Projet de Programmation

Benoit Donnet

Année Académique 2023 - 2024

Agenda

Partie 2: Outils

Agenda

- Chapitre 6: Gestion des Versions
  - Introduction
  - Partage d'un Même Fichier
  - Concepts du SCM
  - GIT

Agenda

- Chapitre 6: Gestion des Versions
  - Introduction
    - ✓ Témoignages
    - ✓ Utilité d'un SCM
    - ✓ Principe de Base
  - Partage d'un Même Fichier
  - Concepts du SCM
  - GIT

Témoignages

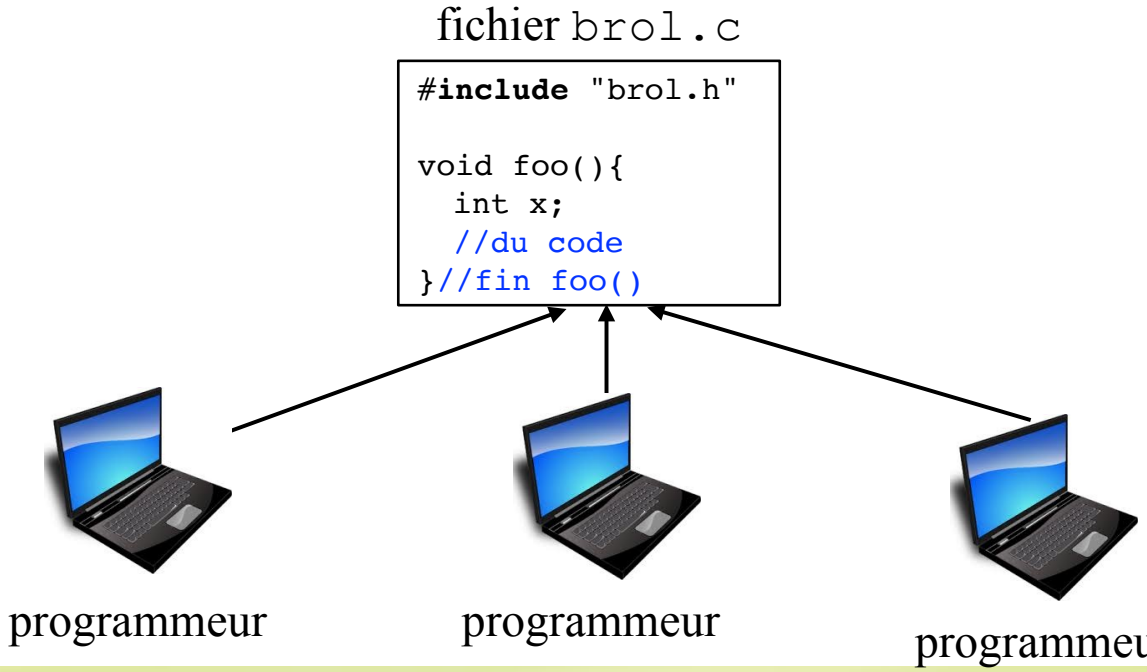
- C’est pas de chance…
  - … mon projet est sur mon portable… qui est tombé
  - … c’est mon binôme qui a le projet sur son portable… il est malade
  - … mon projet fonctionnait bien et j’ai essayé d’implémenter une autre fonction et plus rien ne fonctionne

Témoignages (2)

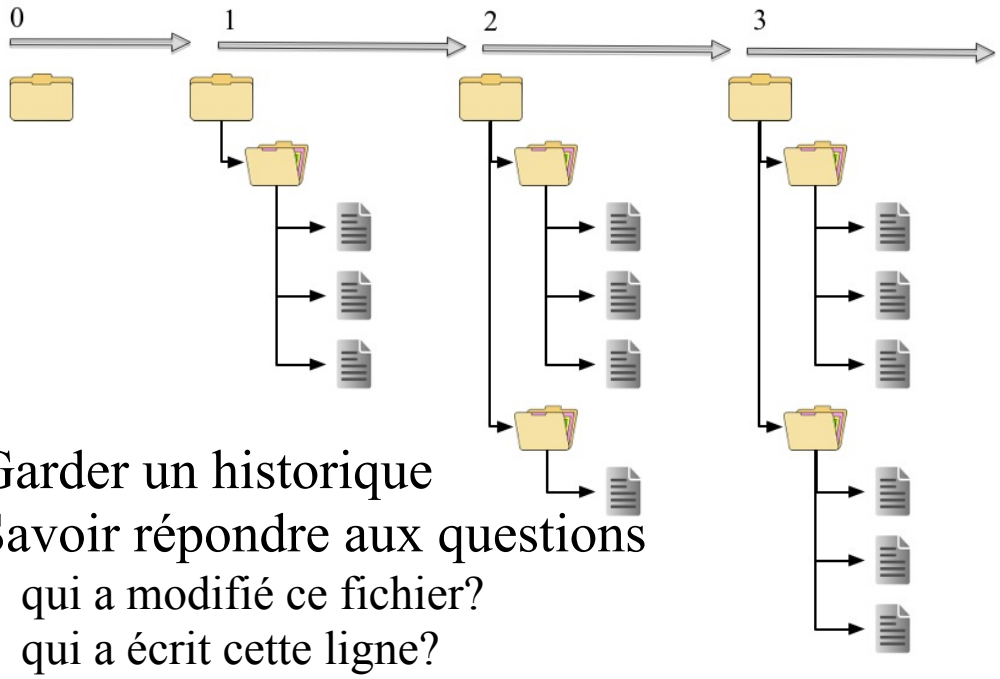
- Travailler à plusieurs, c’est pas facile…
  - ✓ … on s’échange le projet par mail/dropbox et on travaille à tour de rôle dessus. On avance pas vite
  - ✓ … pour que ça compile, j’ai dû écraser les modifications faites par mon binôme… maintenant, il est fâché
  - ✓ … on a travaillé chacun dans notre coin et on n'arrive pas à fusionner notre travail… rien ne fonctionne à la fin

