Les Systèmes de Gestion de Version

Journées Mathrice - Mars 2011 - Dijon-Besançon

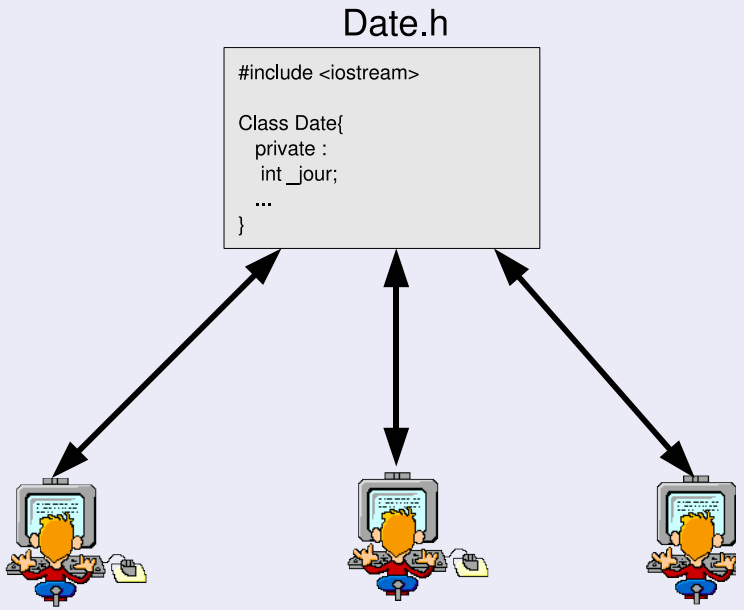
F. Langrognet

PLAN

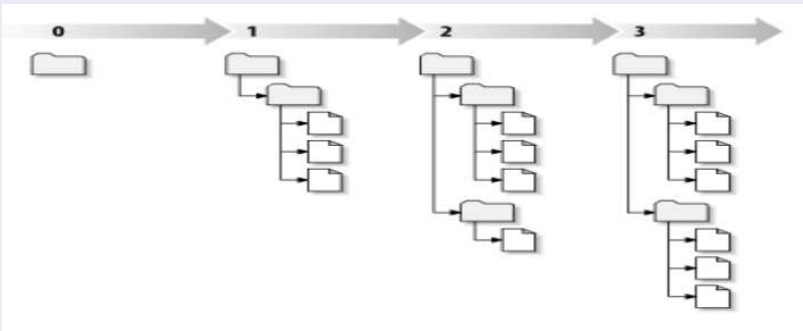
PLAN

Objectifs d'un Système de Gestion de Version

Travailler à plusieurs



Conserver l'historique

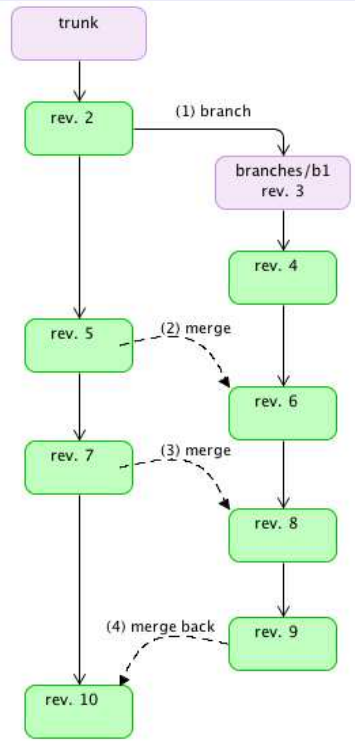


- Pouvoir revenir en arrière
- Qui a modifié pour la dernière fois ce fichier ?
- Quelles sont les différences entre 2 versions de ce fichier ?
- Quelle est la version du 15 mars 2007 ?

Objectifs d'un Système de Gestion de Version (suite)

Et aussi ...

- La gestion des branches
  Objectif : mener en parallèle plusieurs versions (stable, testing, ...)

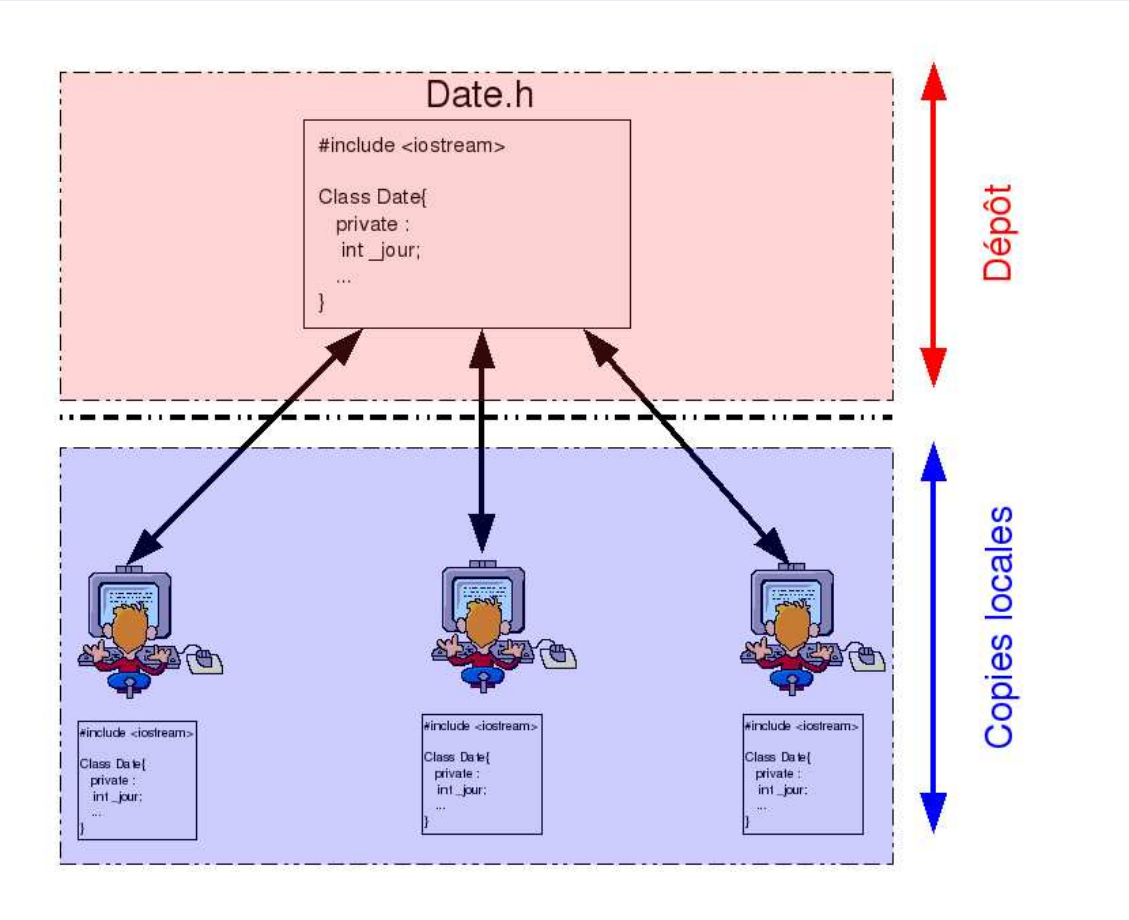


- L'utilisation de *tags*
  Objectif : donner un nom explicite à une version pour pouvoir y accéder facilement

