# Présentation Git

## Un outil de collaboration puissant

---

Denis Pettens  Pablo Gonzalez Alvarez  Gaëtan Cassiers

26 février 2017

Louvain-li-Nux

# Table des matières

# Introduction

# Gérer un projet

Comment gérez-vous actuellement un projet ?

- L'envoyer à travers un message sur Facebook, ... (**Très mauvaise idée**)
- L'envoyer par mail (**Un peu moins**)
- Utiliser une Dropbox, Google Drive, ... (**Déjà mieux mais toujours risqué ou manque de fonctionalités**)

Solution : Utiliser un **système de gestion de version décentralisé** (Distributed Version Control System (DVCS) pour les anglophiles).

# Un DVCS ?

- **Version** Enregistre des « instantanés » du projet.
- **Gestion** Revenir en arrière, voir des différences, fusionner des modifications.
- **Décentralisé** Chacun travaille sur sa copie, et on fusionne les modifications.
- **Projet** n'importe quel répertoire (« dossier »). Donc n'importe quoi !

# Et Git dans tout ça ?

- `Git` a été créé en 2005 par **Linus Torvalds** (auteur de `Linux`) ;

Ses avantages :

- Le plus connu et utilisé (90 % du marché, communauté très présente) ;
- Vitesse ;
- Facile d'utilisation mais aussi très puissant ;
- Distribué (pas besoin de connexion internet tout le temps) ;

Ses inconvénients :

- De nouveaux concepts
- Interface principale en ligne de commande
- Mais il existe aussi des interfaces graphiques

# Instalation et configuration

# Installer Git

## Git

**Ubuntu** : `sudo apt-get install git`

**OS X** : `https://sourceforge.net/projects/git-osx-installer/`

**Windows** : `https://git-for-windows.github.io/`

# Configuration de base

Git a besoin de deux informations de base sur vous pour pouvoir travailler efficacement :

- **Nom et Prénom**

```
git config --global user.name "Jules Dupont"
```

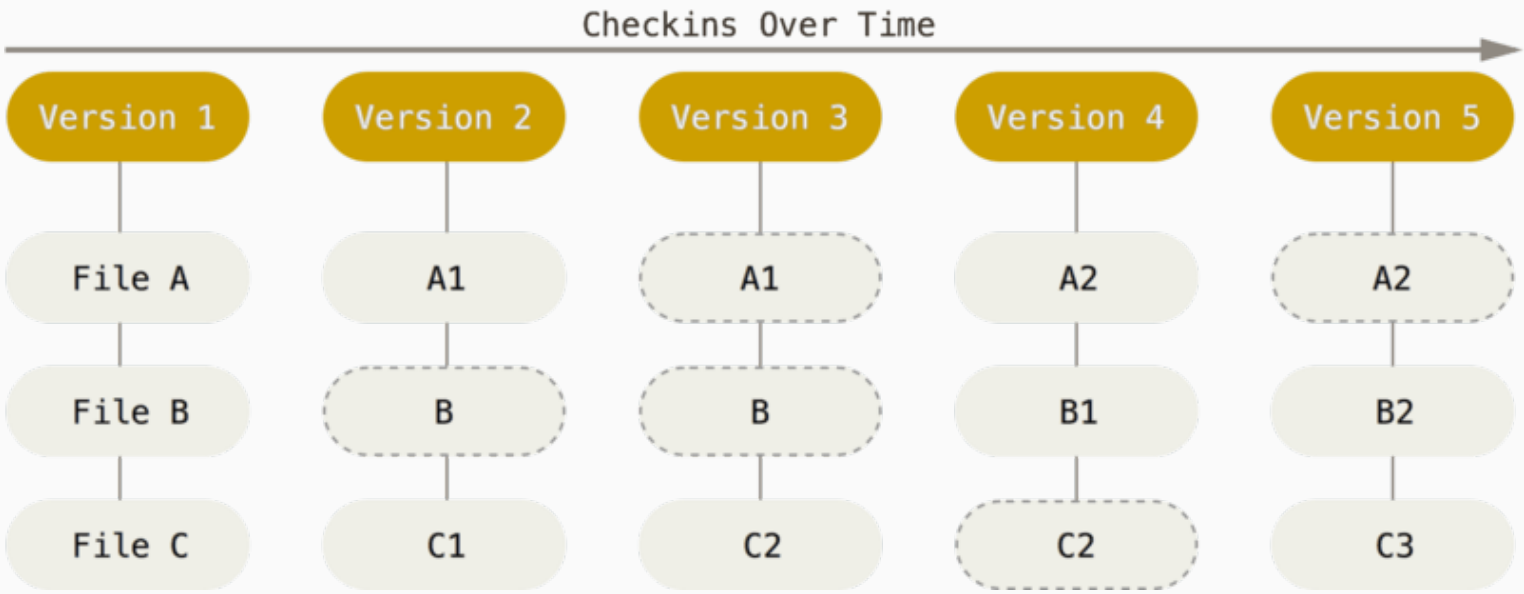
- **Email**

```
git config --global user.email "jules.
    dupont@email.fr"
```

L'option `--global` permet de configurer `git` pour tous vos autres projets sur votre PC.

# Premier pas avec Git

# Concept : le commit




Les illustrations non-sourcées viennent de `https://git-scm.com/book`.

# Commande : git init

- Initialise un dossier en un nouveau dépot `git`.
- Exemple

```
$ mkdir newProject
$ cd newProject
$ git init
```

- Cela crée un sous-dossier `.git` où tout la magie de git se fait
- Vous mettrez tous les fichiers du projet dans `newProject`

# Commande : git status

- `git` vous dit où vous en êtes.
- Exemple

```
$ git status
On branch master
Initial commit
nothing to commit (create/copy files and use
    "git add" to track)
```

- À utiliser sans modération !

# Commande : git add

- Ajoute un fichier dans le projet git.
- Exemple

```
$ vi notes.txt # copier un fichier
$ git status
#[...]
Untracked files:
  (use "git add <file>..." to include in what
       will be committed)
  notes.txt
nothing added to commit but untracked files
   present (use "git add" to track)
$ git add notes.txt
#[...]
Changes to be committed:
  (use "git rm --cached <file>.." to unstage)
  new file:   notes.txt
```

