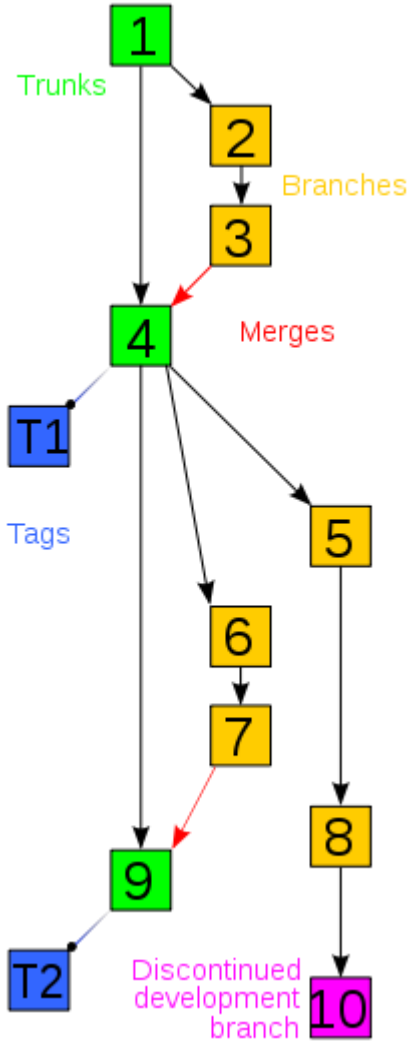
Git & Redmine

Alexandre Abadie

SED Saclay

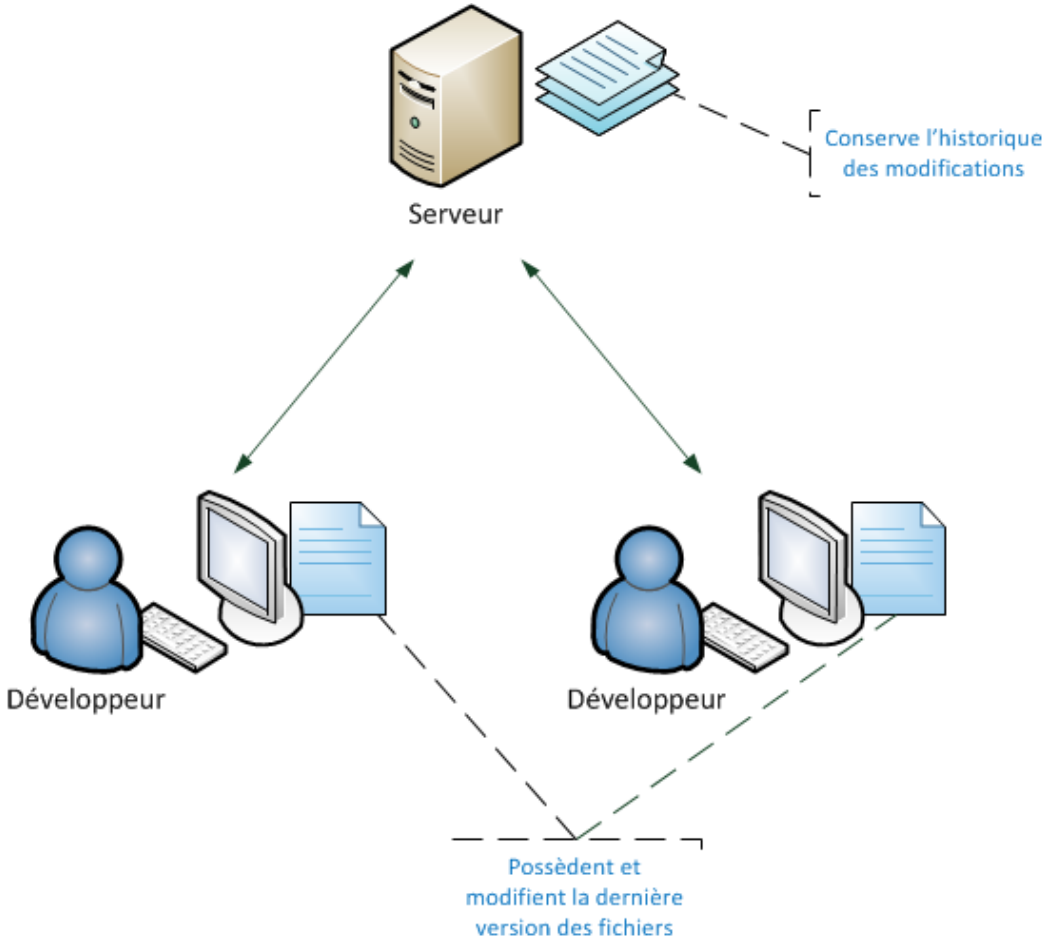
Généralités sur les gestionnaires de version

- Outils pour faciliter le travail en équipe sur des versions successives du code source d'un logiciel
- Les gestionnaires de version peuvent être employés dans d'autres domaines :
  - Gestion de documentation
  - Gestion de configuration (Admin système)
  - Etc
- 2 grandes familles :
  - Systèmes centralisés
  - Systèmes distribués



Systèmes centralisés

Un seul dépôt de versions accessible sur un serveur distant

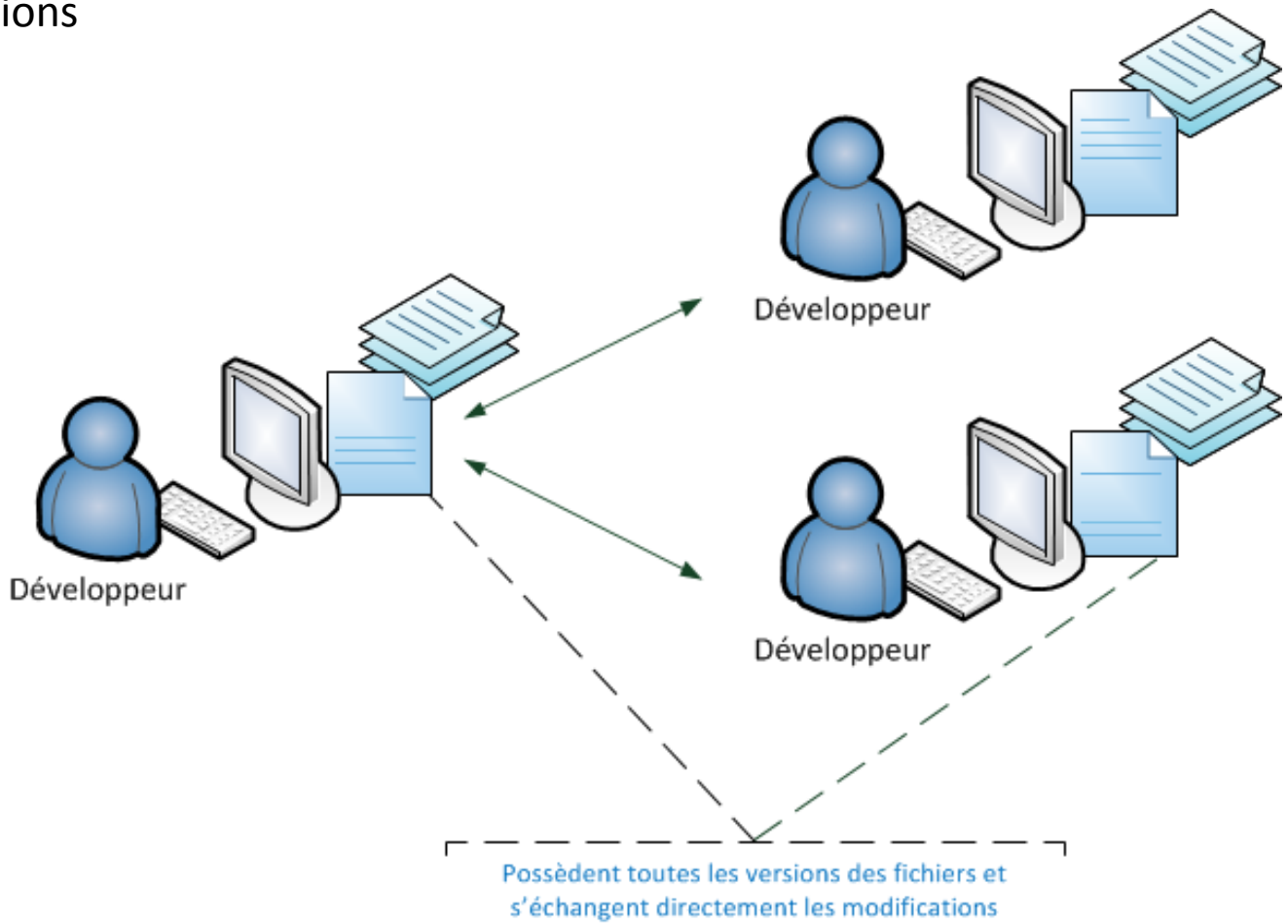


Exemples :