PLAN

Principe de base

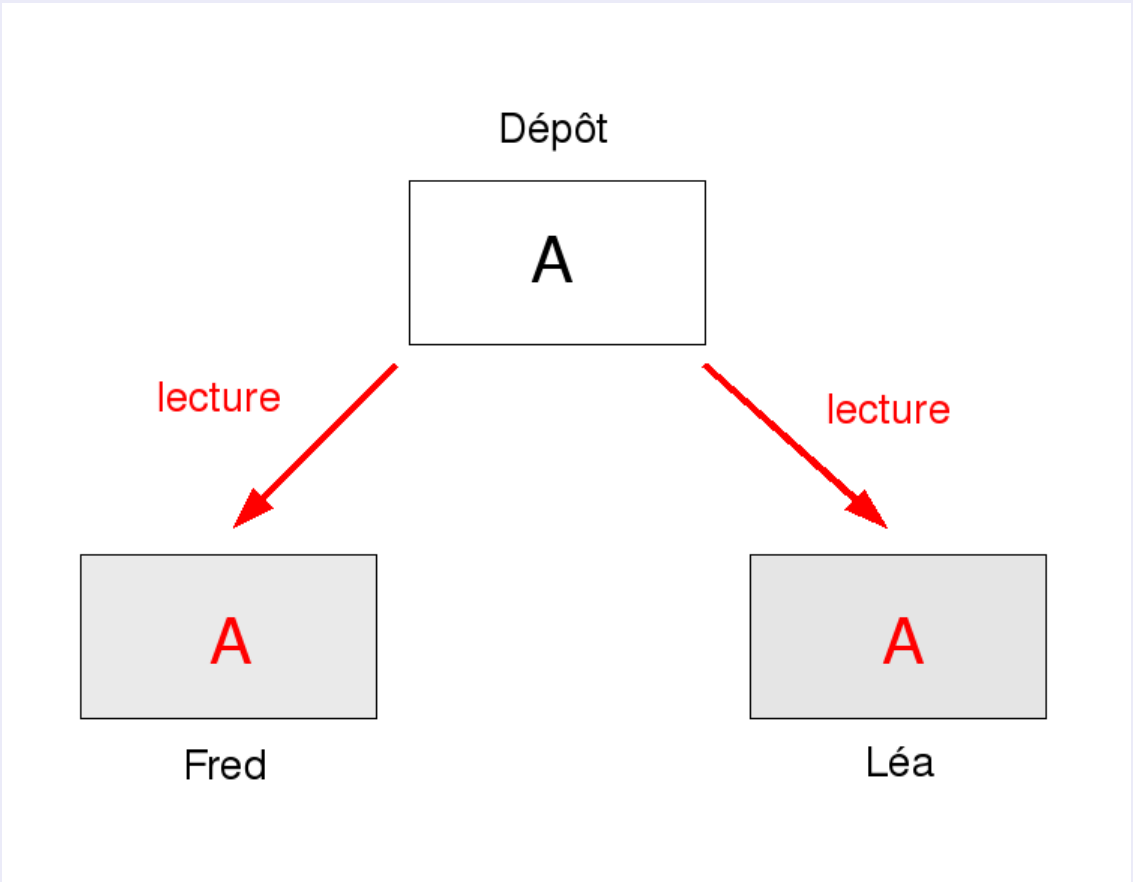
Notion de *dépôt* et *copie locale*



Les accès (écriture/lecture) se font via le système de gestion de version

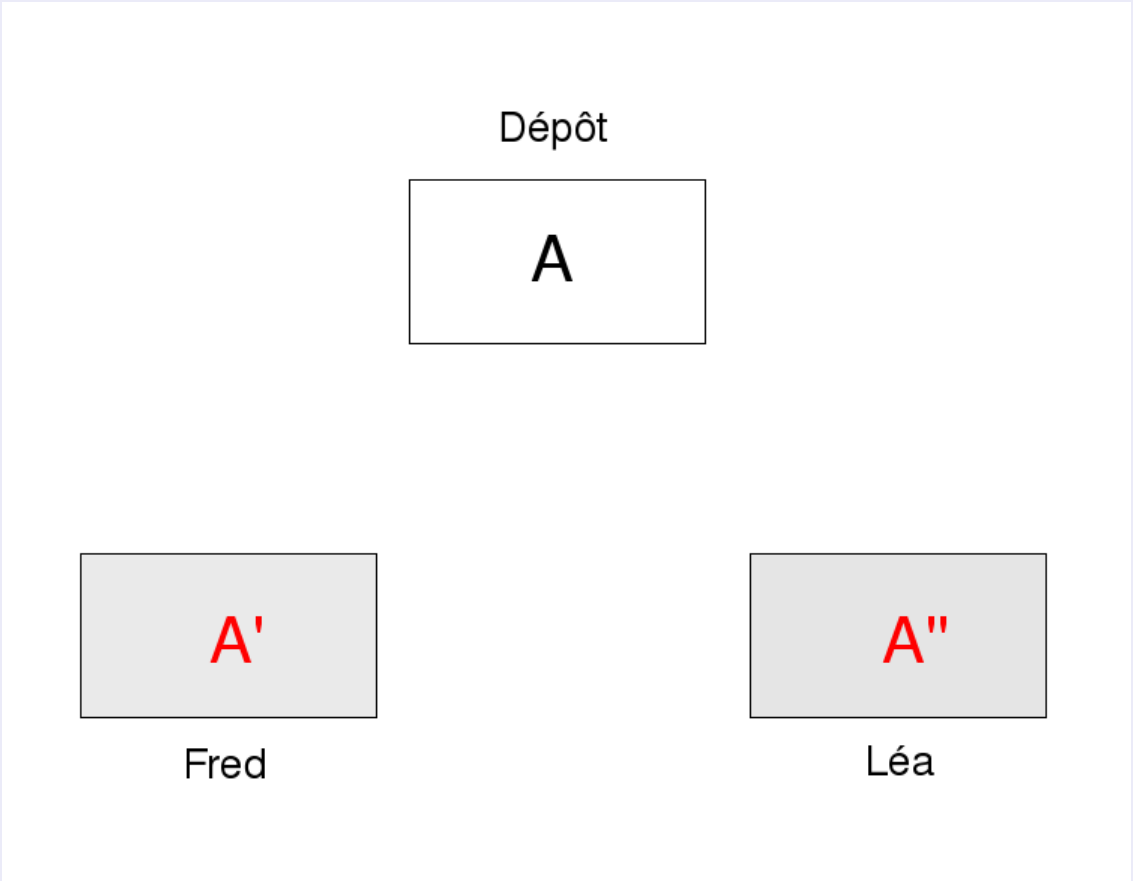
Le problème . . .

Problème



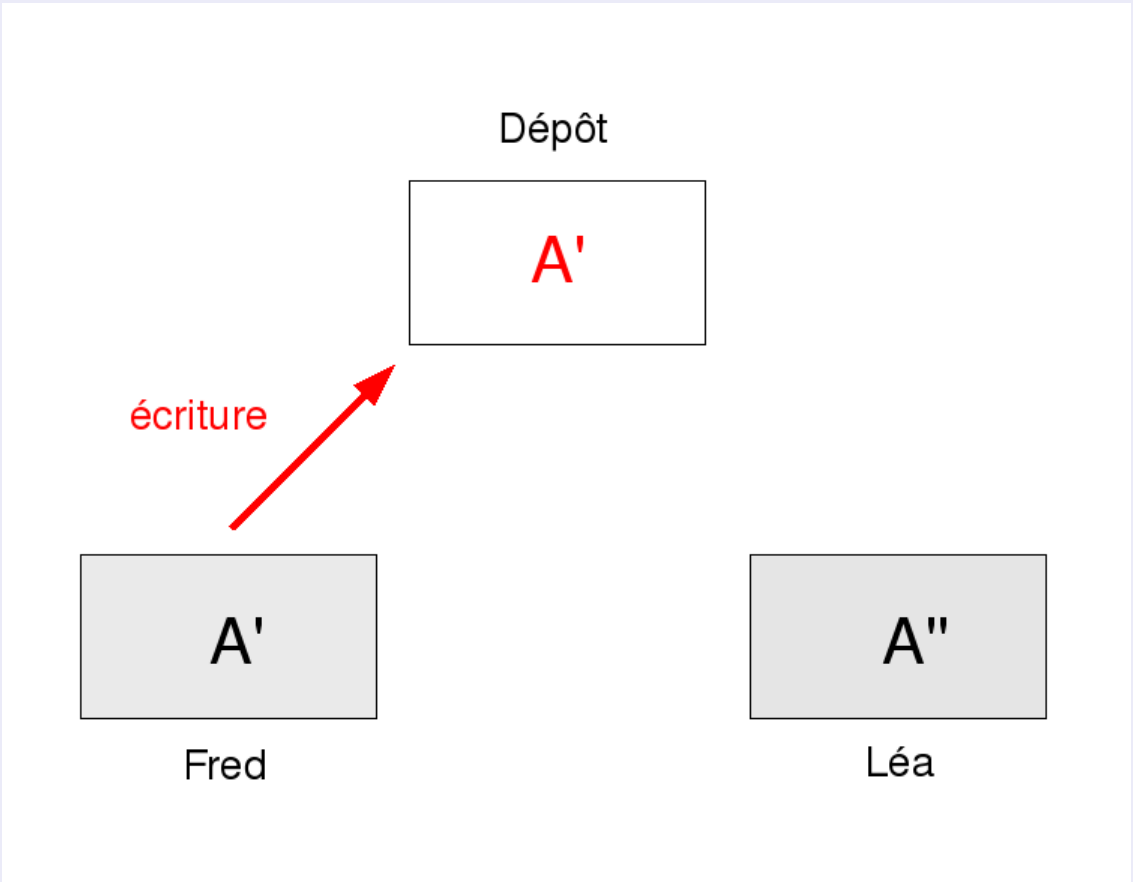
Fred et Léa accèdent au même fichier et le copient chez eux

Problème (suite)



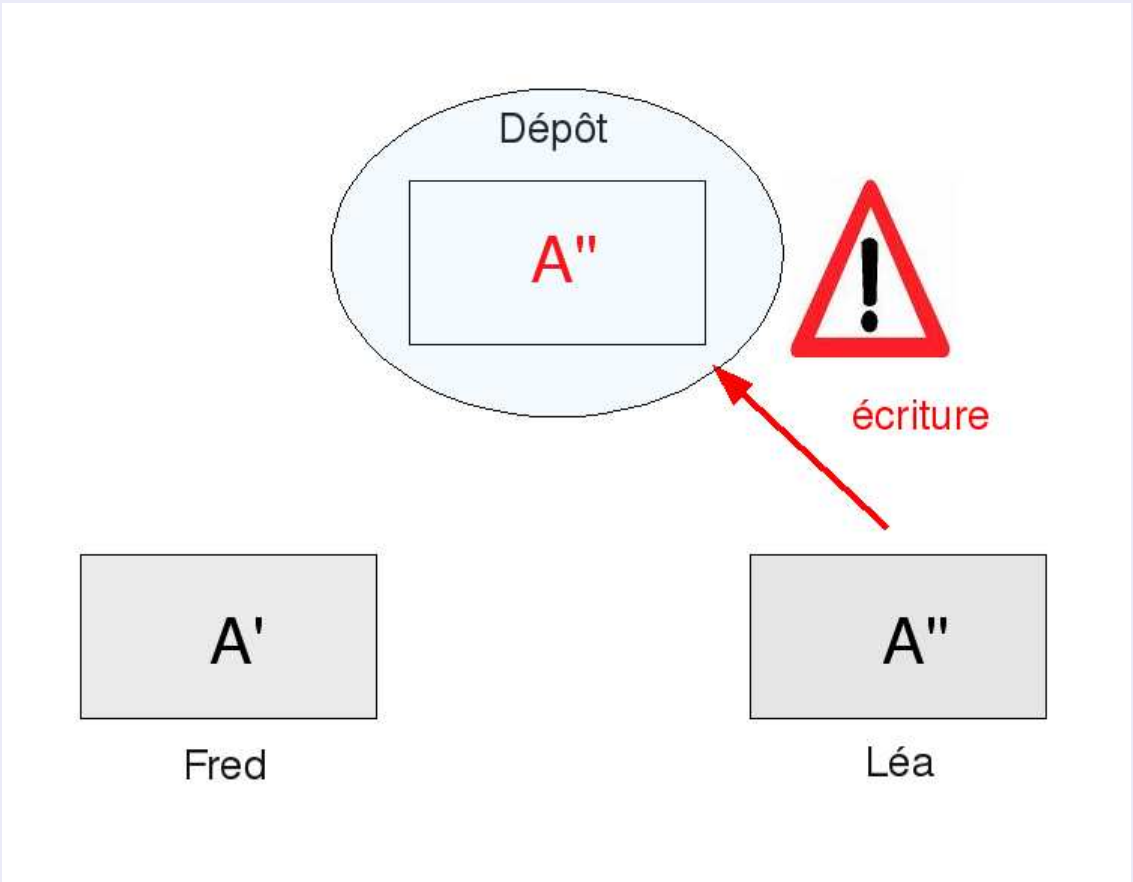
Fred et Léa font chacun des modifications

Problème (suite)



