Systèmes de Gestion de Versions

David Ananda, Pierre-Louis Sergent, Matthieu Kirschleger, Bruno Inec

8 novembre 2017

IUT informatique Lyon1

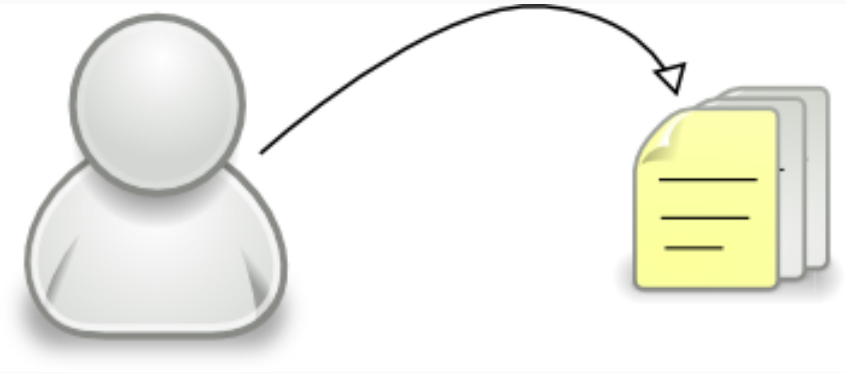
Table des matières

Histoire

SCCS & RCS

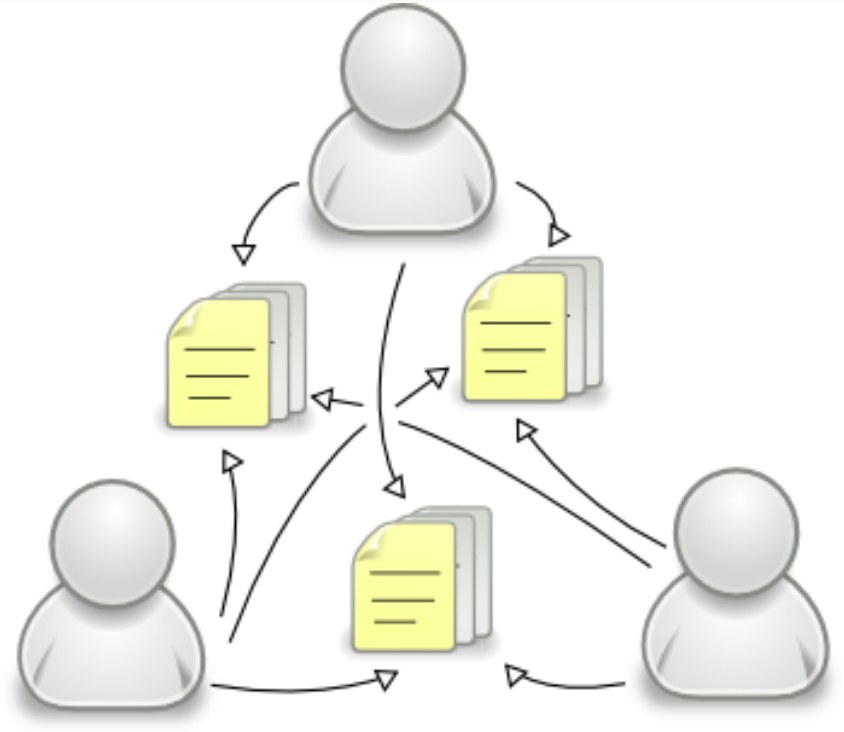
GNU SCCS (Source Code Control System), 1972