- CVS
- Subversion

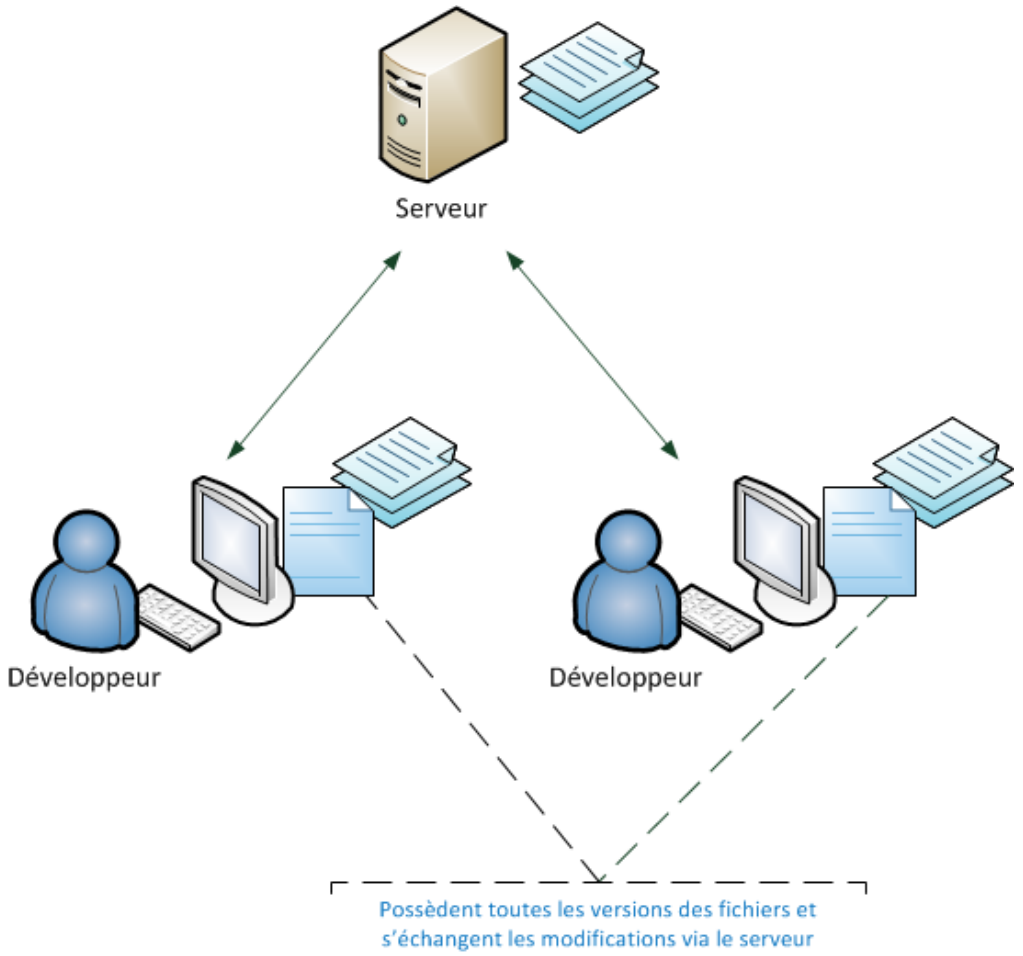
Systèmes distribués

Chaque développeur possède localement un dépôt de versions



Systèmes distribués

Dans la pratique, on utilise un (ou plusieurs) serveur(s) servant d'intermédiaire entre les développeurs



Exemples :

- Git
- Mercurial
- Bazaar
- Arch

Généralités sur Git

- Créé en avril 2005 par Linus Torvald
- Au départ pour gérer le code source du noyau Linux
- Les objectifs :
  - Rapidité
  - Simplicité
  - Système distribué
  - Capable de gérer de très gros projets
  - Capable de supporter des développements non linéaires (branches multiples)
- Outil de gestion de version de plus en plus utilisé aujourd'hui

Installation de Git

Linux :

Debian : **`apt-get install git`**

Fedora : **`yum install git-core`**

Mac OSX :

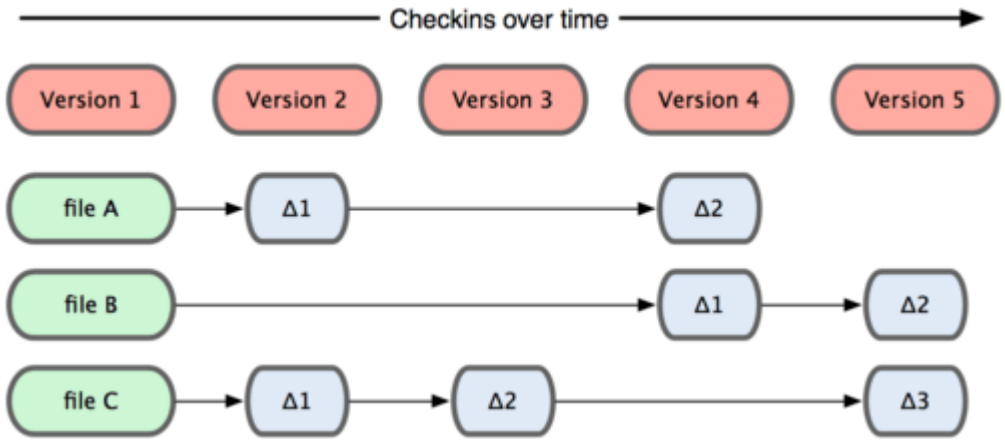
http://code.google.com/p/git-osx-installer

Windows:

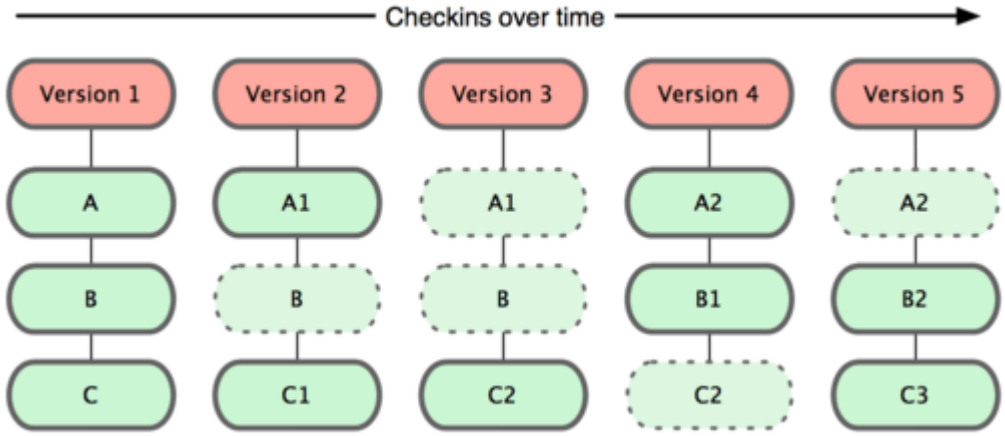
http://msysgit.github.com/

Comparaison avec les autres gestionnaires de version

La plupart des gestionnaires de versions gèrent un historique des différences appliquées aux fichiers