Fred écrit sur le dépôt

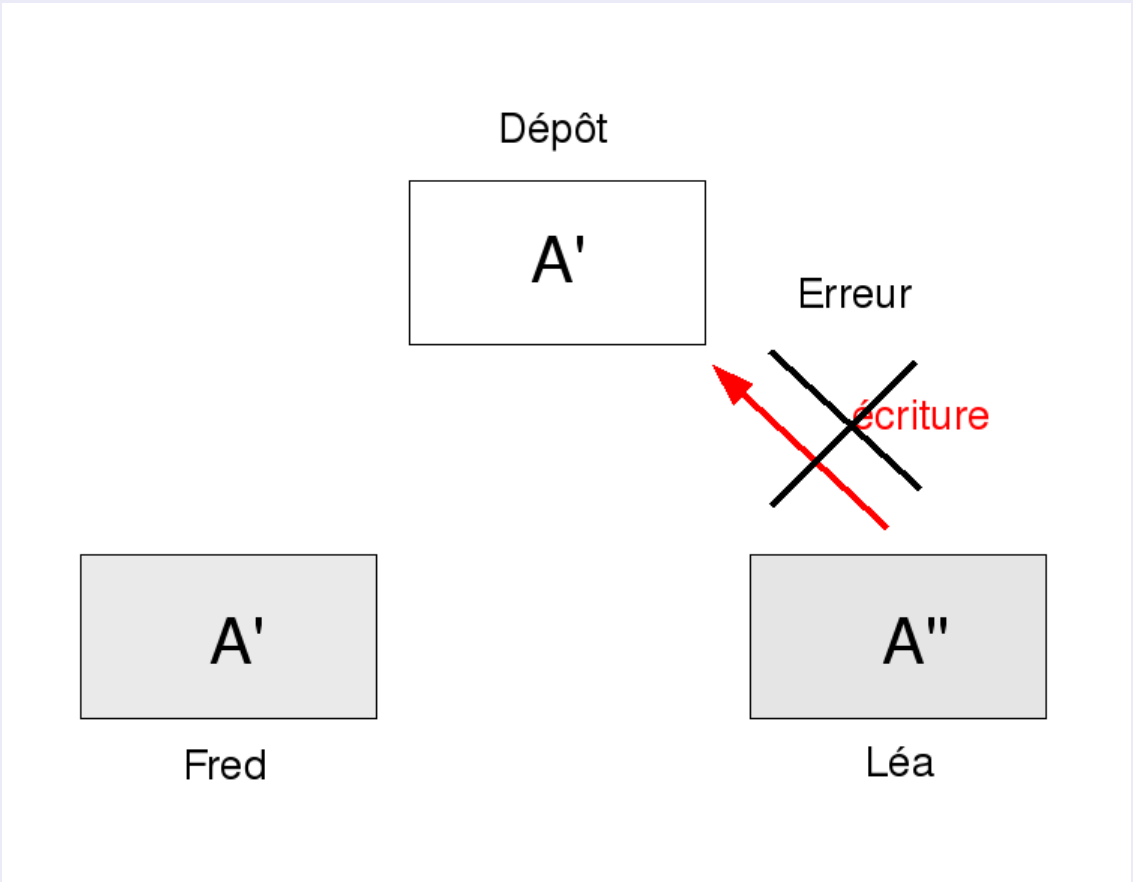
Problème (suite)



Léa écrit sur le dépôt en écrasant la version de Fred

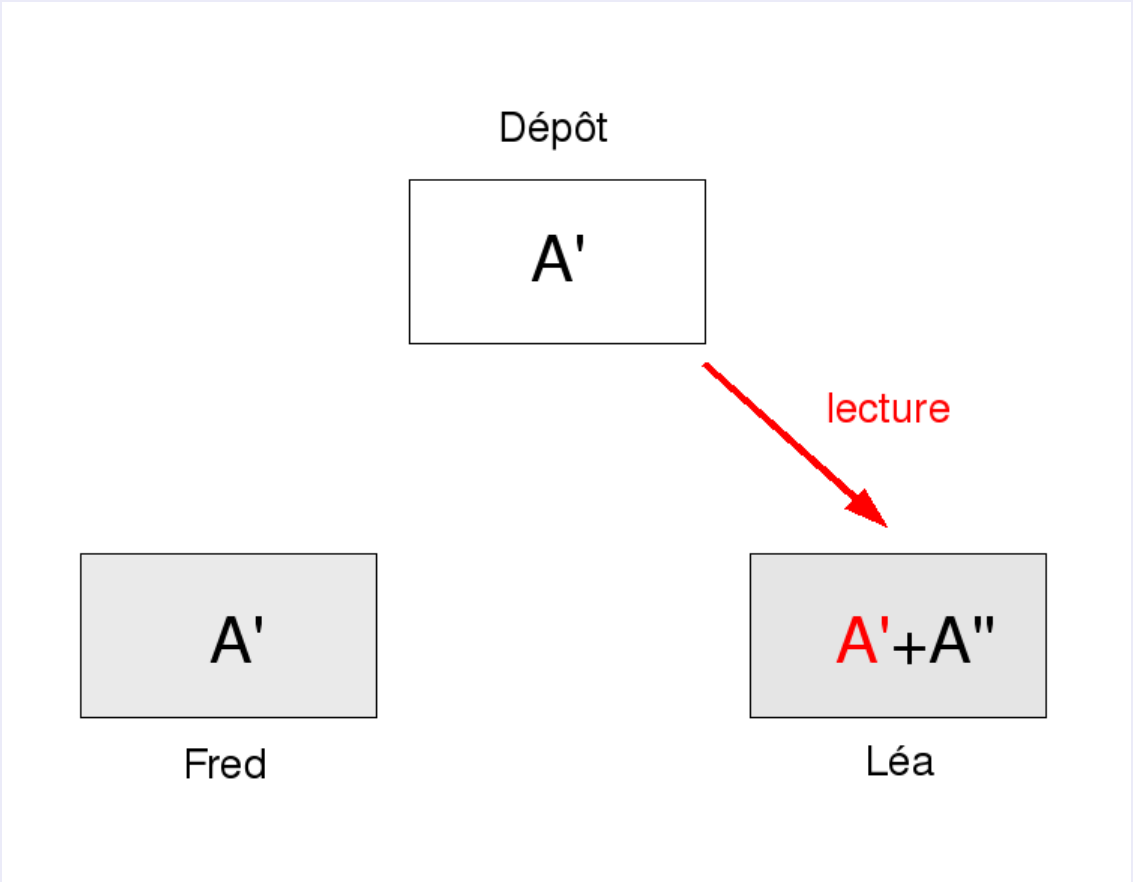
La solution . . .

Solution



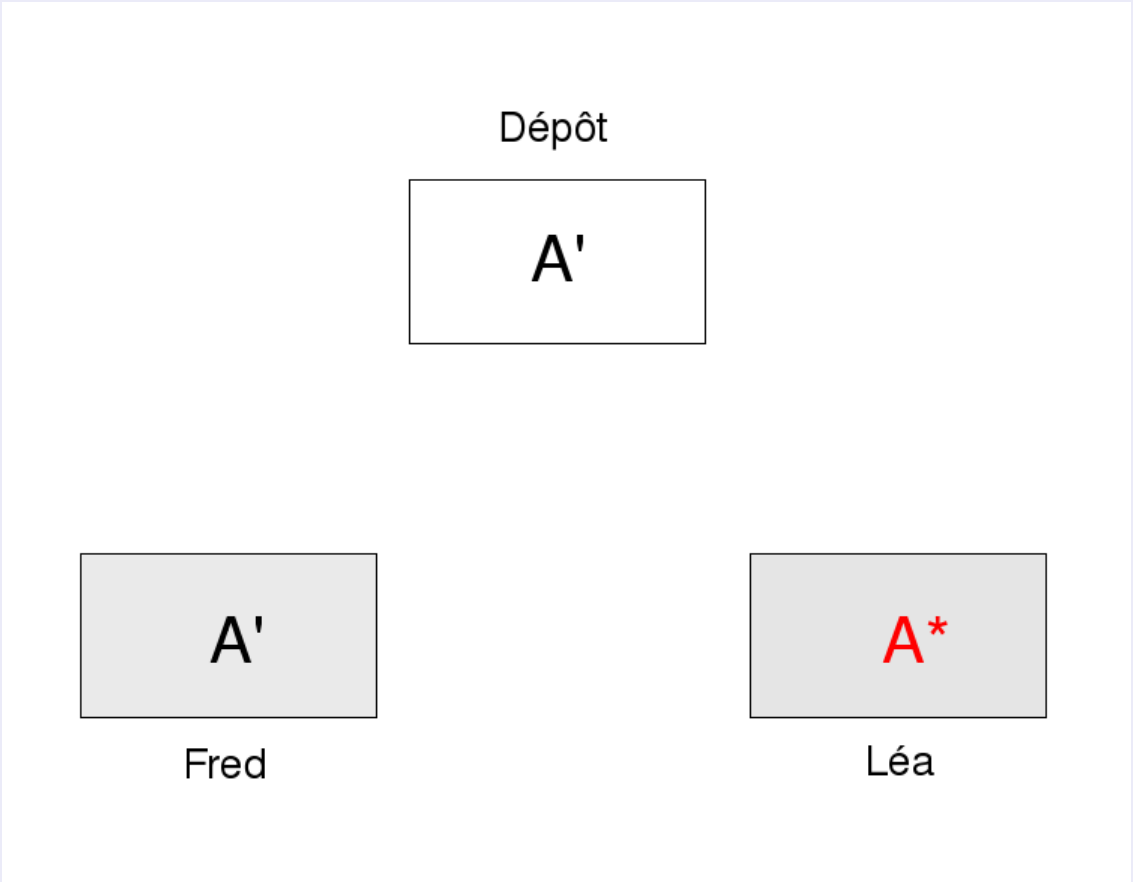
Léa ne peut pas écrire sur le dépôt car sa version n'est pas à jour

Solution (suite)



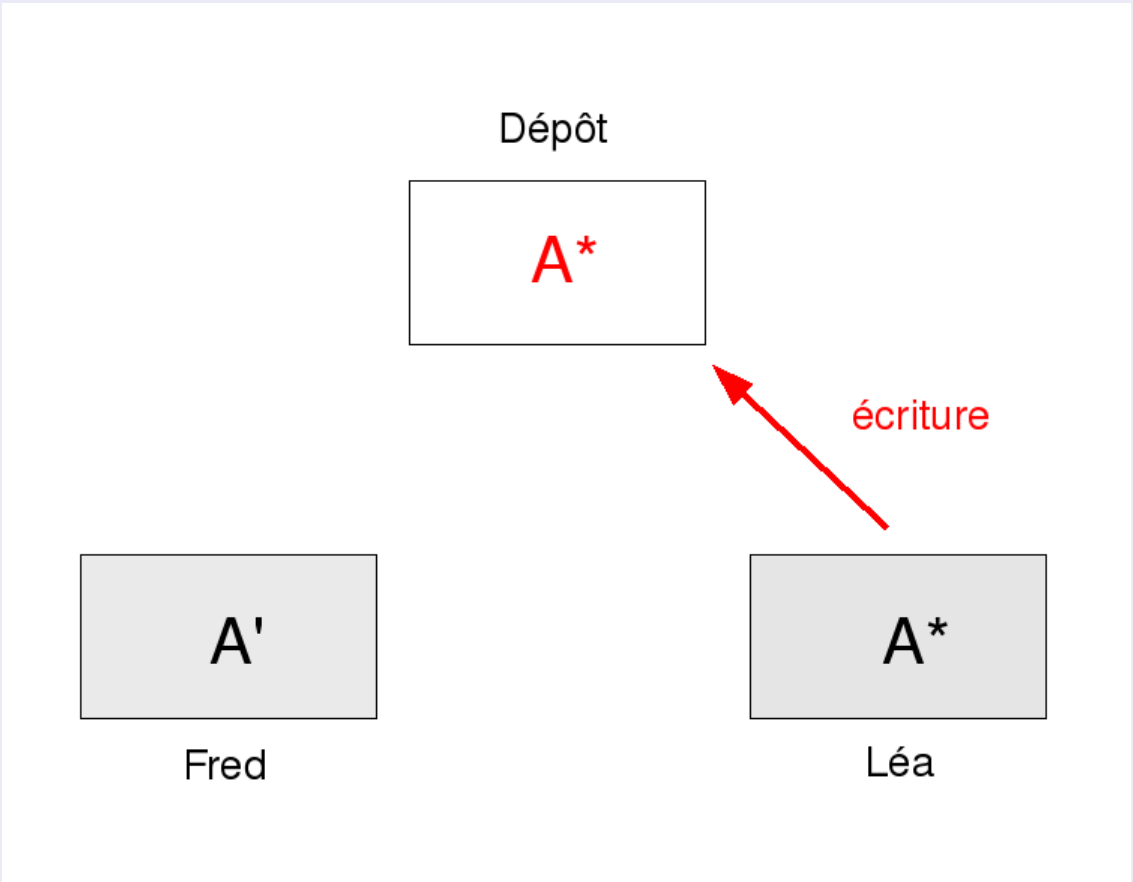
Léa met à jour : elle récupère la version du dépôt sans perdre ses modifications

