UNIVERSITÉ RENNES

GIT – commandes avancées

Branches, Merge, Conflits

https://rtavenar.github.io/tuto-git

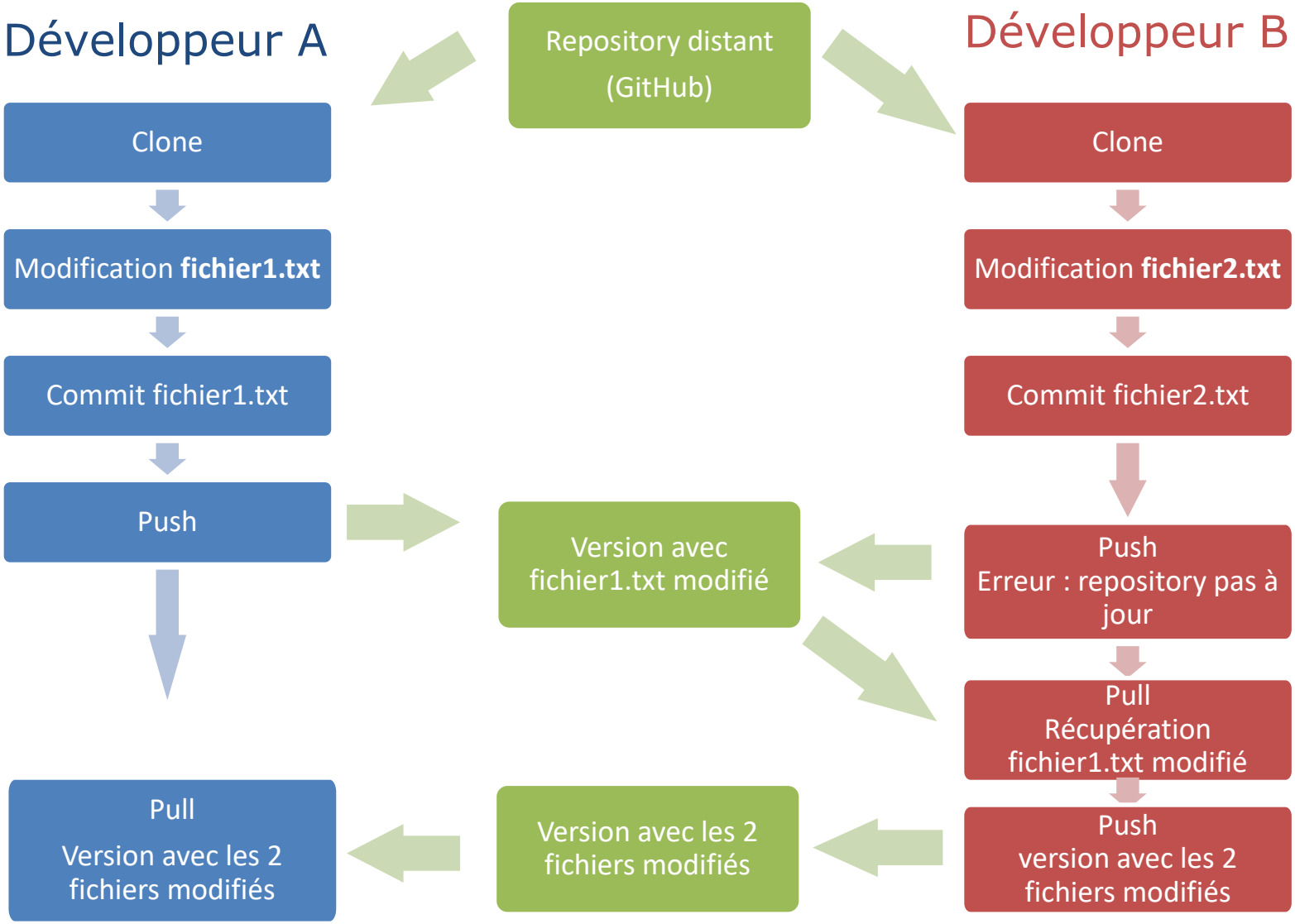
Dans l’épisode précédent…

- Utilisation de git pour un usage local
- Principales commandes
  - git init (initialiser un depot local vierge)
  - git clone (copier des sources distantes)
  - git commit (enregistrer des changements locaux)
  - git push (envoyer les modifications sur le depot distant)
  - git pull (récupérer les modifications du dépot distant)