Git gère une suite de références vers des instantanés du contenu du projet.
Git peut être vu comme un mini système de fichiers



Organisation d'un dépôt Git

`git init` = création d'un dépôt => dossier *.git*

`HEAD branches/ config description hooks/ info/ objects/ refs/`

config : le fichier de configuration de git local au projet

hooks : les scripts automatiques exécutés avant/après certaines actions

info : contient le fichier *exclude* et le fichier *refs*

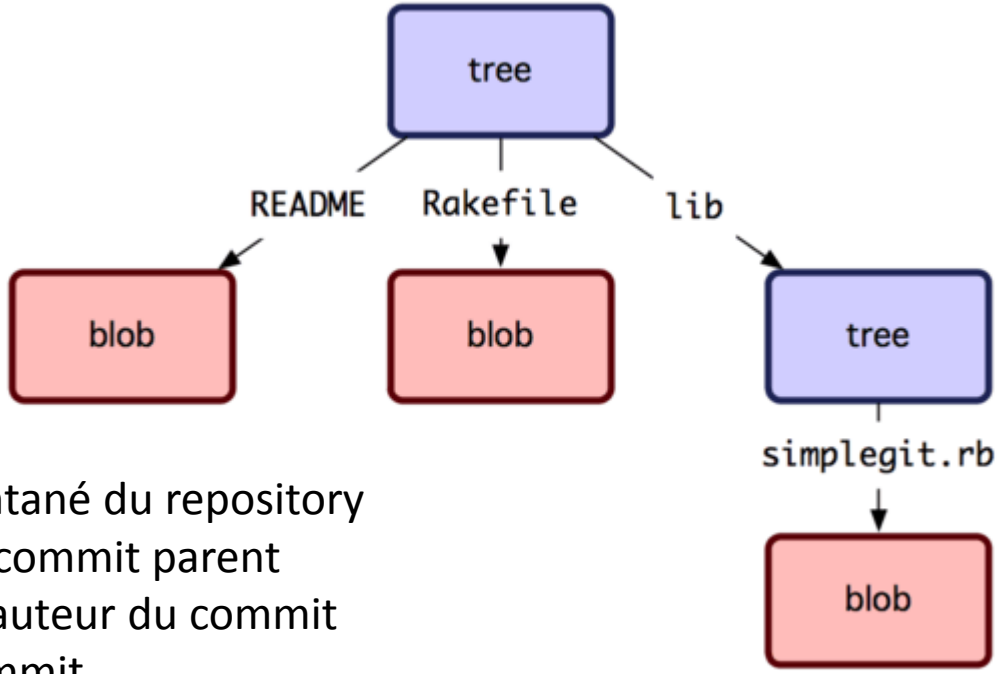
refs : stocke les pointeurs vers les objets commit de la base de données

objects : stocke le contenu de la base de données

index : stocke les informations sur la zone d'attente

HEAD : pointe sur la branche en cours dans le répertoire de travail

Organisation d'un dépôt Git



3 types d'objets :

- Blobs : Contenu d'un fichier
- Tree : Correspond à un répertoire
- Commit :
  - Contient un arbre vers un instantané du repository
  - Contient une référence sur son commit parent
  - Contient des informations sur l'auteur du commit
  - Contient le commentaire du commit

Chaque object est identifié par un hash SHA-1 unique
Exemple : **`9585191f37f7b0fb9444f35a9bf50de191beadc2`**

Exemple : projet *simplegit* https://github.com/schacon/simplegit

Paramétrage de Git

Commande **`git config`**

Lister les paramètres de configuration : **`git config --list`**

3 niveaux de configuration :

- *Système* => pour tous les utilisateurs
  correspond au fichier **`/etc/gitconfig`**

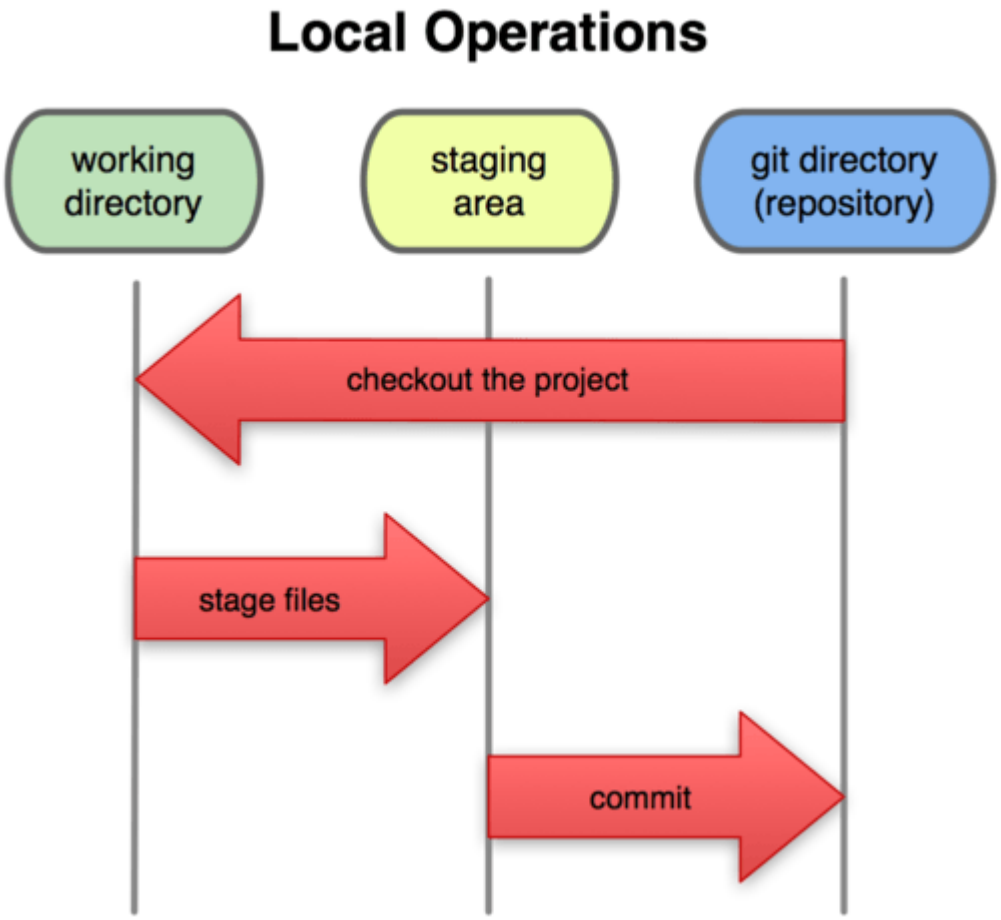
- *Global* => pour tous les projets d'un utilisateur,
  correspond au fichier **`~/.gitconfig`**

- *Projet* => paramètres s'applique uniquement au projet courant
  correspond au fichier **`$PROJECT_PATH/.git/config`**

Configuration des informations utilisateur :

```
git config --global user.name "Alexandre Abadie"
git config --global user.email alexandre.abadie@inria.fr
```

Concepts de base



3 statuts : validé, indexé, modifié

(modified, staged, commited)

