Git

Plan

- Historique
- Bases de Git
- Premiers pas
- Gestion des branches
- Merging
- Remotes
- Workflows

CC BY NC SA

Historique

Qu'est-ce qu'un VCS ?

- Enregistre les différentes version d'un ensemble de fichiers
- Gère la chronologie des différences entre les fichiers
- Retour arrière
- Facilite le travail en équipe

cc BY NC SA

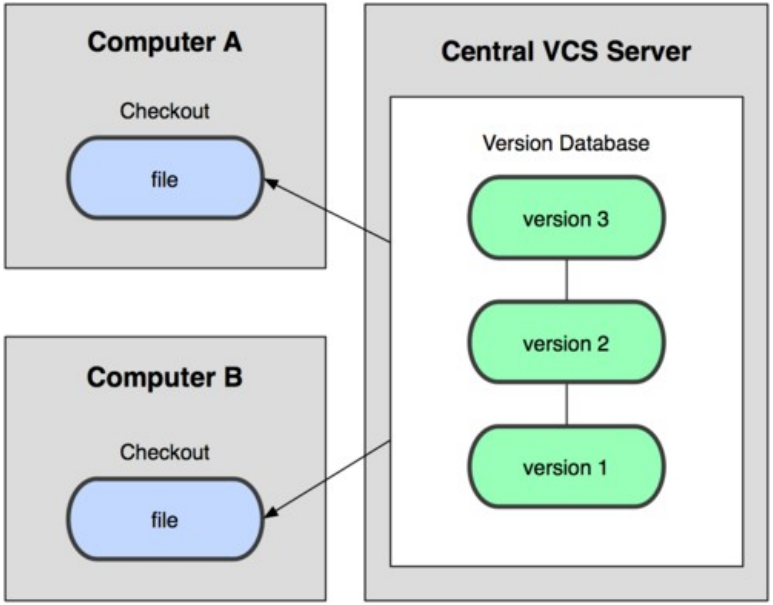
Historique des VCS

- 1990 : CVS
  - Chaque fichier possède sa propre version
- 2000 : Subversion
  - Ensemble des fichiers ont un numéro de révision
  - Commit atomiques
- 2005 : DVCS (Git, Mercurial, Bazaar)
  - Plus de serveur central
  - Gestion facilité des branches et des merge

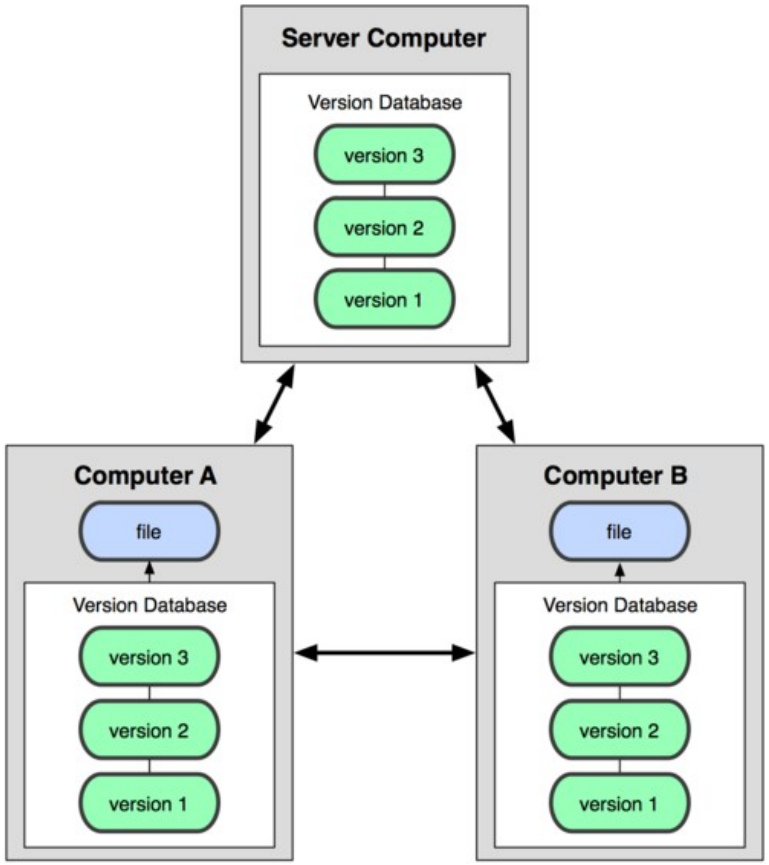
CC BY NC SA

DVCS

Centralisé (CVS, SVN) :



Décentralisé (Git, Hg, Bazaar) :



cc BY NC SA

DVCS

- Chaque personne possède une copie du référentiel
- Rapide car chaque opération se fait en local
- Permet de commiter en étant déconnecté
- Permet d'expérimenter en local
- Plus de point unique de défaillance
- Permet une participation aisée (permissions)

CC BY NC SA

Pourquoi Git ?

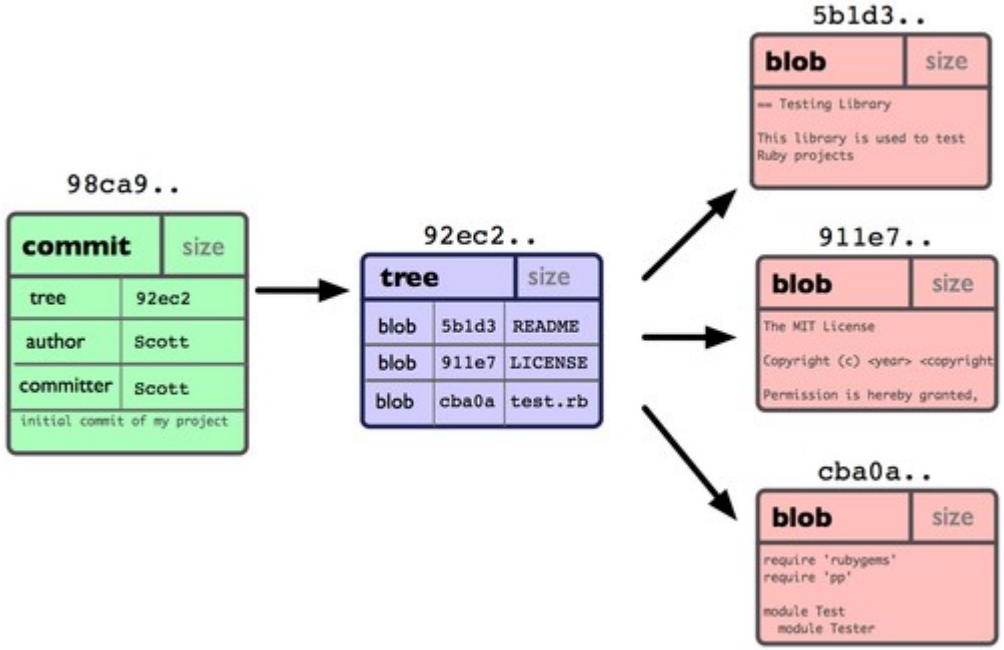
- Utilisé par la plupart des projets OpenSource : Linux, VLC, LibreOffice, Android, Firefox
- Facilité pour la création de branche et merge
- Rapidité
- GitHub, le SourceForge social