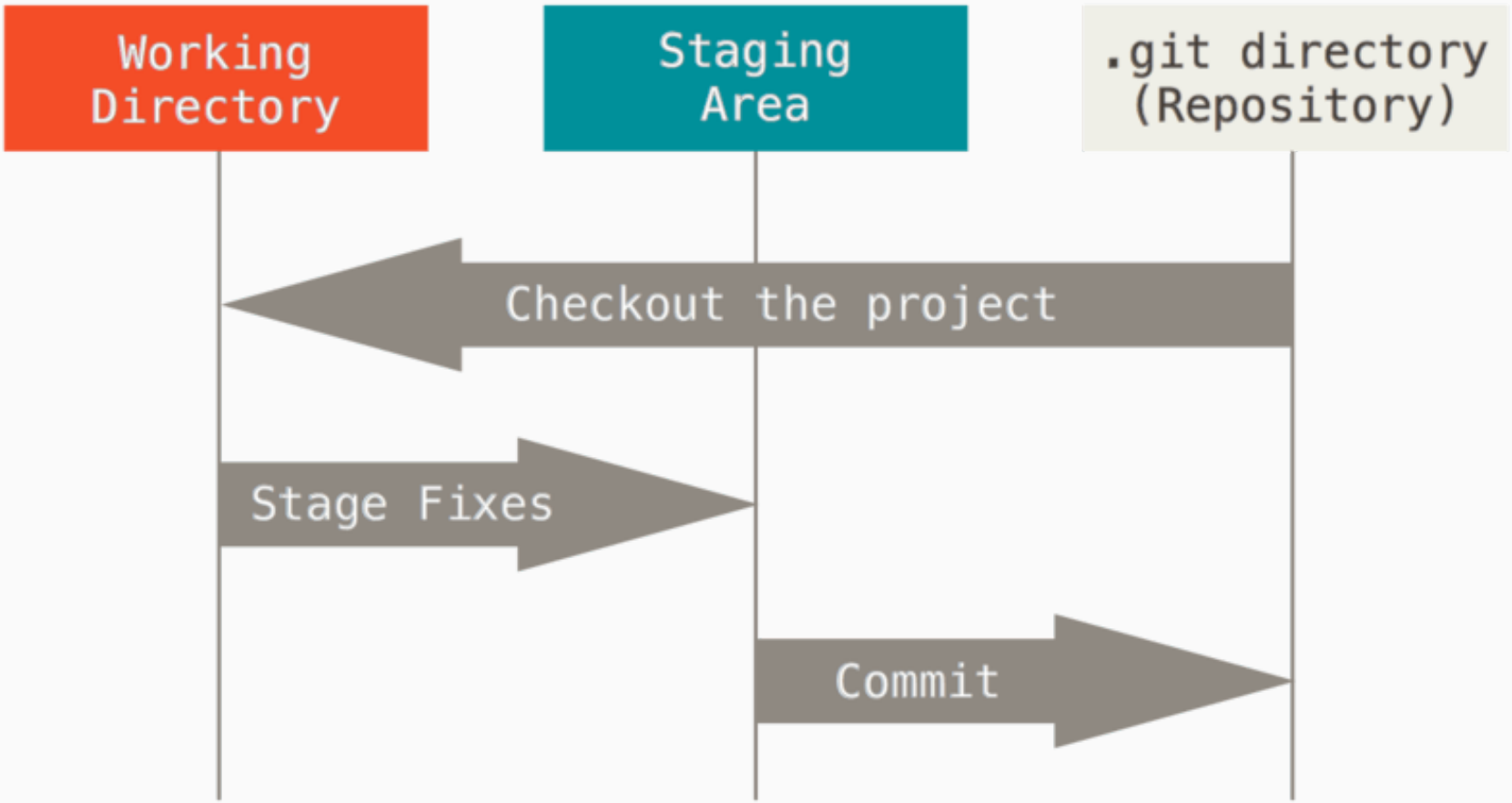
# Commande : git commit

- Crée un commit sur base des fichiers ajoutés.
- Exemple

```
# toujours verifier ce qu'on commit
$ git status
#[...]
Changes to be committed:
  (use "git rm --cached <file>.." to unstage)
  new file:   notes.txt
$ git commit
# Ouvre un editeur de texte
# Editer, sauvegarder et fermer
[master (root-commit) 12f87b9] ajout de le
   premiere note
 1 file changed, 1 insertion(+)
```

- Message de commit : décrit les changements effectués.

# En résumé : le cycle de vie d'un fichier



# Commande : git log

- Visualiser l'historique du projet
- Exemple

```
$ git log
commit 12f87b95caff8cbeb5ce0717528d77e27
Author: Louvain Linux<info@louvainlinux.org>
Date:   Sun Feb 26 17:51:16 2017 +0100

    ajout de le premiere note
```

- Ouvre parfois un pager. Se déplacer avec les flèches haut/bas, quitter avec q.

# Astuce : de l'aide !

On peut trouver de l'aide :

- rapide : `git [command] -h`

```
$ git log -h
usage: git log [<options>] [<revision-range>]
    [[--] <path>...]
   or: git show [<options>] <object>...

    -q, --quiet           suppress diff
        output
    --source              show source
[...]
```

- plus détaillée : `git [command] --help`

# Exercice 1

```
$ mkdir newProject
$ cd newProject
$ git status
$ # Creer un fichier
$ git add monfichier.txt monfichier2.png
$ git commit
$ # Editer le message de commit
$ git log
```

Utile :

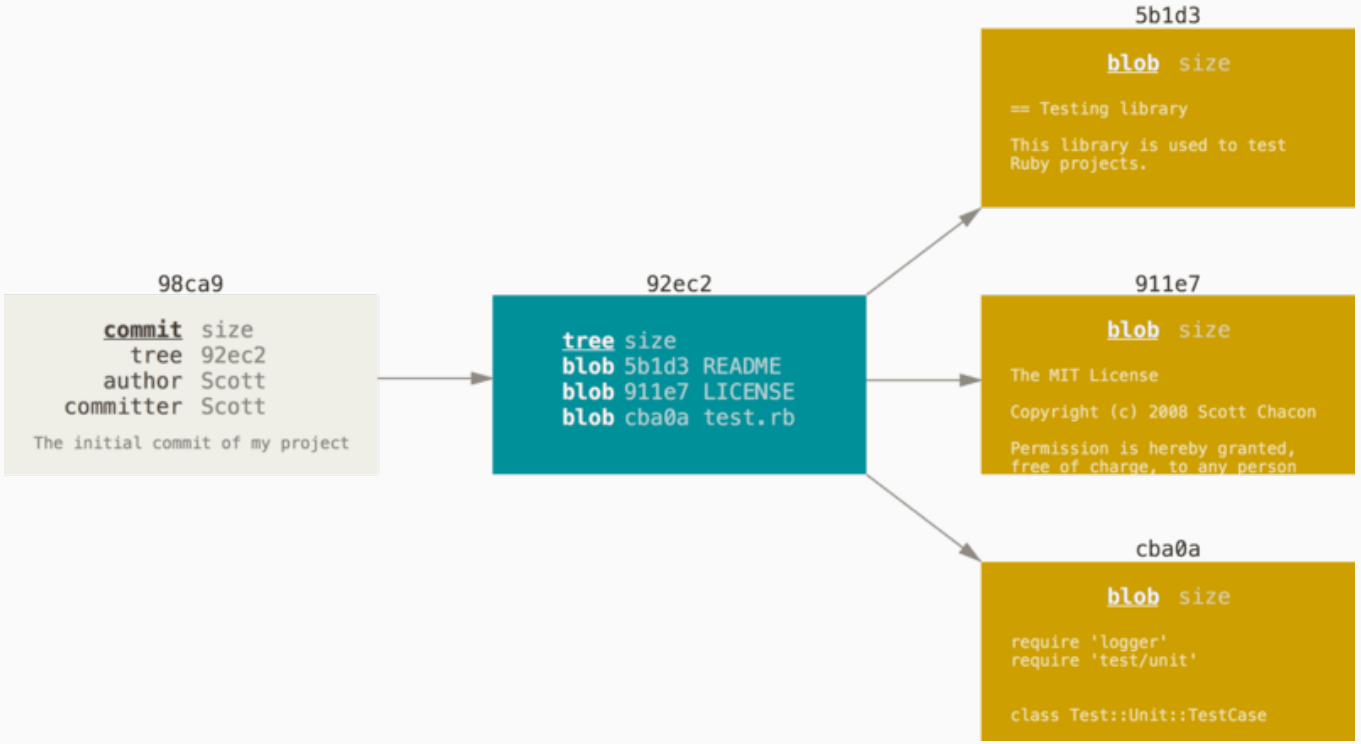
```
$ git --help # liste des commandes git
$ git [commande] --help
```

Bonus : regardez l'aide de `git mv` et de `git rm`.