Les commandes de base

Première étape : le repository

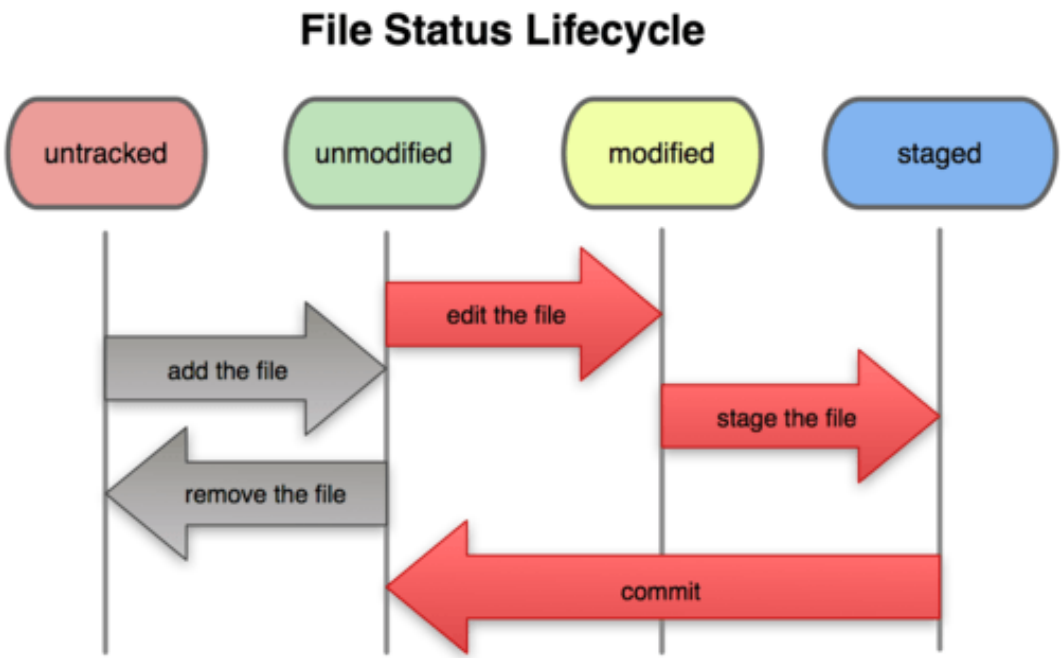
- Soit on le crée à partir d'un projet existant :

```
git init
touch README
git add README
git commit –m "initialisation de mon repository"
```

- Soit on récupére le repository d'un projet existant

```
git clone git@github.com:progit/progit.git
```

Les commandes de base



Les commandes de base : gérer les modifications

Vérifier l'état des fichiers : **`git status`**
Inspecter les différences éventuelles : **`git diff`**
Inspecter les différences indexées : **`git diff --cached`** (ou --staged)

Ajouter des fichiers à l'index : **`git add *fichier*`**
Valider les fichiers de l'index : **`git commit –m "commentaire"`**
Supprimer des fichiers : **`git rm *fichier*`**
Supprimer des fichiers de l'index : **`git rm --cached *fichier*`**
Renommer un fichier : **`git mv ancien nouveau`**

Note :

- On peut "sauter" la phase d'index avec l'option "-a" :

  `git commit -a -m "commentaire"`
- On peut ajouter des sous parties d'un ou plusieurs fichiers à l'index :

  `git add -p *fichier*`

Visualiser l'historique des validations :

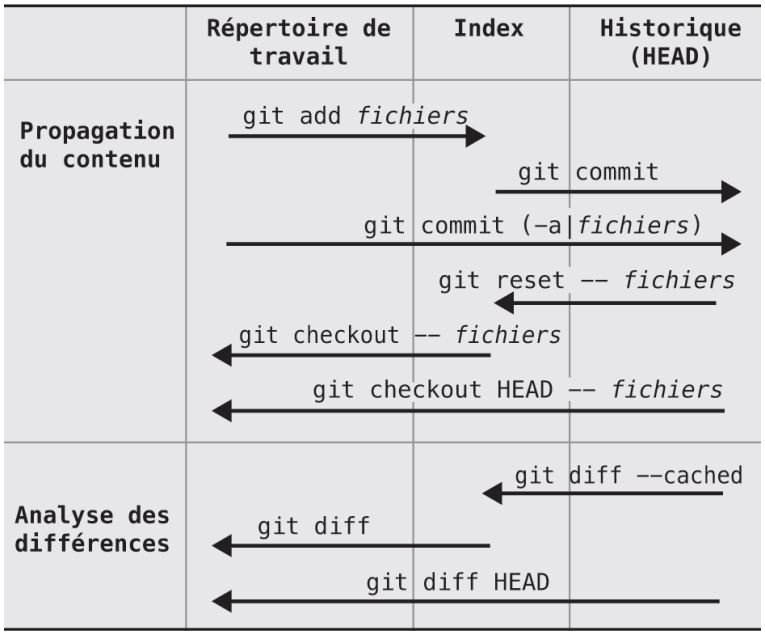
- En ligne de commande : **`git log`**
- Avec un outil graphique : **`gitk`**

Les commandes de base : annuler des actions

Modifier le dernier commit : **`git commit --amend`**

Enlever un fichier de l'index : **`git reset fichier`**

Annuler les modifications sur un fichier : **`git checkout fichier`**



Mettre de côté des modifications (non ajoutées à l'index) : **`git stash`**

Appliquer les modifications mises de côté : **`git stash apply`**

Les commande de bases : les depôts distants

Dépôt origin : **`git clone git@github.com:aabadie/simplegit.git`**

Lister les dépôts distants : **`git remote -v`**

Ajouter un remote : **`git remote add <nom> <url>`**

Supprimer un remote : **`git remote rm <nom>`**

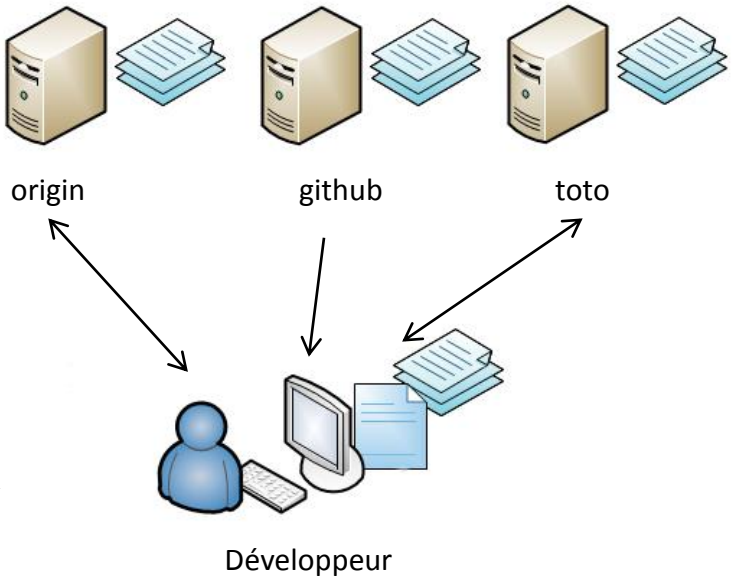
Renommer un remote :
`git remote mv <ancien> <nouveau>`