CC BY NC SA

Bases de Git

Bases de Git

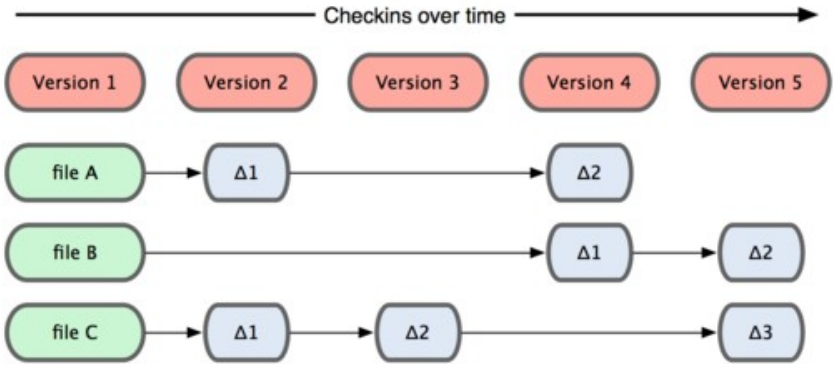
- 3 types d'objets



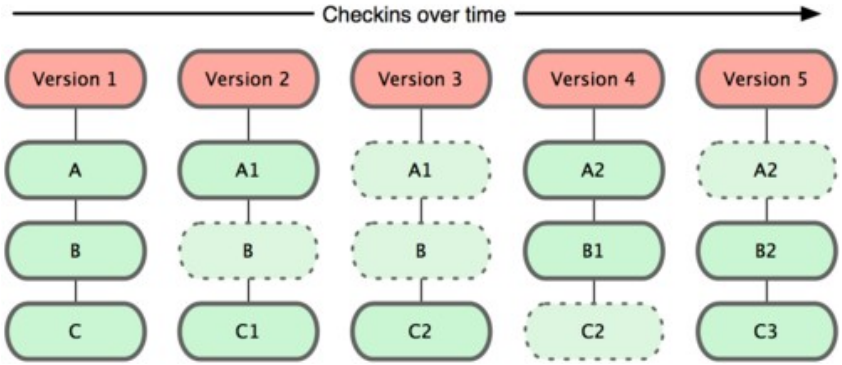
- Chaque objet est représenté par un SHA-1

Bases de Git

- Les principaux VCS enregistrent les différences :



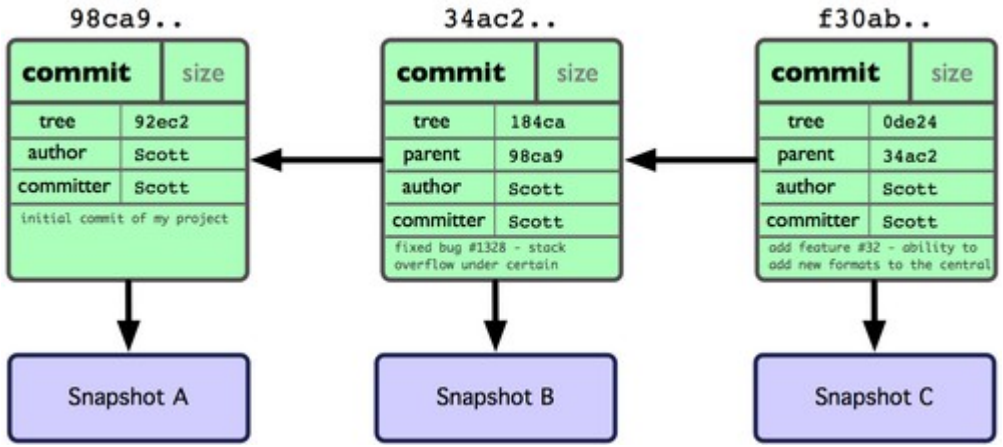
- Git enregistre des snapshots du référentiel :



CC BY NC SA

Bases de Git

- Après plusieurs commit :



Premiers pas

Initialisation du référentiel

git init

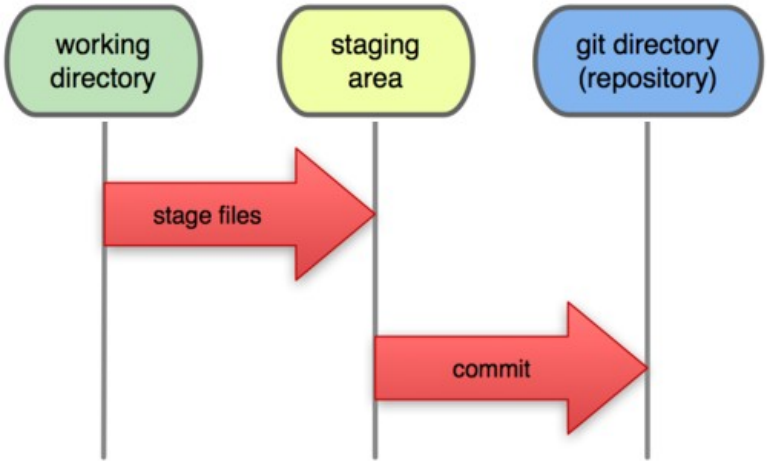
- Création d'un sous-répertoire *.git* unique à la racine du projet

cc BY NC SA

Ajout de fichiers

git add

- Ajoute un ou plusieurs fichiers à l'index



CC BY NC SA

Commit

git commit

- Enregistre les modifications présentent dans l'index
- Message de commit obligatoire

cc BY NC SA

Commit (amend)

