

# Git et Gitlab au quotidien

## Commandes Principales et Workflows

### Merge - Rebase

Benoît Bayol

June 9, 2015

# Outline

Licence

git-merge

git-cherry-pick

git-rebase

# Topic

## Licence

git-merge

git-cherry-pick

git-rebase

- ▶ 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-merge**

git-cherry-pick

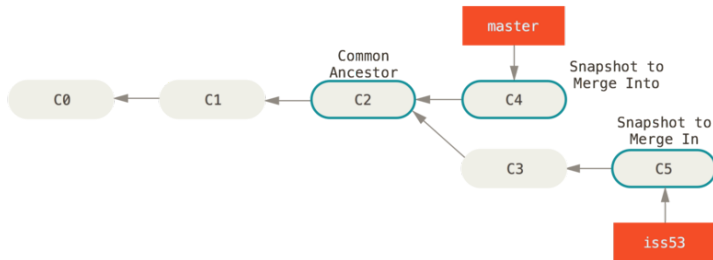
git-rebase

- ▶ un merge se fait sur une branche et la fusionne avec une autre en ajoutant un commit de merge
- ▶ de temps en temps ce merge sera fast-forward (sans commit de merge) car la branche hôte n'a pas évolué depuis la bifurcation
- ▶ c'est le mécanisme le plus répandu
- ▶ c'est sécurisé comme méthode
- ▶ la seule contrainte c'est les conflits mais c'est normal :)

# Précautions

- ▶ Un merge c'est prendre le travail d'une branche et le fusionner avec la branche en cours
- ▶ Merger quand tout est bien validé dans les différentes branches.
- ▶ Ne pas laisser des choses en état modifié en attente cela peut causer des ennuis difficiles à réparer

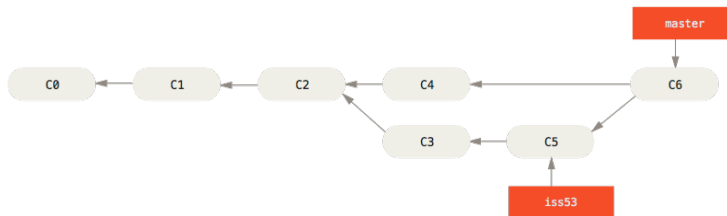
## Avant le merge





# Après le merge

```
1 git checkout master
2 git merge iss53
```



# Cas particulier

- ▶ evil merge introduire des changements qui n'ont aucun parent (ex: ajout de modifications alors que `--no-commit`)
- ▶ octopus plusieurs merge en même temps
- ▶ conflits
  - ▶ 1:file ancetre commun
  - ▶ 2:file nos modifications
  - ▶ 3:file modifications de la branche mergée
- abort
- ▶ fast-forward
  - ▶ ne produit pas de commit de tête car l'ancetre commun est toujours le HEAD de la branche qui attend la branche à merger.

# Options intéressantes

- ▶ `-dry-run` simulation
- ▶ `-abort` annule merge en cours si ce n'est pas un fast-forward (pratique en cas de conflit)
- ▶ `-squash` squash tous les commits de la branche en un
- ▶ `-no-commit` ne crée pas le commit de merge
- ▶ `-no-ff` n'accepte pas le fast-forward automatiquement

# Conflicts

- ▶ `git show 1:file`
- ▶ `git show 2:file`
- ▶ `git show 3:file`
- ▶ `git checkout --theirs file`
- ▶ `git checkout --ours file`
- ▶ `git add & git commit`

# Topic

Licence

git-merge

git-cherry-pick

git-rebase

## git cherry-pick

- ▶ le cherry-pick est une sorte de mini-merge puisque l'on descend au niveau du commit
- ▶ on peut donc prendre un commit de n'importe quelle branche et le réappliquer sur une autre
- ▶ il faut que le commit soit "cohérent" bien évidemment

# Topic

Licence

git-merge

git-cherry-pick

git-rebase

- ▶ le rebase permet de prendre une branche et de la dérouler à partir d'un point donné plus haut
- ▶ cela permet notamment de faire des fast-forward plus facilement et donc d'éviter les commits de merge
- ▶ cela rend plus lisible l'historique MAIS cela a des inconvénients
- ▶ ON NE REBASE JAMAIS UN CODE DÉJÀ PUSHÉ
- ▶ tous les sha1 de la branche changent dans un rebase
- ▶ s'il y a des conflits on les corrige au fur et à mesure du rebase
- ▶ options intéressantes
  - ▶ -abort annule le rebase
  - ▶ -continue continue le rebase après résolution des conflits
  - ▶ -i permet de faire un rebase interactif