# Les branches

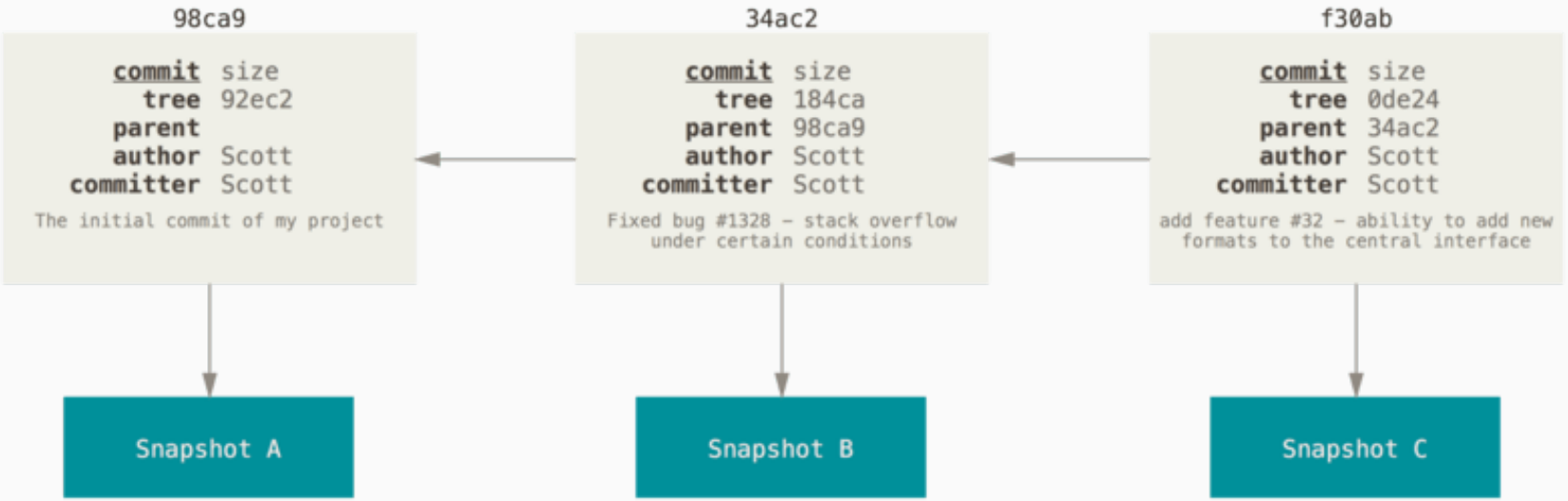
# De derrière : les objets git

- Chaque commit a un identifiant : **12f87**b95caff8cbeb5ce0717528d77e27db5669c.



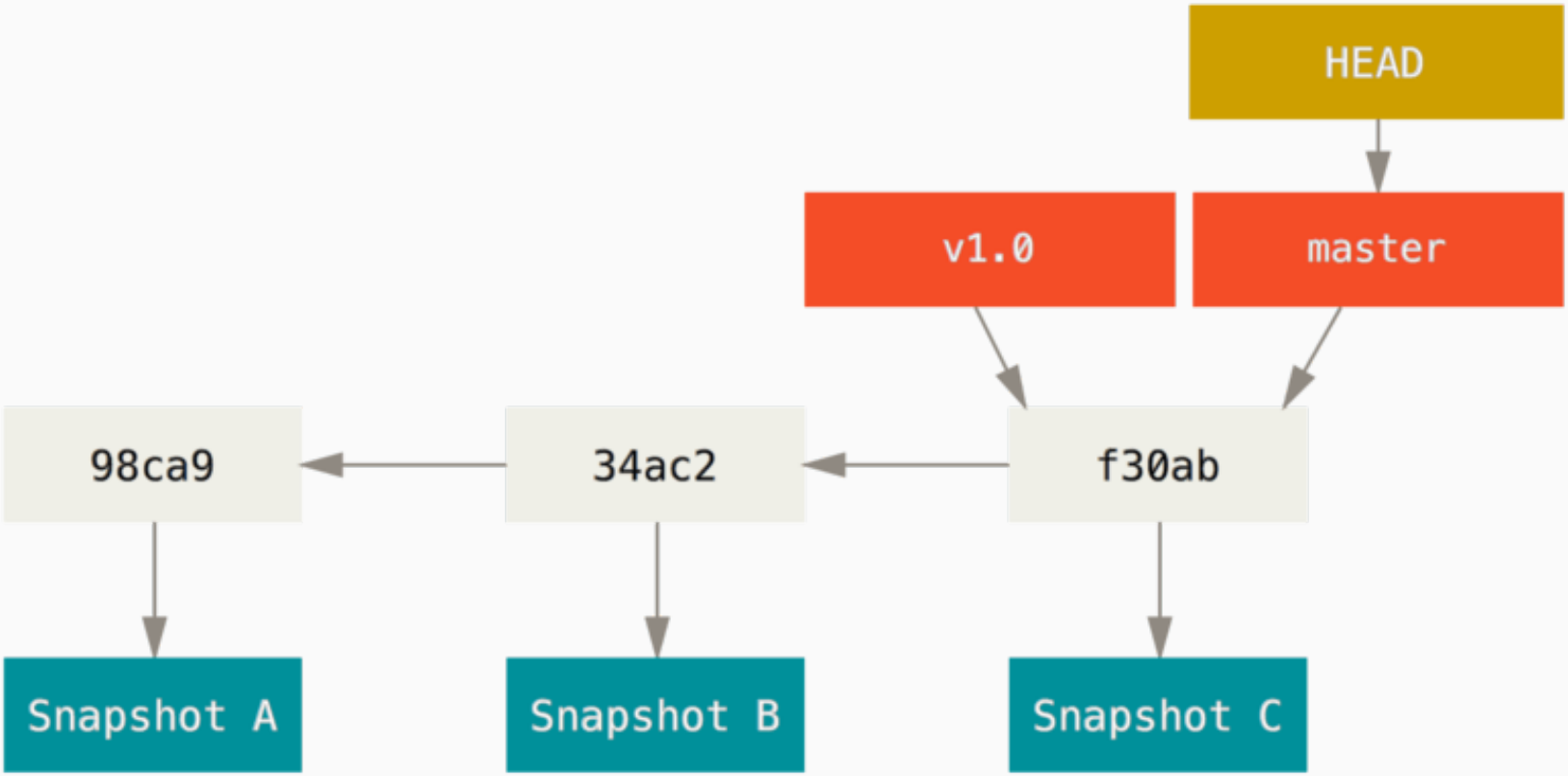
# De derrière : les parents

- Chaque commit a un parent.