Solution (suite)



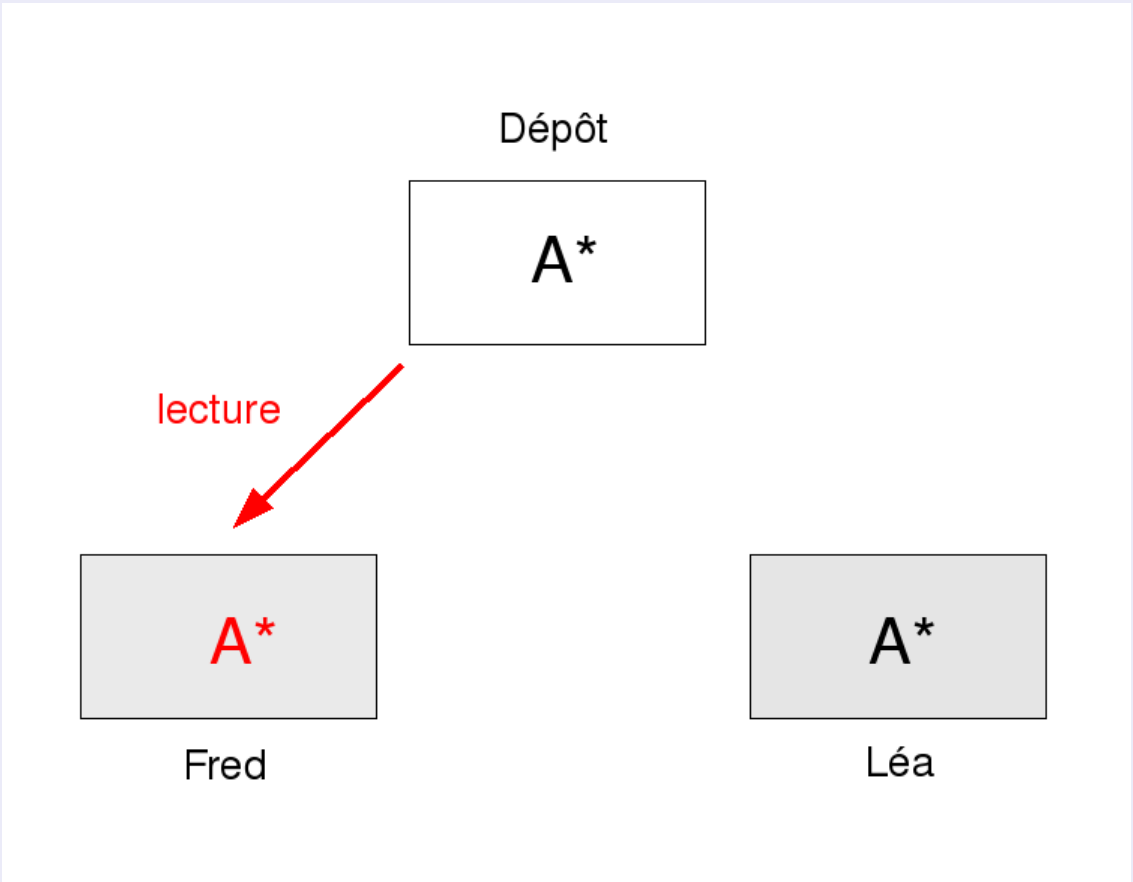
Léa fusionne la version du dépôt (A') avec sa version (A")
A',A" -> A*

Solution (suite)



Léa peut écrire sur le dépôt

Solution (suite)



Fred récupère la nouvelle version

Système de Gestion de Version

Système de Gestion de Version

Un SGV gère le mécanisme de lecture-fusion-écriture

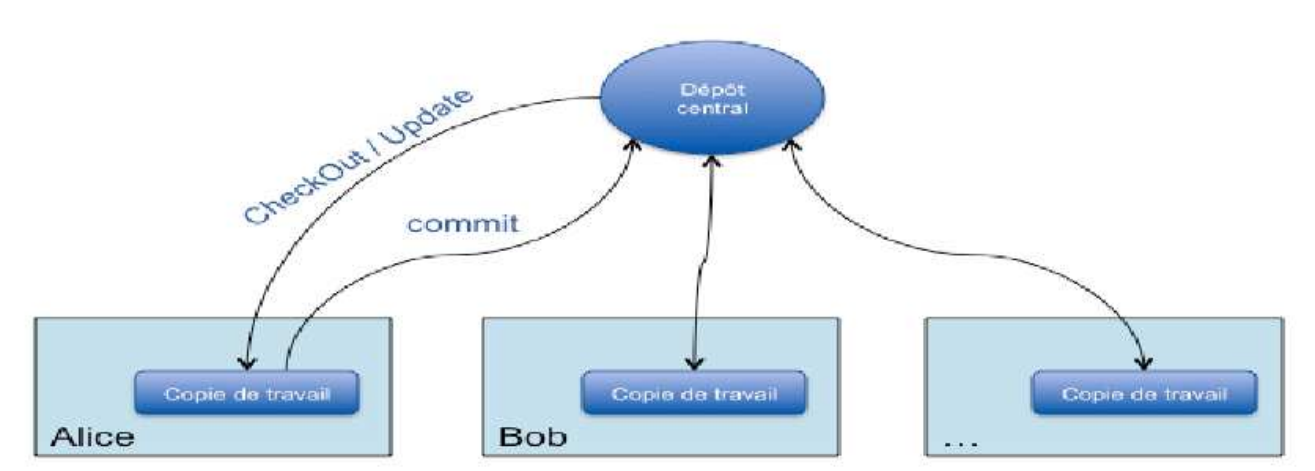
- Les demandes de lecture, écriture se font via le SGV
- La fusion automatique est possible si
  - il s'agit d'un fichier *texte (diff)*
  - *les modifications sont assez éloignées les unes de autres*
- Le SGV conserve l'historique
- Et aussi : gestion des branches, tags, ...

PLAN

2 grandes catégories de SGV

1. Les systèmes centralisés

Un seul dépôt centralisé



Des qualités . . .

- Technologie éprouvrée
- Largement disponible (IDE, Forges)
- Portabilité
- Sécurité

et des défauts !

- Echange entre les dépôts impossible
- Echange entre les copies locales impossible
- Travail hors connexion impossible
- Temps de mise à jour long pour de gros projet
- Et si le serveur tombe en panne ?

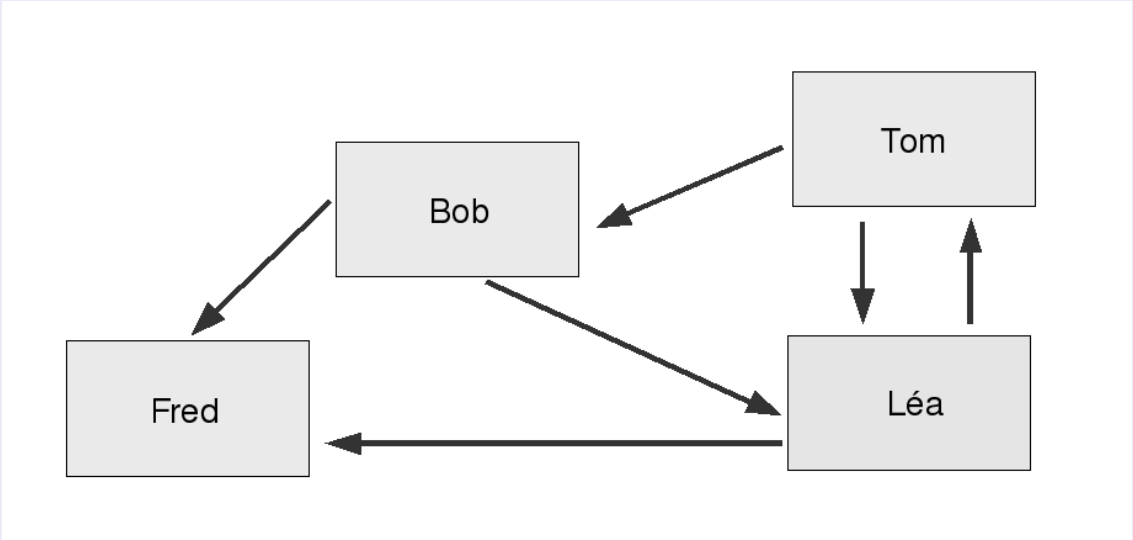
2 grandes catégories de SGV (suite)

2. Les systèmes décentralisés

Objectifs : pallier les problèmes des systèmes centralisés

- Pouvoir utiliser ce système *hors connexion*
- Ne pas être dépendant d'un dépôt centralisé (panne, temps, . . .)
- Pouvoir échanger ses fichiers avec une partie des développeurs
- . . .