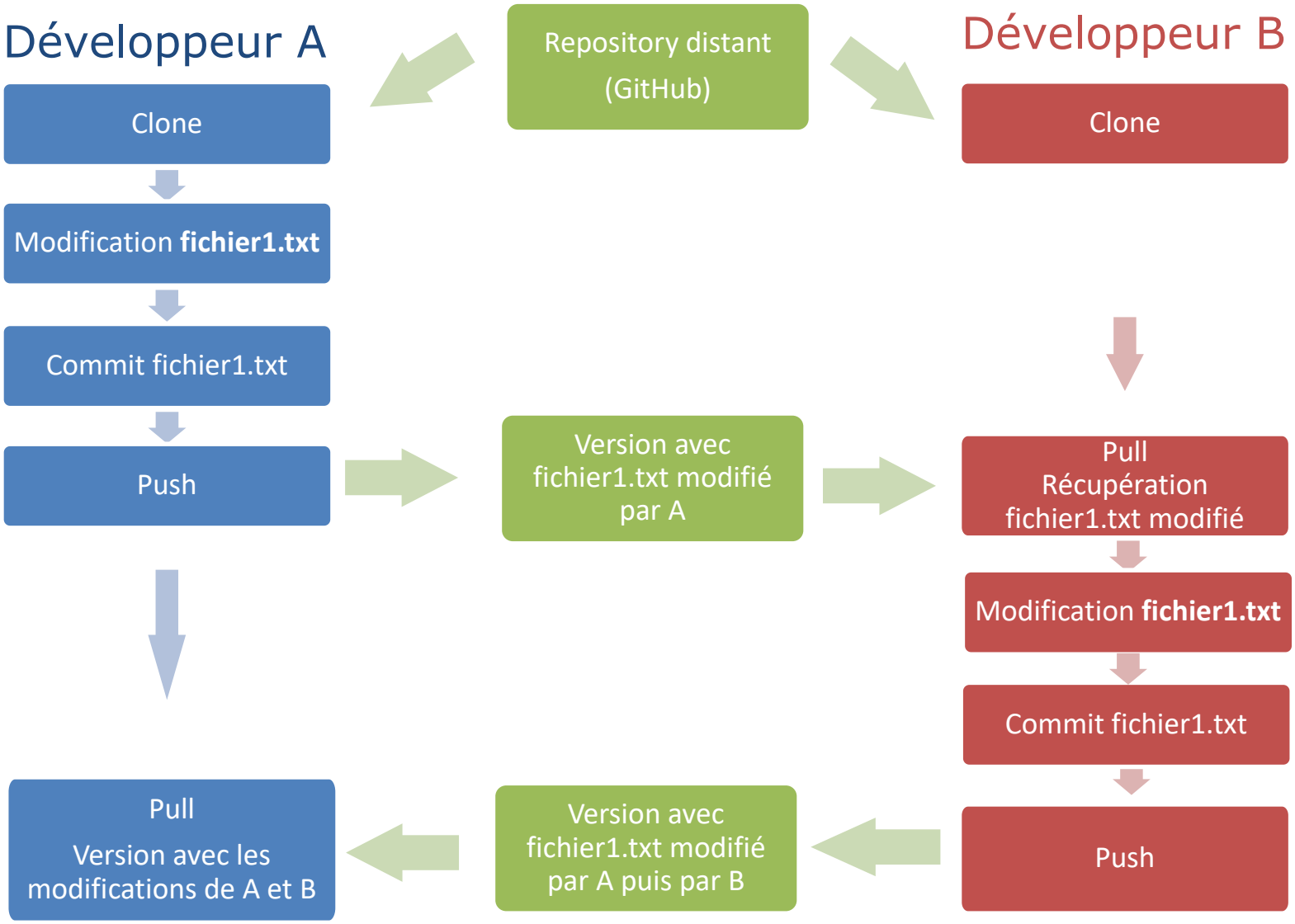
Scénario 2

- Travail collaboratif sur un même projet
  - Travail sur des fichiers différents
  - Travail sur des fichiers identiques successivement
  - Travail sur un même fichier sans conflit
  - Travail sur un même fichier avec conflit
- Travail avec des branches
  - Objectif : ne pas casser la branche principale