# De derrière : les étiquettes

- On peut mettre des étiquettes sur des commits.
- `HEAD` est la position actuelle.

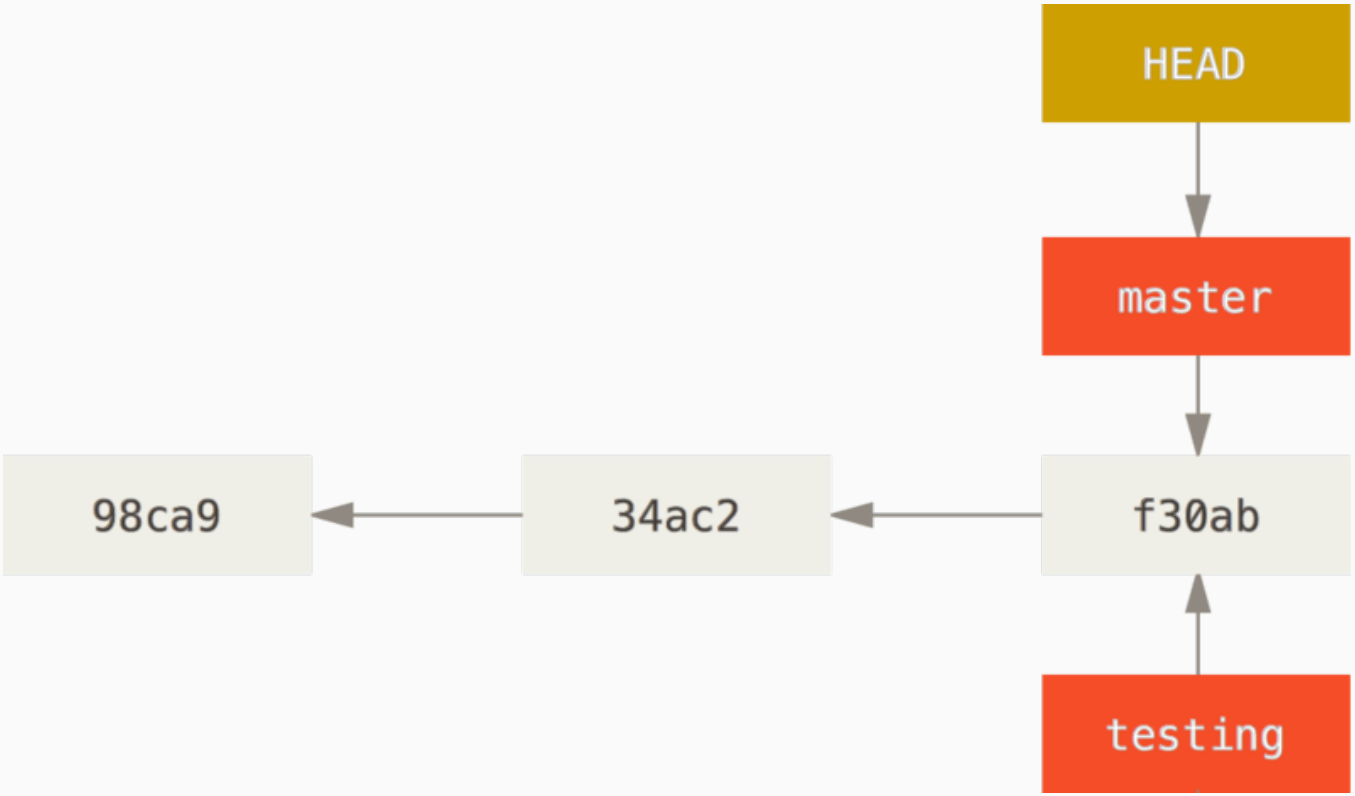


# Commande : git branch

- Une branche est une nouvelle étiquette.

```
$ git branch testing
```

- La branche par défaut est `master`.

