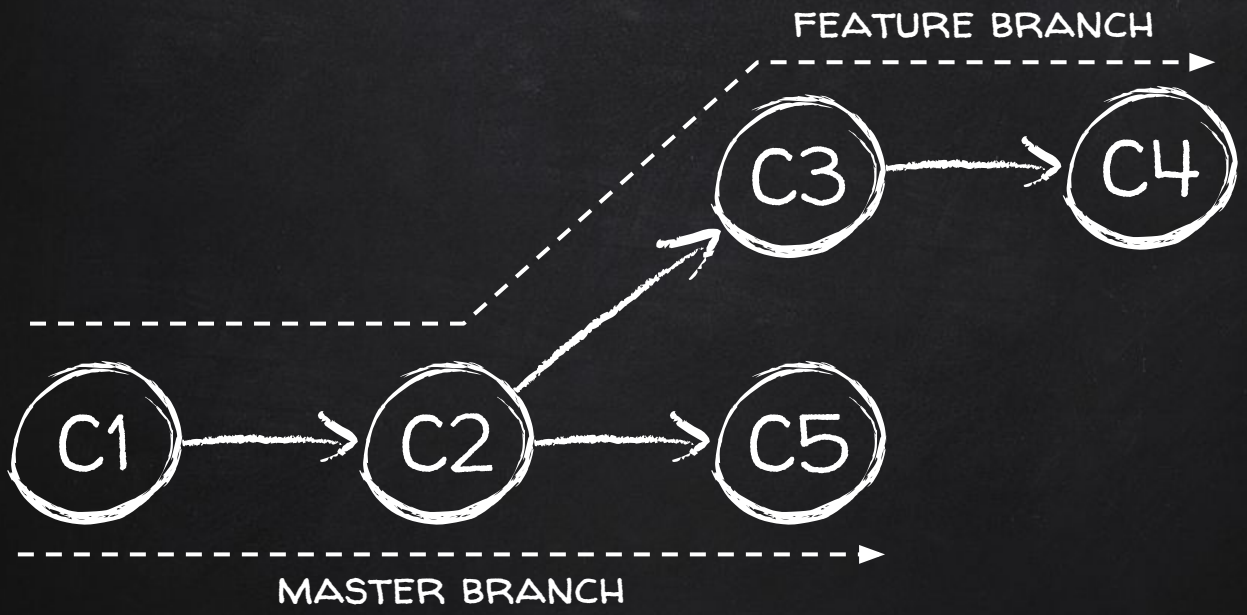


AVEC UN MERGE

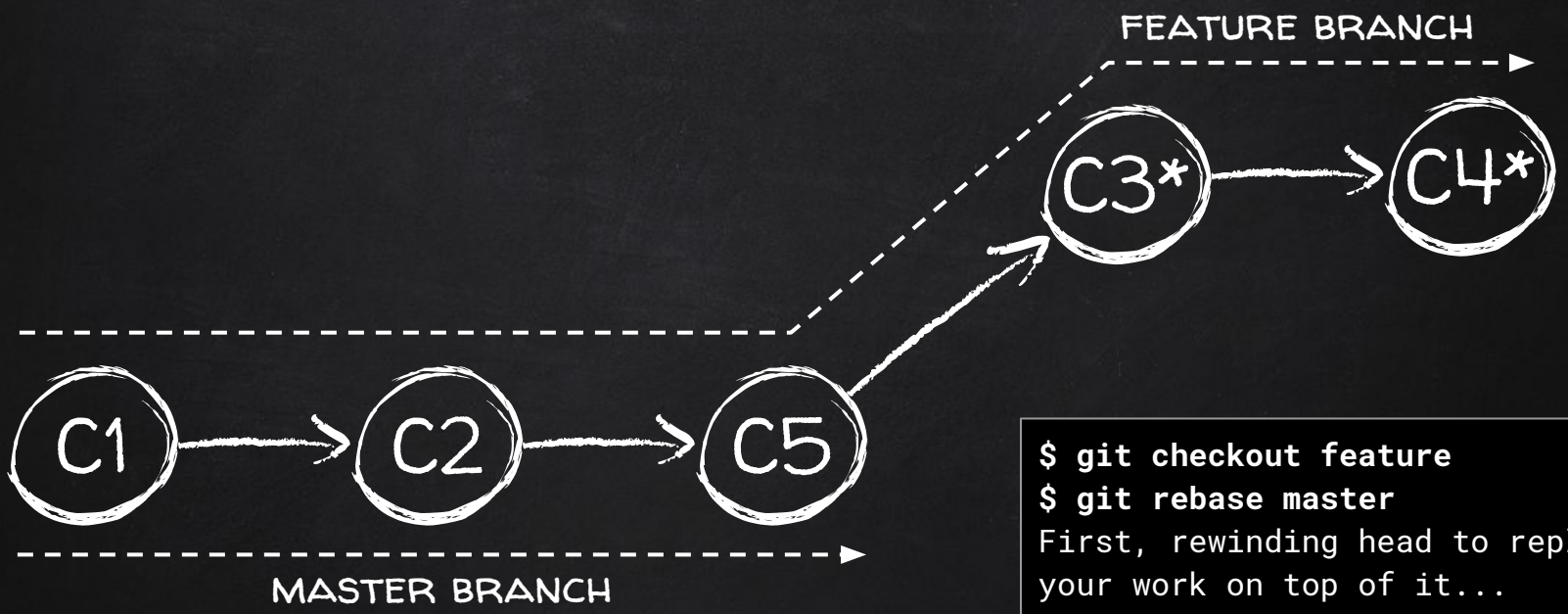


MERGE COMMIT

AVEC UN REBASE



AVEC UN REBASE




```
$ git checkout feature
$ git rebase master
First, rewinding head to replay
your work on top of it...
Applying: Message du commit C3
Applying: Message du commit C4
```

Quand rebaser ?