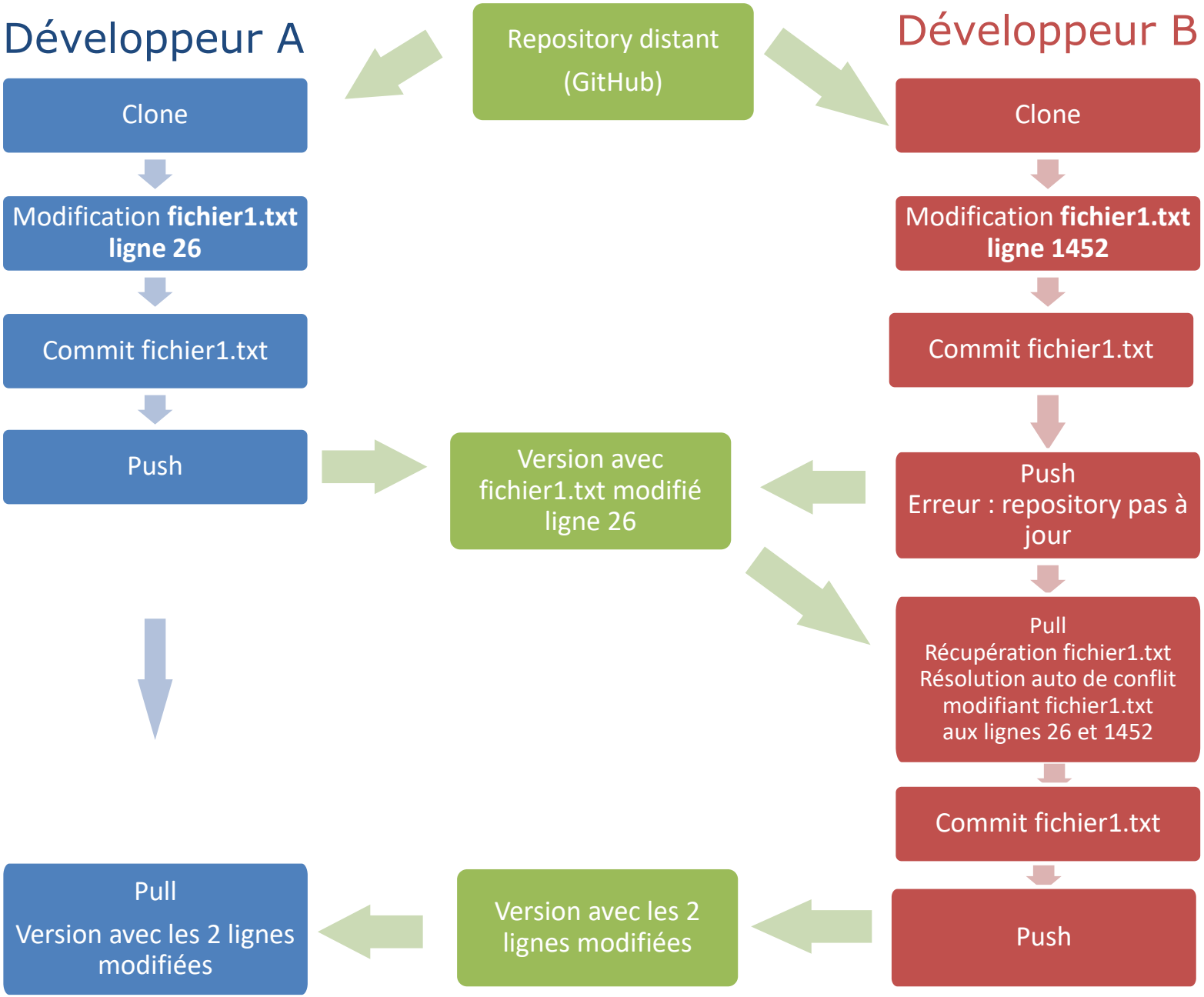
Travail sur des fichiers différents



Travail successif sur des fichiers identiques



Travail conjoint sur un même fichier sans conflit



Exemple de fusion sans conflit

git pull appelle implicitement git merge

Version initiale du fichier

La première ligne
La seconde ligne
La troisième ligne

Version distante du fichier

La première ligne
La seconde ligne
La troisième ligne en mieux

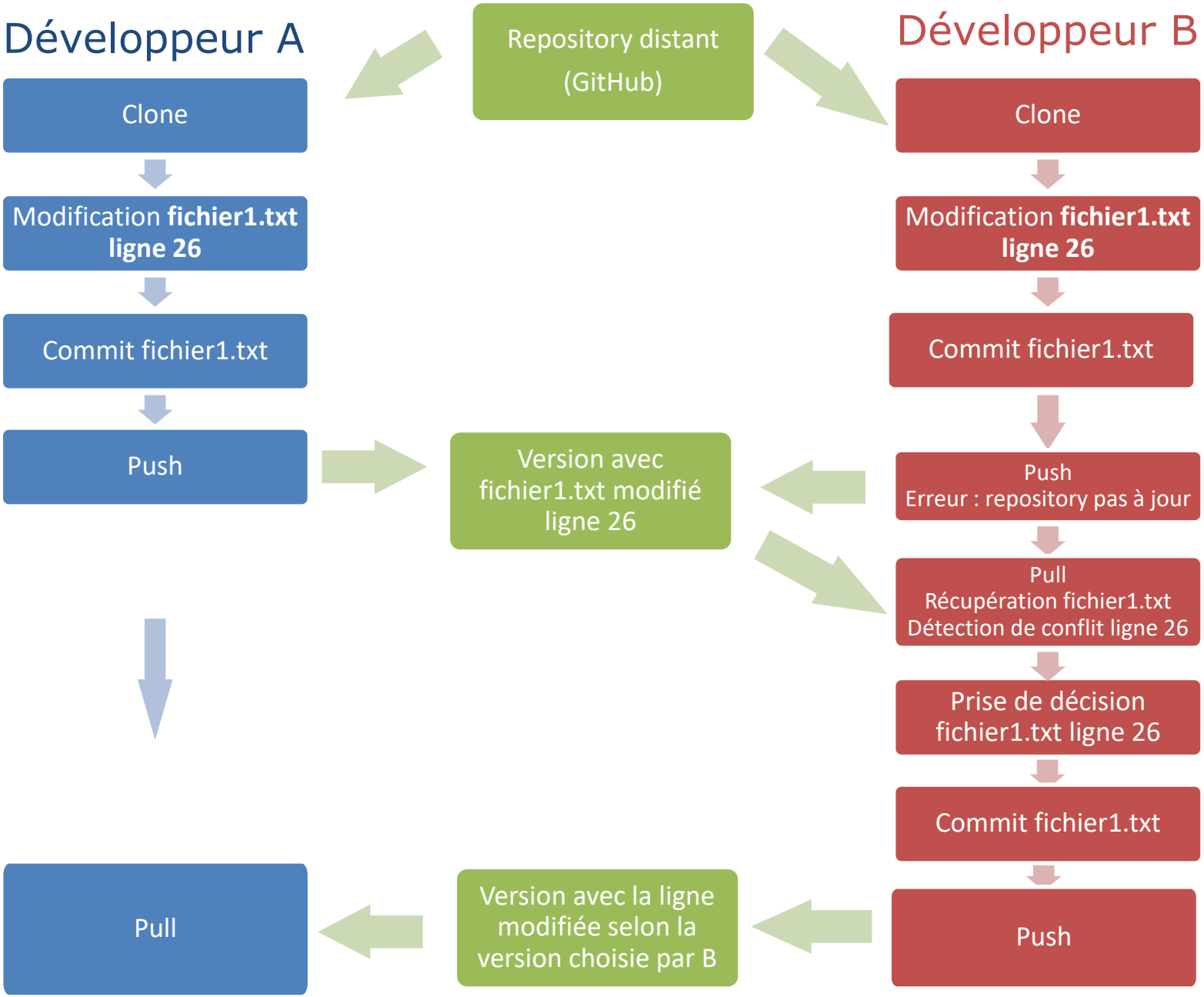
Version locale du fichier

La première ligne en mieux
La seconde ligne
La troisième ligne

Version fusionnée par Git

La première ligne en mieux
La seconde ligne
La troisième ligne en mieux

Travail conjoint sur un même fichier avec conflit



Exemple de fusion avec conflit

Version initiale du fichier

La première ligne
La seconde ligne
La troisième ligne

Version distante du fichier

La première ligne
La seconde ligne en mieux
La troisième ligne

Version locale du fichier

La première ligne
La seconde ligne en beaucoup mieux
La troisième ligne

Version fusionnée par Git

La première ligne
<<<<<<< HEAD
La seconde ligne en beaucoup mieux
=======
La seconde ligne en mieux
>>>>>>> REPOSITORY
La troisième ligne