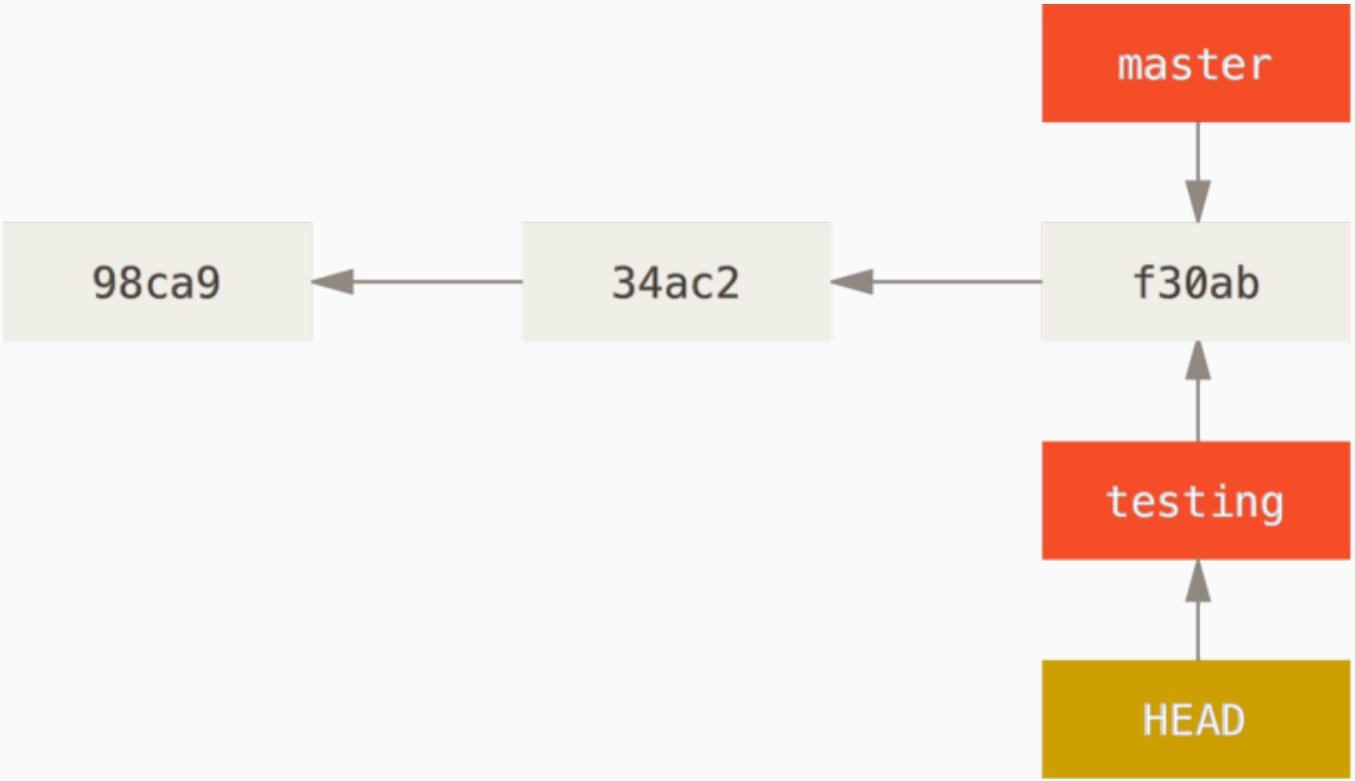


# Commande : git checkout

- Permet de changer de branche.

```
$ git checkout testing
```

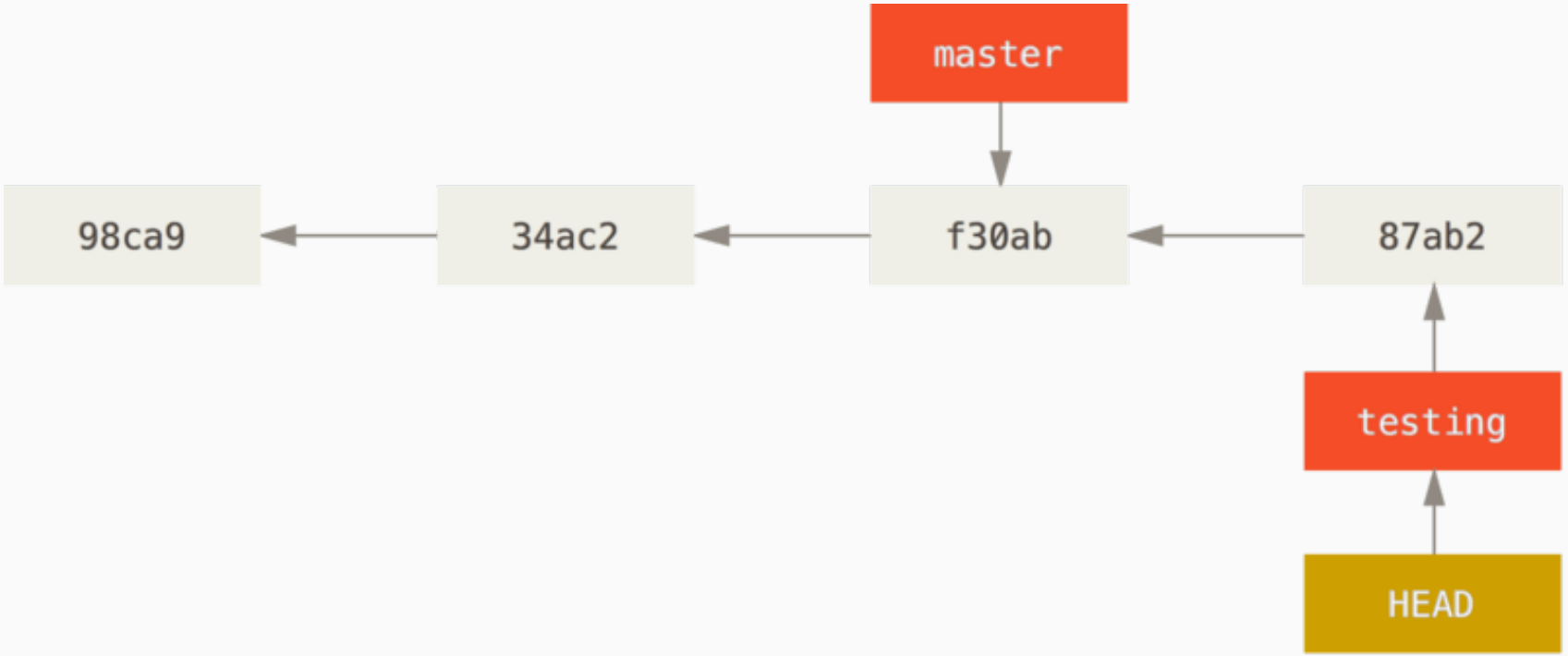
- La branche courante est celle qui suit les nouveaux commits.



# Commande : git checkout (2)

- La branche courante est celle qui suit les nouveaux commits.

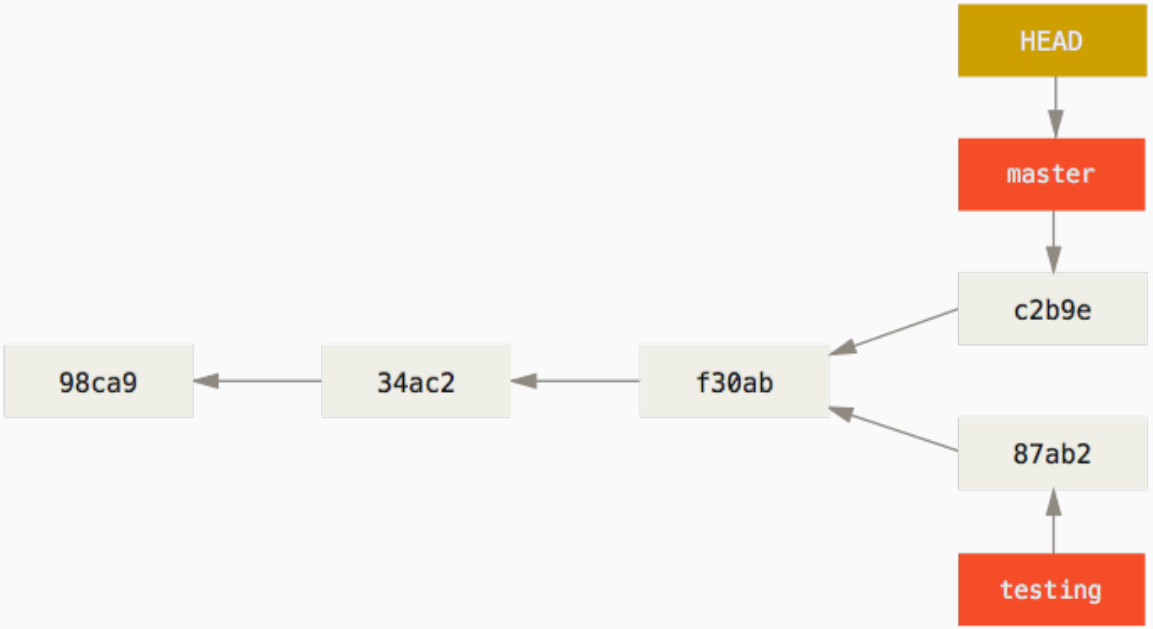
```
$ [Quelques changements]
$ git commit
```



# Branches divergentes

- Utilité : travailler sur des modifications indépendantes.