Utilité

1. Travailler à plusieurs



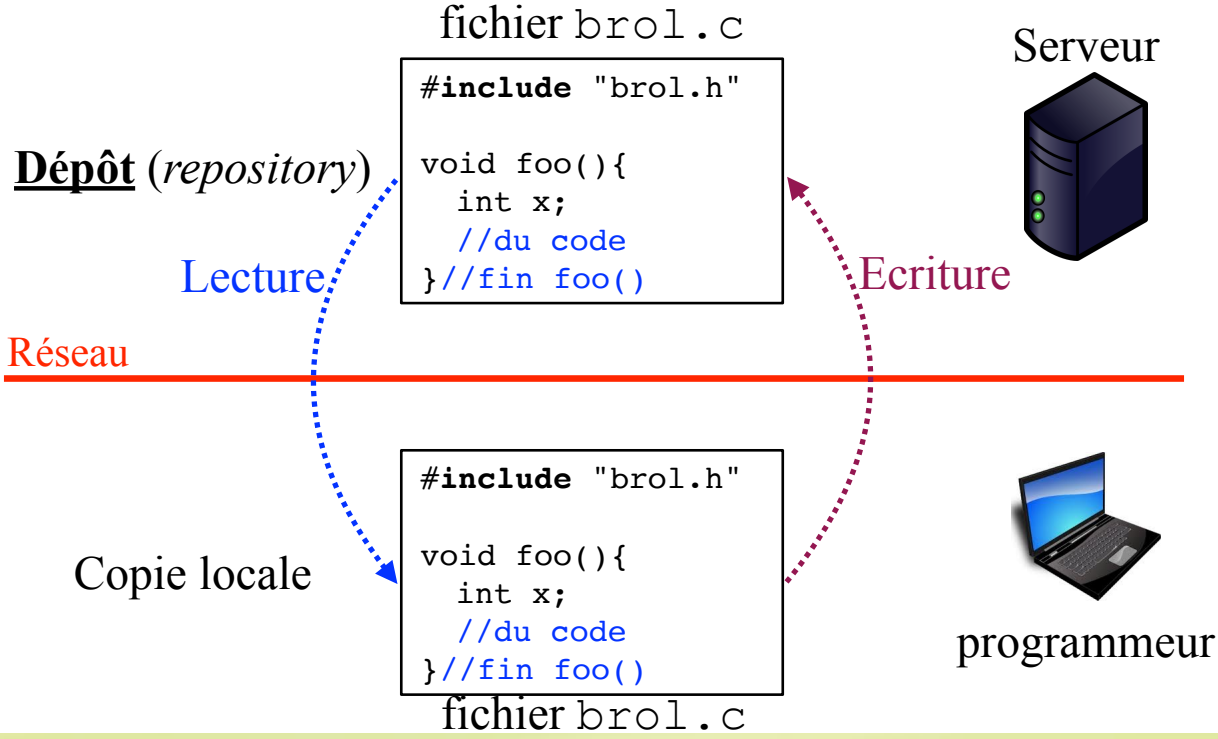
Utilité (2)



2. Garder un historique
3. Savoir répondre aux questions
   - qui a modifié ce fichier?
   - qui a écrit cette ligne?
   - quelle était la version précédente de ce fichier?
   - quels fichiers avait-on le 31 janvier 2015?

Principe de Base

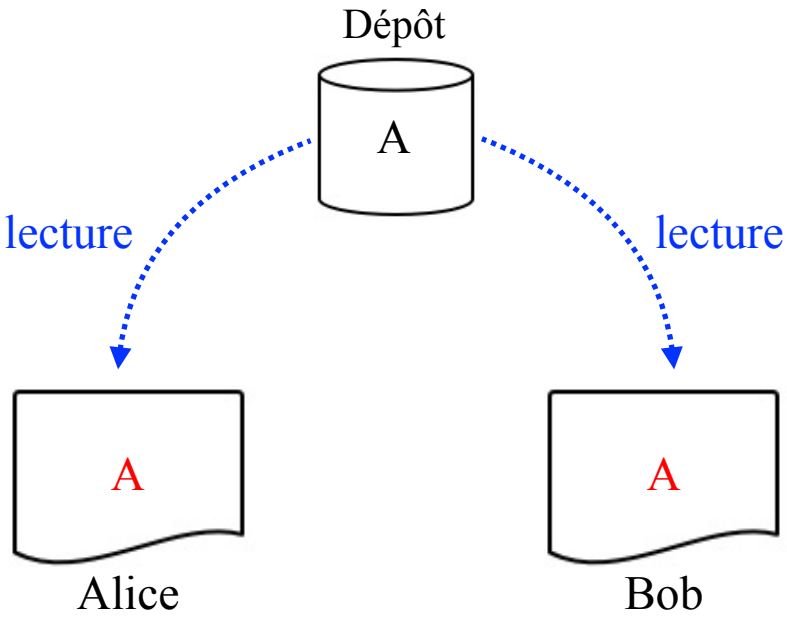
- Dépôt et copie locale



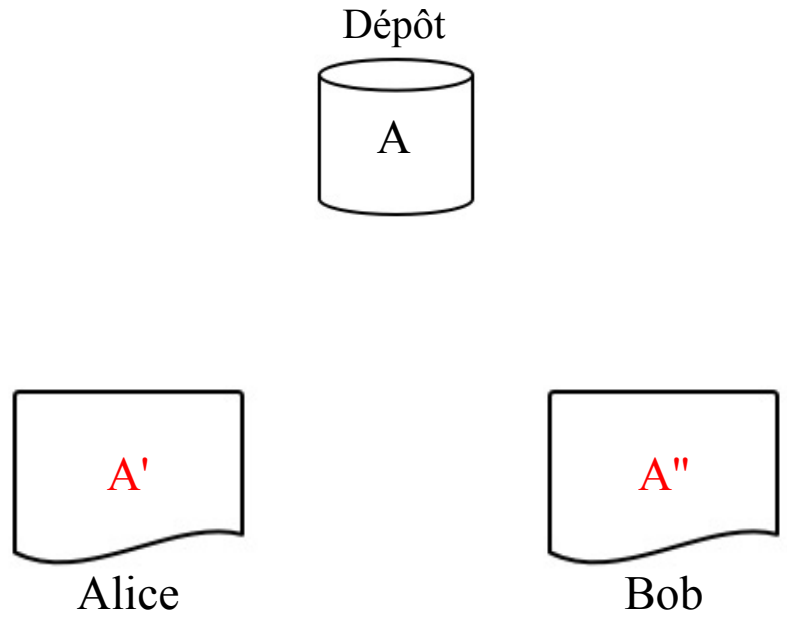
Agenda

- Chapitre 6: Gestion des Versions
  - Introduction
  - Partage d'un Même Fichier
    - ✓ Problème
    - ✓ Solution Simple
    - ✓ Copie-Modification-Fusion
  - Concepts du SCM
  - GIT