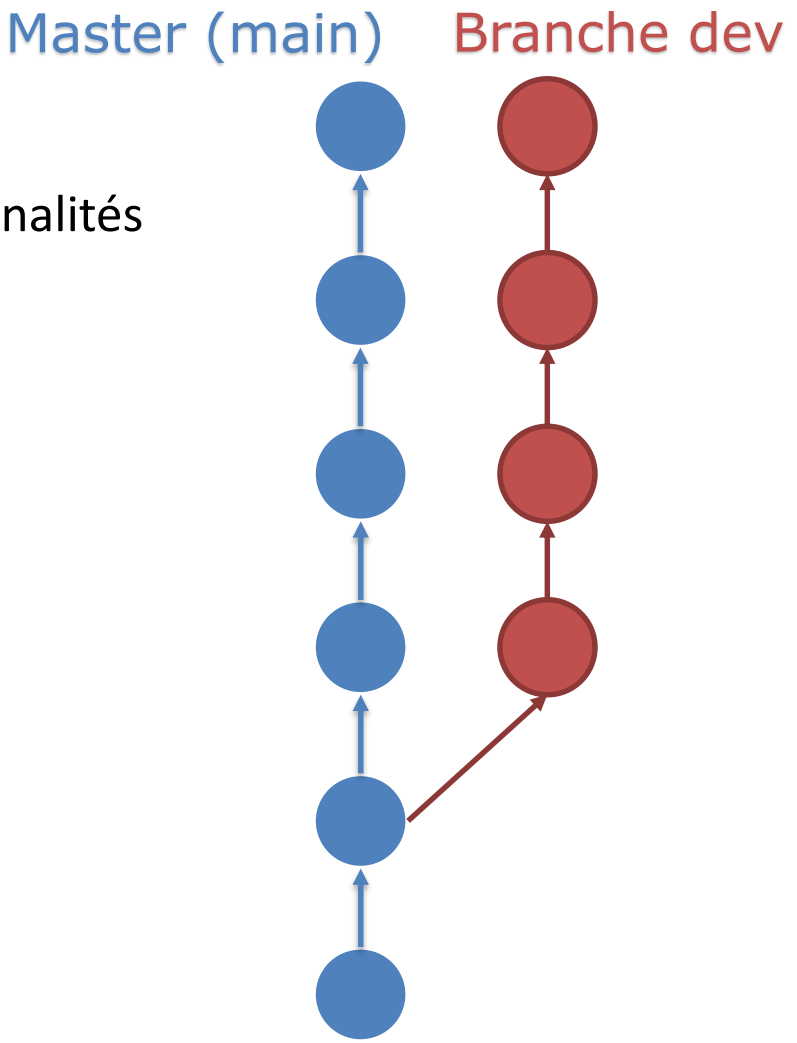
Conflits repérables grâce aux chevrons

Travail avec des branches

- Intérêt
  - Générer des variantes de codes