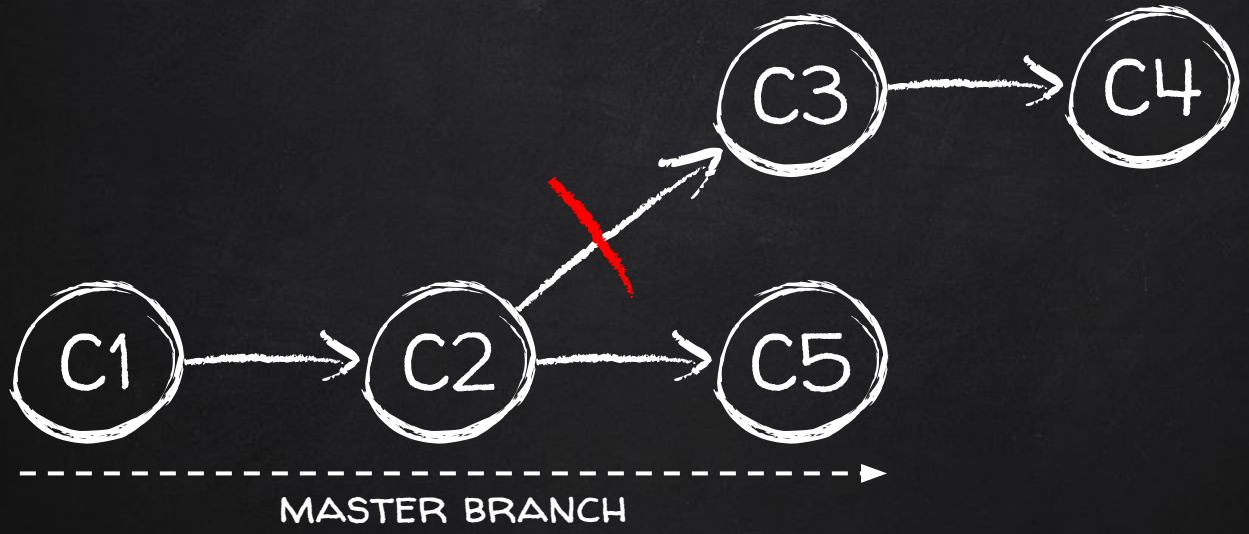
- pour actualiser une branche avec les avancées de sa branche source
- pour réécrire l'historique des commits sur une branche

Quand rebaser ?

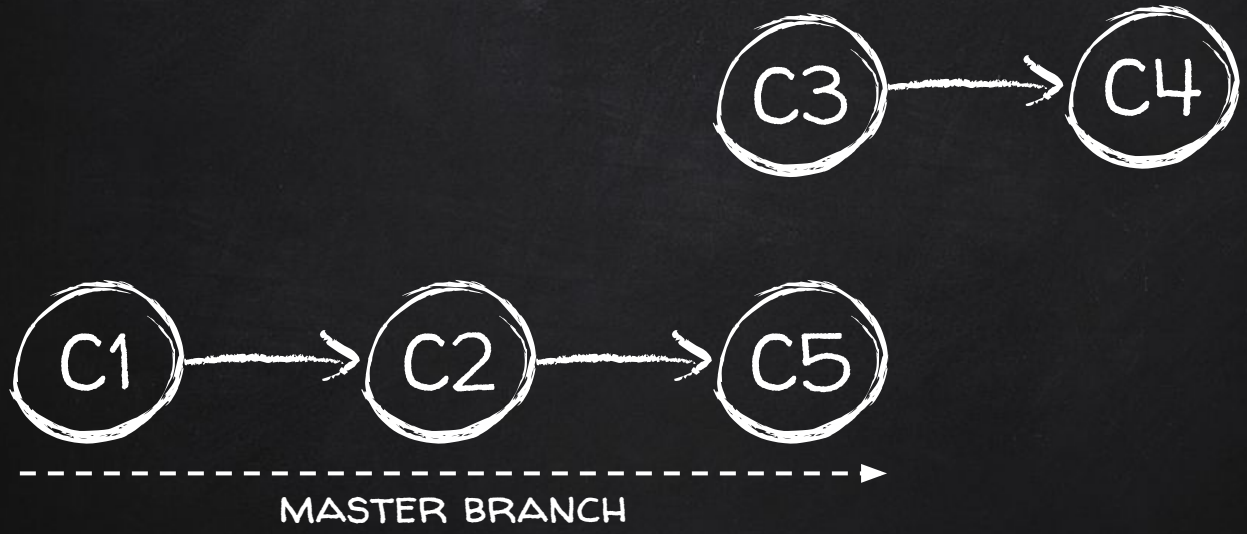
- pour actualiser une branche avec les avancées de sa branche source
- pour réécrire l'historique des commits sur une branche