Inspecter un dépôt distant : **`git remote show <nom>`**

Se synchroniser avec un remote : **`git fetch <nom>`**

Récupérer et fusionner une branche distante : **`git pull`**

Synchroniser son repository sur un remote : **`git push <nom> <branche>`**

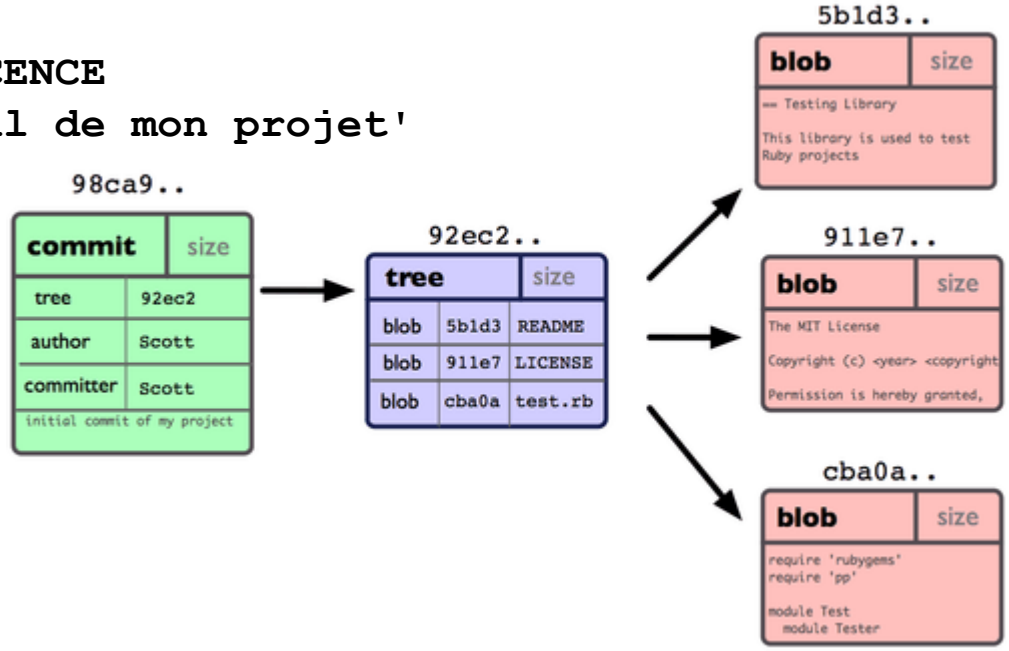


Les branches

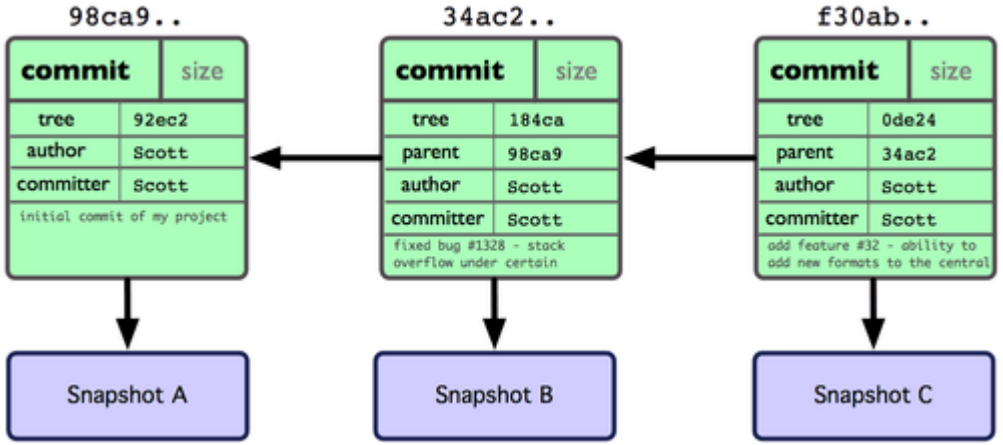
Exemple :

```
git add LISEZMOI test.rb LICENCE
git commit -m 'commit initial de mon projet'
```

Contenu d'un commit :

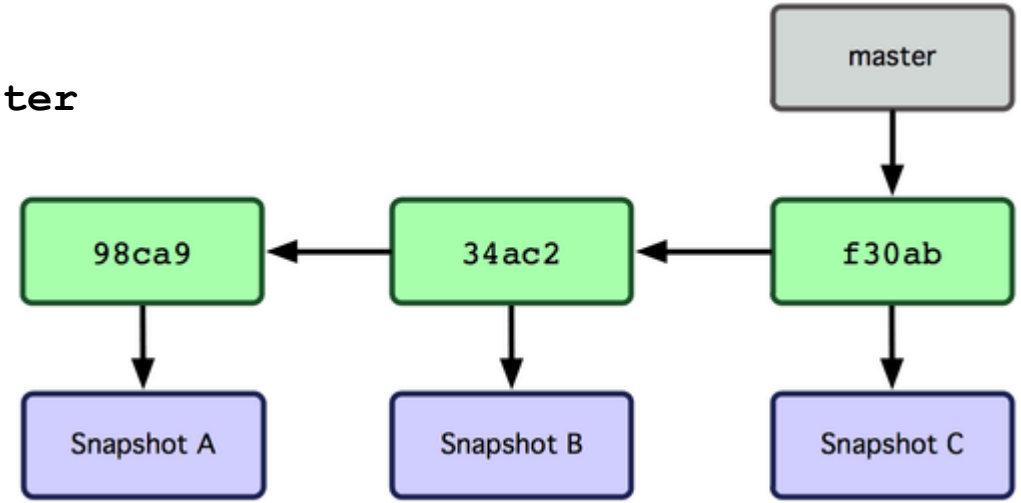


Historique des commits :

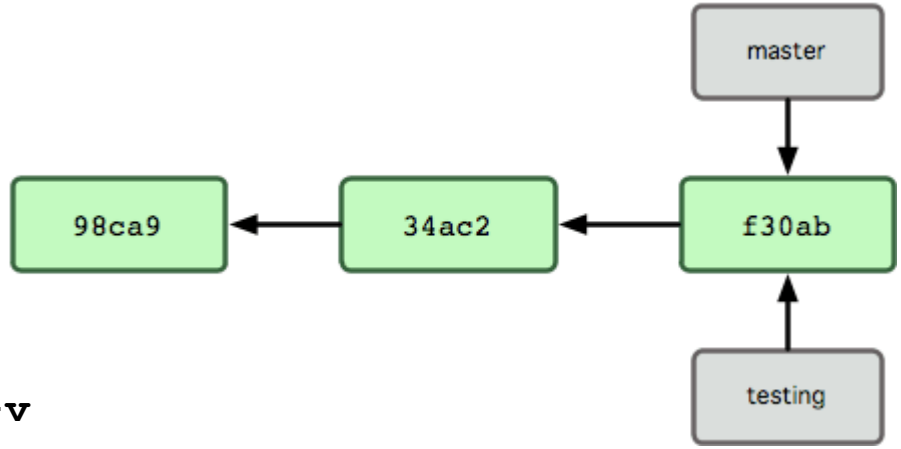


Les branches

La branche par défaut : **`master`**



Nouvelle branche : **`git branch testing`**



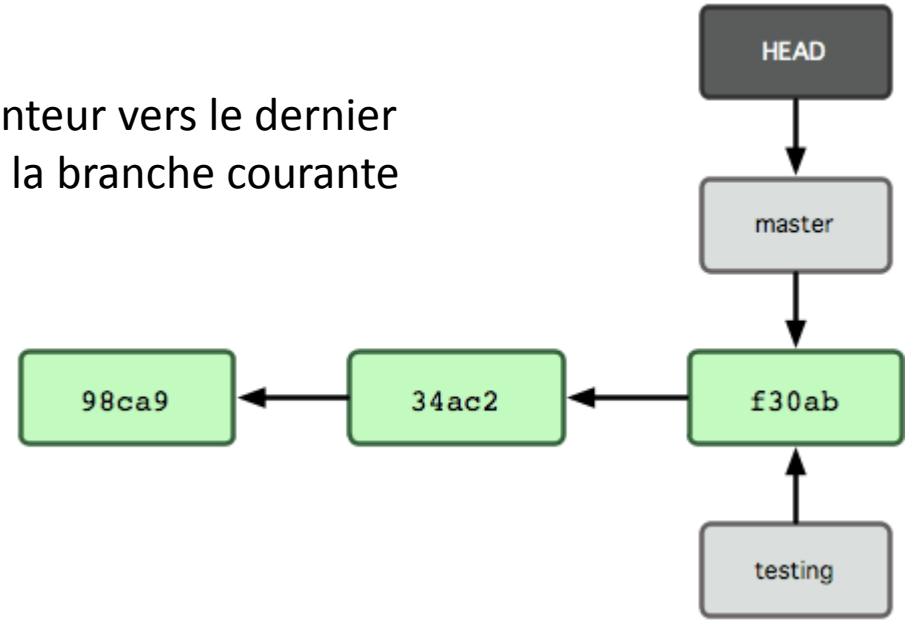
Note :