GNU RCS (Revision Control System), 1982

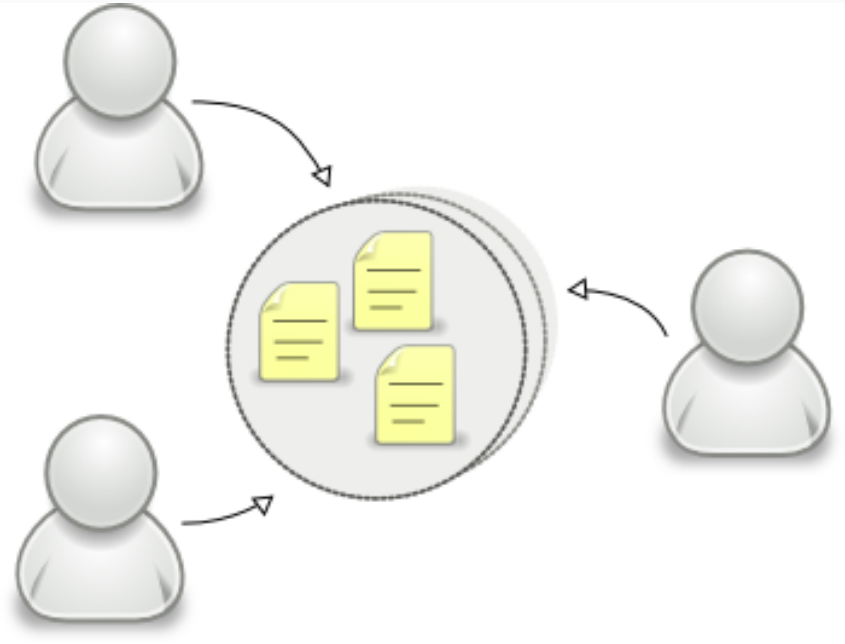


CVS & SVN

CVS (Concurrent Versions System), 1990



SVN (Subversion), 2000

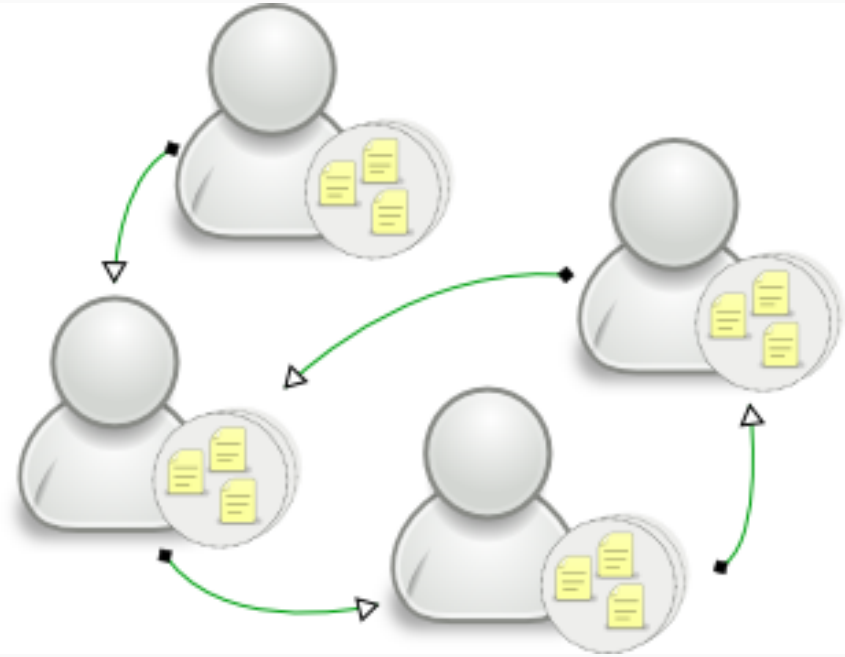


Git, Mercurial & Bazaar

Git, 2005

Mercurial, 2005

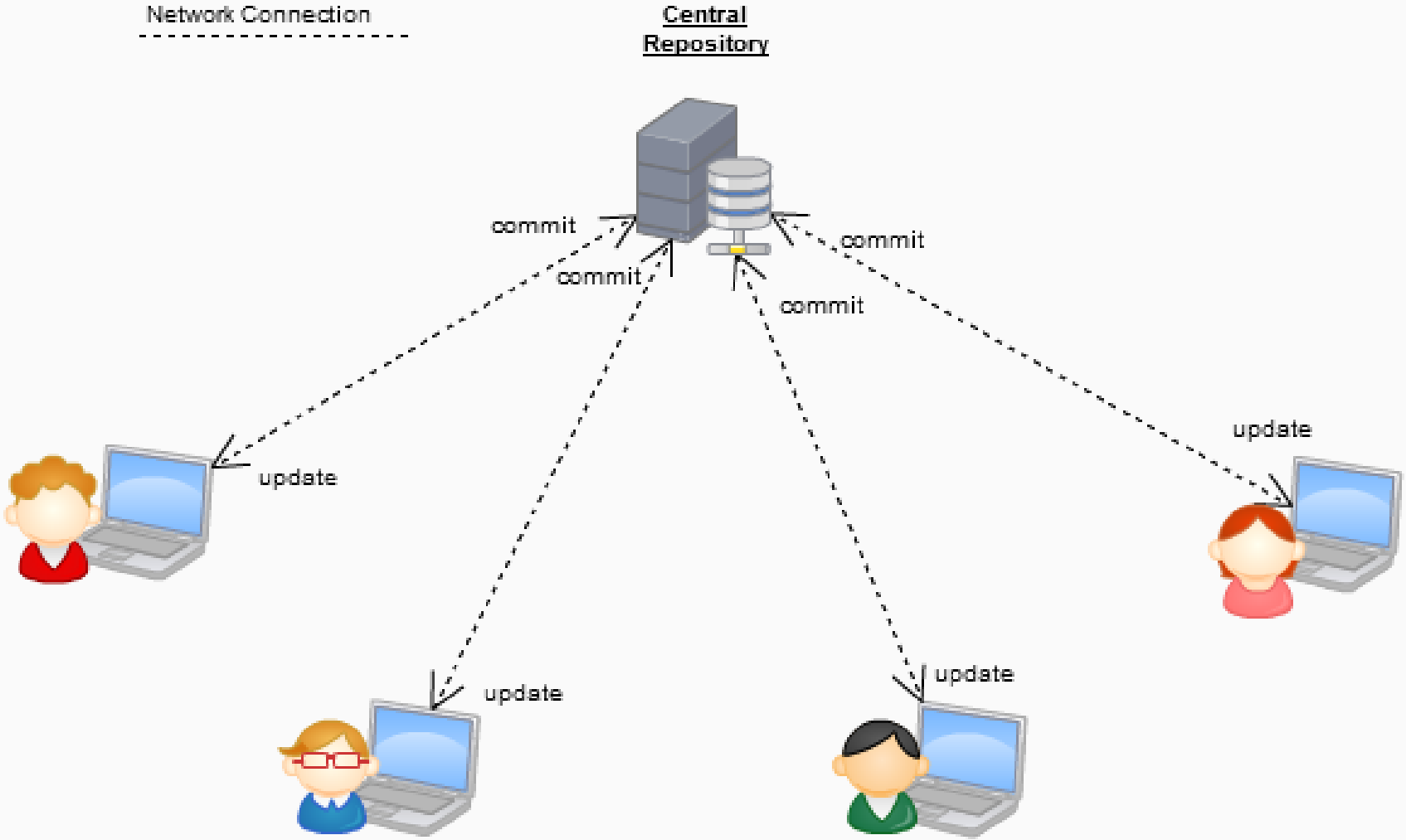