Toujours sur une branche où l'on travaille seul

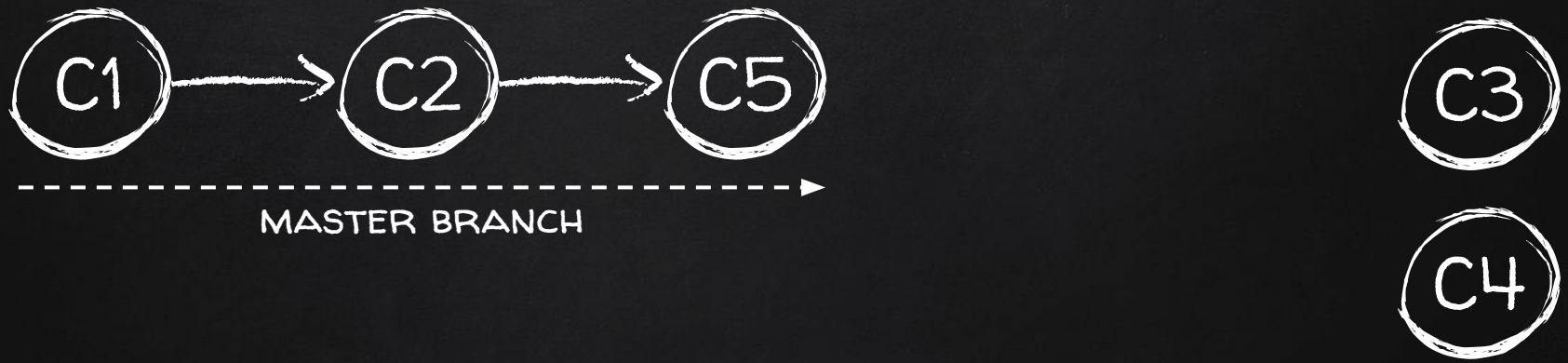
REBASE



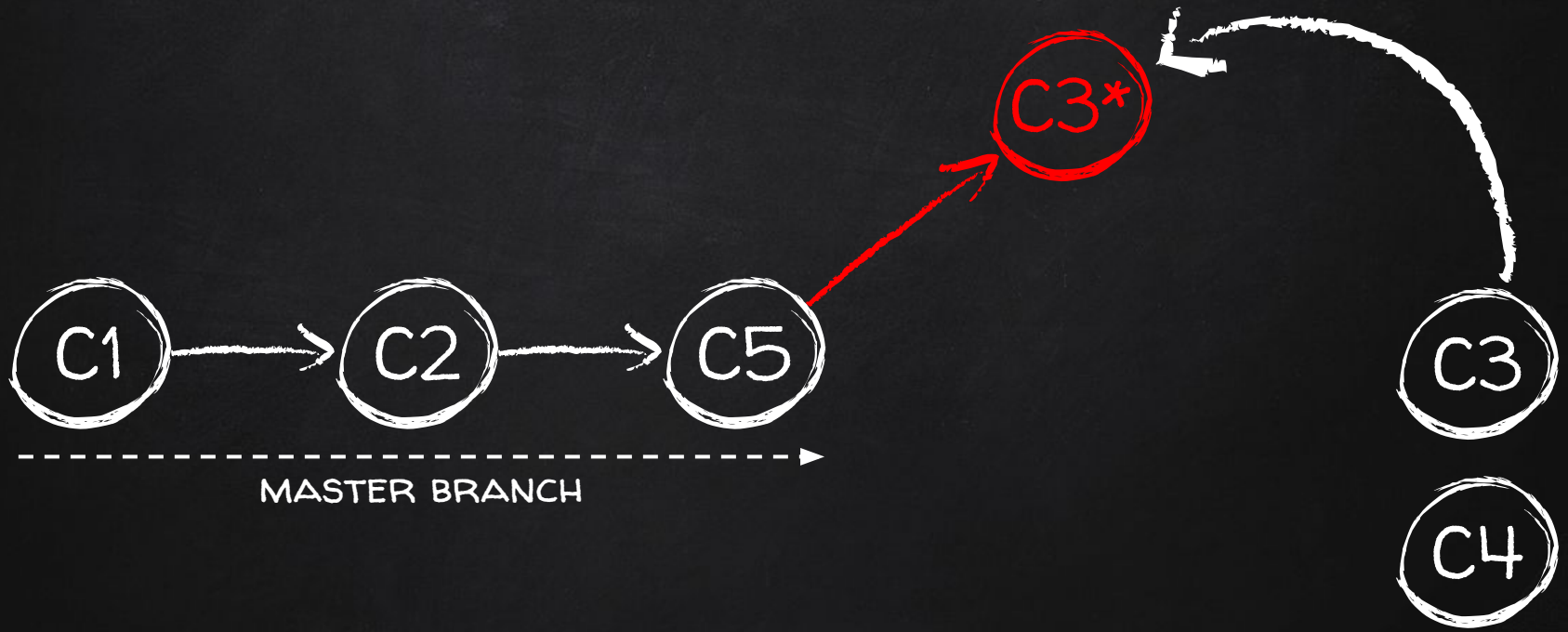
REBASE