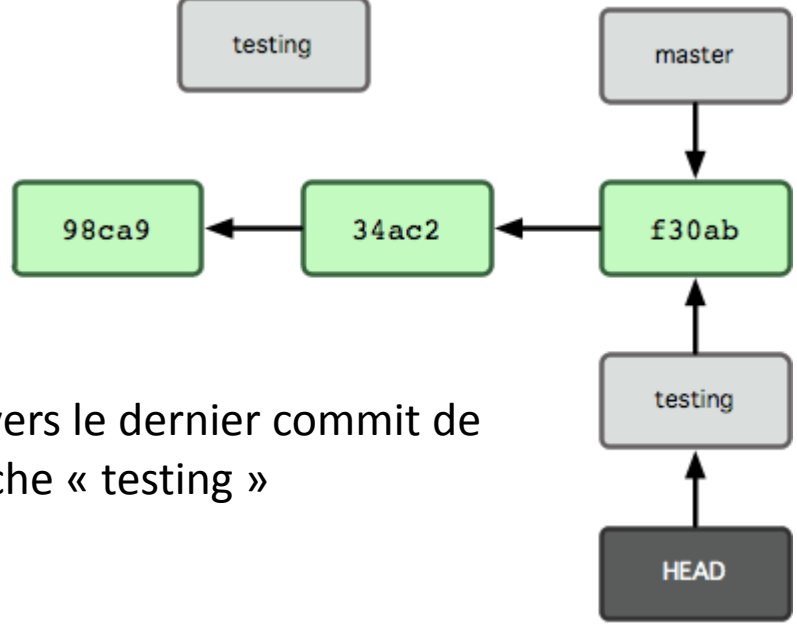
Lister les branches : **`git branch`**

`git branch -v`

Les branches

HEAD : pointeur vers le dernier commit de la branche courante





`git checkout testing` : déplace HEAD vers le dernier commit de la branch « testing » ⇔ bascule vers la branche « testing »

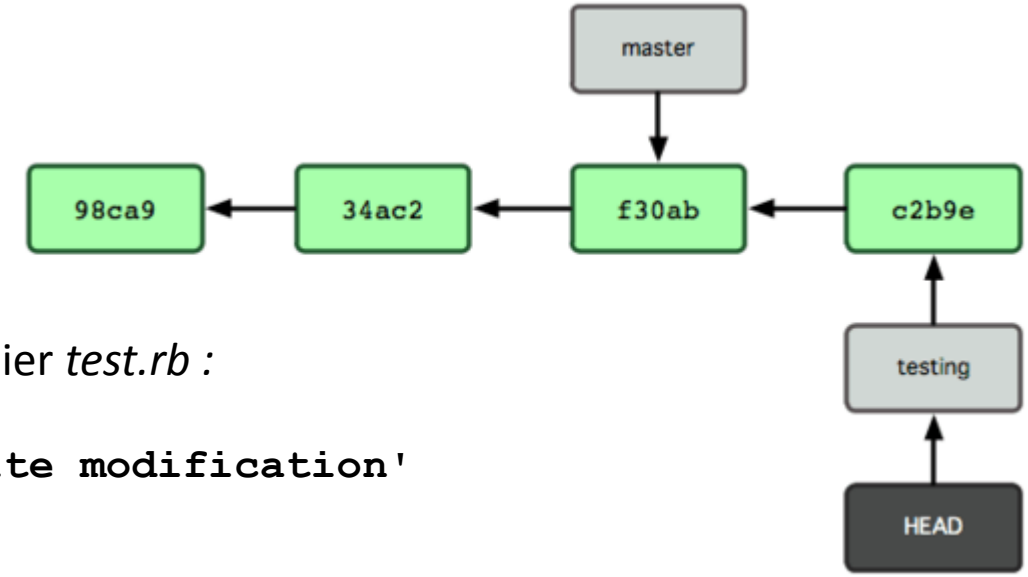
Les branches

Créer une nouvelle branche :

```
git branch <branche>
git checkout <branche>
```

ou en une seule commande

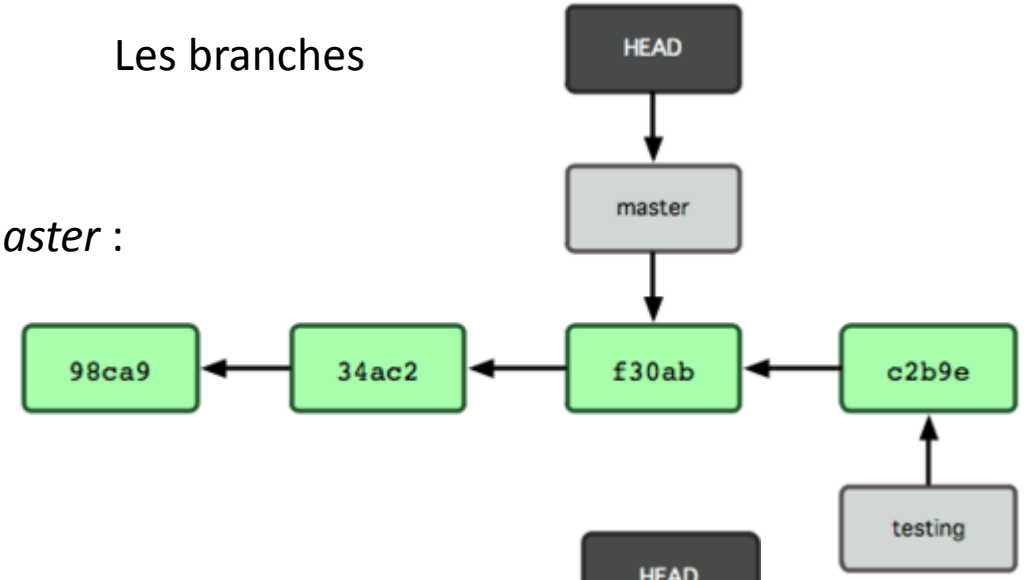
```
git checkout -b <branche>
```



Modifions maintenant le fichier *test.rb :*

```
vim test.rb
git commit -a -m 'petite modification'
```

