# Les branches

Romain THERRAT

POCKOST

14 décembre 2020

# Sommaire

1 Fusion

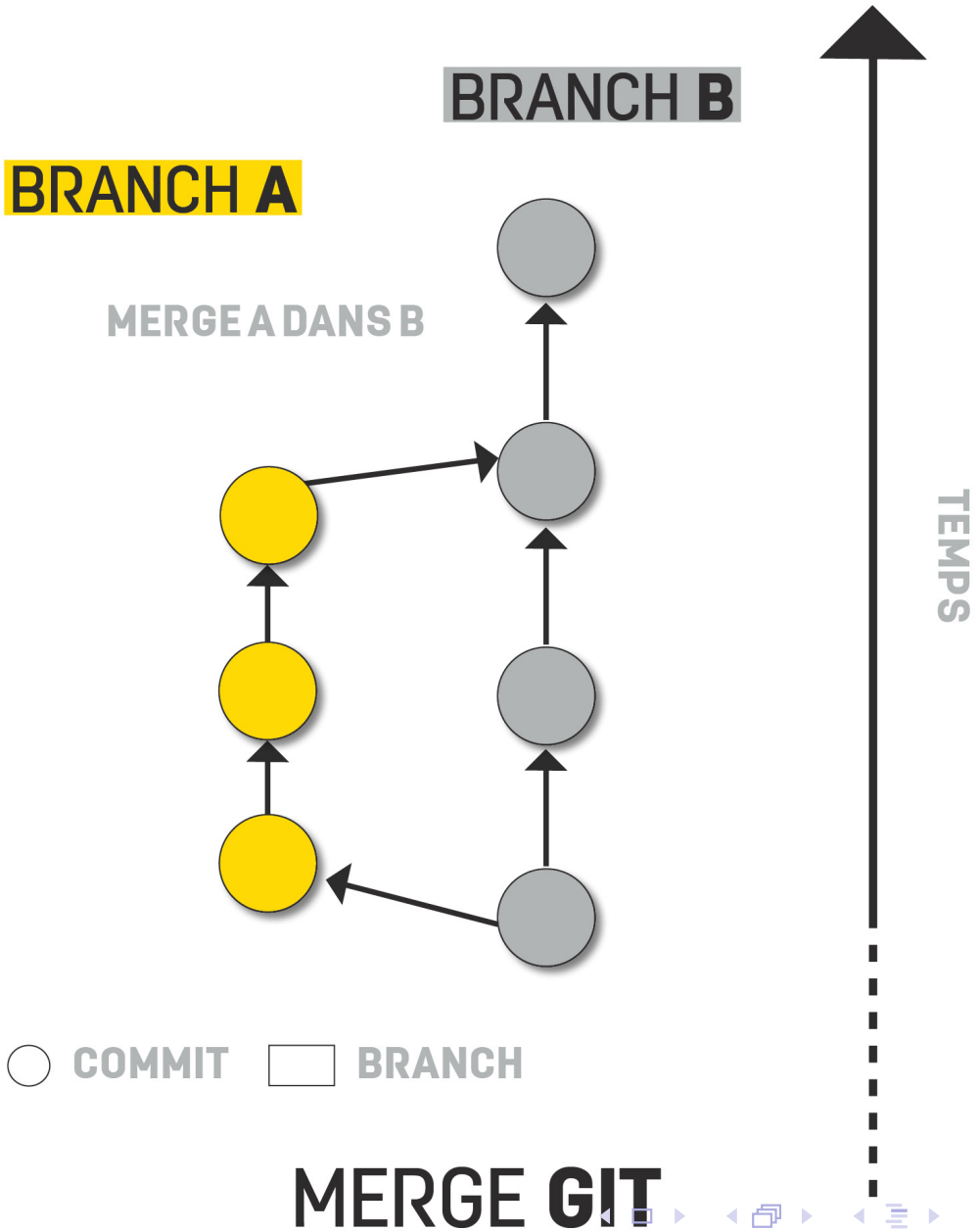
2 Rebase

3 Labs

# Merge et rebase

- Les branches peuvent être
  - fusionnées (`merge`)
  - rebasés (`rebase`)
- La fusion est utile pour
  - Intégrer une nouvelle fonctionnalité en préprod
  - Passer la préprod en production
  - Appliquer un correctif urgent en production
  - ...