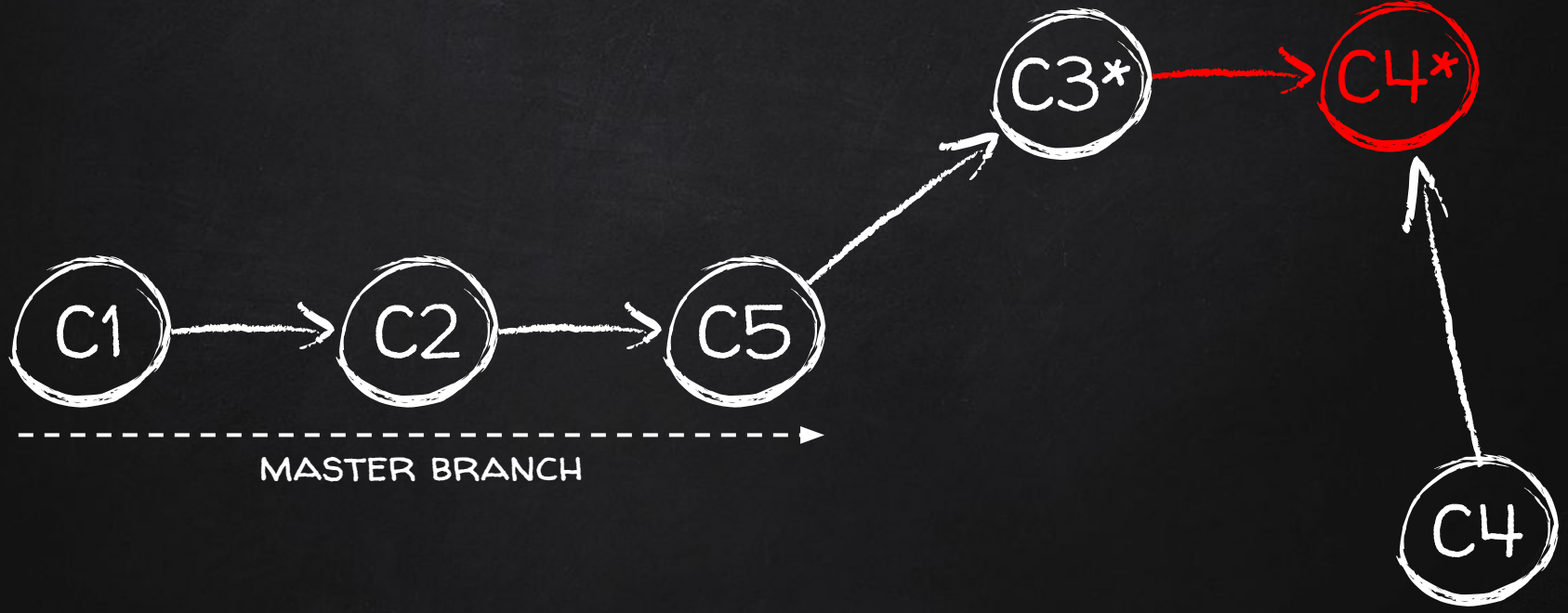
REBASE



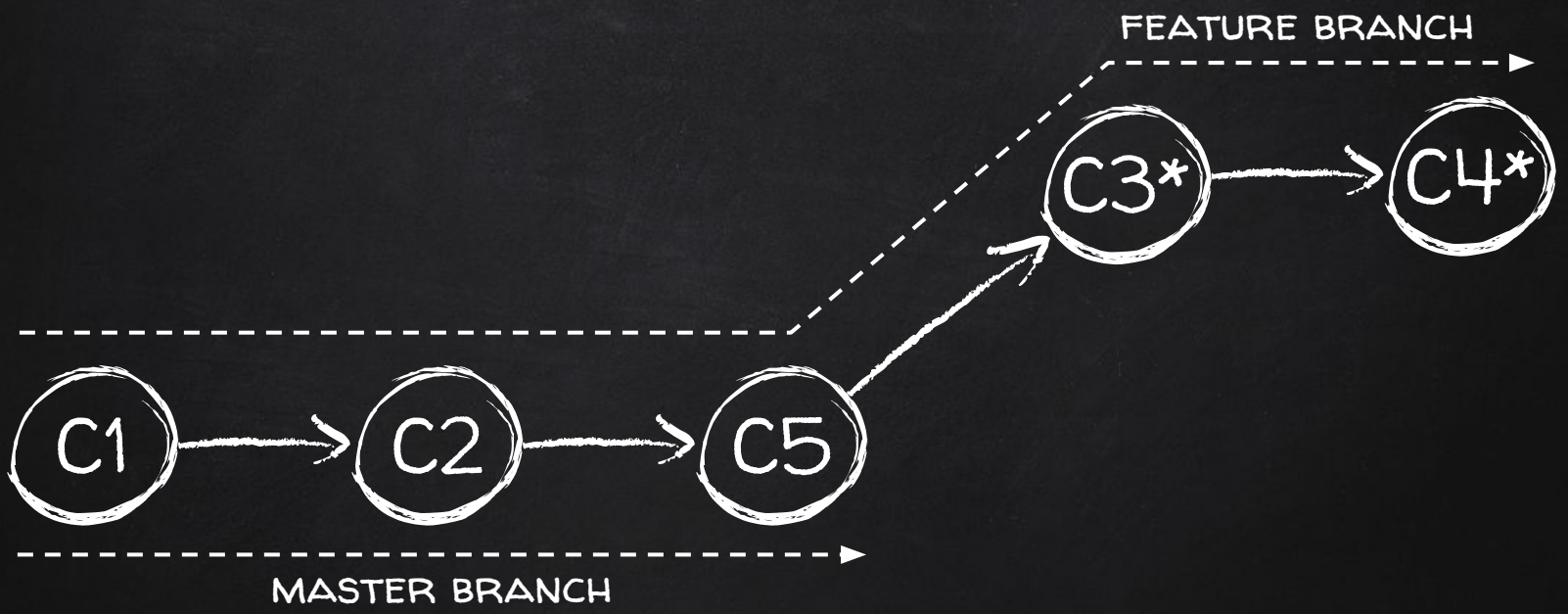
REBASE



REBASE

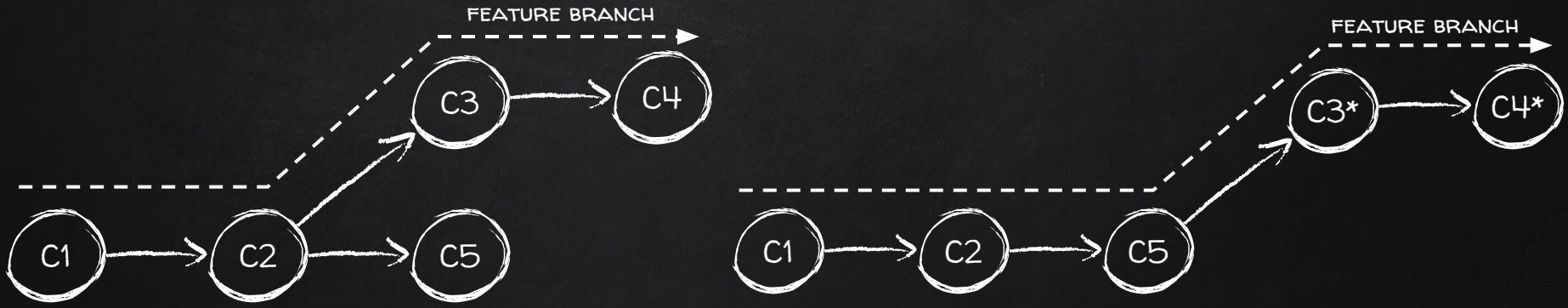


AFTER REBASE