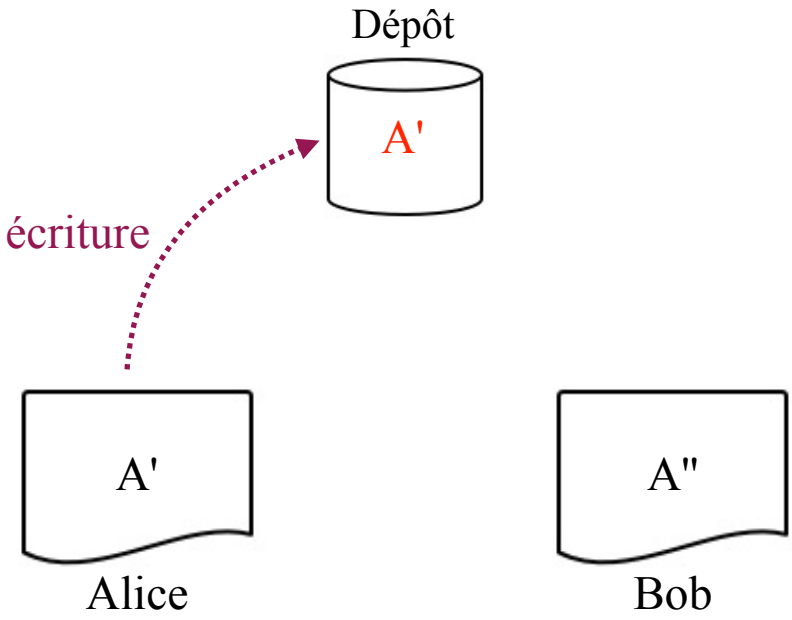
Problème



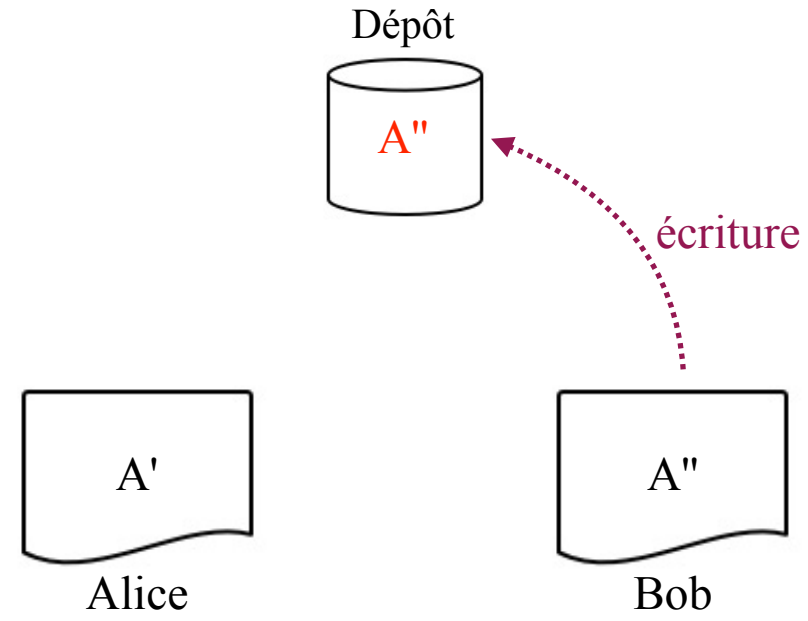
Problème (2)



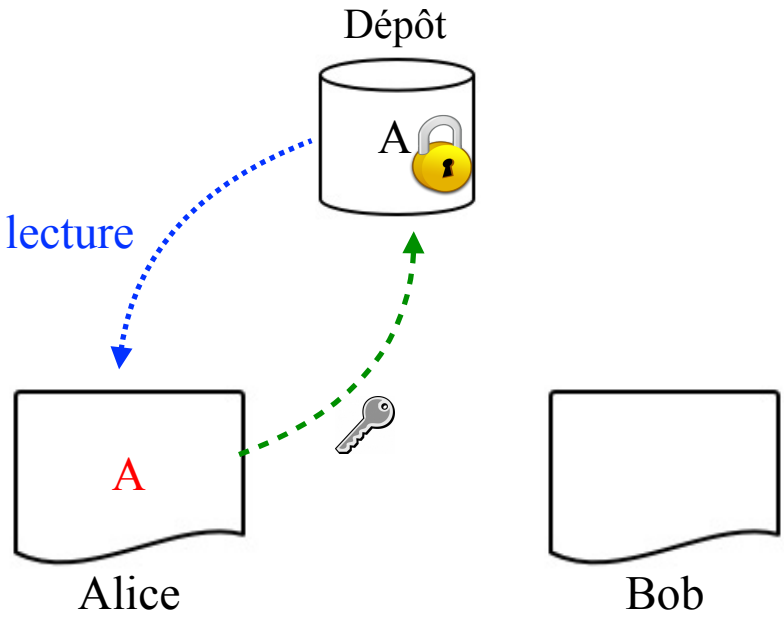
Problème (3)



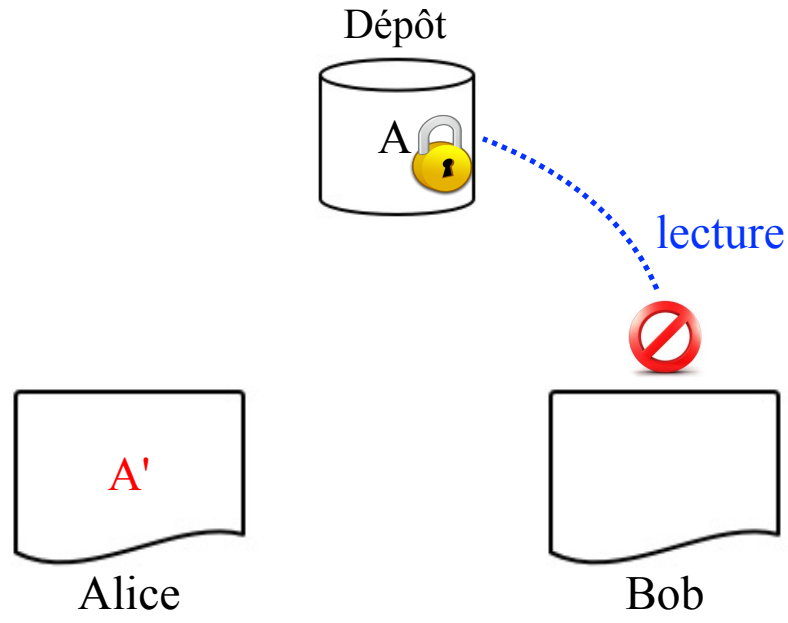
Problème (4)