git commit --amend

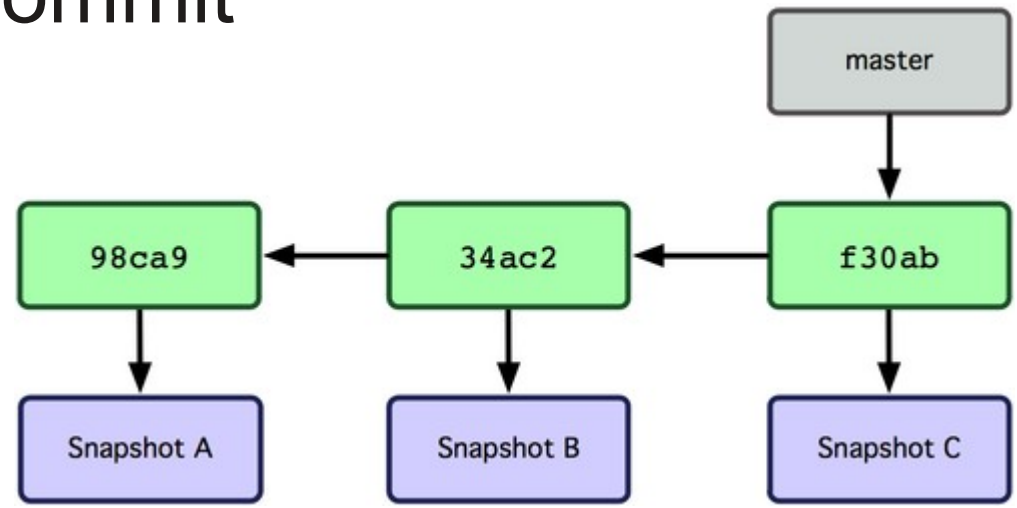
- Met à jour le dernier commit
- Possible uniquement s'il n'a pas déjà été partagé sur un serveur

CC BY NC SA

Gestion des branches

Gestion des branches

- Une branche est un simple pointeur vers un commit spécifique
- La branche par défaut est *master*
- Le pointeur est avancé automatiquement à chaque commit

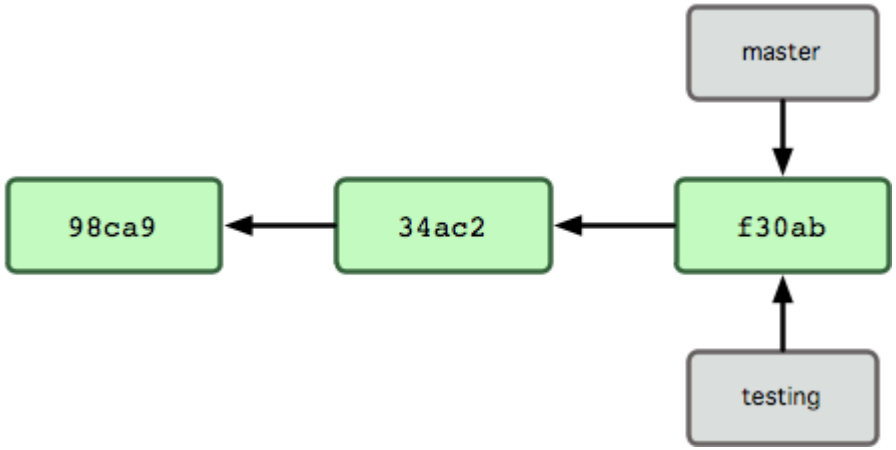


CC BY NC SA

Création d'une branche

git branch *branch-name*

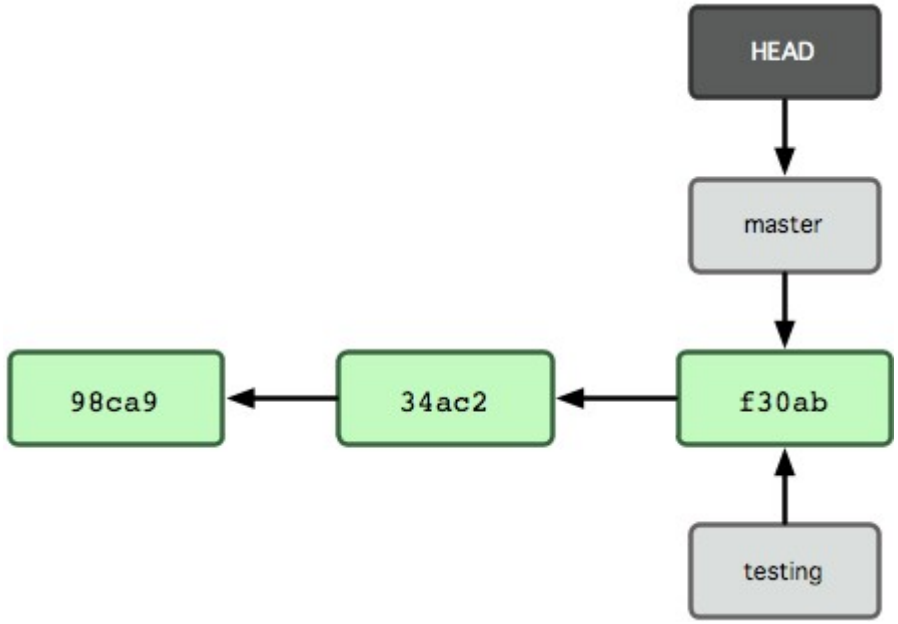
- Création d'un nouveau pointeur *branch-name* sur le commit courant



CC BY NC SA

Branche courante

- Git gère un pointeur spécifique : *HEAD*
- *HEAD* pointe sur la branche courante