Les branches

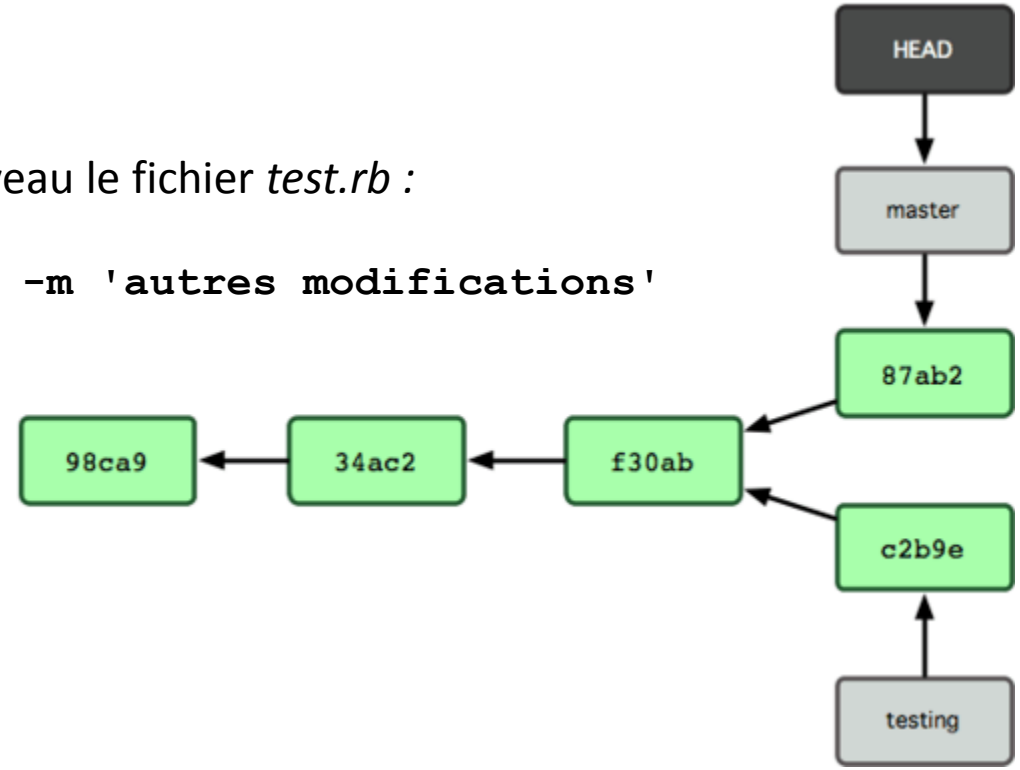


Rebasculons sur la branch *master* :

```
git checkout master
```

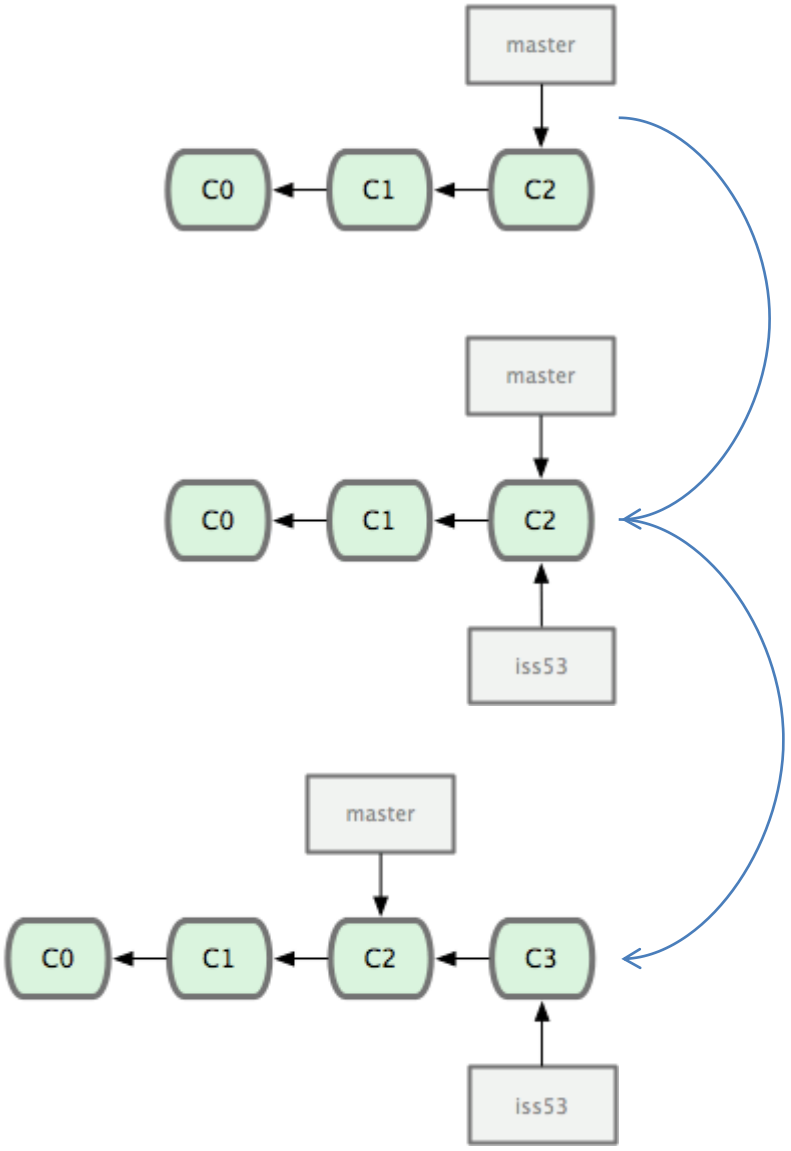
Modifions à nouveau le fichier *test.rb :*

```
vim test.rb
git commit -a -m 'autres modifications'
```



Les branches ont divergé

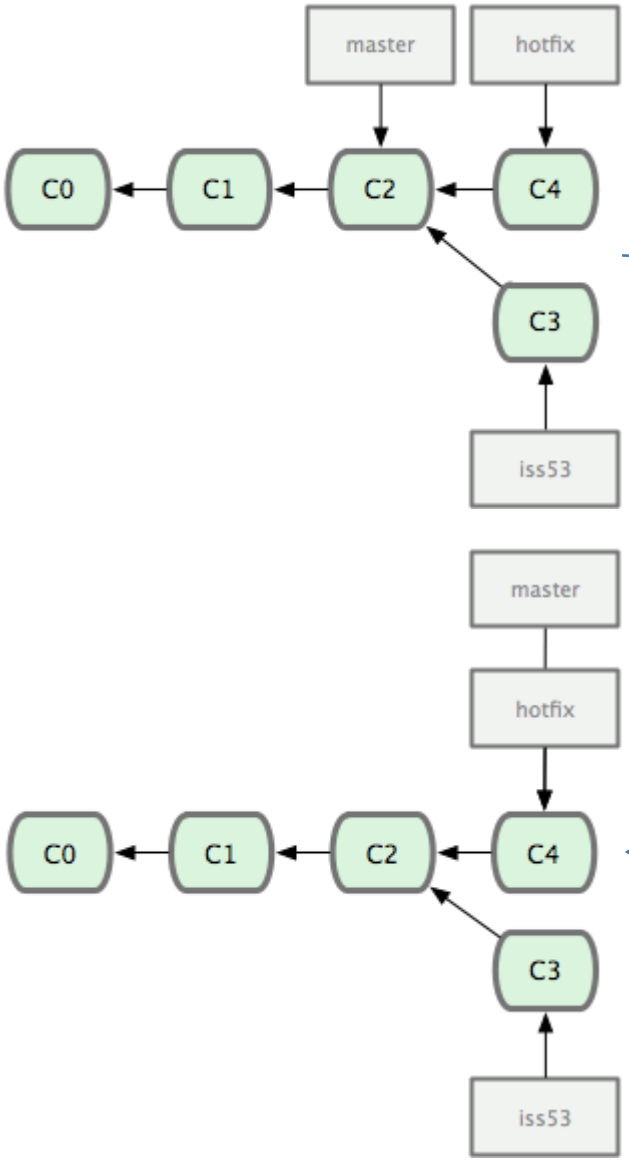
Fusionner les branches




```
git checkout -b iss53
```

```
vim index.html
git commit -a -m 'ajout d'un pied de
page'
```

Fusionner les branches




```
git checkout master
git checkout -b hotfix
vim index.html
git commit -a -m "correction urgente"
```