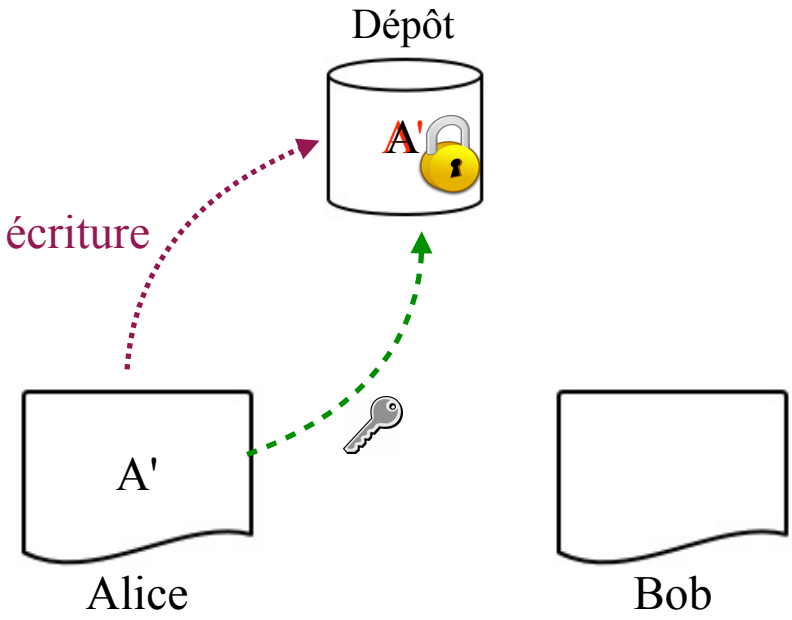
Solution Simple



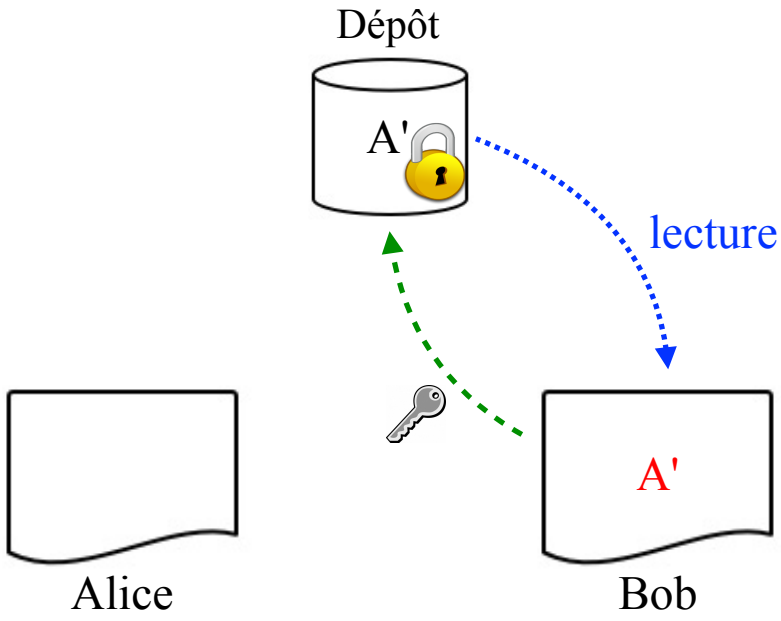
Solution Simple (2)



Solution Simple (3)



Solution Simple (4)



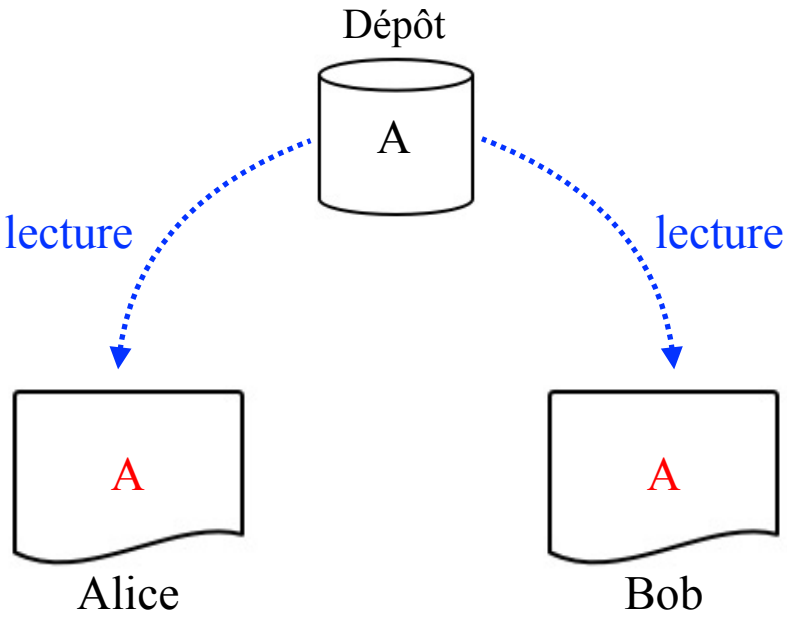
Solution Simple (5)

- Solution simple à mettre en oeuvre
- Mais…
  - si Alice verrouille le fichier et l'oublie, Bob restera bloqué
  - deux utilisateurs ne peuvent pas modifier, en même temps, des endroits différents d'un gros fichier

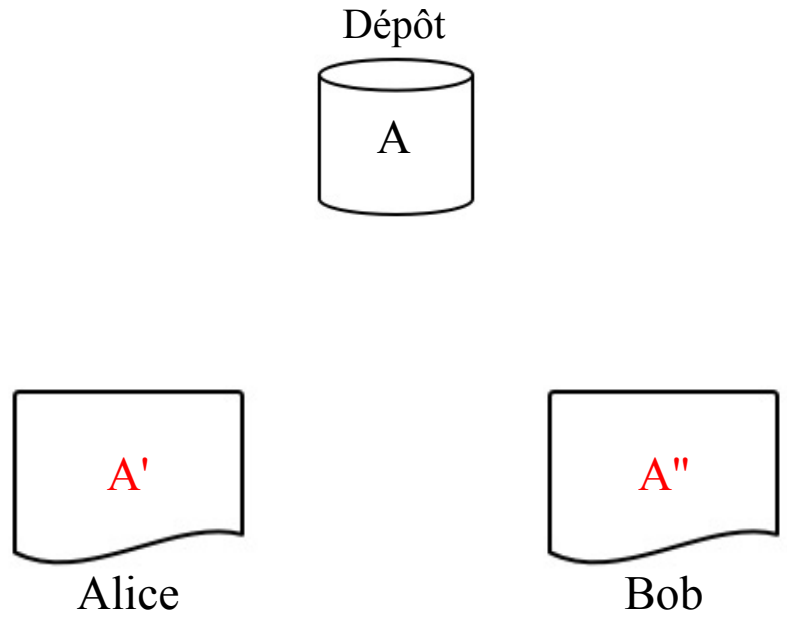
Solution Simple (6)

- Il faut trouver une autre solution
  - *copie-modification-fusion*
  - pas de verrou persistant