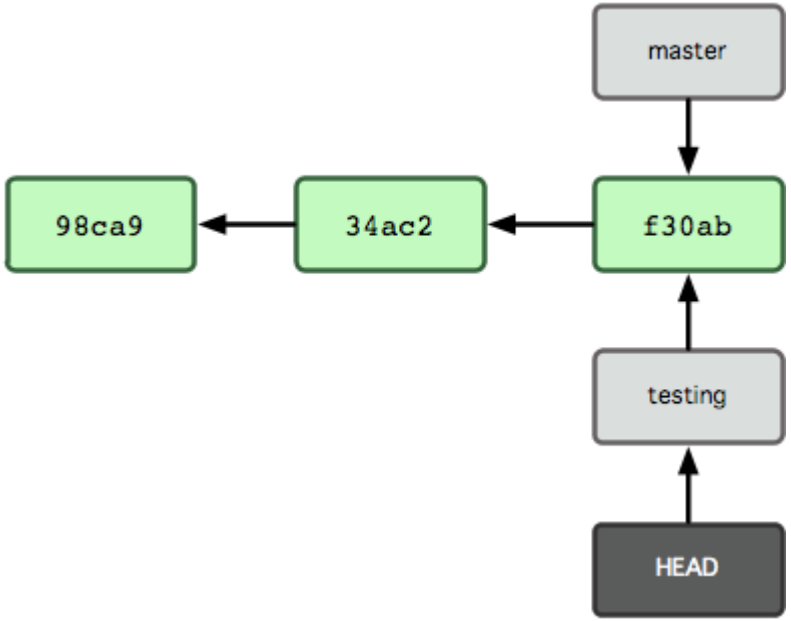


CC BY NC SA

Changement de branche

git checkout *branch-name*

- Déplacement de *HEAD* sur *branch-name*

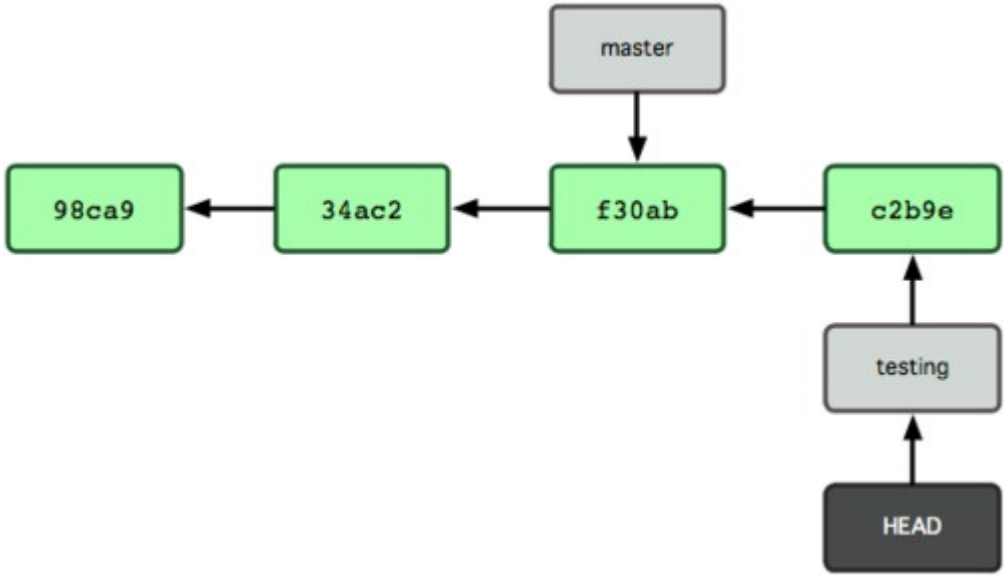


CC BY NC SA

Modifications sur les branches

git commit

- Déplacement du pointeur courant sur le nouveau commit



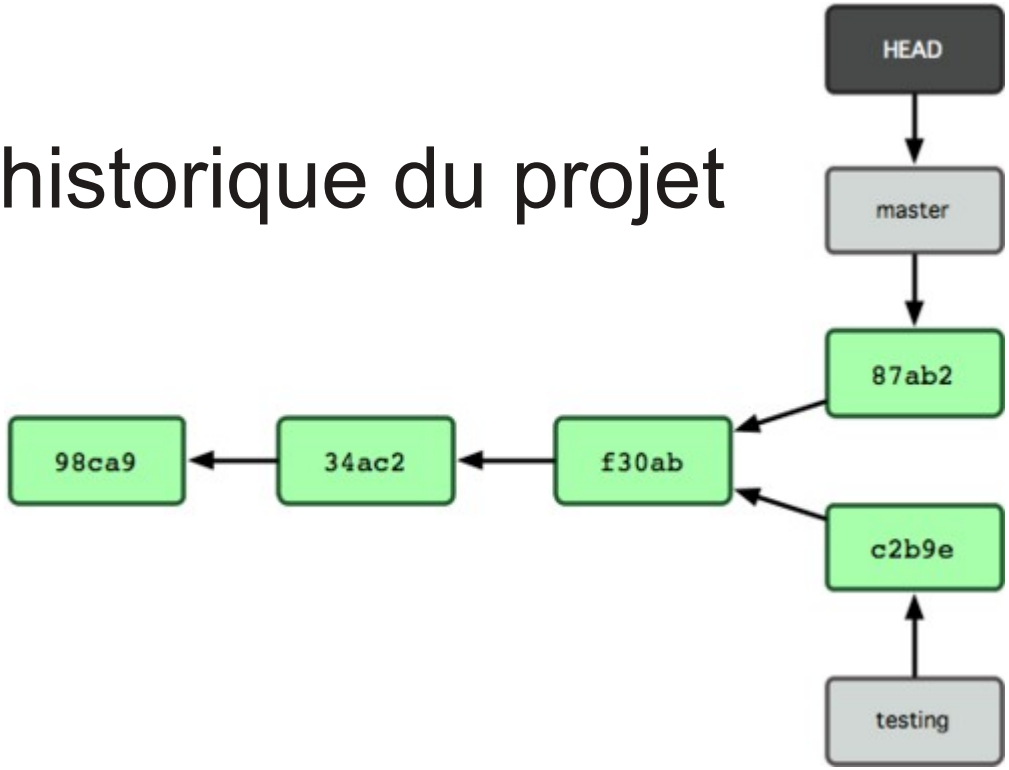
CC BY NC SA

Modifications sur les branches

git checkout master

git commit

- Divergence de l'historique du projet

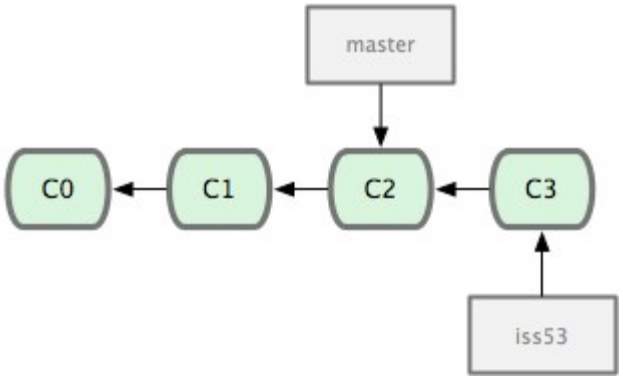


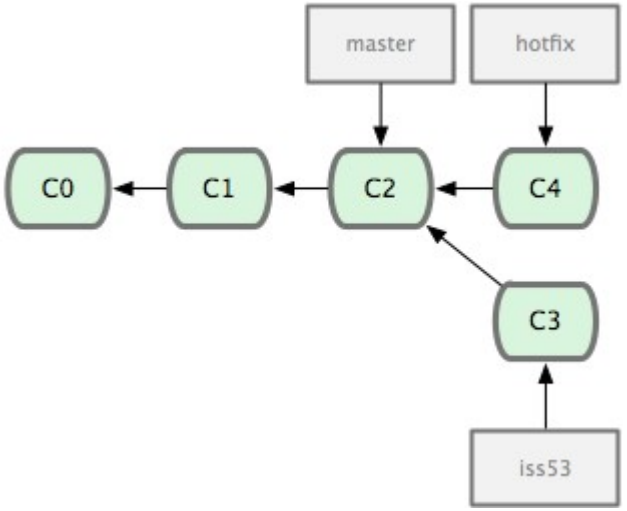
CC BY NC SA

Merging

Situation initiale

- 3 branches
  - master : version courante
  - iss53 : travail en cours sur un ticket
  - hotfix : bug bloquant à corriger