GNU Bazaar, 2005



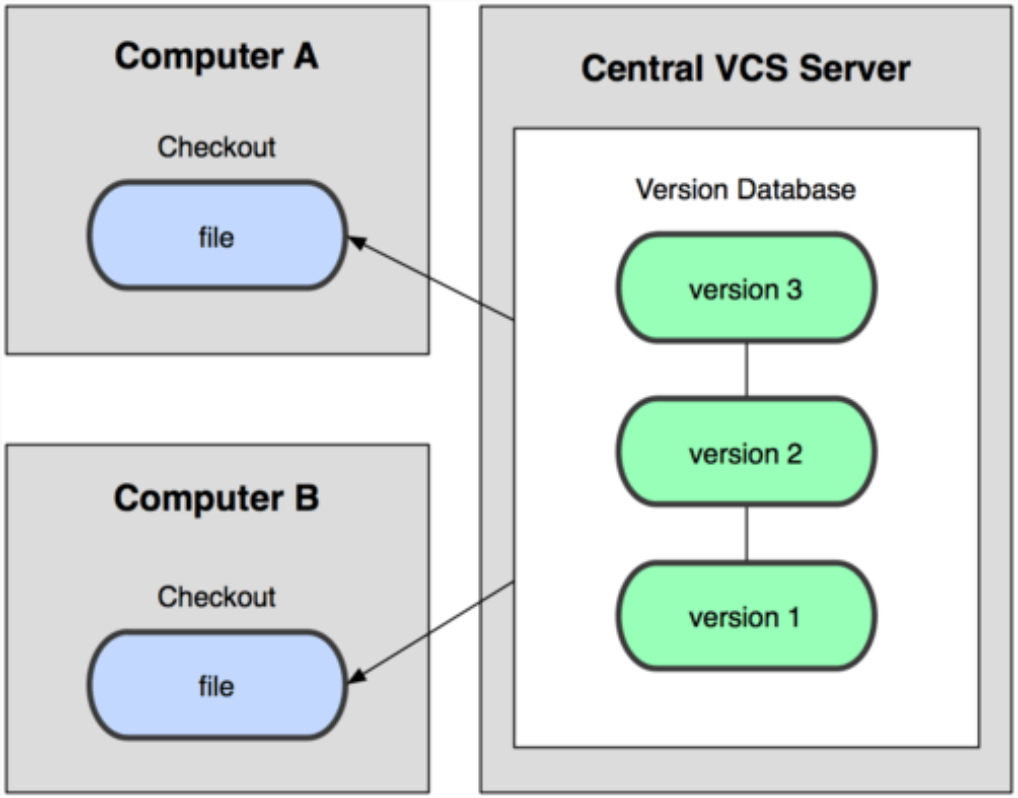
Décentralisé vs Centralisé

Principe de base d'un VCS

Gérer le mécanisme lecture-fusion-écriture



Gestionnaire de versions centralisée CVCS



- Un seul dépôt de référence (serveur)
- Les utilisateurs travaillent sur une copie

Gestionnaire de versions centralisée CVCS

Qualités :

- technologie éprouvée
- largement disponible
- sécurisé

Défauts :

- échange entre les dépôts impossible
- échange entre les copies locales impossible
- travail hors connexion impossible
- dépendant du serveur