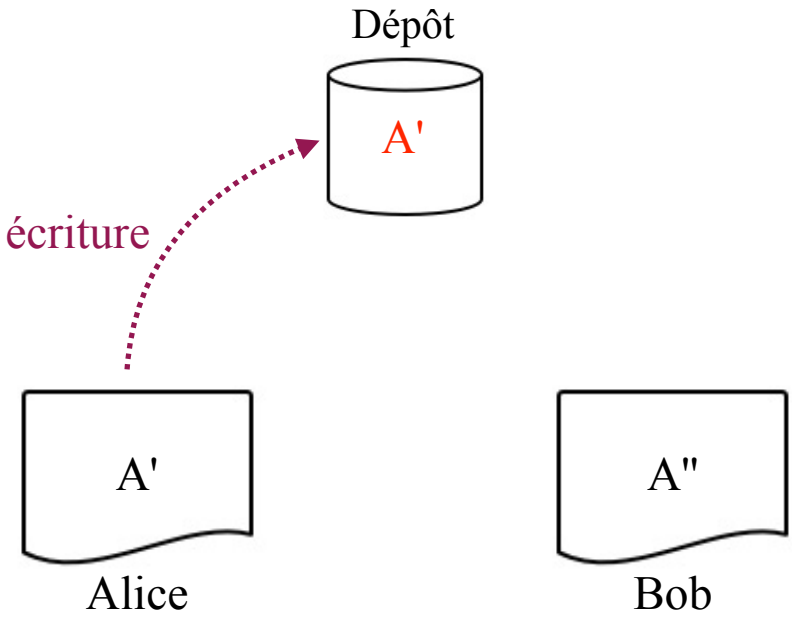
C-M-F



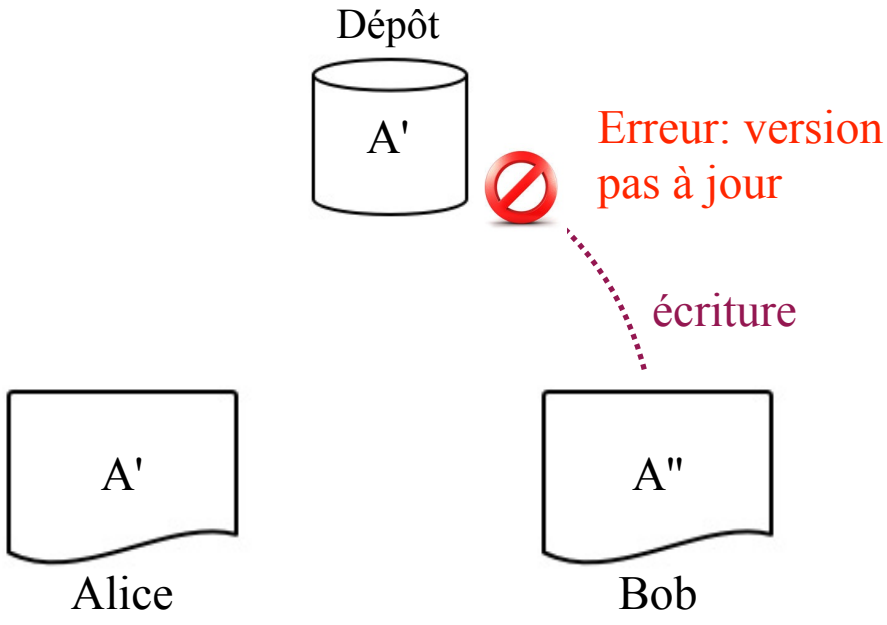
C-M-F (2)



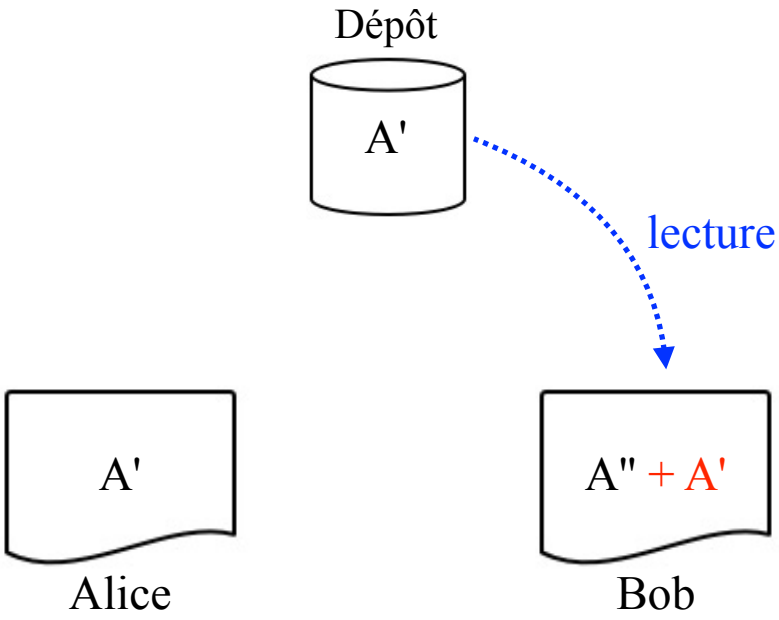
C-M-F (3)


Dépôt
A'
écriture
A'
Alice
A"
Bob

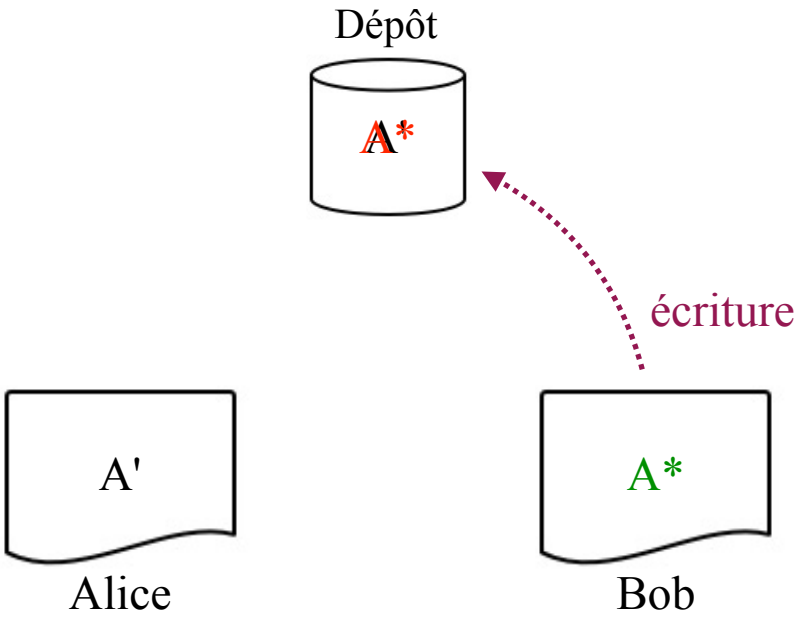
C-M-F (4)


Dépôt
A'
Erreur: version
pas à jour
écriture
A'
Alice
A"
Bob

C-M-F (5)



