
Logiciels de gestion de versions

Contexte/Problématique

- 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, mais il est malade
 - Mon projet marchait bien, et j'ai essayé d'implémenter une autre fonction et plus rien ne marche
- Travailler à plusieurs c'est pas facile :
 - « On s'échange le projet par mail et on travaille à tour de rôle dessus. => on n'avance pas vite »
 - « Pour que ça compile, j'ai du écraser les modifications de mon binôme... maintenant il est fâché »
 - « On a travaillé chacun dans notre coin et on n'arrive pas à fusionner notre travail. => rien ne marche à la fin »

Gestionnaire de versions et développement collaboratifs

- Gestionnaire de versions (fonction « Undo »):
 - Historique un peu évolué (annotation).
 - Exemple : les Wiki
- Outils collaboratifs :
 - Framapad (cf. cours de C2I)
 - Google docs/ * Office,... (cf. cours de C2I)
 - Dropbox, ... (OK si vous êtes tout seul)
- Outils collaboratifs sans gestionnaire de version ?
 - => L'équipe explose très vite...

WIKIPÉDIA
L'encyclopédie libre

- Portails thématiques
- Index alphabétique
- Article au hasard
- Contacteur Wikipédia

- [Premiers pas](#)
- [Aide](#)
- [Communauté](#)
- [Modifications récentes](#)
- [Faire un don](#)

Lance Armstrong : Historique des versions

— Naviguer dans l'historique

Outils ext

Liste des auteurs - Rechercher l'auteur d'un passage de l'article - N

Suppression - Neutralité - Droit d'auteur - [Article de loi](#)

Légende : (actu) = différence avec la version actuelle - (diff)

Comparer les versions sélectionnées

20-june-2012 à 15:00 - Ann. Gabriel (disputed contribution)

- (actu | diff) 30 janvier 2013 à 13:02 Ange Gabriel (discuter | contributions)
- (actu | diff) 27 janvier 2013 à 14:40 Lomita (discuter | contributions) **m** ...
version de Axel33430)) (défaire)
- (actu | diff) 27 janvier 2013 à 14:39 81.53.247.187 (discuter) ... (107 940 oct)
- (actu | diff) 27 janvier 2013 à 14:33 Axel33430 (discuter | contributions) ...
dernière version de Freva)) (défaire)
- (actu | diff) 27 janvier 2013 à 14:32 81.53.247.187 (discuter) ... (107 921 oct)
- (actu | diff) 21 janvier 2013 à 17:25 Freva (discuter | contributions) **m** ... (1)
(défaire)
- (actu | diff) 24 janvier 2013 à 00:25 76eBot (discuter | contributions) ...

Public Document

Options

Import/Export

Saved revisions

Time Slider

1 Warum sollte man Etherpad mit //SEIBERT/MEDIA im
eigenen Unternehmen installieren?

2

3

- Die Software ist sicher und der Quellcode liegt offen
(Open Source).
- Die Installation ist einfach und schnell durchgeführt.
- Auf Wunsch //SEIBERT/MEDIA Schulungen vor Ort
durch und hilft dabei, den Einsatz im Unternehmen zu
gewährleisten.
- Wir bieten auch eine Online-Installation auf unseren
Sernern, die über eine SSL-Verschlüsselung und eine IP-
Beschränkung nur für Ihr Unternehmen zugänglich
gemacht wird.
- Wir beraten Sie beim Einsatz mit anderen Web 2.0-
Applikationen wie Wikis, Microblogs und Jira im
Unternehmen.
- Außerdem helfen wir Ihnen dabei eine sinnvolle
Abgrenzung der Systeme zu schaffen. Wir
programmieren bei Bedarf Schnittstellen zu bestehender
Software.

4

5

6

7

8

9

10

11

Zoom: 115%

Martin Seibert

Paul Herwarth

Sebastian Preuss

Share this

May 15, 2010

Sebastian Preuss: Über den Chat können alle 22:42
Teilnehmer leicht und schnell mit einander über den
Text kommunizieren.

Paul Herwarth: Die Zusammenarbeit muss 22:43
nicht zwangsläufig in Echtzeit erfolgen. Pads können
auch asynchron bearbeitet werden.

Martin Seibert: Etherpad ist heute die 22:44
kostengünstigste und ausgereifteste Echtzeit-
Kommunikationsplattform, die für Unternehmen
verfügbar ist.

Chat:

Sidebar

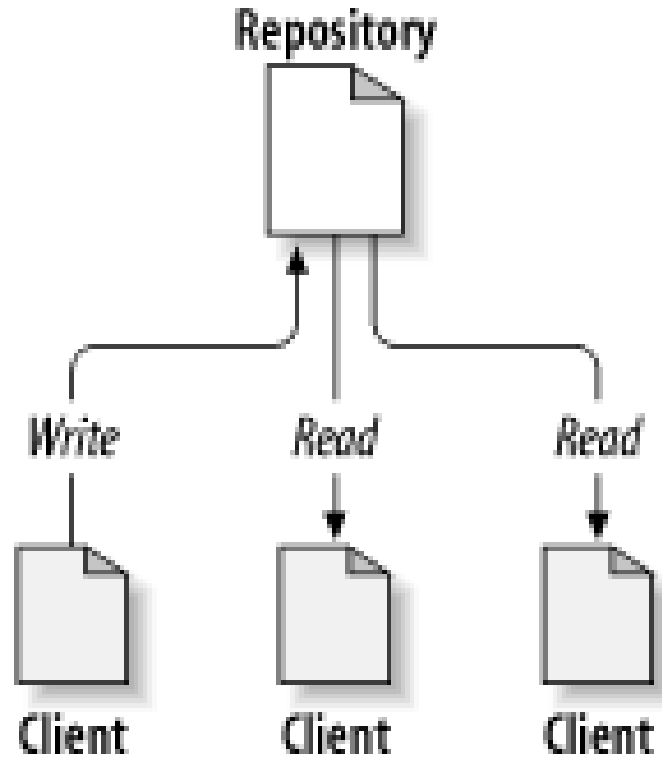
Full Window

Gestionnaires de version pour le développement

- Centralisés :
 - RCS (mort ?)
 - CVS (presque mort)
 - **subversion** (mature, largement déployé)
 - ...
- Décentralisés :
 - **git** (mature, croissance ... mais plus complexe)
 - Mercurial
 - Quelques autres...