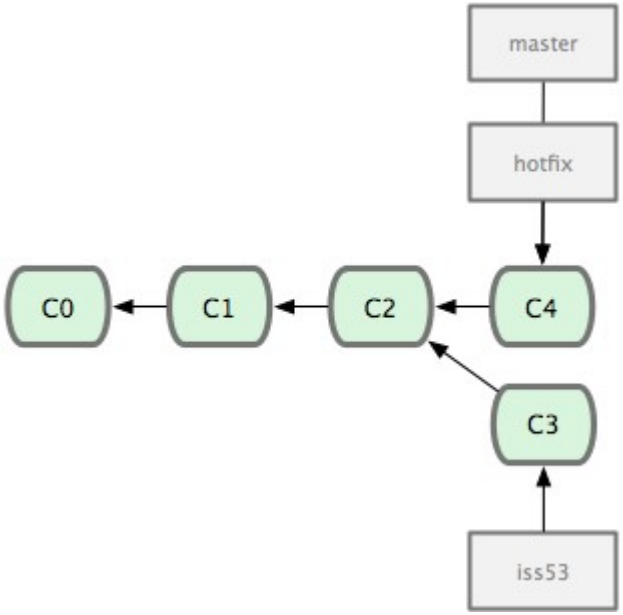


CC BY NC SA

Merge « fast forward »

git checkout master

git merge hotfix



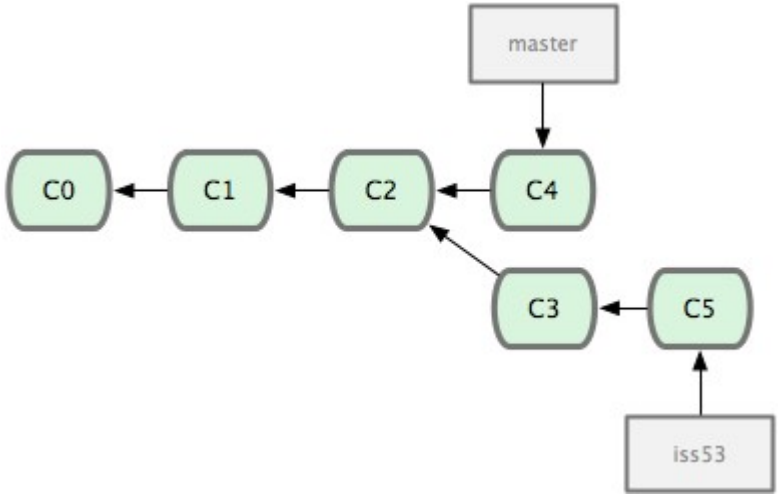
CC BY NC SA

Merge « fast forward »