C-M-F (6)



C-M-F (7)

- La fusion automatique est possible si
  - il s'agit d'un fichier texte
  - et les modifications sont à des endroits éloignés
    - ✓ quelques lignes
- Unité de suivi
  - le fichier
  - aucun problème si les fichiers modifiés sont différents

Agenda

- Chapitre 6: Gestion des Versions
  - Introduction
  - Partage d'un Même Fichier
  - Concepts du SCM
    - ✓ Vocabulaire
    - ✓ Opérations Importantes
    - ✓ SCM Centralisé
    - ✓ SCM Distribué
  - GIT

Vocabulaire

- Objets de base?
  - fichier(s)
  - dossier(s)
- Points clés?
  - fusion +/- automatique possible pour les fichiers texte
    - ✓ code source
    - ✓ LaTeX
    - ✓ HTML
    - ✓ …
  - analyse ligne par ligne
  - espaces pris en compte
    - ✓ attention à la ré-indentation
  - renommage/copie explicite ou implicite

Vocabulaire (2)

- **Etat d'un projet**
  - ensemble de fichiers/dossiers constituant le projet pour une personne à un moment donné
  - un état peut être sauvé dans le SCM ou pas
- **Patch** ou **diff**
  - ensemble de modifications entre un état et un autre

Vocabulaire (3)

- **Commit**
  - état associé à plusieurs méta-données
    - ✓ Author
      - personne ayant créé les modifications
    - ✓ AuthorDate
      - date de création des modifications
    - ✓ Parent(s)
      - commit(s) précédent(s) dans l'historique des versions du projet
    - ✓ Message
      - description des modifications
- Les SCMs stockent souvent un diff par rapport au (premier) commit parent plutôt que l'état complet du projet
  - mais avec les méta-données

Vocabulaire (4)

- **Working directory**
  - état du projet en cours de travail
  - peut correspondre ou non à un commit
  - dérive d'un ou plusieurs commit(s)
- **Branche**
  - ensemble de commits partant d'un état initial vide et conduisant au commit final/courant de cette branche
  - une branche est généralement nommée
- **Historique du Dépôt**
  - ensemble de branches
    - ✓ généralement non disjointes
      - i.e., avec des commits appartenant à plusieurs branches
  - généralement non manipulable directement

Opérations Importantes

- **checkout**
  - transfert d'un état de l'historique dans le répertoire de travail
- **commit**
  - enregistrement de l'état courant du répertoire de travail dans un nouveau commit
- **merge**
  - fusion (automatique ou manuelle) de l'état d'une branche et de l'état courant (*fusion locale*) ou de l'état de deux branches (*fusion de branches*)
- **clone**
  - duplication d'un historique (souvent associé à un checkout de la branche principale)

Opérations Importantes (2)

- **push**
  - envoi des commits d'un historique local dans un historique distant
- **fetch**
  - récupération des commits d'un historique distant dans l'historique local
- **pull**
  - fetch + merge
- **log**
  - affichage de l'historique
- **status**
  - affichage de la situation des fichiers dans l'état courant (modifié, effacé, …)

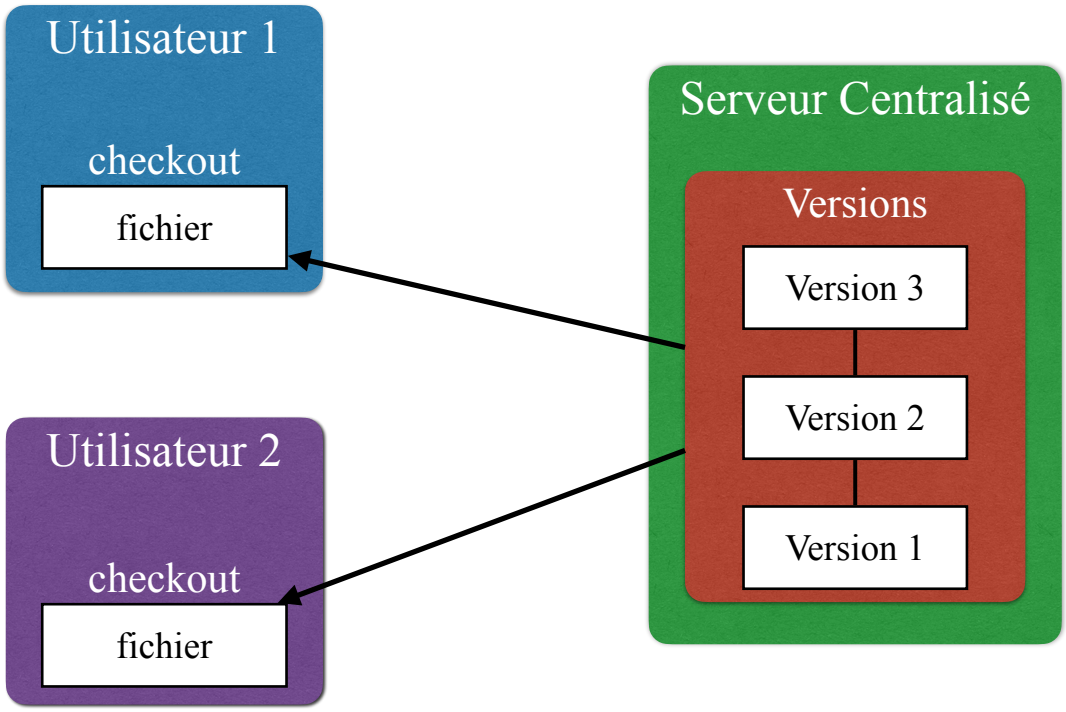
SCM Centralisé

- Un seul dépôt mais plusieurs répertoires de travail
- Opérations importantes
  - checkout
  - commit
  - merge (local et branches)
- Exemples
  - CVS
  - *SVN*
  - Perforce

SCM Centralisé (2)