  - Mettre au point des nouvelles fonctionnalités

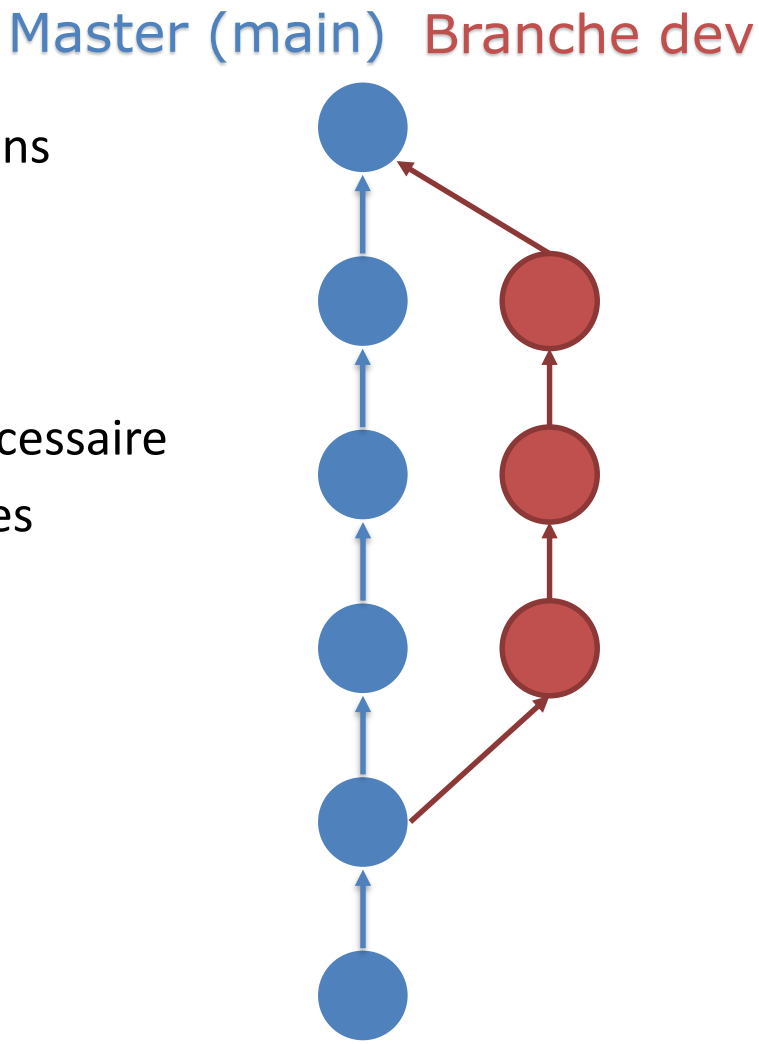
  Sans perdre la branche principale !

