

# Git et Gitlab au quotidien

## Commandes Principales et Workflows

Branch - Checkout - Stash - Reset - Revert - Tag

Benoît Bayol

June 9, 2015

# Outline

Licence

git-branch

git-checkout

git-stash

git-reset

git-reflog

git-revert

git-tag

# Topic

## Licence

git-branch

git-checkout

git-stash

git-reset

git-reflog

git-revert

git-tag

- ▶ Les graphiques présents dans les slides sont extraits du livre "Pro Git 2" qui est aussi présent dans l'archive.
- ▶ La licence de ce document est :  
<https://creativecommons.org/licenses/by-nc-sa/3.0/>

# Topic

Licence

**git-branch**

git-checkout

git-stash

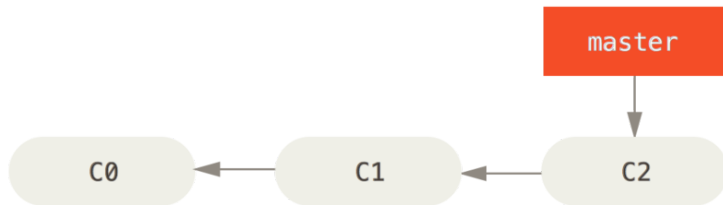
git-reset

git-reflog

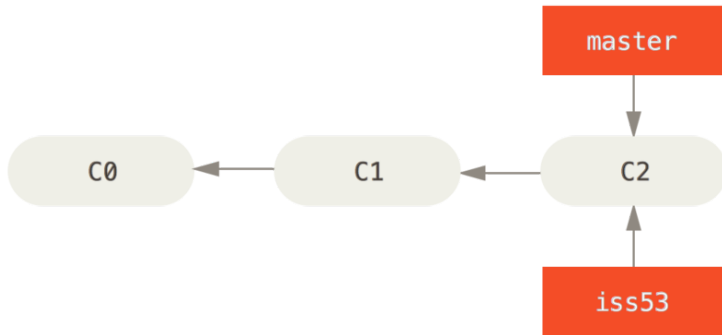
git-revert

git-tag

- ▶ une branche est un pointeur sur un commit
- ▶ cette commande permet de créer une branche mais est souvent surchargée par git checkout -b
- ▶ options intéressantes:
  - ▶ -a affiche toutes les branches
  - ▶ -D supprime une branche