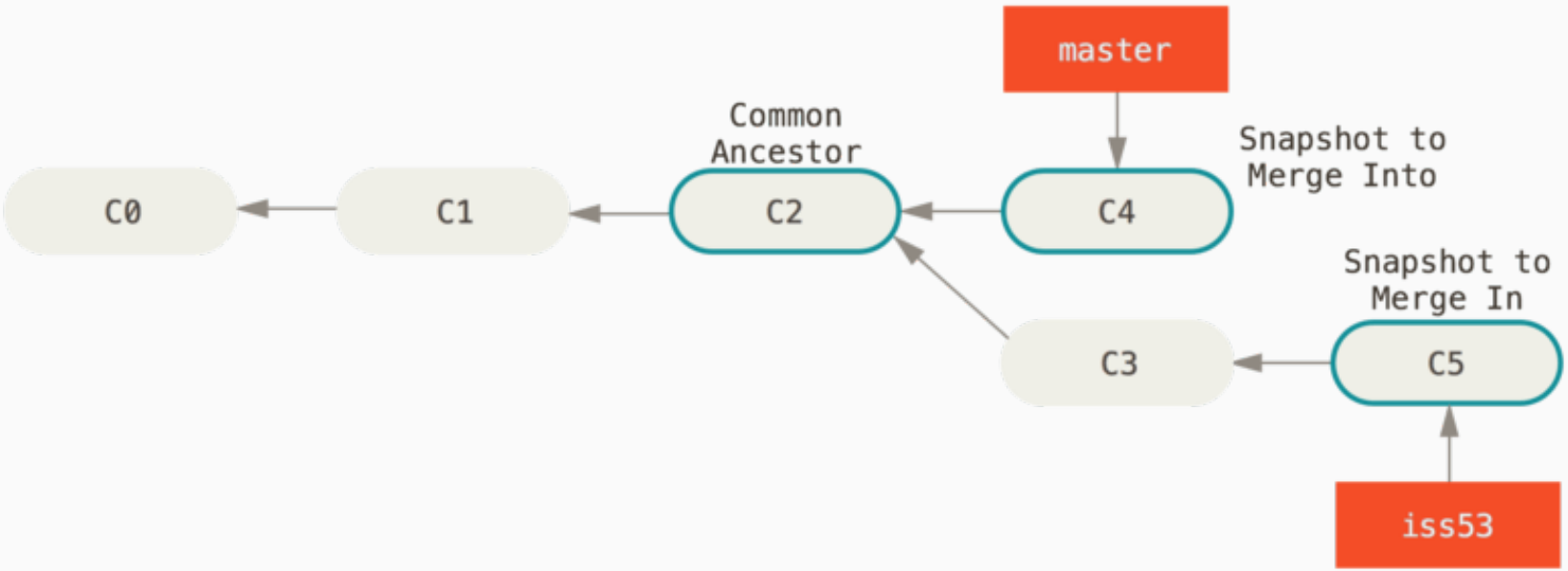
```
$ git checkout master
$ [Quelques changements]
$ git commit
```



```
$ git log --oneline --decorate --graph --all
* c2b9e (HEAD, master) made other changes
| * 87ab2 (testing) made a change
|/
* f30ab add feature #32 - ability to add new
   formats to the
* 34ac2 fixed bug #1328 - stack overflow under
   certain conditions
* 98ca9 initial commit of my project
```

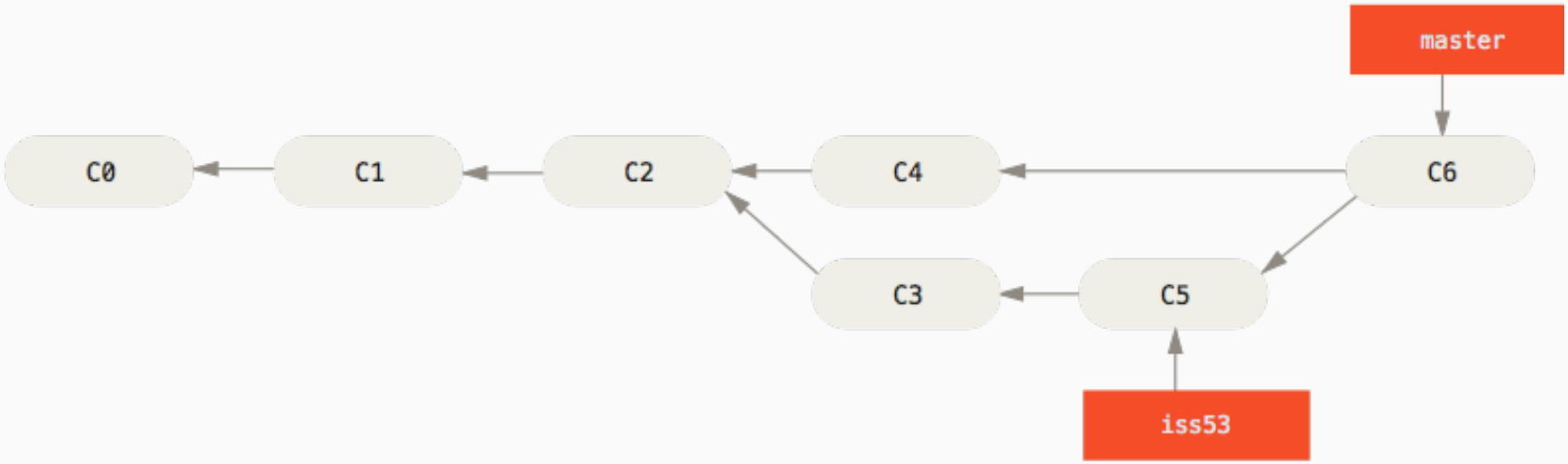
# Commande : git merge : fusionner des modifications

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```



# Commande : git merge (2)

```
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```



# Conflits

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then
    commit the result.
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:      index.html
no changes added to commit (use "git add" and/or
    "git commit -a")
```