Chaque développeur possède son propre dépôt

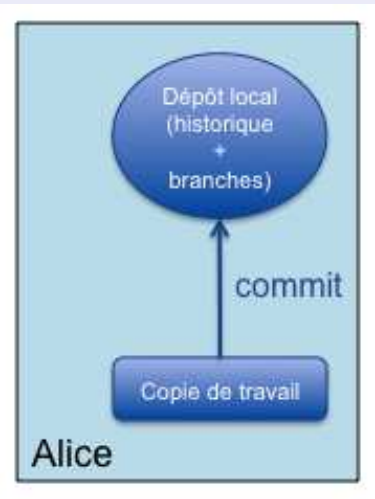


Les sytèmes décentralisés

Les avantages d'un système centralisé (en local)

Chaque développeur a son propre dépôt et sa copie de travail

Il peut donc utiliser un SGV décentralisé pour (par exemple)
conserver l'historique ou gérer des branches en local

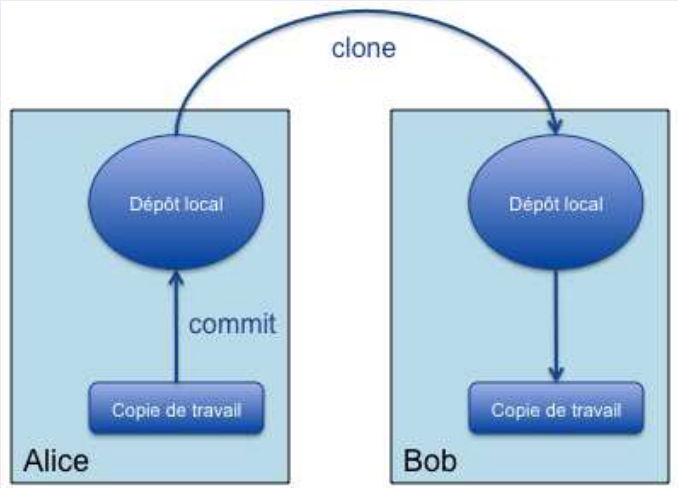


Les sytèmes décentralisés (suite)

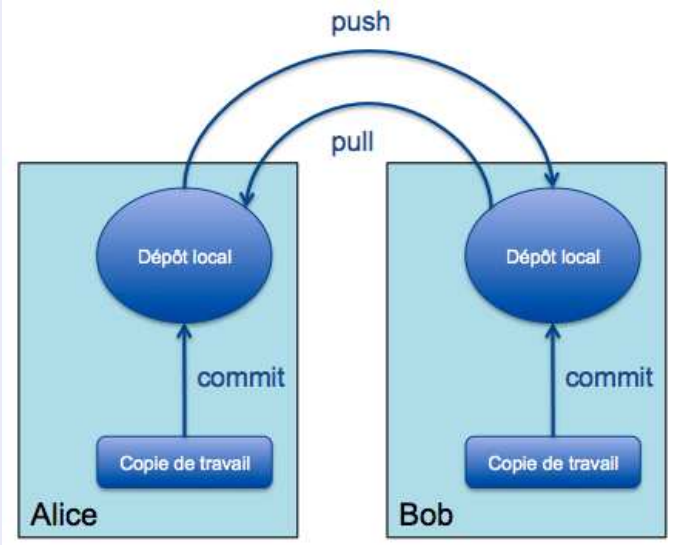
Travail entre dépôts

Les dépôts locaux peuvent communiquer

Clone d'un dépôt vers un autre (initialisation)



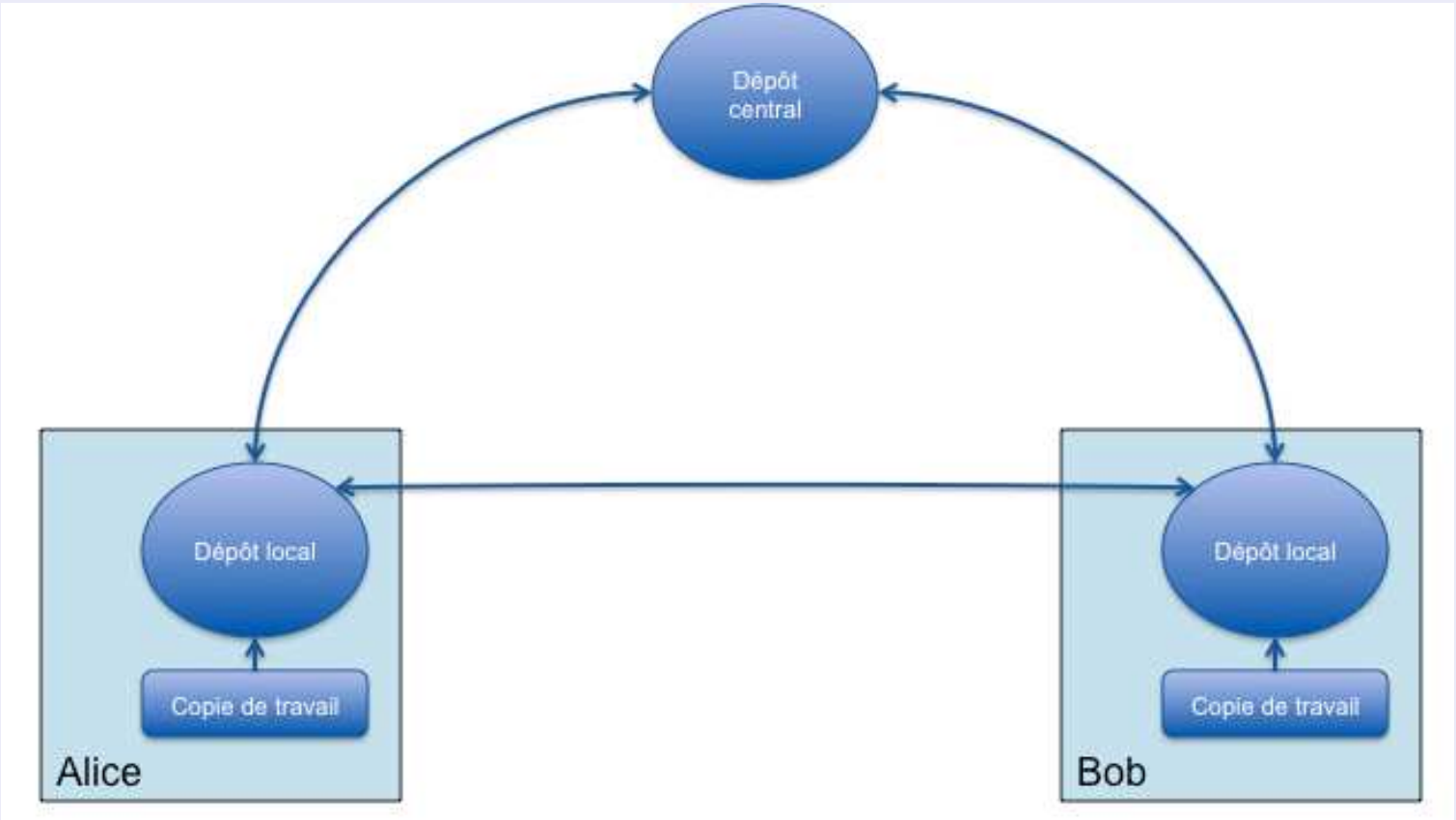
Ecriture/Lecture d'un dépôt vers un autre



Les sytèmes décentralisés (suite)

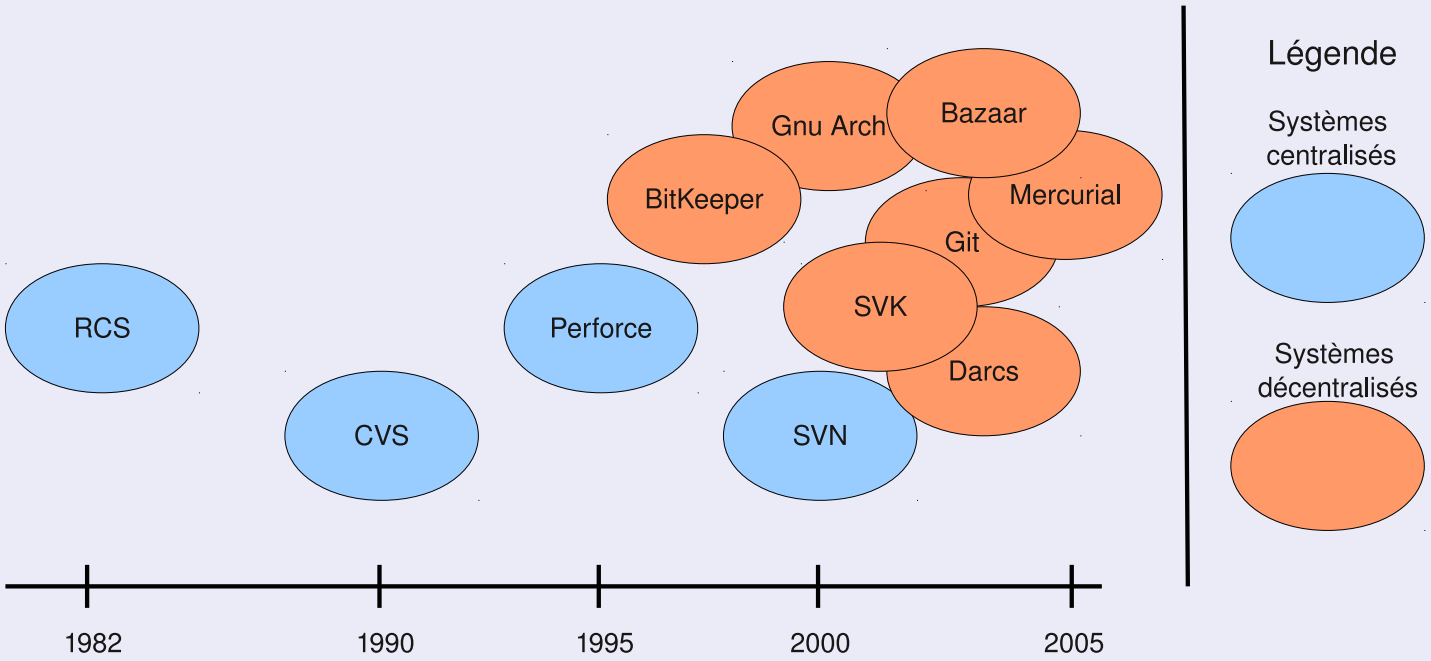
Avec un dépôt central ?

Si le nombre d'utilisateurs est grand, il peut être utile de mettre en pace un dépôt central pour stocker la version la plus à jour du système



Quel SGV choisir ?

Cartographie (incomplète) des SGV



Quel SGV choisir ?