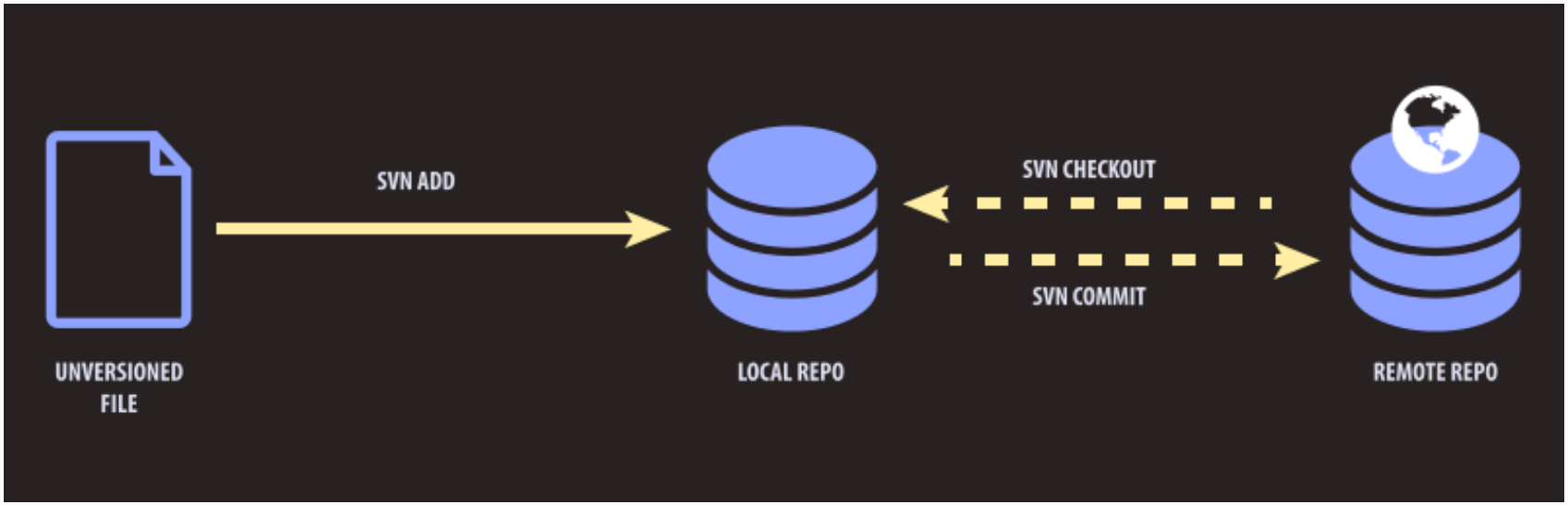
Présentation SVN

Serveur centralisé et unique :

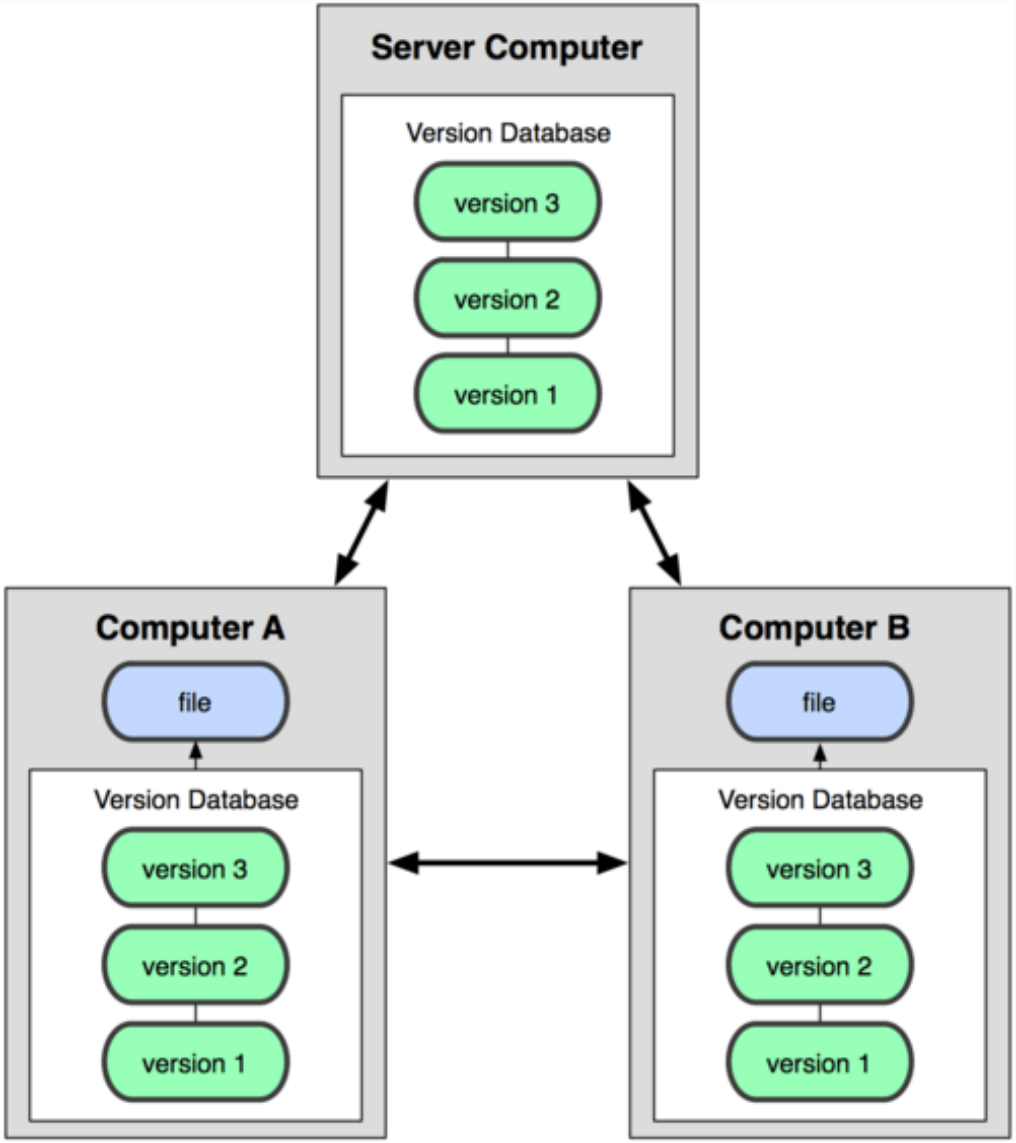
- les fichiers de références (le dépôt ou repository)
- un logiciel serveur SVN tournant en tâche de fond