git branch -d hotfix

git checkout iss53

git commit

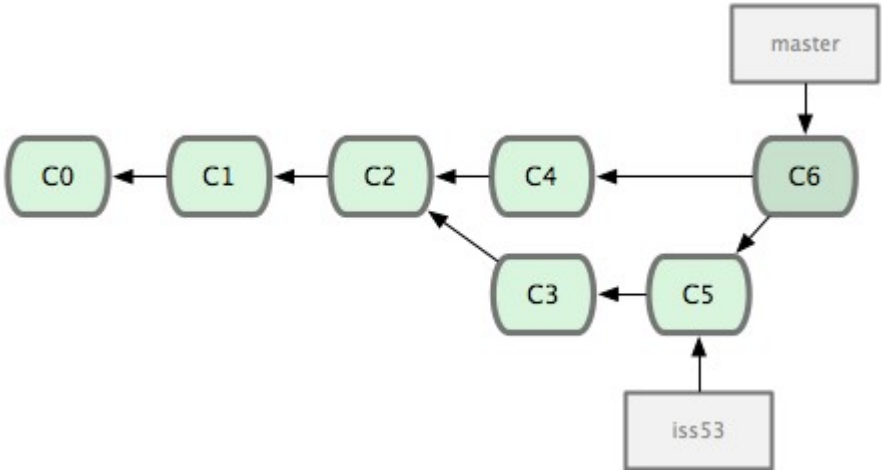


cc BY NC SA

Merge classique

git checkout master

git merge iss53

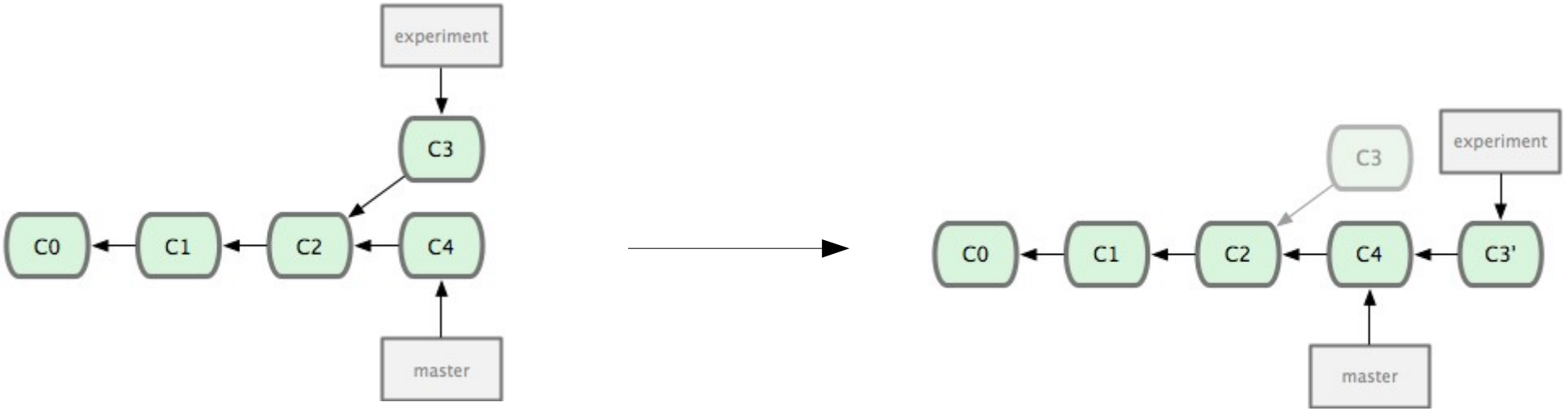


cc BY NC SA

Rebase

git checkout experiment

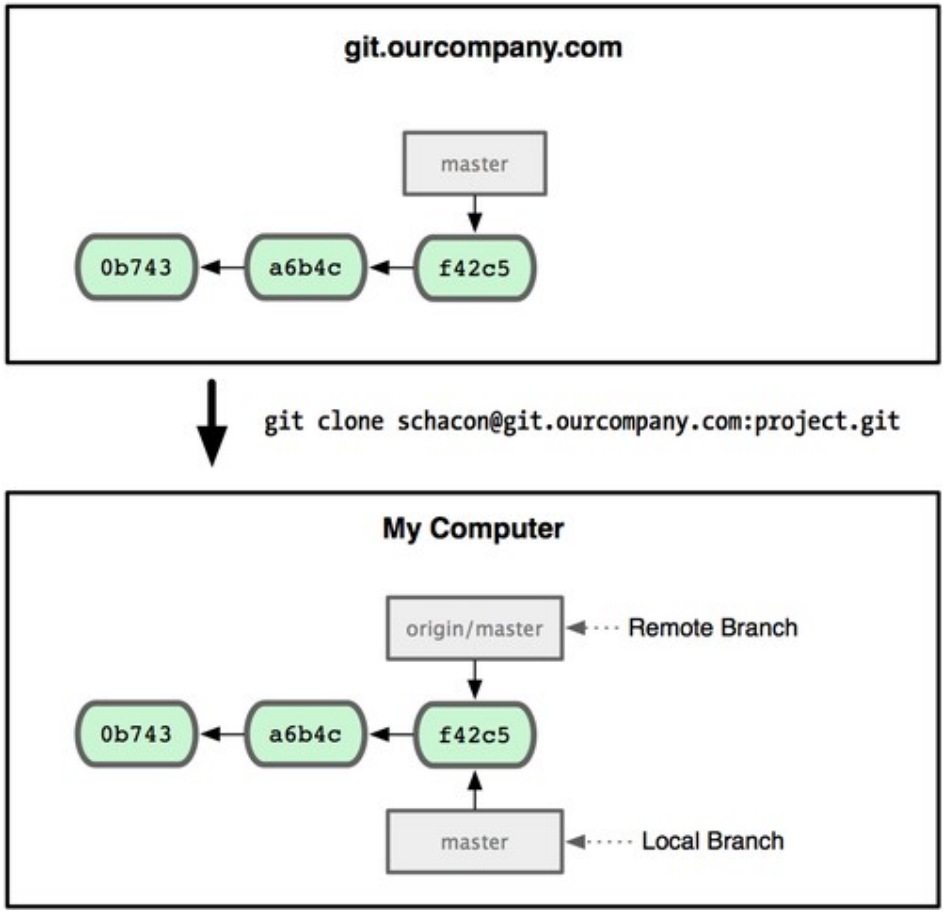
git rebase master



Remotes

Clone

git clone *repository*



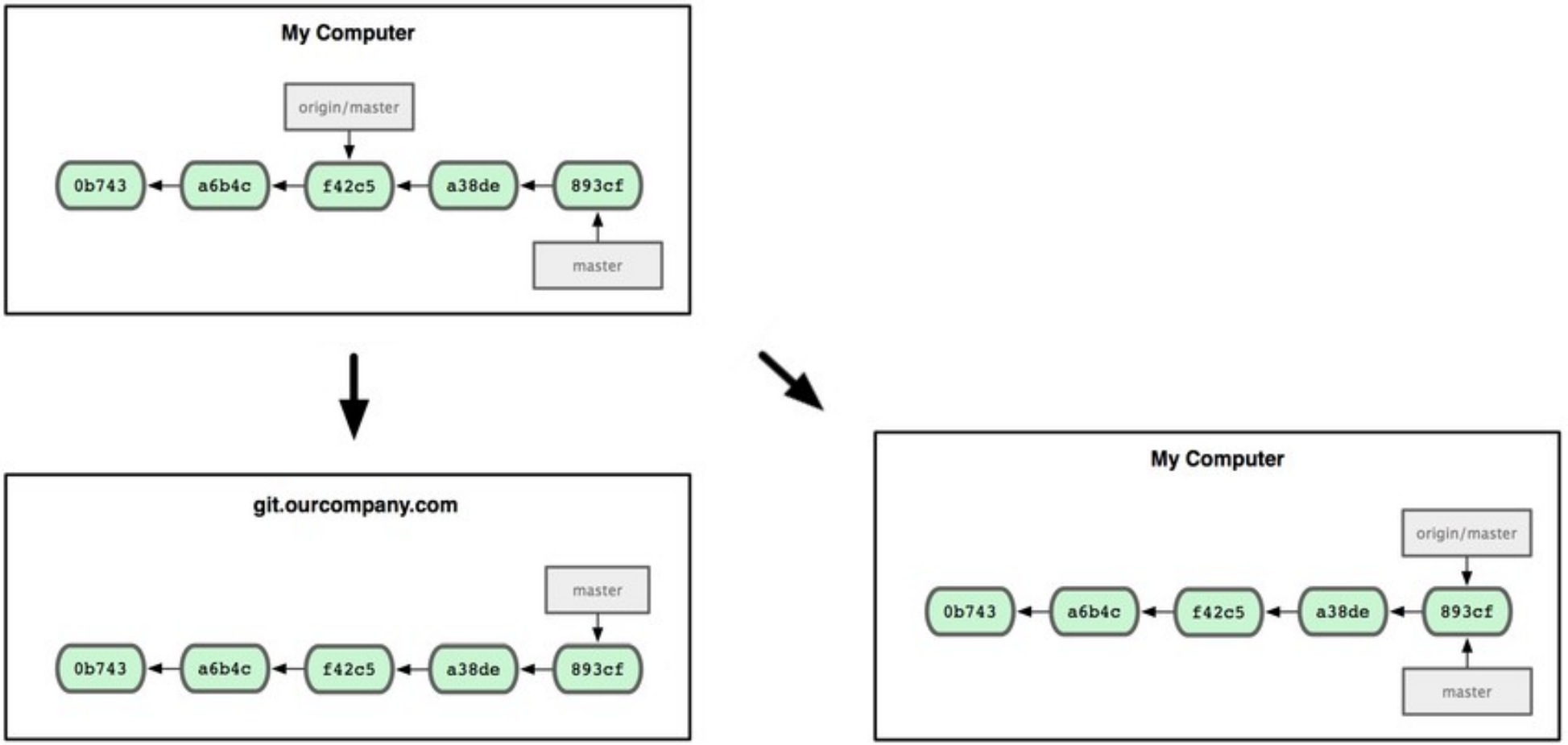
Clone

- Récupération de tous les objets du référentiel distant
- Création d'une branche locale : origin/master
- Pointeur vers la branche master du serveur
- Impossible de déplacer la branche