Vaste choix

- Technologie en pleine évolution
- De nouveaux systèmes apparaissent régulièrement

Elements à prendre en compte

- Pérénité : systèmes *leaders* vs. systèmes *émergents*
- Intégré dans des *IDE*
- Proposé par des *Forges*
- Interfaces graphiques
- Portabilité (multi OS)
- Sécurité
- Documentations abondantes
- Outils connexes (ex : cvs2svn)

PLAN

L'exemple de Subversion (SVN)

Un SGV très répandu

- Documentations très riches, forums actifs
- Interfaces graphiques
  - Linux : rapidsvn, kdesvn, esvn, Qsvn, ...
  - Windows : intégré à l'explorateur via le plugin TortoiseSVN
- Proposé dans les Forges et intégré dans certains IDE (Eclipse, Kdevelop)

Le *successeur* de CVS

Reprend le modèle de CVS en comblant certains manques :

- Renommage et déplacement de fichiers sans perte de l'historique
- Gestion des répertoires
- commits atomiques
- Gestion des *metadonnées* (ex : permissions)
- Possibilité de migrer de CVS vers SVN sans perte de l'historique (cvs2svn)
- Protocoles réseaux sécurisés (HTTPS)

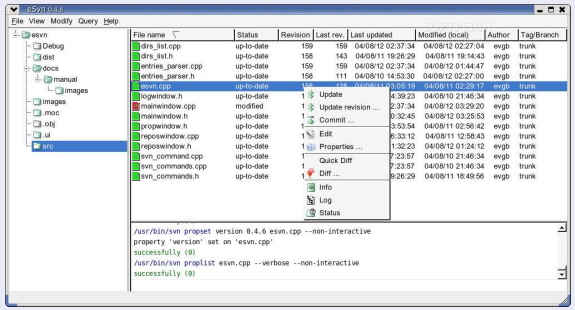
L'exemple de Subversion (suite)

Principales commandes

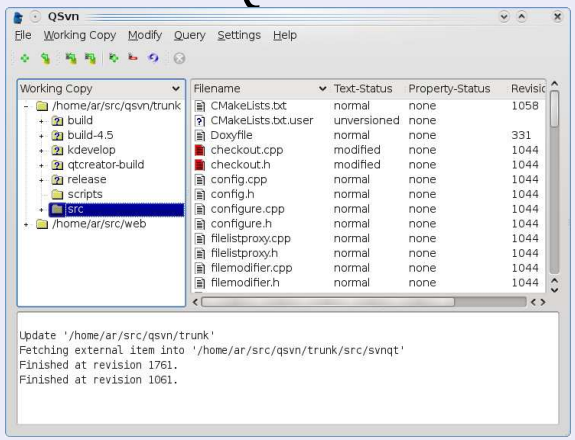
- svnadmin create : créer un nouveau dépôt
- svn import : importer un projet dans le dépôt
- svn checkout : lire tout un projet
- svn update : lire/mettre à jour depuis le dépôt
- svn commit : écrire/modifier le dépôt (nouvelle révision)
- svn status : état de la copie locale
- svn add : ajouter un fichier
- svn rm : enlever un fichier
- svn mv : déplacer un fichier

Interfaces graphiques

esvn



Qsvn



PLAN

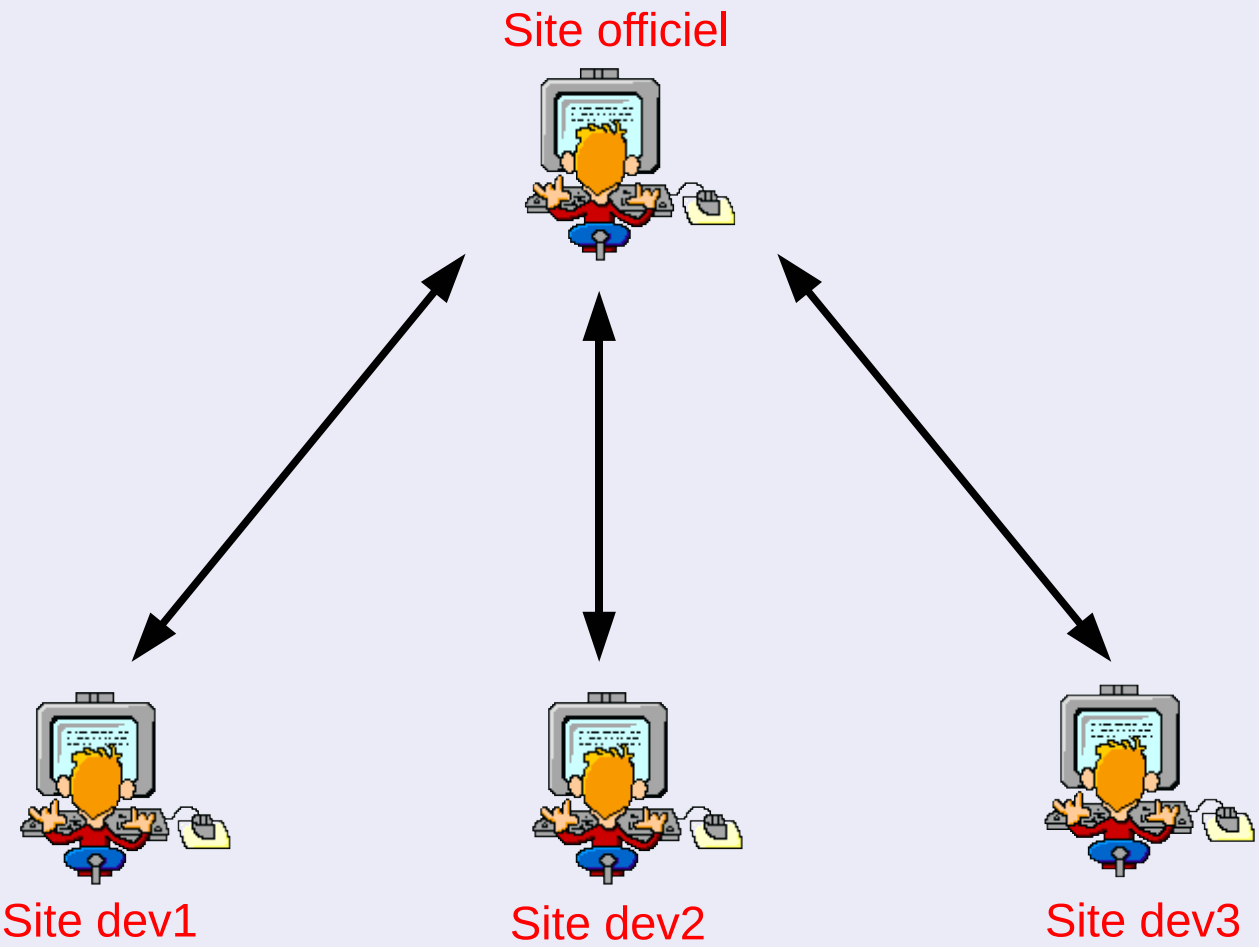
Utiliser un SGV pour le développement web

Objectifs

- Développer, tester sur plusieurs sites de développement sans risque
- Travailler à plusieurs
- Conserver l'historique des modifications
- Publier sur le site officiel quand on veut

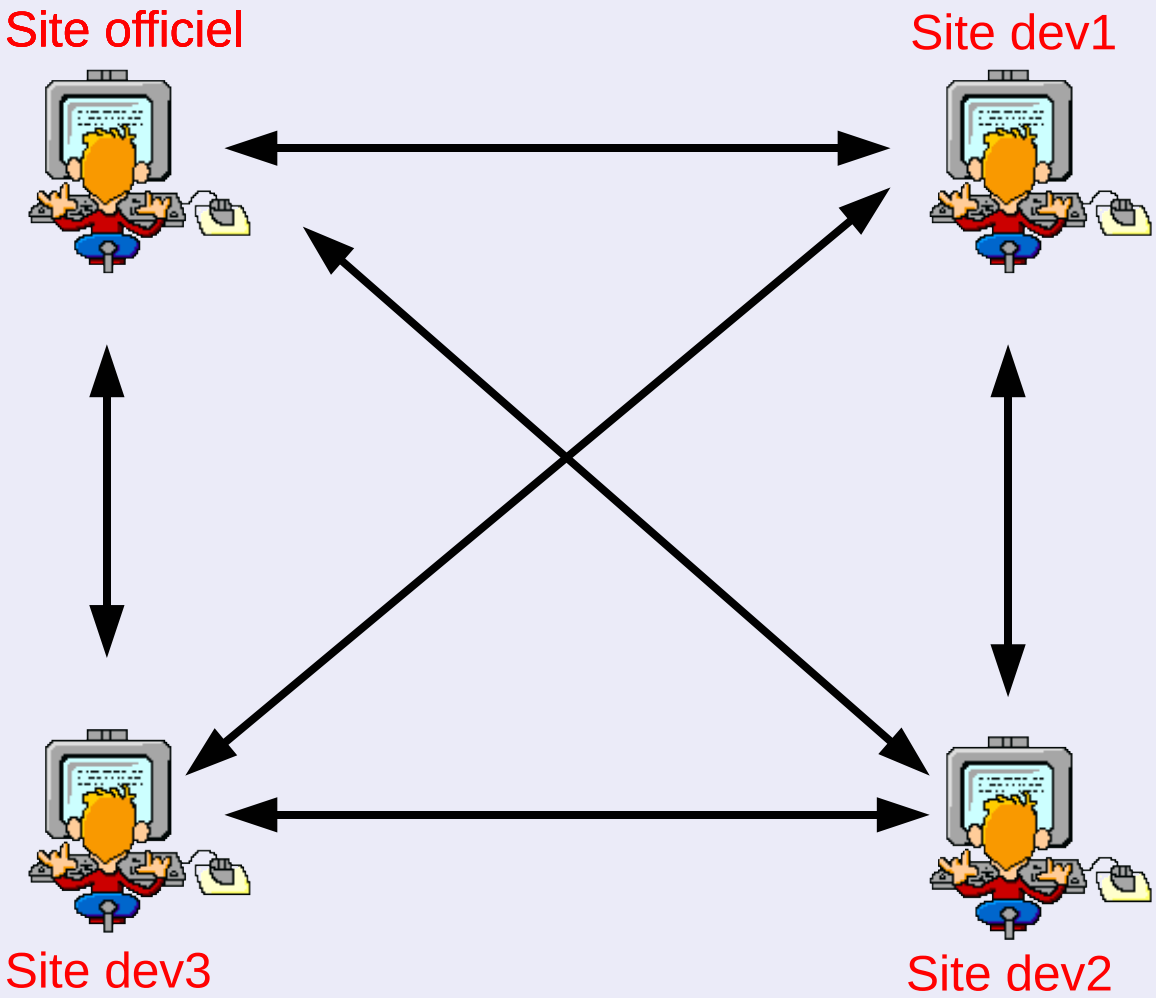
Quel SGV (1) ?