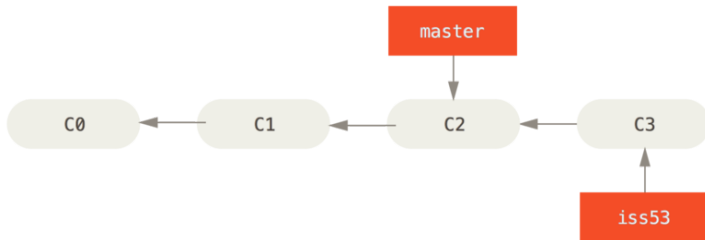


git checkout -b iss53

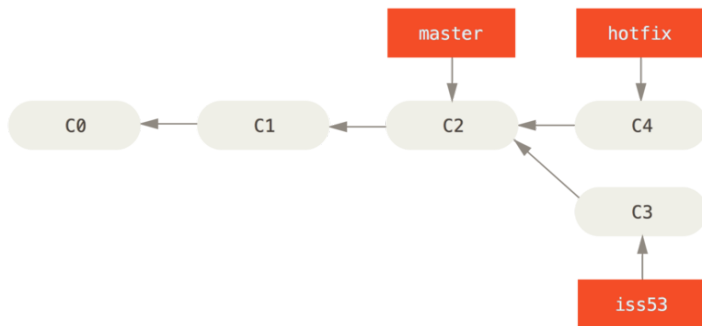




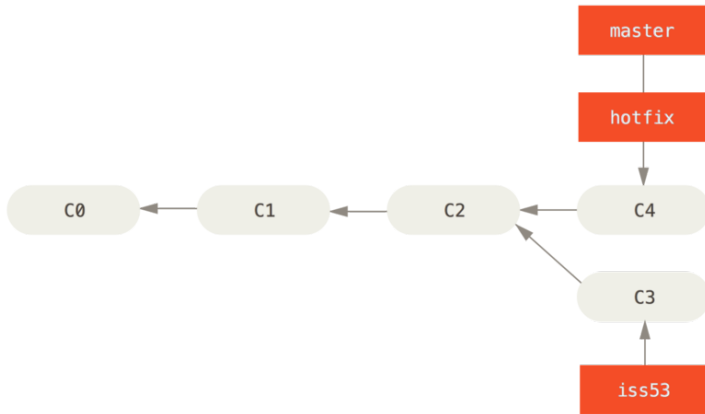
## git commit



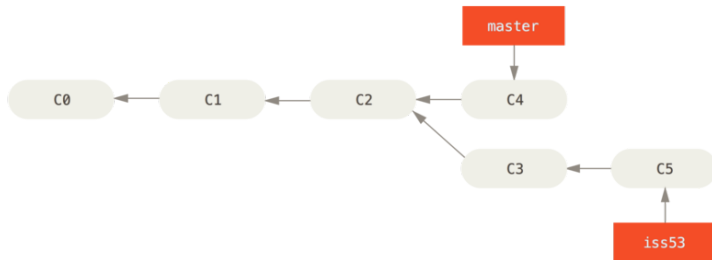
`git checkout master && git checkout -b hotfix && git commit`



git checkout master && git merge hotfix



## git branch -D hotfix



# Topic

Licence

git-branch

**git-checkout**

git-stash

git-reset

git-reflog

git-revert

git-tag

- ▶ permet de changer le HEAD sur un commit en particulier
- ▶ ce commit peut être donné par son sha1 ou alors le nom de la branche
- ▶ si c'est un commit on passe alors en "detached state"
- ▶ sinon on est sur une branche

# Topic

Licence

git-branch

git-checkout

**git-stash**

git-reset

git-reflog

git-revert

git-tag

- ▶ la stash est un lieu où l'on peut mettre temporairement du travail afin de ne pas créer un commit spécialement pour lui
- ▶ exemple
  - ▶ vous travaillez sur une branche
  - ▶ un rapport de bug arrive
  - ▶ vous stashez votre travail et changer pour la master
  - ▶ vous faites une branche pour corriger le bug
  - ▶ vous merger
  - ▶ vous revenez sur votre branche de départ
  - ▶ vous réappliquez la stash



# Topic

Licence

git-branch

git-checkout

git-stash

**git-reset**

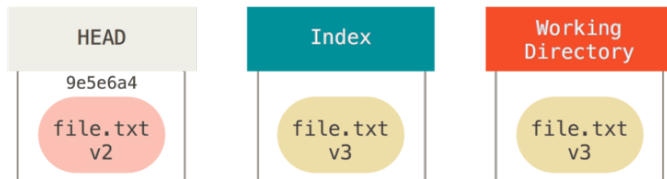
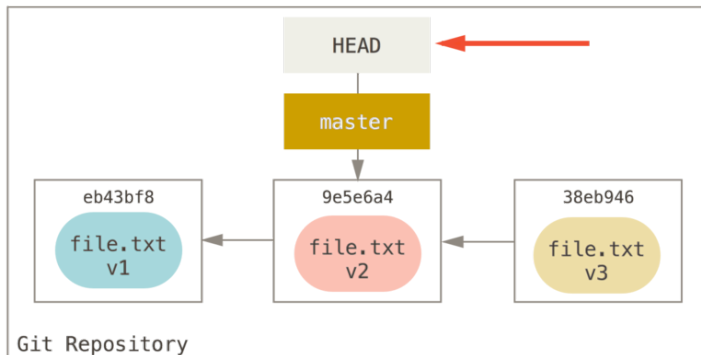
git-reflog