CC BY NC SA

Push

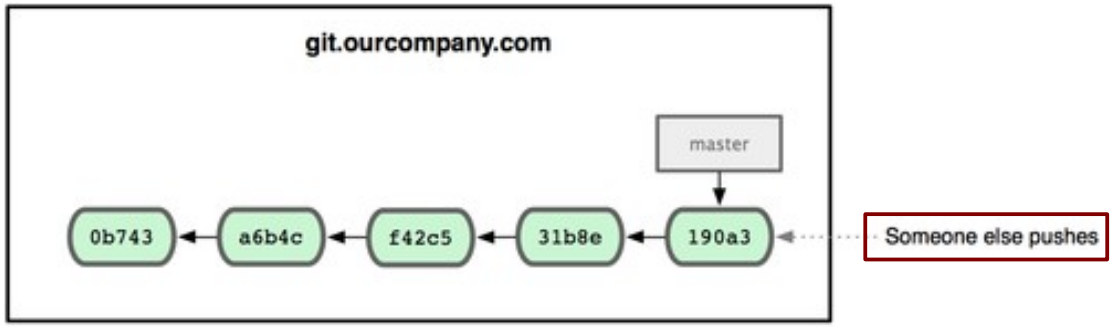
git push *remote branch*

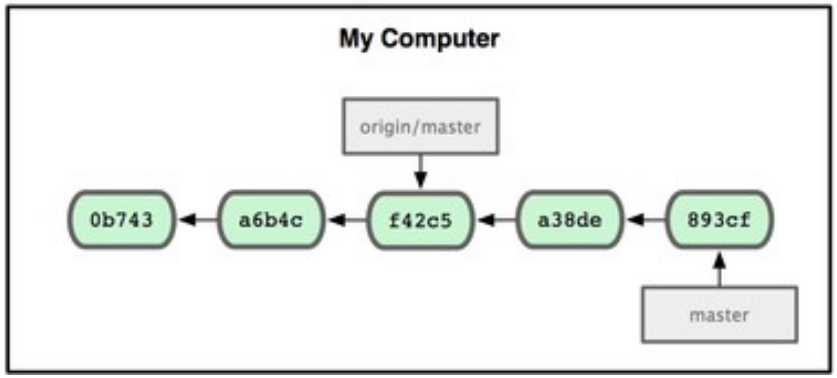


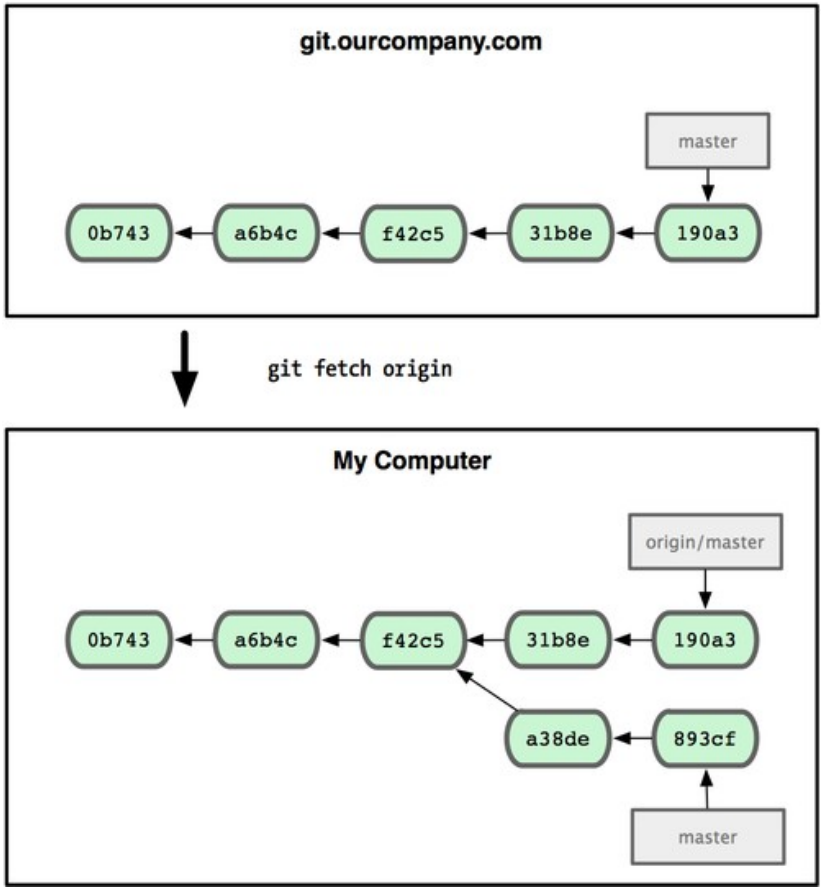
CC BY NC SA

Fetch

git fetch *remote*







Pull

git pull *remote*

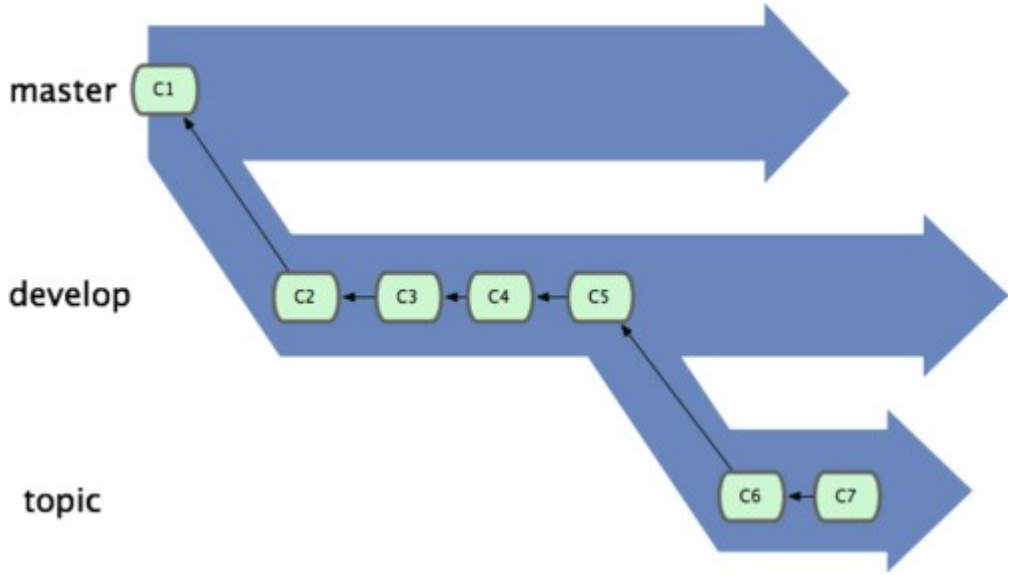
- git fetch + git merge
- utile si aucune modification faite en locale