# Conflits : résolution

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.
    com</div>
=======
<div id="footer">
 please contact us at support@github.com
>>>>>>> iss53:index.html
```

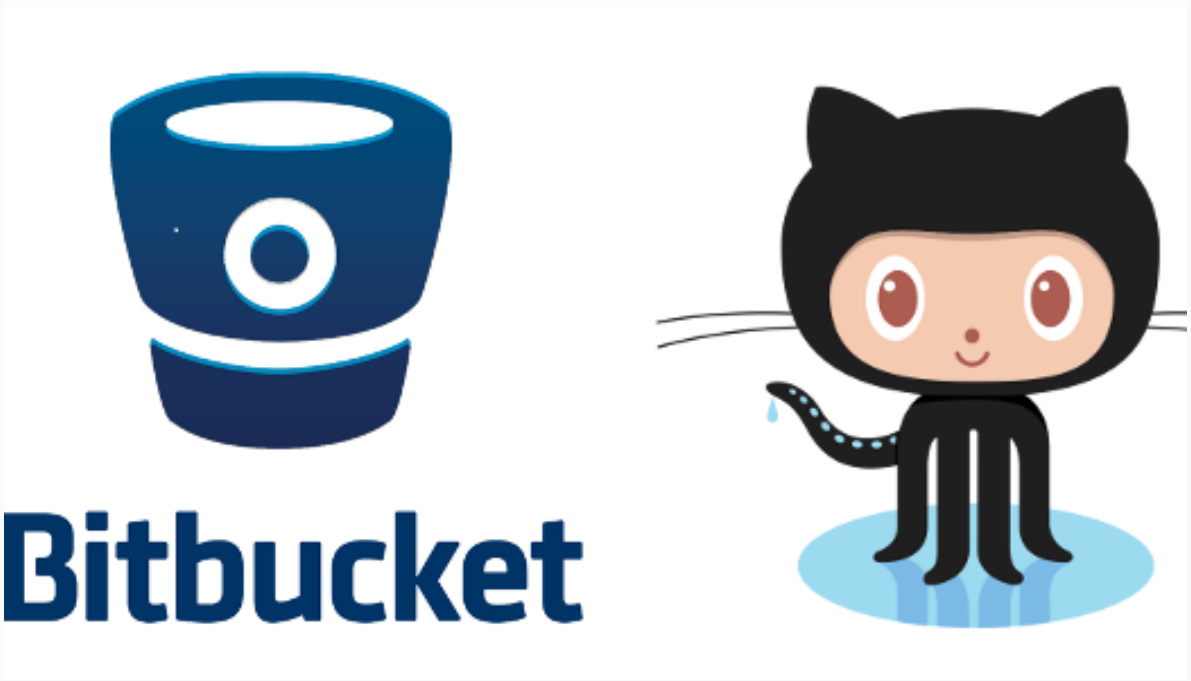
**Editer le fichier**, ou (**Attention** : supprime les modifications de la branche mergée !) `$ git checkout -- [fichier en conflit]`.
Puis