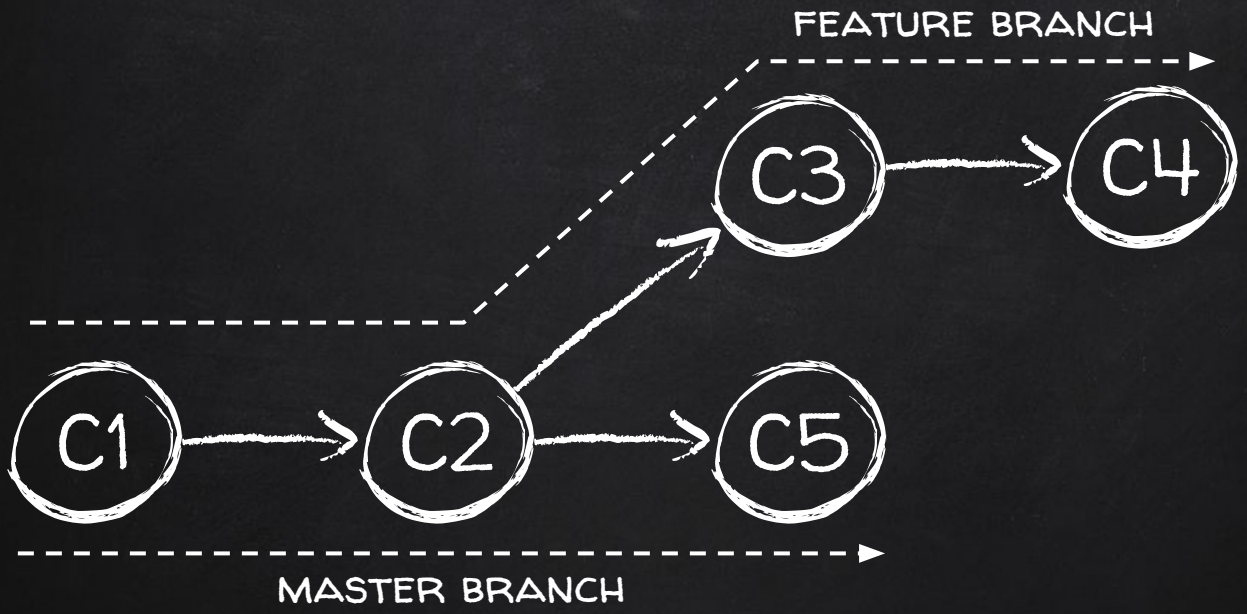


PUSHER APRÈS UNE REBASE

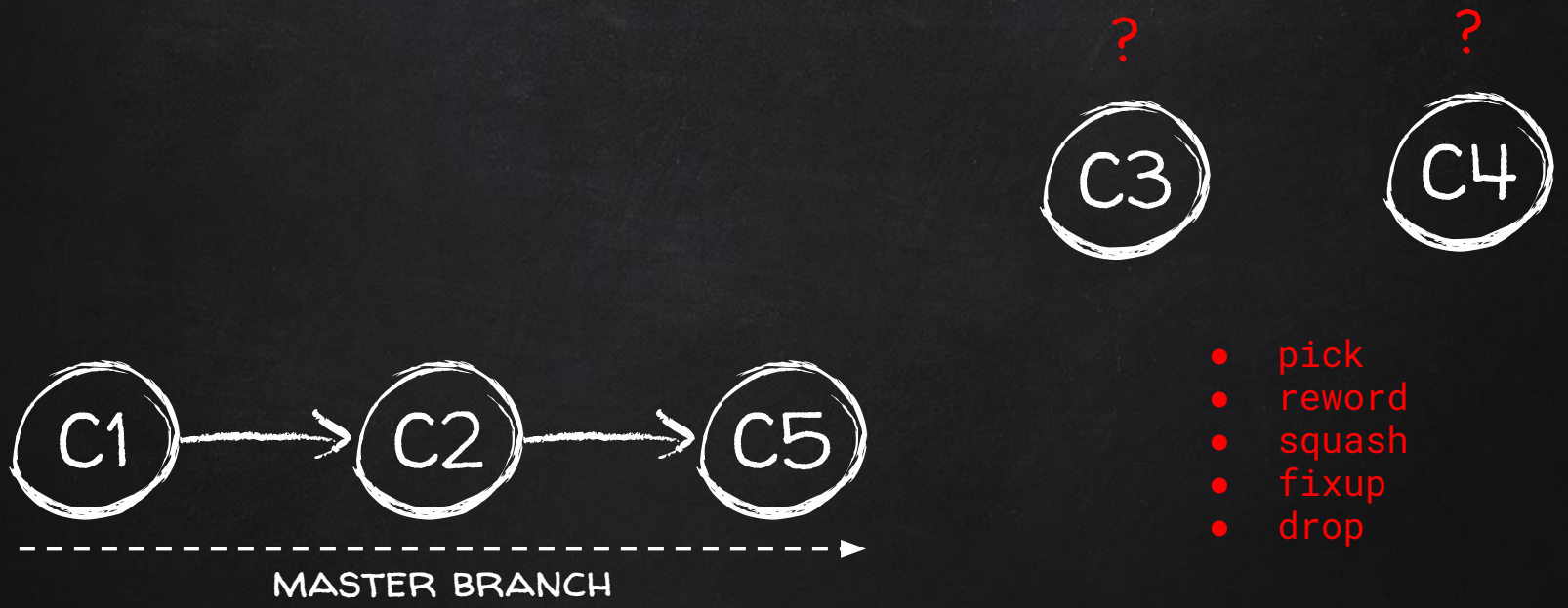
```
$ git push --force origin feature
```