# Merge


BRANCH B
BRANCH A
MERGE A DANS B
TEMPS
COMMIT
BRANCH
MERGE GIT

# git merge

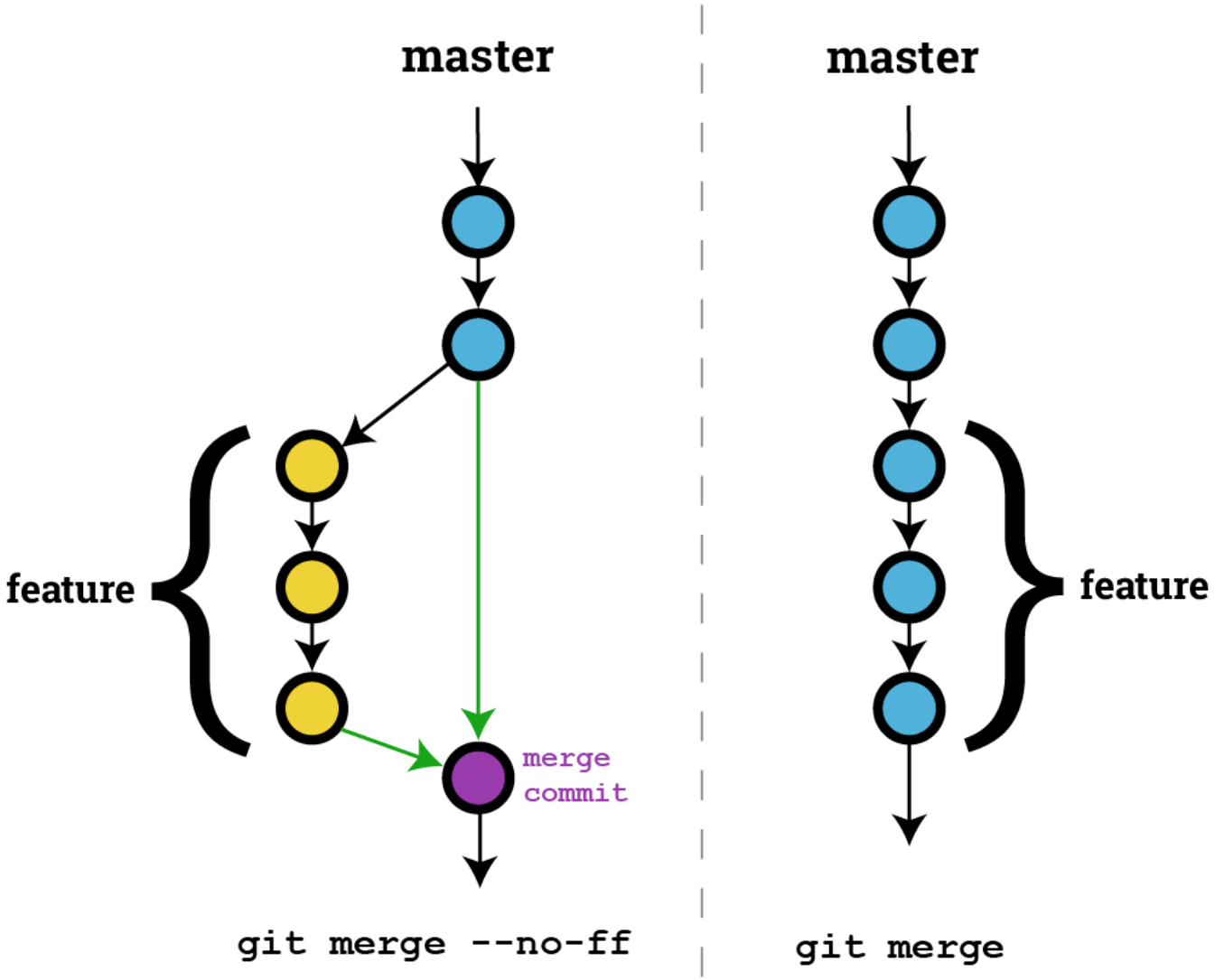
```
    $ git branch
  dev
* master
  testing
$ git merge dev
Merge made by the 'recursive' strategy.
 new-file-dev.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 new-file-dev.txt
```

# Type de merge

Il existe deux type de merge

- Avec un `fast-forward` (défaut)
- Appliquer tous les changements sur la branche de destination
- Sans `fast-forward`
- Créer un commit avec toutes les modifications