Postes clients :

- copie locale du repo, éventuellement modifié
- logiciel client permettant la synchronisation manuelle et/ou automatisée entre chaque client et le serveur de ref



Gestionnaire de versions décentralisée DVCS



- Dépôt propre à chaque développeur
- Copie propre à chaque développeur

Gestionnaire de versions centralisée CVCS

Qualités :

- communication possible entre les dépôts
- possibilité de mettre en place un dépôt central (serveur)
- travail hors connexion possible
- indépendant du serveur
- gestion des branches
- gestion des merges

Défauts :

- complexité (Git)

Git vs Mercurial

Format du dépôt

Mercurial

A tout misé sur les logs en append-only, optimiser la recherche sur le disque de nos machines

Git

Stocke chaque commit/fichier dans un simple dépôt de ‘hash’ de documents, chaque commit finira dans ce dépôt comme une entité séparée

Réécrire l'historique

Mercurial

- Édition difficile des commits passés or il est plus facile de voir les chgmts en examinant l'histoire par les commits
- Mercurial Queus permet d'empiler des pré-commits de sorte à pouvoir les réorganiser jusqu'à votre commit final
- Extension équivalente de "interactive rebase" = "histedit" → append-only → génère un fichier de

Git

- Il est facile de « remonter le temps » pour éditer les commits précédents si nécessaire → les logs de commit Git peuvent devenir des récits soigneusement élaborés (voir Mercurial Queus)
- Vous arriverez au moment où vous avez besoin de faire un changement dans votre historique de commits ; vous n'aurez vraiment qu'une seule

Les branches

Git

- Namespace du serveur indique clairement qui est qui

Mercurial

- Extension Bookmark → clone direct des branches Git, au début on pouvait pas faire le push des bookmarks sur le serveur
- Problème : les bookmarks partagent le même namespace

Staging (zone de transit)

Git

- Tout ce qu'on ajoute à un commit passe par cette zone de transit : commande "git add" → appelée aussi index
- On ajoute des modifications, pas aux fichiers eux-mêmes, mais au reflog

Mercurial

- Extension Record
- Doit copier les modifications 'dans un emplacement temporaire → mettre à jour les fichiers de stockage, commit → enfin annuler les modifications
- Erreur = on recommence tout