git-revert

git-tag

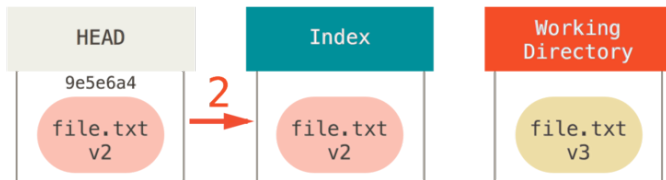
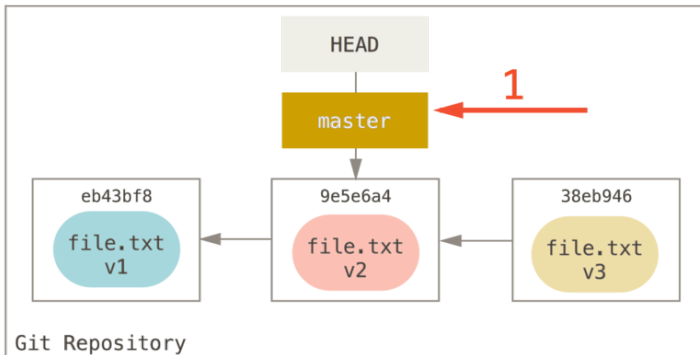
- ▶ permet de remettre la branche en cours sur un commit en particulier
- ▶ reset possède 3 modes : soft, mixed (par défaut), hard
  - ▶ soft : revient à l'état "changes to be committed"
    - ▶ ça peut s'appliquer à une situation "ah mince j'ai oublié d'ajouter ce fichier dans le commit"
  - ▶ mixed : reset l'index (il n'y a plus rien) mais ne touche pas au working tree
    - ▶ ça peut s'appliquer à une situation "mince je viens de compiler et tester ça tourne pas car j'ai oublié de préciser ce petit truc"
  - ▶ hard : reset l'index et le working tree
    - ▶ ça peut s'appliquer à une situation "bon je devais pas être bien réveiller"

git-reset soft



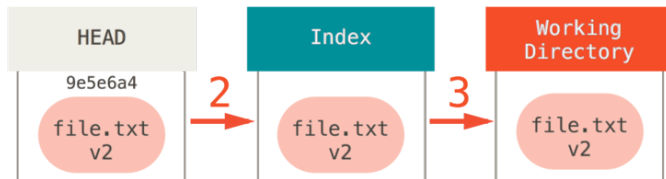
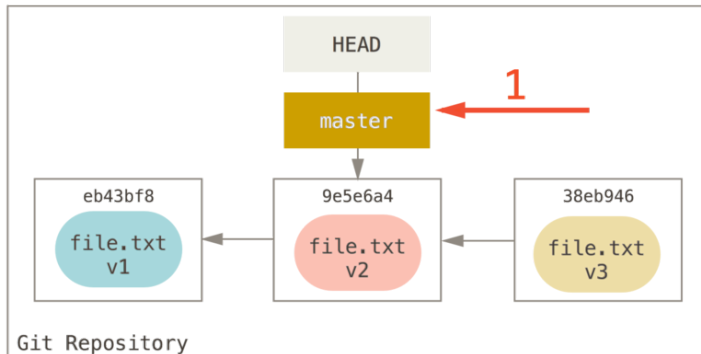
git reset --soft HEAD~

git-reset soft



git reset [--mixed] HEAD~

git-reset soft



git reset --hard HEAD~

# Topic

Licence

git-branch

git-checkout

git-stash

git-reset

**git-reflog**

git-revert

git-tag

- ▶ Garde en mémoire les évènements git
- ▶ Permet par exemple de retrouver un commit perdu à cause d'un git-reset trop vite fait

# Topic

Licence

git-branch

git-checkout

git-stash

git-reset

git-reflog

**git-revert**

git-tag



## git-revert

- ▶ annule un commit en un créant un autre
- ▶ les + deviennent des - et inversement
- ▶ s'applique surtout pour des commits déjà partagés (pusher)

# Topic

Licence

git-branch

git-checkout

git-stash

git-reset

git-reflog

git-revert

**git-tag**

- ▶ permet de tagger une version en particulier avec un nom spécifique
- ▶ cela ne rentre pas dans le namespace des branches
- ▶ idéal pour gérer des versions de type X.Y.Z
- ▶ idéal pour tagger une version d'un logiciel qui a servi à produire un papier scientifique