  => Plusieurs versions du code coexistent



Branches

- Branche = lignée généalogique des modifications
- Branche principale sous Git :
  - branche ***main***
  - ou ***master***
- Possibilité de créer autant de branches que nécessaire
- Possibilité de fusionner des branches entre elles



Gérer des variantes

Exemple : CV

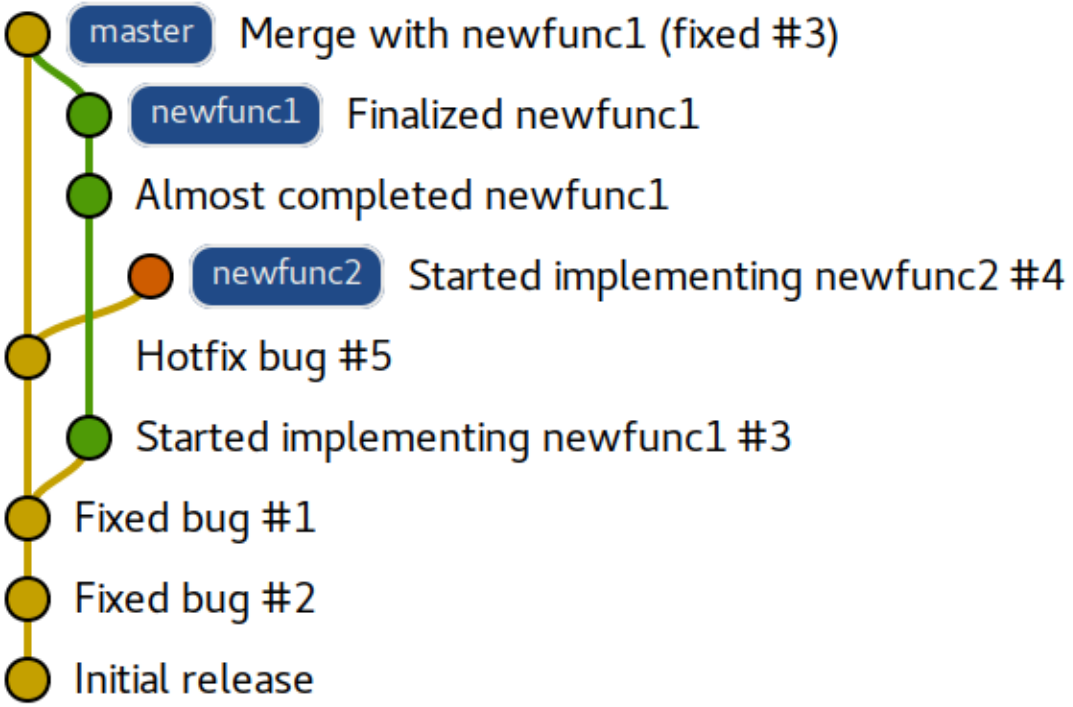
- Une version principale maintenue à jour
- Des variantes adaptées pour chaque demande d'emploi

https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/

Mettre au point des fonctionnalités

Exemple : logiciel

- Une version stable, avec correction de bugs
- Une ou plusieurs versions expérimentales
  - Implémentation de nouvelles fonctionnalités
  - Intégration à la version stable après finalisation



Principales commandes

- Créer une nouvelle branche
  - git branch <nom_nouvelle_branche>

  Attention : ne se place pas sur la branche créée

- Créer une nouvelle branche et s'y placer
  - git checkout –b <nom_nouvelle_branche>

Principales commandes

- Visualiser les branches
  - git branch
- Changer de branche
  - git checkout <nom_branche>
- Fusionner deux branches
  - Se placer dans la branche destination
  - git merge <nom_branche_a_importer>

SOS

- J’ai créé une branche par erreur
  - git branch –d <branche_a_supprimer>
- J’ai fait des modifications sur une mauvaise branche
  - Remiser les changements git stash
  - Créer une branche et se mettre dessus
  - Ré-appliquer les changements git stash apply
- Le message de mon commit est erroné
  - git commit --amend -m "Votre nouveau message de commit"