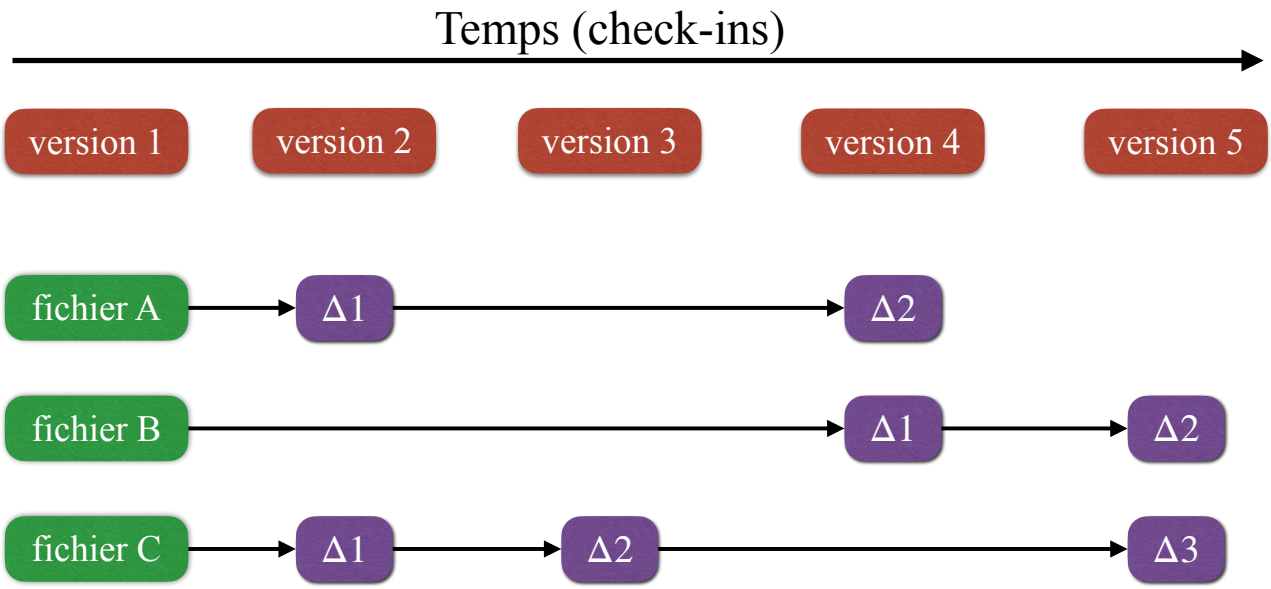
- Le serveur central conserve la copie officielle du projet
- On réalise des "checkouts" de la copie officielle vers la copie locale
  - les modifications sont effectuées en local
  - les changements ne sont pas versionnés
- Quand on est prêt, on "check in" vers le serveur
  - le check in incrémenté la version officielle sur le serveur

SCM Centralisé (3)



SCM Centralisé (4)

- Un SCM centralisé conserve un tracking des versions pour chaque fichier individuel



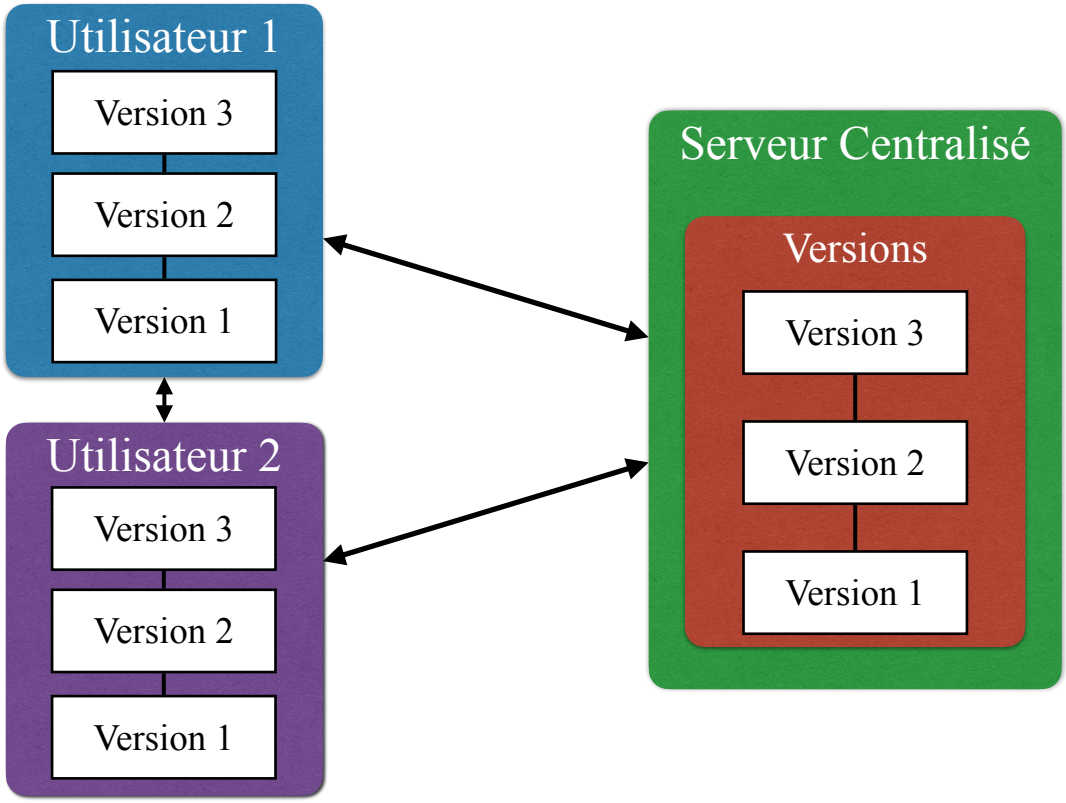
SCM Distribué

- Chaque répertoire de travail contient aussi l'historique
- Opérations importantes
  - pull
  - commit
  - merge (local et branches)
  - clone
  - push
  - fetch
- Exemples
  - *git-hub*
  - mercurial
  - bazaar
  - …

SCM Distribué (2)

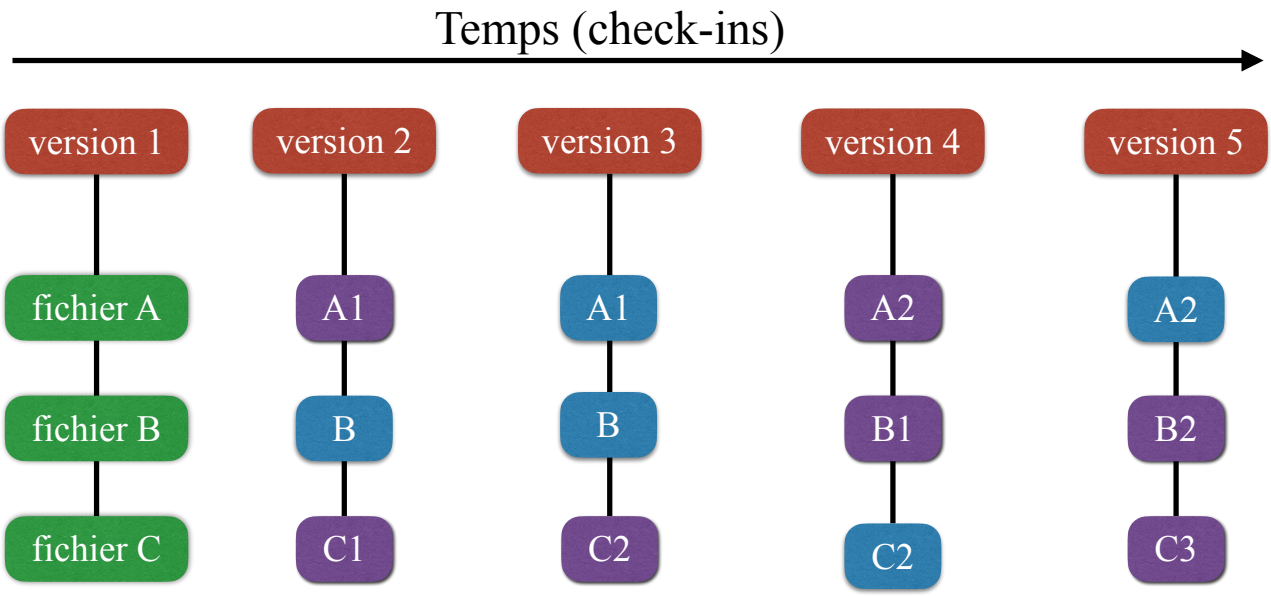
- On ne fait pas de checkout depuis un repo central
  - on le clone et récupère (pull) les changements
- La copie local est une copie complète de tout ce qu'il y a sur le serveur
- Beaucoup d'opérations sont locales
- Quand on est prêt, on pousse (push) les changements sur le serveur