cc BY NC SA

Workflows

Branches

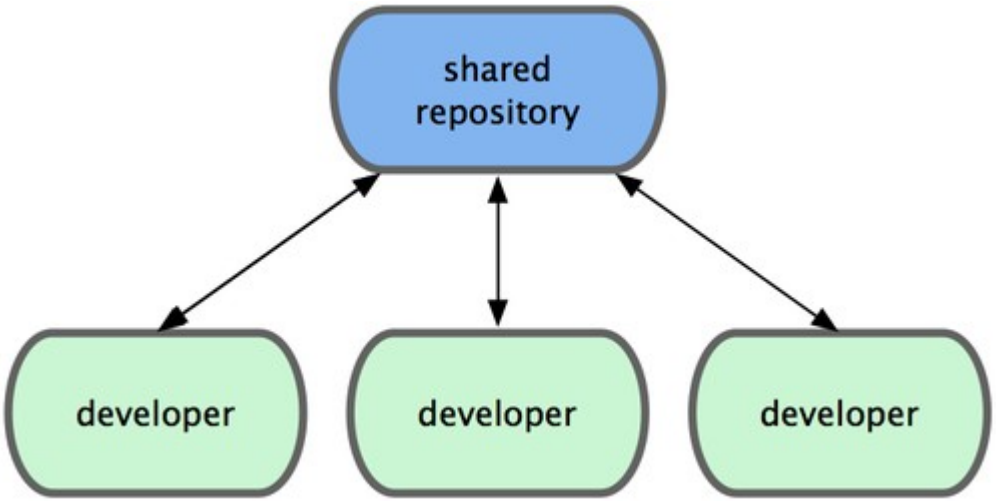
- une branche master stable avec tags
- une branche de développement
- une branche par fonctionnalité



CC BY NC SA

Centralisé

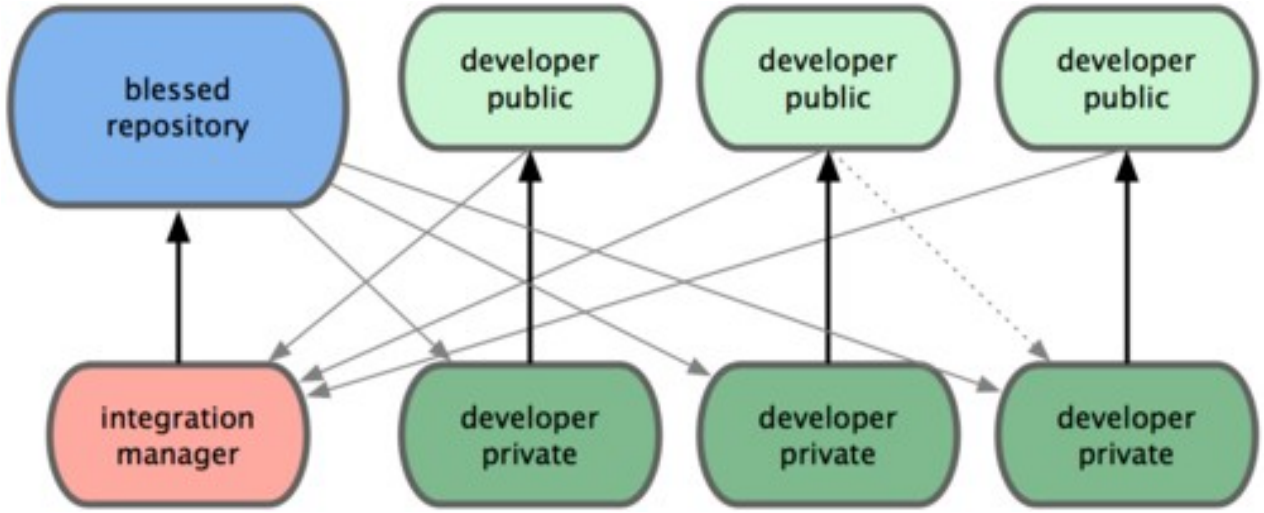
- chaque développeur utilise un référentiel partagé (similaire à SVN)



CC BY NC SA

Integration manager (GitHub)

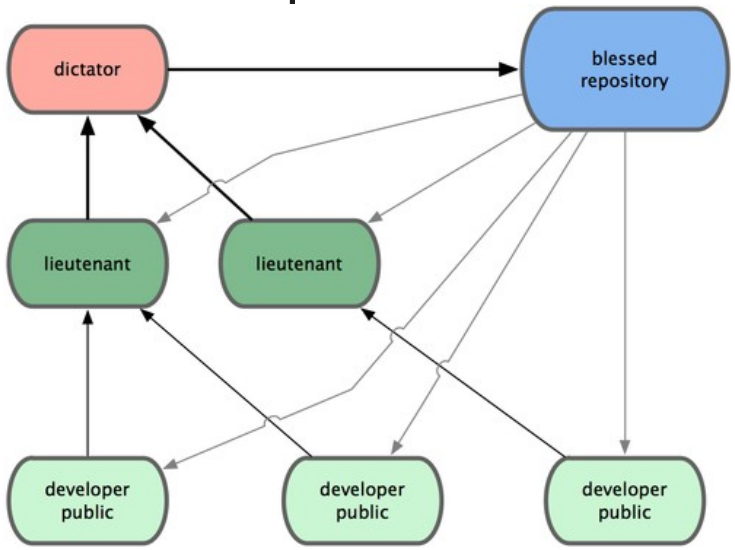
- le mainteneur du projet publie dans un repository de référence
- un contributeur clone ce référentiel et publie ses modifications dans son référentiel public
- le contributeur envoi un e-mail au mainteneur pour qu'il récupère ses changements
- le mainteneur intègre les modifications du contributeur dans son référentiel local
- le mainteneur publie dans le repository de référence



CC BY NC SA

Dictateur et lieutenants (Linux)

- les développeurs travaillent sur leurs branches et se synchronise avec le repo de référence
- les lieutenants mergent le travail des développeurs dans leur branche master
- le dictateur merge les branches master des lieutenants
- le dictateur publie dans le repo de référence



CC BY NC SA

Conclusion

Git, l'essayer c'est l'adopter

- Rapide
- Branche et merge facilité
- Historique clair

- Difficulté de prise en main

CC BY NC SA

Q&A

Crédits

- Images tirées du site : http://git-scm.com/book
- Template inspiré de http://antoniogoncalves.org/

cc BY NC SA