REBASE INTERACTIF



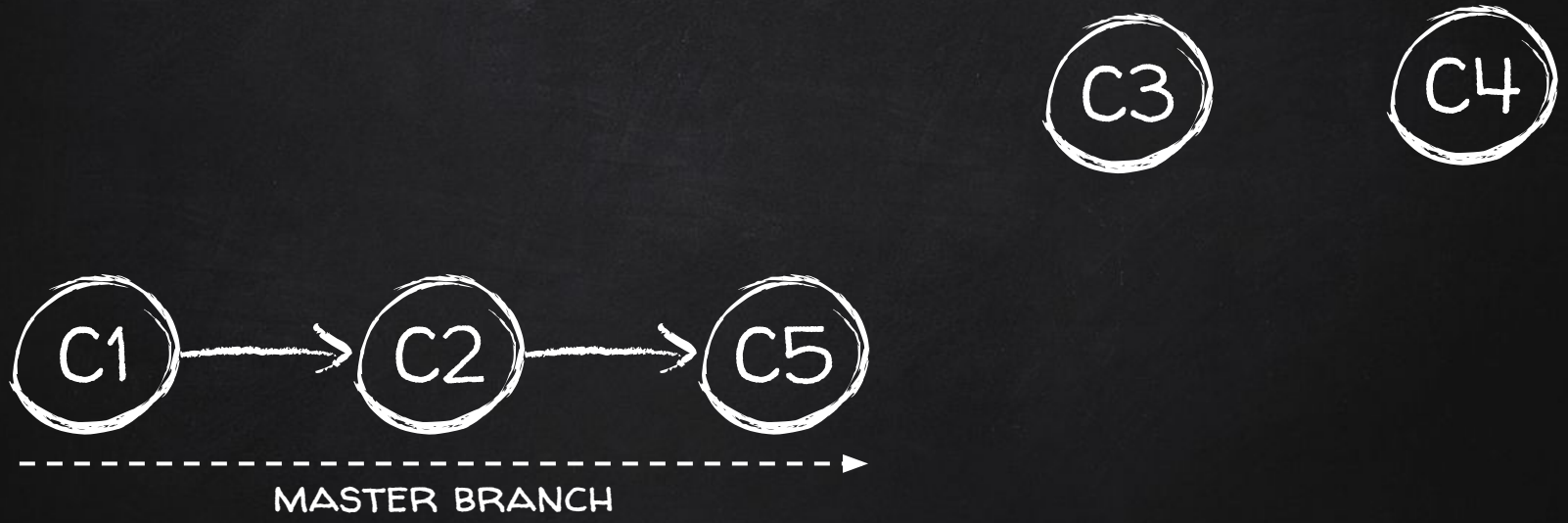
REBASE INTERACTIF



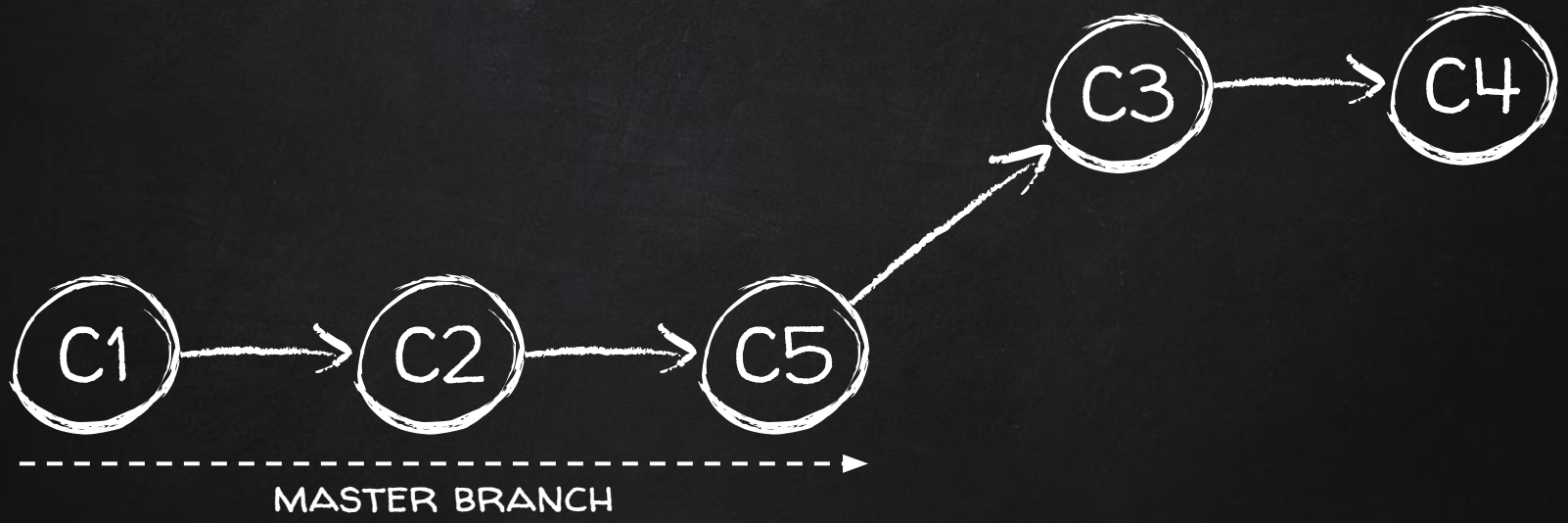
- pick
- reword
- squash
- fixup
- drop

REBASE INTERACTIF

pick pick



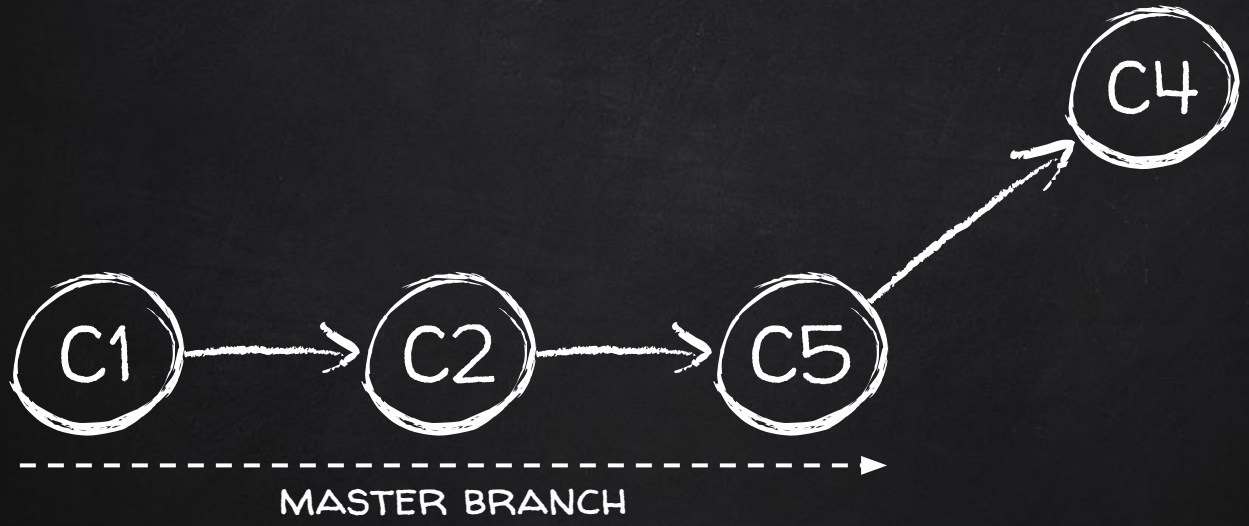
REBASE INTERACTIF



REBASE INTERACTIF

drop pick

REBASE INTERACTIF



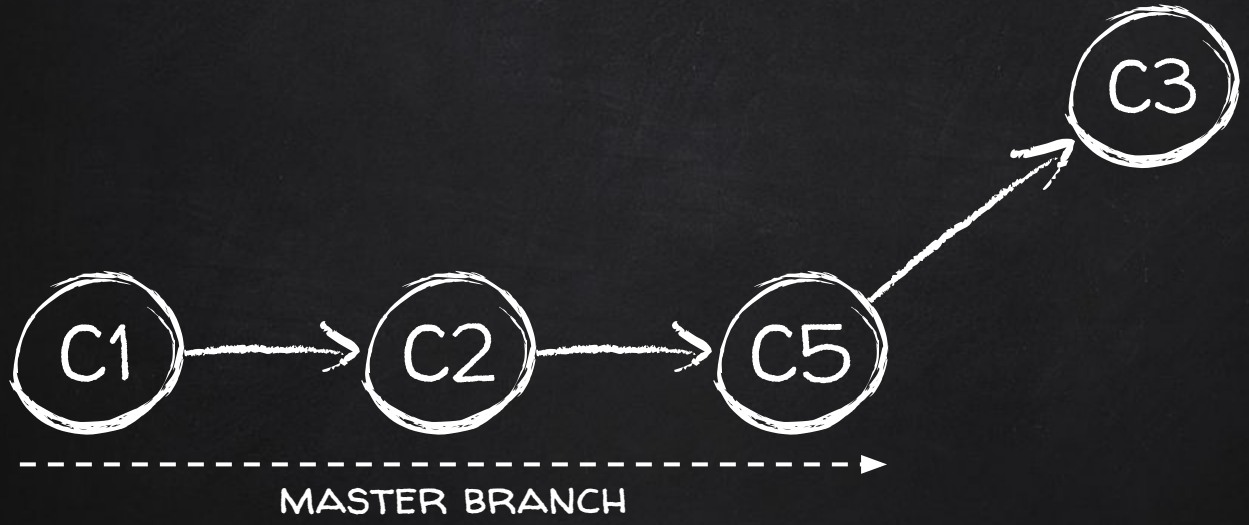
REBASE INTERACTIF

pick

fixup

REBASE INTERACTIF

contient les changements
introduits par C3 et C4

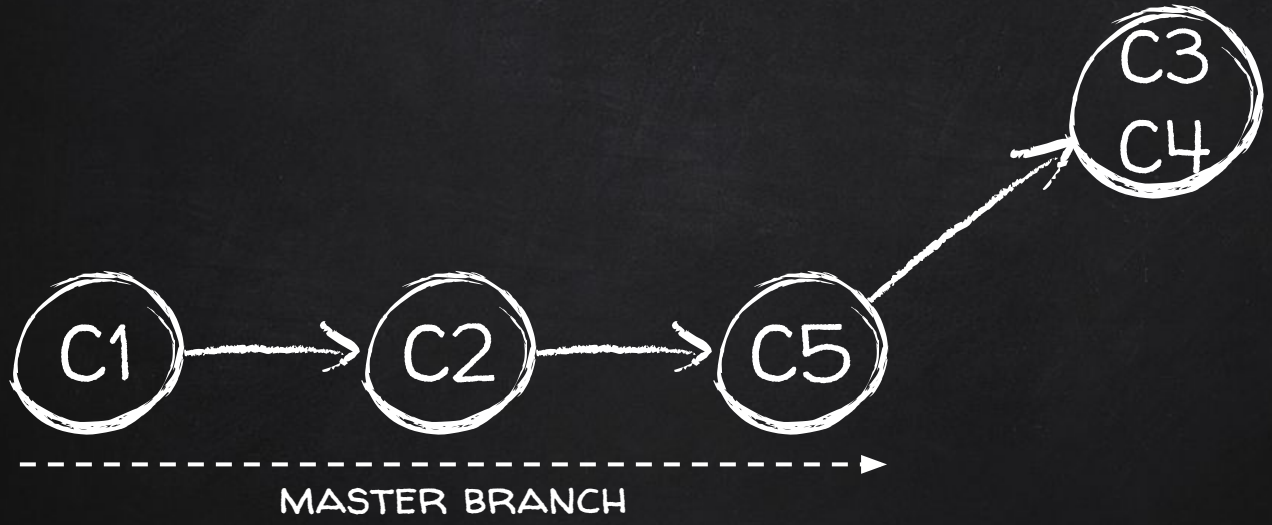


REBASE INTERACTIF

pick

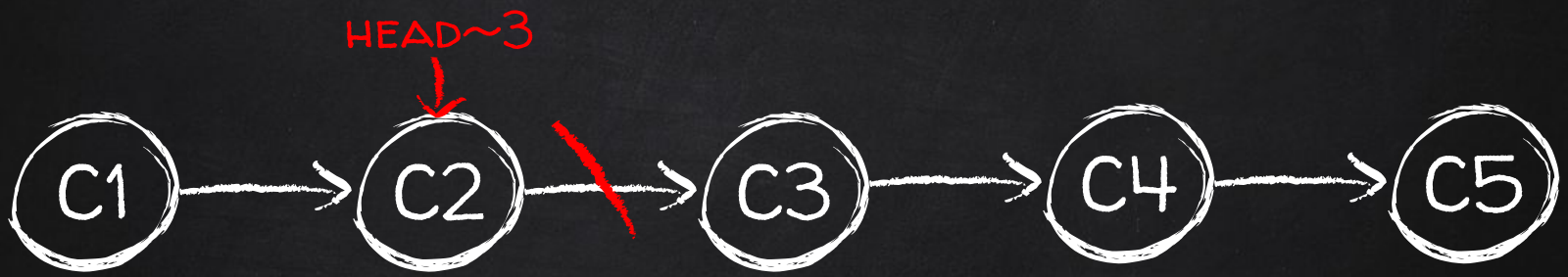
squash

REBASE INTERACTIF



REBASE INTERACTIF

```
$ git rebase -i HEAD~3
```



REBASE INTERACTIF

```
$ git rebase -i HEAD~3
```

REBASE INTERACTIF

```
$ git rebase -i HEAD~3
```

contient les
changements introduits par
C3, C4 et C5

$ git revert

```
$ git revert C2
$ git push origin <branch>
```

$ git stash

```
$ git status -s
 M Dockerfile
 M README.md

$ git stash
Saved working directory and index state WIP on feature-b: e389ac4
Modification du README.md

$ git status -s
//rien
$ git stash show
 Dockerfile | 1 +
 README.md  | 2 ++
 2 files changed, 3 insertions(+)
```

$ git stash

```
$ git stash list
stash@{0}: WIP on feature-b: e389ac4 Modification du README.md
stash@{1}: WIP on master: a092f1c Modification Dockerfile

$ git stash
Saved working directory and index state WIP on feature-b: e389ac4
Modification du README.md

$ git stash drop //supprime le dernier stashé
$ git stash clear //supprime tous les niveaux de stash

$ git stash pop //réintègre les modifications stashées à la copie locale
$ git stash apply stash@{1}
```