SCM Distribué (3)



SCM Distribué (4)

- Un SCM distribué conserve un snapshot de l'état entier du projet



Agenda

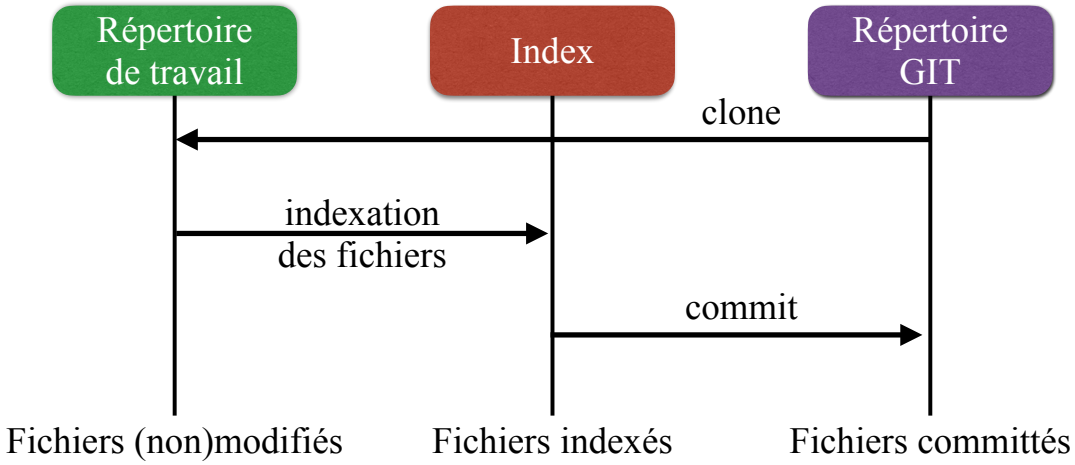
- Chapitre 6: Gestion des Versions
  - Introduction
  - Partage d'un Même Fichier
  - Concepts du SCM
  - GIT
    - ✓ Commandes Principales
    - ✓ En Local
    - ✓ Flux de Base
    - ✓ Revenir en Arrière
    - ✓ Branches

Commandes Principales

Commande	Description
`$>git clone url [dir]`	copie en local un repo git distant
`$>git add fichier`	ajoute le fichier dans l'index
`$>git commit -m "message"`	enregistre un snapshot dans l'index
`$>git status`	examine le status des fichiers dans le répertoire de travail et l'index
`$>git diff`	montre un diff entre ce qui est indexé et ce qui est modifié mais non indexé
`$>git help [commande]`	obtenir de l'aide pour une commande spécifique
`$>git pull`	récupère un repo distant et essaie de le fusionner dans la branche courante
`$>git push`	pousse les nouvelles branches et données vers le repo distant

En Local

- Dans la copie locale, les fichiers peuvent être
  - récupérés et modifiés mais pas encore committés
    - ✓ copie de travail
  - dans un entre-deux (**staging** area -- **index**)
    - ✓ fichiers indexés sont prêts à être committés
    - ✓ un commit sauve un snapshot de tous les états indexés



Flux de Base

- Quatre étapes
  1. modification des fichiers dans le répertoire de travail
     - ✓ status du fichier: *untracked*
  2. indexation des fichiers et ajout de snapshots dans l'index
     - ✓ status du fichier: *staged*
  3. commit, pour transférer les fichiers de l'index vers le repo GIT
     - ✓ status du fichier: *committed*
  4. pousser, pour transférer le snapshot vers le serveur distant

```
[alice@laptop ~]$>edit code.c &
[alice@laptop ~]$>git add code.c
[alice@laptop ~]$>git commit -m "mon code..."
[alice@laptop ~]$>git push
```

Revenir en Arrière

- Quatre commandes permettent d'annuler des modifications
  - `reset`, `revert`, `checkout`, `restore`
- Retirer un fichier de l'index
  - `git reset -HEAD code.c`
  - status du fichier: *staged* → *modified*
- Annuler des modifications sur un fichier
  - `git checkout -- code.c`
  - `git restore code.c`
  - status du fichier: *committed*
  - contenu ramené à l'état du dernier commit

Revenir en Arrière (2)

- Inverser un commit qui vient d'être réalisé et ajouter un "contre-commit" dans l'historique
  - `git revert HEAD`
- Autres méthodes
  - `git reset --soft HEAD 1`
    - ✓ annule le dernier commit
    - ✓ ne touche pas aux fichiers
  - `git reset --mixed HEAD 1`
    - ✓ annule add et commit
    - ✓ ne touche pas aux fichiers
  - `git reset --hard HEAD 1`
    - ✓ ramène à l'état du commit précédent
    - ✓ modifie les fichiers
- Attention
  - `git reset HEAD n`
    - ✓ permet de revenir en arrière de *n* commits

Branches

- Utiles dans un contexte collaboratif
- Objectif(s)?
  - isoler son travail du développement principal
  - tester plusieurs "pistes"
  - maintenir une version "stable" et une version "en développement"
  - ...
- Commandes
  - `git branch MaNouvelleBranche`
    - ✓ créer une nouvelle branche
  - `git checkout MaBranche`
    - ✓ basculer sur *MaBranche*
  - `git checkout master`
    - ✓ revenir sur la branche principale
  - `git branch -d MaBranche`
    - ✓ supprimer *MaBranche*
  - `git branch`
    - ✓ lister toutes les branches

Branches (2)

- Pour fusionner *MaBranche* avec le master
  - `git checkout master`
  - `git merge MaBranche`