# Type de merge



# Sommaire

1 Fusion

2 Rebase

3 Labs

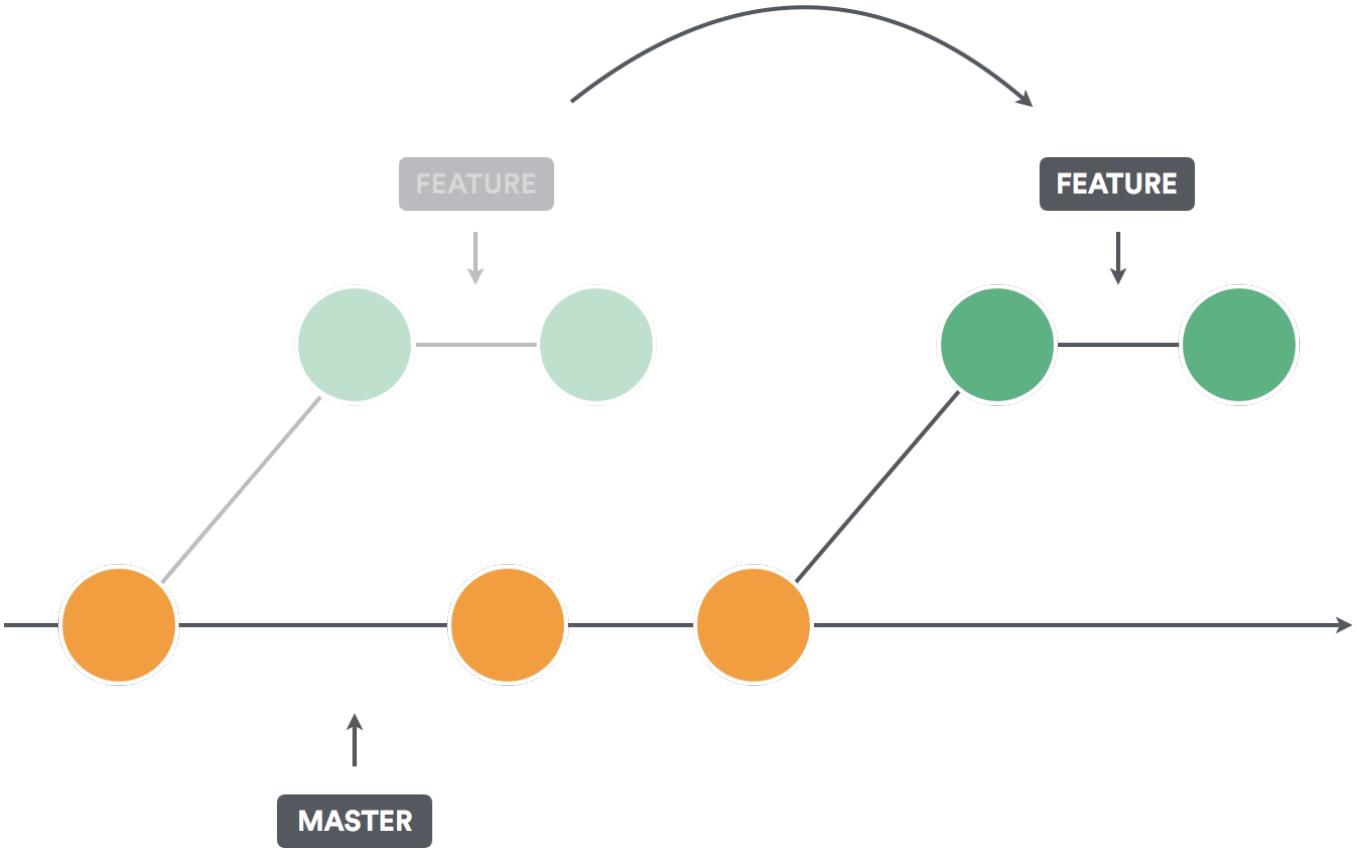
# Rebase

Pourquoi utiliser le rebase ?

- la branche de base (`master`?) a évolué
- Récupérer une nouvelle fonctionnalité

L'ensemble de nos commits vont être réapliqués

# Rebase



# Rebase

```
$ git branch |tee
* dev
  master
  testing
$ git rebase master
    First, rewinding head to replay your work on top of
    Applying: Some comment (dev)
```

# Sommaire

1 Fusion

2 Rebase

3 Labs

# Labs

Nous allons reprendre notre exercice précédent pour merger nos branches. Nous avons actuellement deux branches :

- master
- develop

Le fichier `contact.php` n'est présent que sur la branche develop

- Merger votre branche develop dans master
- Constater dans l'historique que le commit ajoutant `contact.php` a été réintégré
- Observer l'arbre sous `gitk`

# Labs

Nous allons maintenant créer deux nouvelles branche pour les faire évoluer indépendamment. L'objectif étant de pouvoir utiliser un `rebase`

- Créer une branche `feature-about` basée sur master
- Créer une branche `feature-contact` basée sur master
- Placer vous dans la branche `feature-about` et ajouter un fichier `about.php`
- Placez-vous maintenant dans la branche `feature-contact` et ajouter quelques lignes au fichier `contact.php`
- Constater que vos commits ne sont présent que dans les branches où ils ont été créés
- Merger `feature-about` dans master
- Constater que le fichier `about.php` n'est présent que dans `master` et pas dans `feature-contact`
- Noter le numéro du dernier commit de `feature-contact`
- Utiliser un `rebase` pour rebaser la branche `feature-contact` sur `master`
- Comparer le numéro du dernier commit de `feature-contact`

# Labs

Nous allons refaire le même exercice mais en utilisant l'option `--no-ff`.

- Créer deux branches `feature-design` et `feature-logo` basées sur `master`
- Faite des modifications dans ces deux brancher
- Merger la branche `feature-design` dans master avec l'option `--no-ff`
- Constater que le numéro de commit de votre modification est le même et qu'un commit de merge a été créé
- Faire un `rebase` de `master` dans `feature-logo`

# Labs

Est ce que je vous ai déjà parlé de ▸ git-school.github.io/visualizing-git/ ?