$ git reflog

```
$ git rev-parse master
3ab24295dc6fdfe3c0339155cbb85051d1fda741
```

```
$ git reflog
71f9f4f1 (HEAD -> master) HEAD@{0}: checkout: moving from feature-a to master
0ed8e3c6 (fix-export-pdf) HEAD@{1}: checkout: moving from master to feature-a
71f9f4f1 (HEAD -> master) HEAD@{2}: checkout: moving from dev to master
3ab24295 (dev) HEAD@{3}: rebase -i (finish): returning to refs/heads/dev
3ab24295 (dev) HEAD@{4}: rebase -i (pick): Ajout route /user/create
0c0ba220 HEAD@{5}: rebase -i (start): checkout HEAD~1
```

$ git bisect

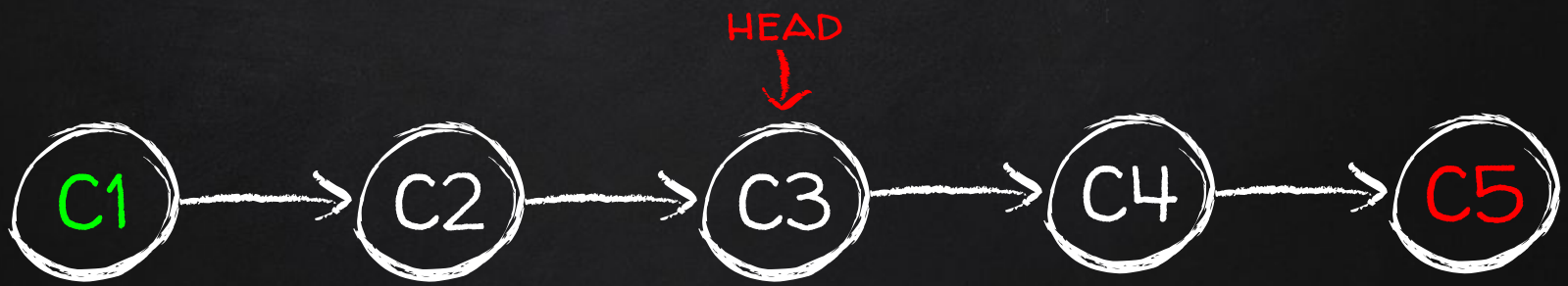
```
$ git bisect start
```

$ git bisect

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good C1             # C1 is known to be good
```

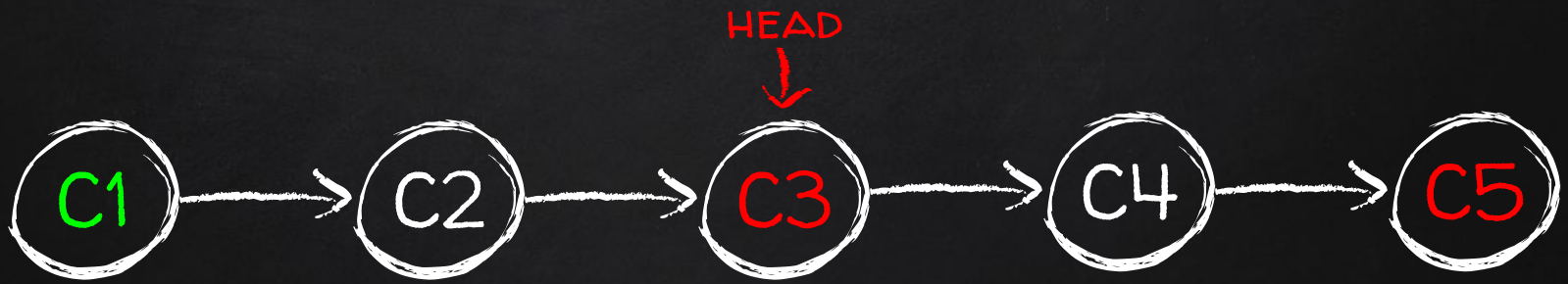
$ git bisect

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good C1             # C1 is known to be good
```



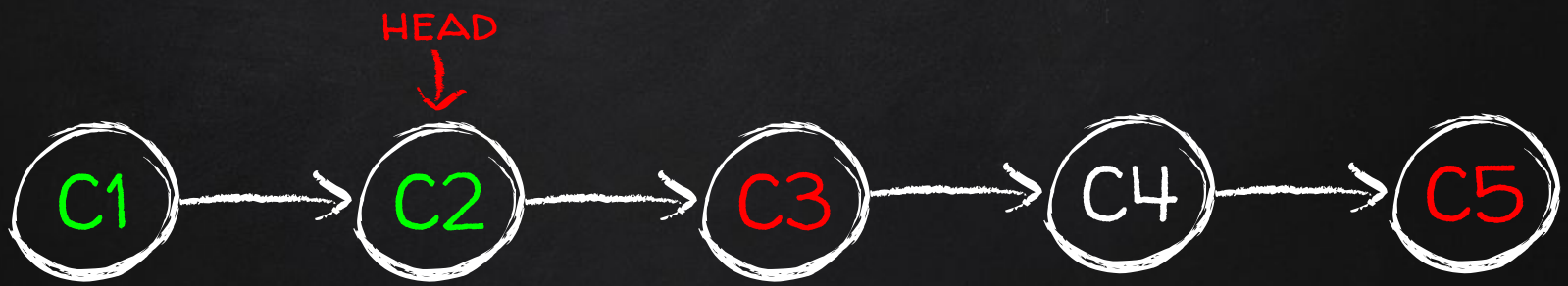
$ git bisect

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good C1             # C1 is known to be good
$ git bisect bad
```



$ git bisect

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good C1             # C1 is known to be good
$ git bisect bad
$ git bisect good
```



$ git bisect

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good C1             # C1 is known to be good
$ git bisect bad
$ git bisect good
```

GIT HOOKS

```
$ ls -l .git/hooks
applypatch-msg.sample
commit-msg.sample
post-update.sample
pre-applypatch.sample
pre-commit.sample
pre-push.sample
pre-rebase.sample
prepare-commit-msg.sample
update.sample
```

$ git help [command]

```
$ git help
(liste des commandes)

$ git help push
(man page de git push)
```

Merci!

Questions?

@meucaa
lucas.vuillier@gmail.com