Un SGV centralisé



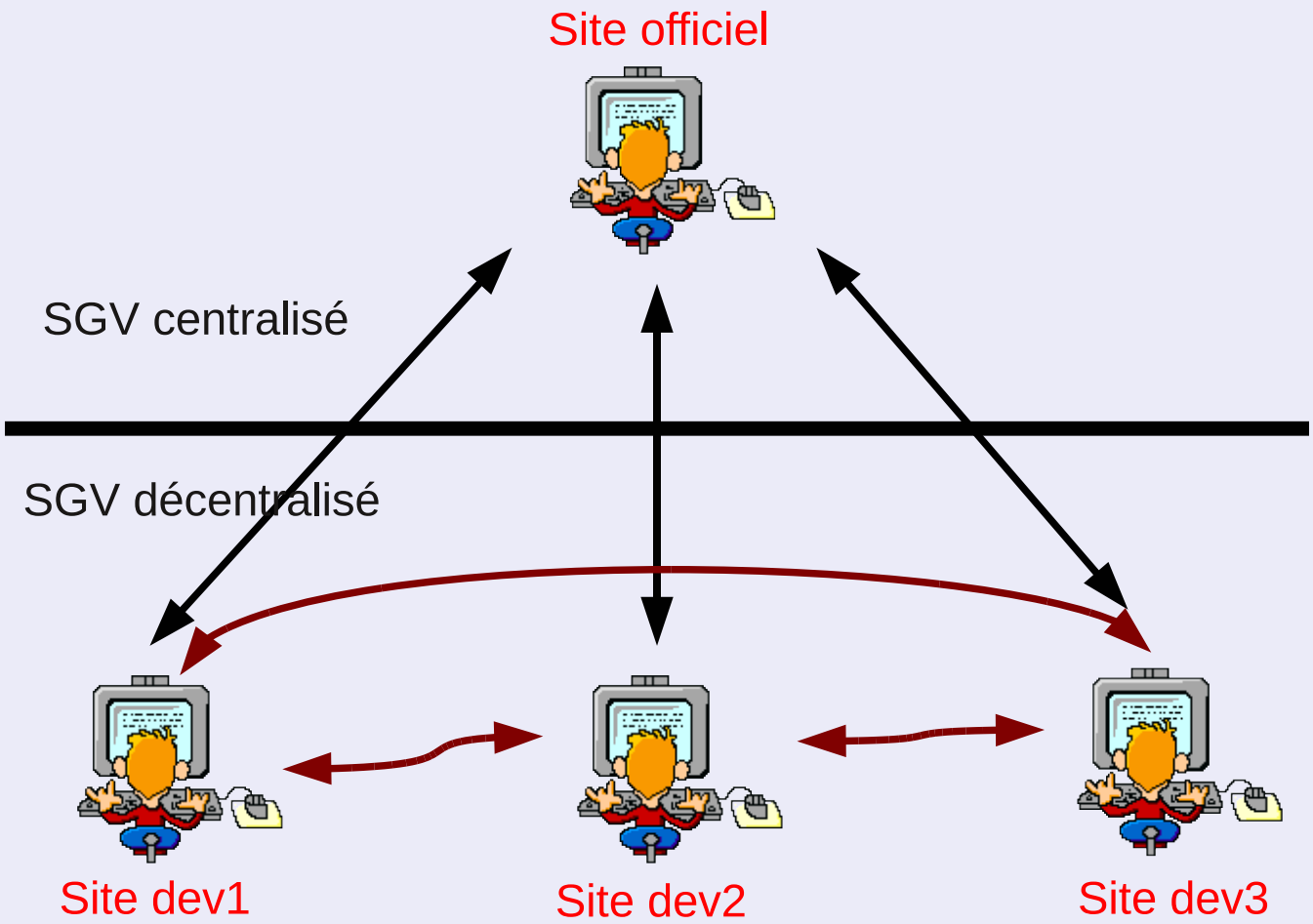
Quel SGV (2) ?

Un SGV décentralisé



Quel SGV (3) ?

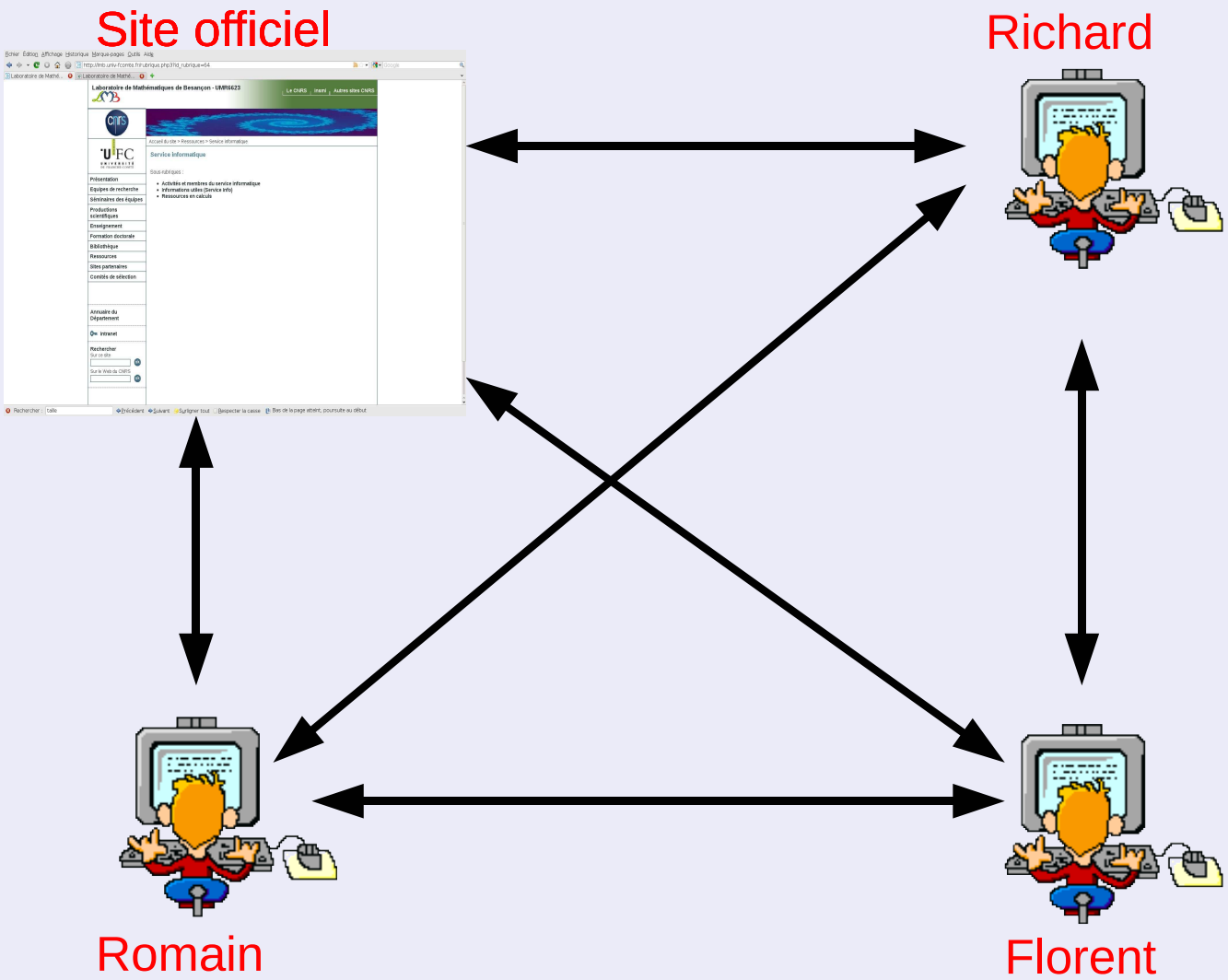
Un SGV centralisé et un SGV décentralisé



PLAN

Utilisation de git

4 sites web sur 4 PC



Mise en place d'un projet git (1)

Sur le site officiel : initialisation

- Initialisation

```
webmaster$ cd /var/www
webmaster$ git init
Initialized empty Git repository in /var/www
```

- Le dépôt (local) est dans le répertoire .git

```
webmaster$ ls -a
.git      intranet      readMe.txt      site_lmb
```

- Il faut ensuite ajouter les fichiers au dépôt

```
webmaster$ git add *
webmaster$ git commit -m 'ajout de tous les fichiers du répertoire /var/www'
```

Mise en place d'un projet git (2)

Sur les PC de développement : clone et configuration

- Clone

```
florent$ cd /var
florent$ git clone ssh ://webmaster@lmb/var/www
Initialized empty Git repository in /var/www/.git/
remote : Counting objects : 195, done.
remote : Compressing objects : 100% (144/144), done.
remote : Total 195 (delta 19), reused 0 (delta 0)
Receiving objects : 100% (195/195), 36.25 KiB, done.
Resolving deltas : 100% (19/19), done.
```

- Le dépôt (local) est dans le répertoire .git

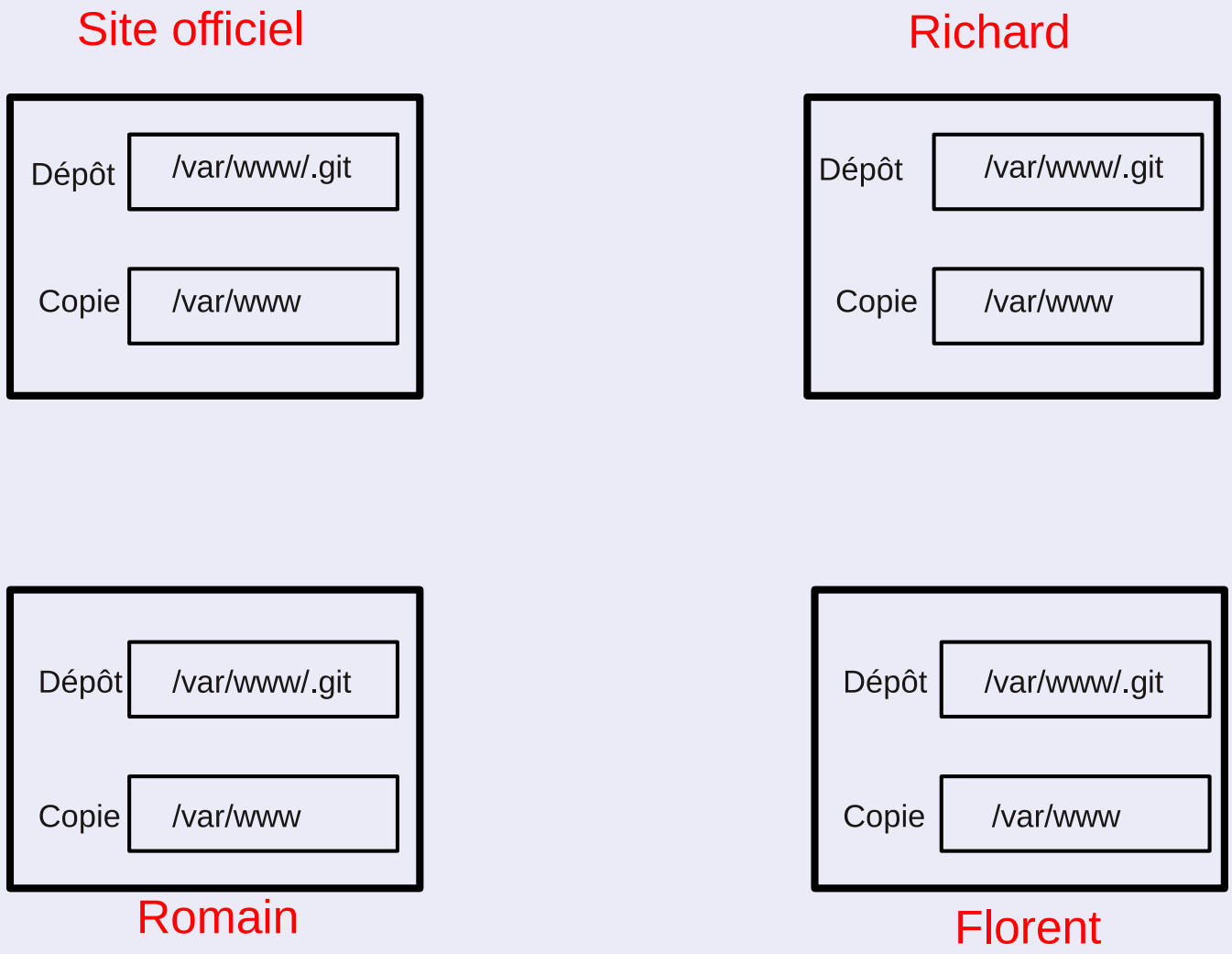
```
florent$ cd www
florent$ ls -a
.git      readMe.txt      intranet      site_lmb
```

- Configuration (cosmétique)

```
florent$ git config –global user.name "Florent Langrognet"
florent$ git config –global user.email "florent.langrognet@univ-fcomte.fr"
```

Dépôts locaux et copies de travail

1 dépôt et 1 copie sur chaque PC



Travail en local...