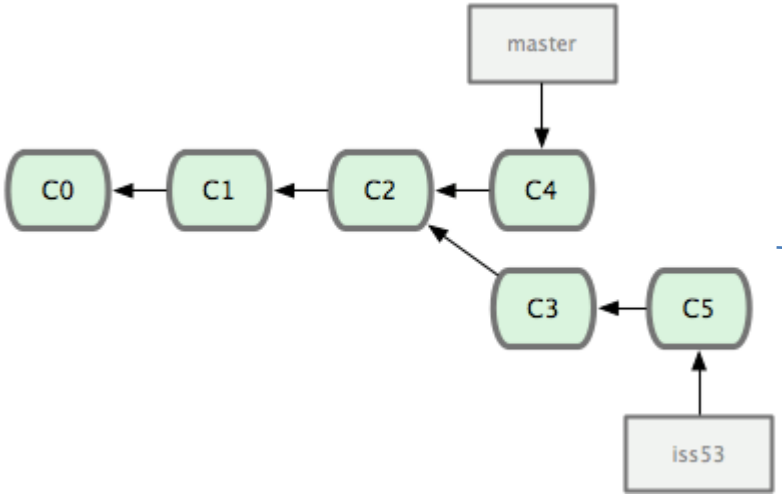
```
git checkout master
git merge hotfix
```

=> Avance rapide (Fast-forward)

Pour supprimer hotfix (qui est inutile maintenant) :

```
git branch -d hotfix
```

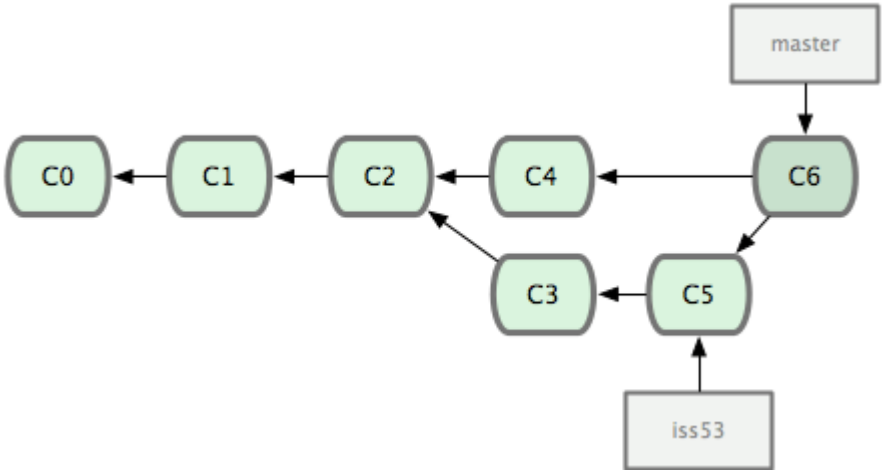
Fusionner les branches




```
git checkout iss53
vim index.html
git commit -a -m 'Travail terminé'
```

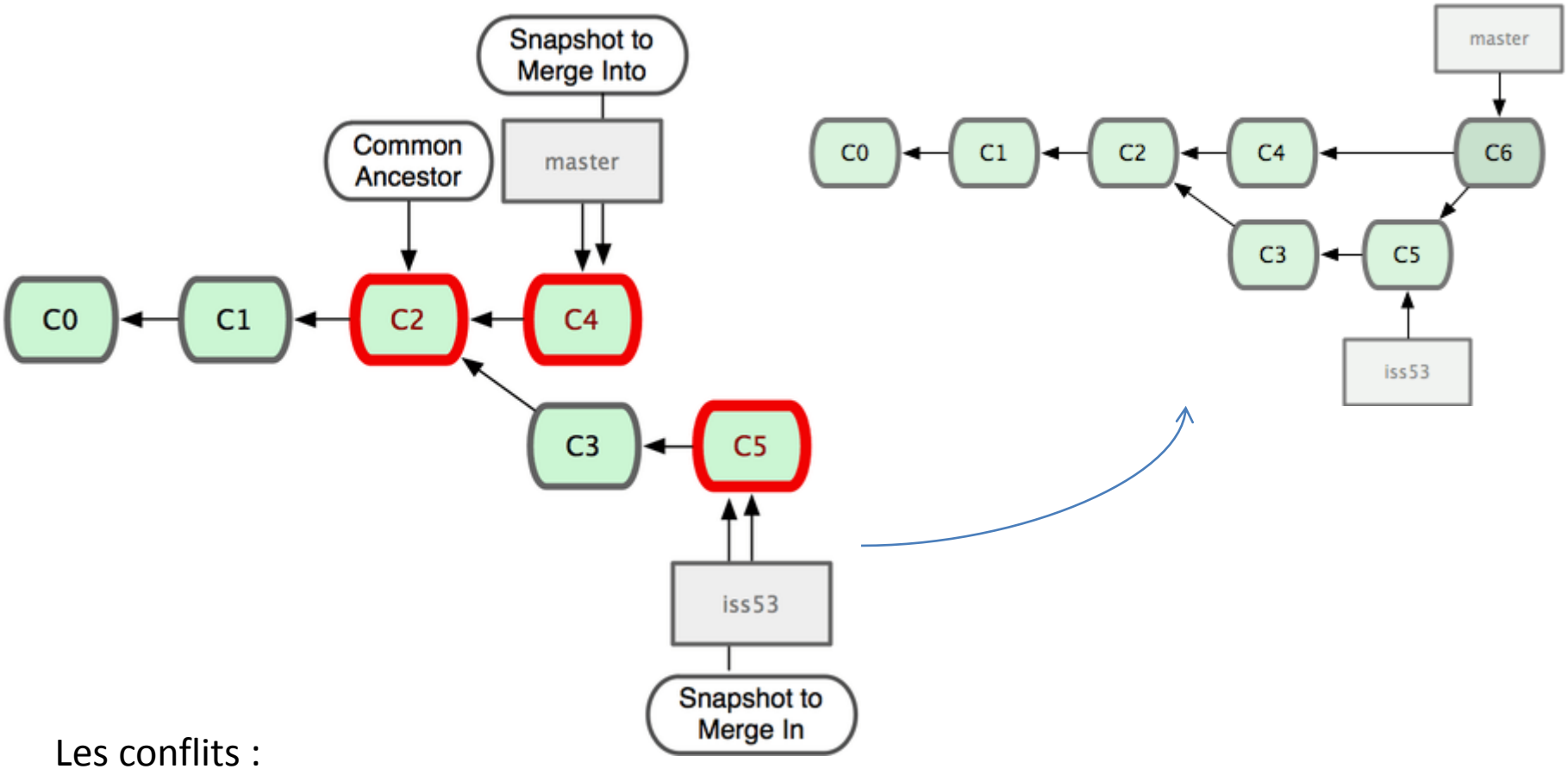
On veut maintenant fusionner iss53 dans master

```
git checkout master
git merge iss53
```



C6 : *commit* spécial de fusion (plusieurs parents) à 3 branches (C2, C4 et C5)

Fusionner les branches



Les conflits :

- Résolution “à la main” : **`git mergetool`**
- Quand le conflit est corrigé : **`git commit`**

Sources

http://git-scm.com/book/fr/
http://fr.openclassrooms.com/informatique/cours/gerez-vos-codes-source-avec-git
http://try.github.io

Comment/où héberger son projet Git

https://gforge.inria.fr/

https://github.com

https://bitbucket.org

https://gitorious.org

Utilisent le mécanisme des forks et merge/pull requests

Comment héberger son projet Git

Soi-même !

http://gitlab.org

http://getgitorious.com/

http://www.redmine.org/ (+ plugin redmine git hosting)

Redmine

Généralités

- Gestionnaire de projet flexible type bug tracker
- Application web Ruby on Rails
- Open source GPLv2
- Multi-plateforme et multi base de données
- Outils utilisé par de nombreux projets et entreprises
  http://www.redmine.org/projects/redmine/wiki/WeAreUsingRedmine

Fonctionnalités multiples

- Support multi-projets
- Diagramme de Gantt
- Calendrier
- Gestionnaire de News, de documents, de fichiers
- Wiki
- Forum
- Integration des systèmes de gestion de version classiques (svn, git, mercurial, etc)
- Support de l'authentification LDAP
- Configuration très très flexible

Démo

http://carnegie.saclay.inria.fr

Merci