```
$ git add [fichier en conflit]
$ git commit
```

# Exercice : les branches

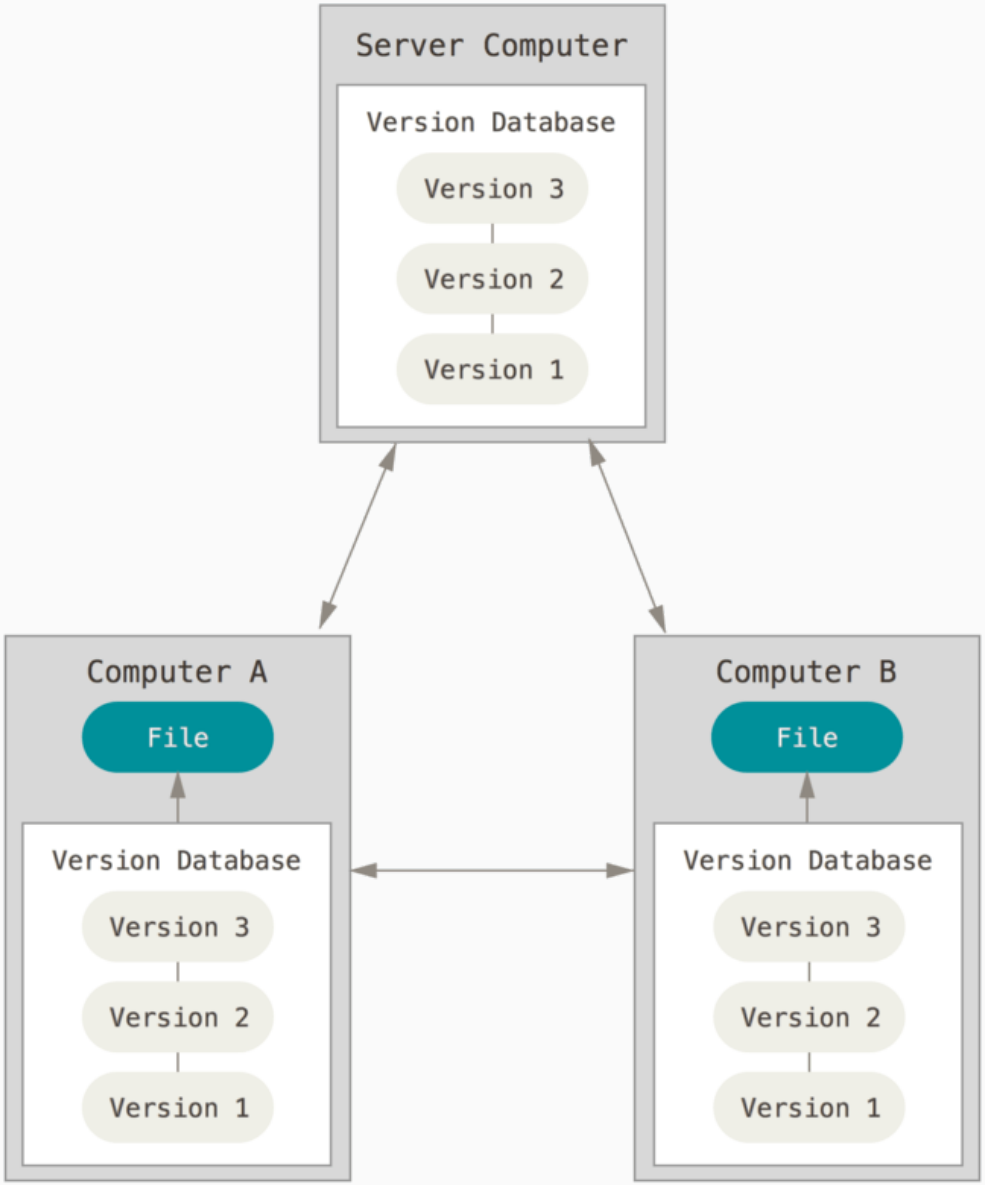
# Le travail en groupe

# Github, Bitbucket, Gitlab

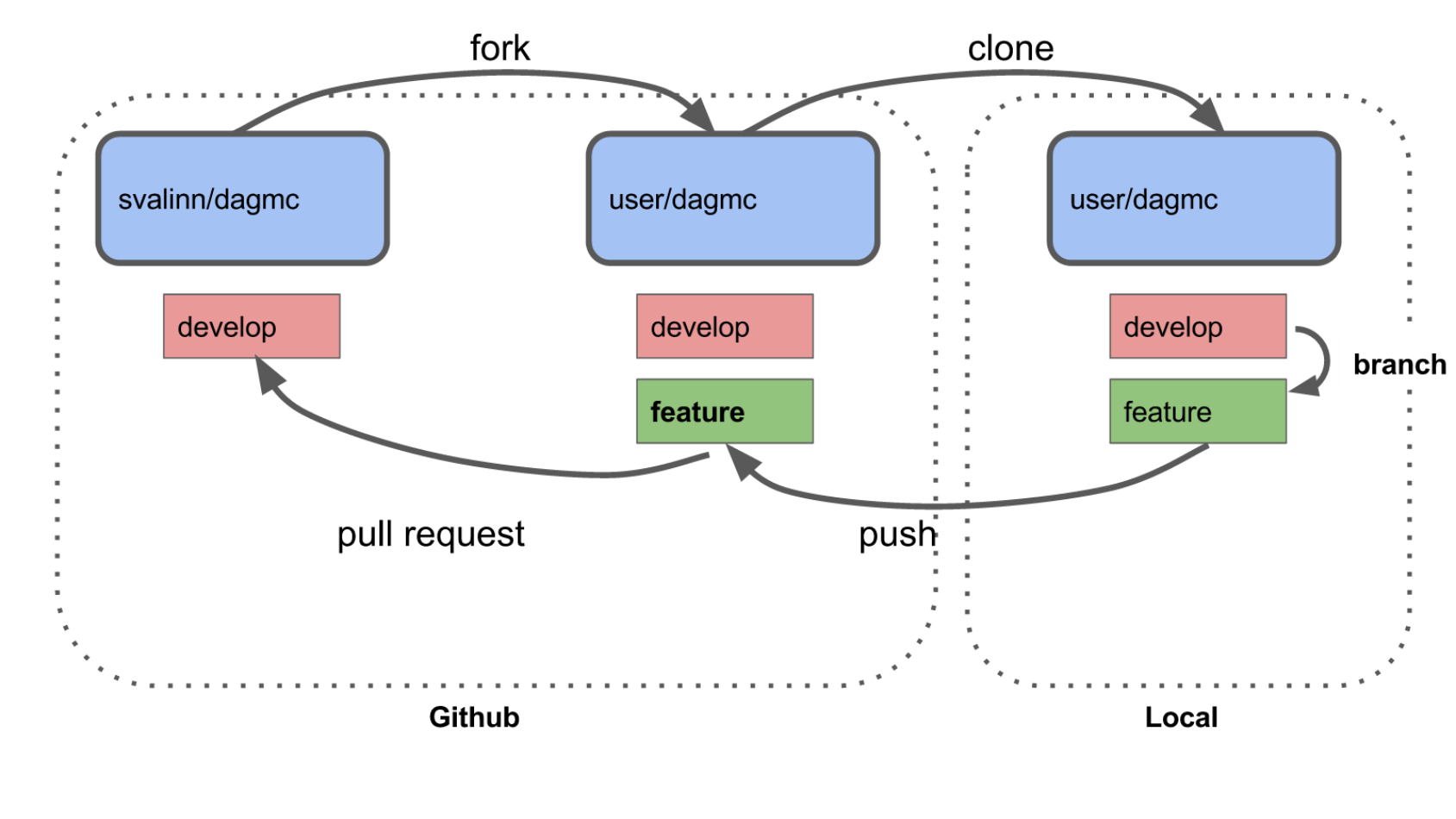



GitLab

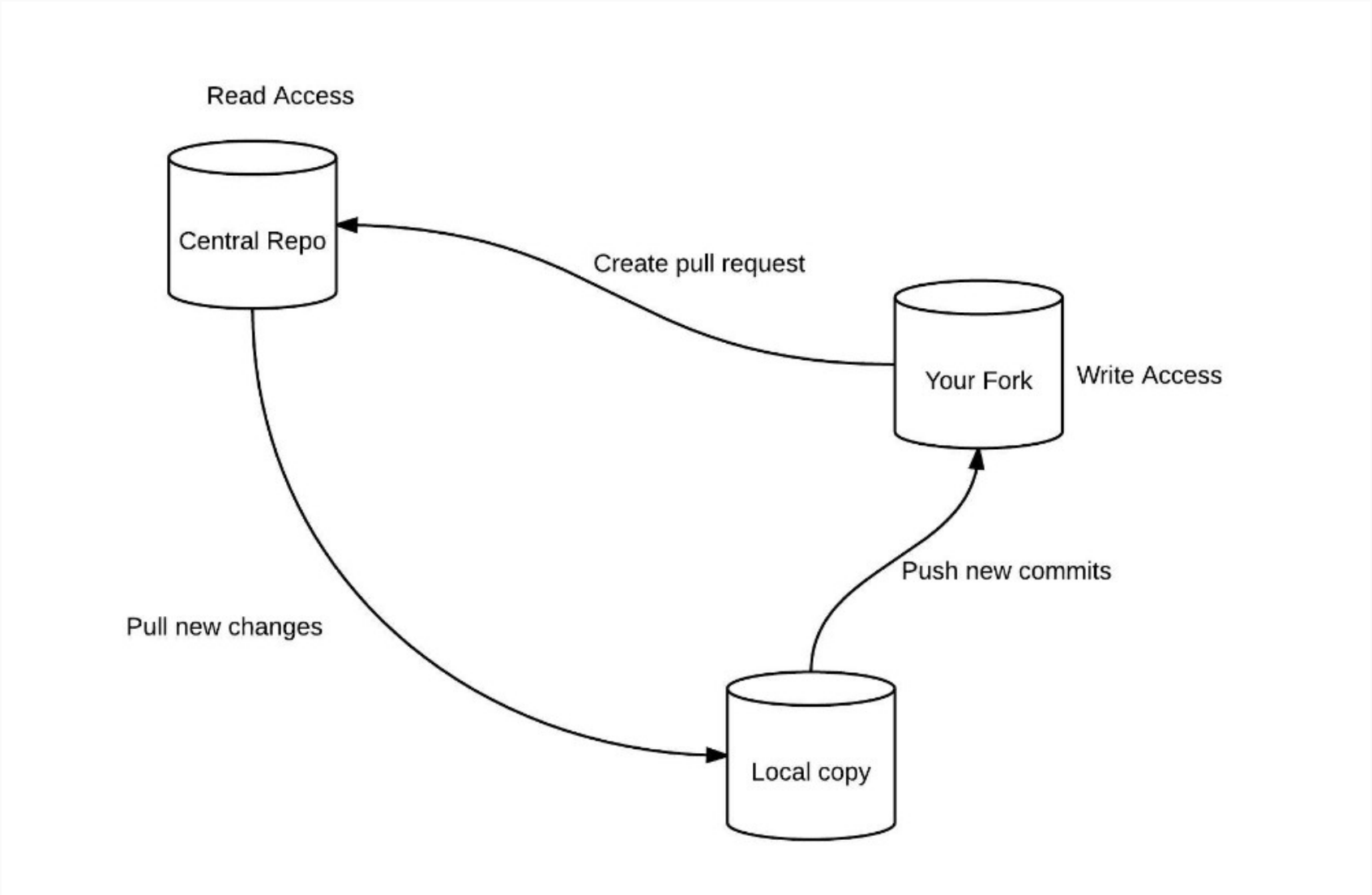
# Distribué... comment se synchroniser ?



# Mise en place



# Méthode de travail



# git clone

- Cloner un répertoire `git` depuis un serveur principal
- Exemple

```
git clone <url>
```

# git remote

- Ajouter un serveur distant à votre répertoire `git`
- Exemple

```
git remote add origin <url>
```

# git pull

- Récupérer les dernières modifications depuis le serveur principal
- Exemple

```
git pull origin
```

# git push

- Envoyer les dernières modifications locales sur le serveur principal
- Exemple

```
git push origin master
```

# Pour aller plus loin ...

# Idées intérésantes

- git flow
- Github student pack
- Travis CI
- ...

# Liens utiles

- https://git-scm.com/
- http://rogerdudler.github.io/git-guide/
- https://fr.atlassian.com/git/tutorials/
- https://www.grafikart.fr/formations/git

# Questions ?