... entre son dépôt et sa copie

- Modification d'un fichier :

```
florent$ kate readMe.txt
florent$ git status -s
M readMe.txt
```

- Mise à jour du dépôt :

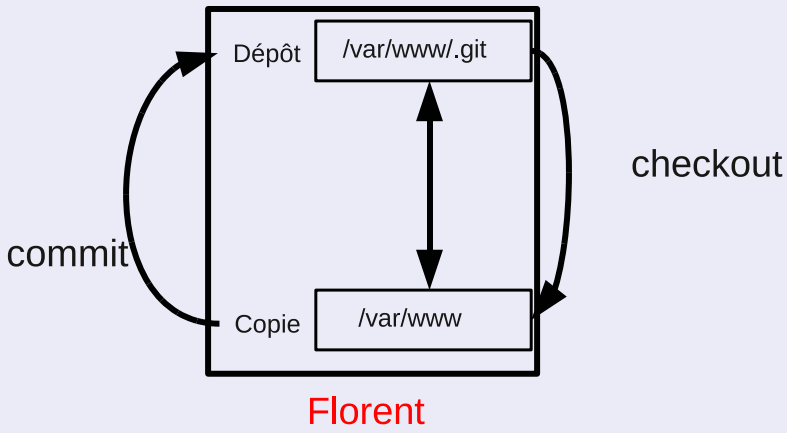
```
florent$ git commit -a -m 'initialisation de readMe.txt'
master 507f3b9 initialisation de readMe.txt
1 files changed, 1 insertions(+), 0 deletions(-)
```

- add, mv, rm via git

```
florent$ git mv f1 f2
```

plutôt que :

```
florent$ mv f1 f2
```



Partage entre 2 développeurs (1)

Partage

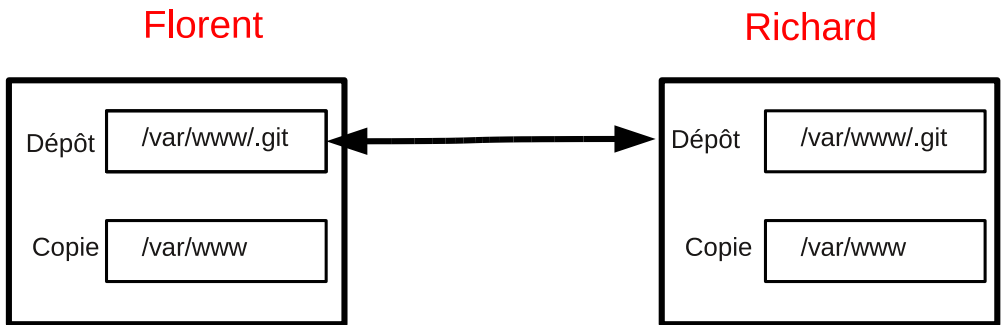
- Liste des machines avec lesquelles je peux travailler :

```
florent$ git remote -v
origin ssh ://webmaster@lmb/var/www (fetch)
origin ssh ://webmaster@lmb/var/www (push)
```

- Ajouter le site de Richard à cette liste :

```
florent$ git remote add site_richard ssh ://florent@pc_richard/var/www
florent$ git remote -v
origin ssh ://webmaster@lmb/var/www (fetch)
origin ssh ://webmaster@lmb/var/www (push)
site_richard ssh ://florent@pc_richard/var/www (fetch)
site_richard ssh ://florent@pc_richard/var/www (push)
```

Partage entre 2 développeurs (2)



Récupérer les développements d'un autre développeur

- Récupérer les modiciations apportées par Richard sur son dépôt : git pull = git fetch + git merge

```
florent$ git pull site_richard master
remote : Counting objects : 5, done.
remote : Compressing objects : 100% (2/2), done
remote : Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects : 100% (3/3), done.
From ssh ://pc_richard/var/www
* branch master -> FETCH_HEAD
Merge made by recursive. file1.txt | 1 +
1 files changed, 1 insertions(+), 0 deletions(-)

florent$ ls -a
.git    file1.txt    intranet    readMe.txt    site_lmb
```

Partage entre 2 développeurs (3)

Pousser ses développements vers un autre dépôt

- Pousser ses propres modifications sur le dépôt de Richard ? (git push)

```
florent$ git push site_richard
fatal : bad config value for 'receive.denycurrentbranch' in ./config
fatal : The remote end hung up unexpectedly
```

Erreur

- Richard doit autoriser les modifications dans son dépôt
  - Configuration git :

```
richard$ git config receive.denycurrentbranch warn
```

  - Droit sur son répertoire .git : ajout dans un groupe autorisé par exemple

- Pousser ses propres modifications sur le dépôt de Richard : git push

```
florent$ git push site_richard
Counting objects : 23, done.
...
remote : warning : updating the current branch
To ssh ://florent@pc_richard/var/www
bc69e0c..139768b master -> master
```

Politique d'autorisation entre les dépôts

Entre les développeurs

Exemples :

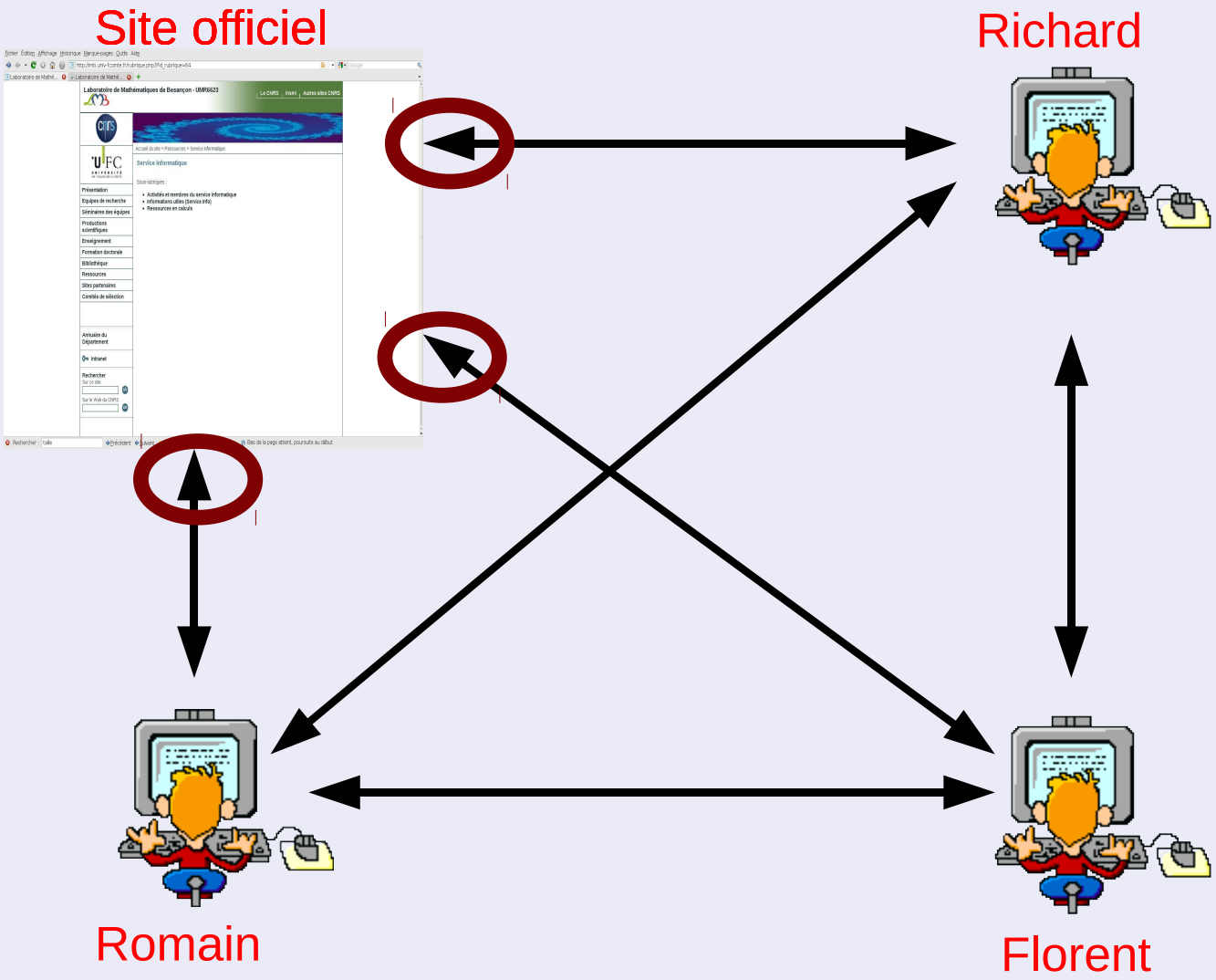
- Pull et Push autorisés
- Pull autorisés et Push interdits

Mise à jour du site officiel

Mise à jour sécurisée

Push interdits - Pull autorisés

git et la sécurité (1)



git et la sécurité (2)

Protocole ssh

- ssh à privilégier plutôt que git, http ou local
  git clone ssh ://user@server/fichier.git
- ssh : protocole par défaut
- démons SSH déjà disponibles sur nos serveurs

Données authentifiées, chiffrées et compressées

Droits et accès ssh

- Git utilise les droits et permissions du système de fichier Le répertoire .git est accessible (par défaut) en lecture (uniquement)
- Possibilité de créer un utilisateur 'git' et autoriser la clé publique de chaque participant sous /.ssh/authorized_keys

Générer des clés publiques et autoriser ces clés entre les co-développeurs

git et la sécurité (3)

Git et infrastructure réseau

- Ouvrir une route si la machine distante n'est pas sur le même réseau !
- Moins rapide si plusieurs réseaux interconnectés et filtrés...

Volume et sauvegarde

Répertoire .git

- Volume de données peut être important
- Sauvegarde de ce sous-répertoire

Les systèmes de gestion de version

Merci à Romain Pacé et Richard Ferrere

FIN

Merci de votre attention