Modèle client-serveur

Les informations partagées sont stockées dans un dépôt (« repository ») sur le serveur central. Chaque utilisateur travaille dans une copie locale.

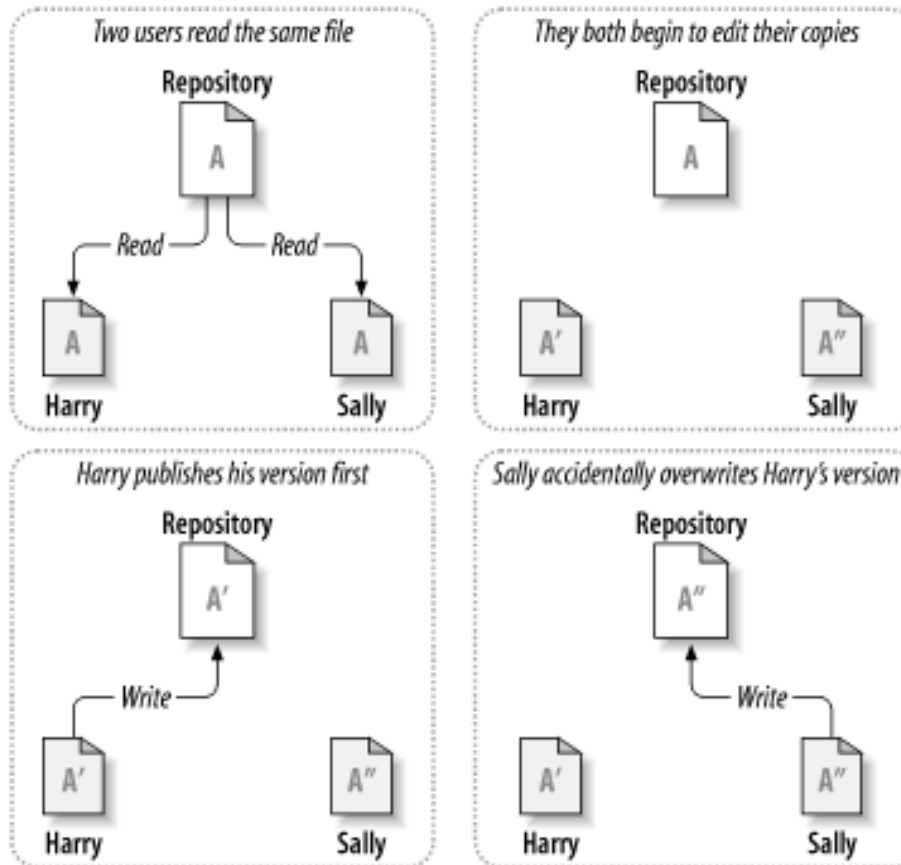


Source : "Version Control with Subversion"

Modèle client-serveur

Les problèmes arrivent vite...

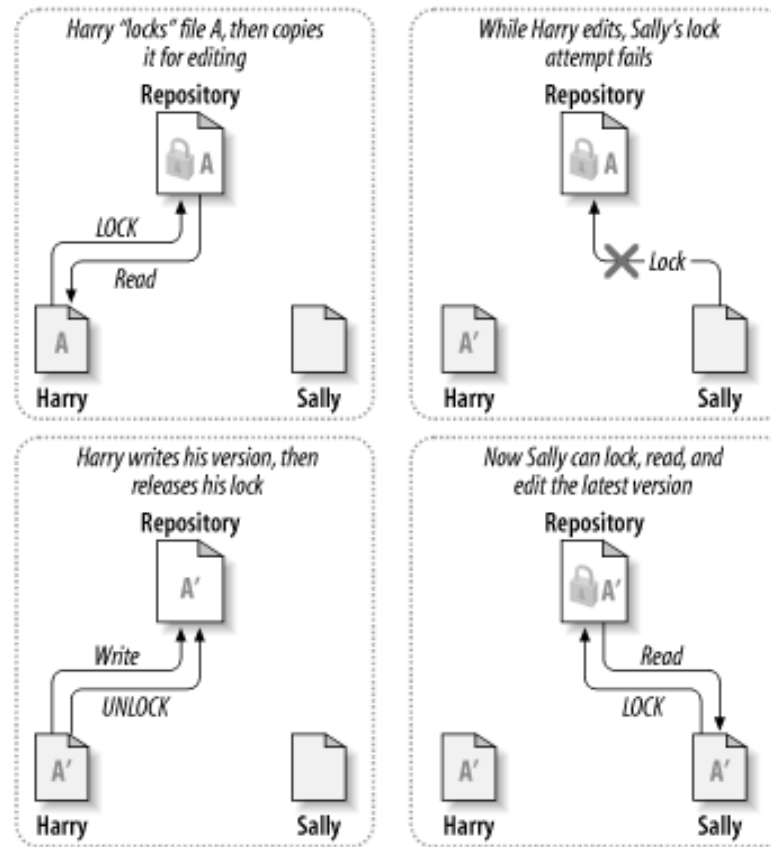
Exemple de scénario où une partie du travail est perdue :



Source : "Version Control with Subversion"

Gestionnaire de versions

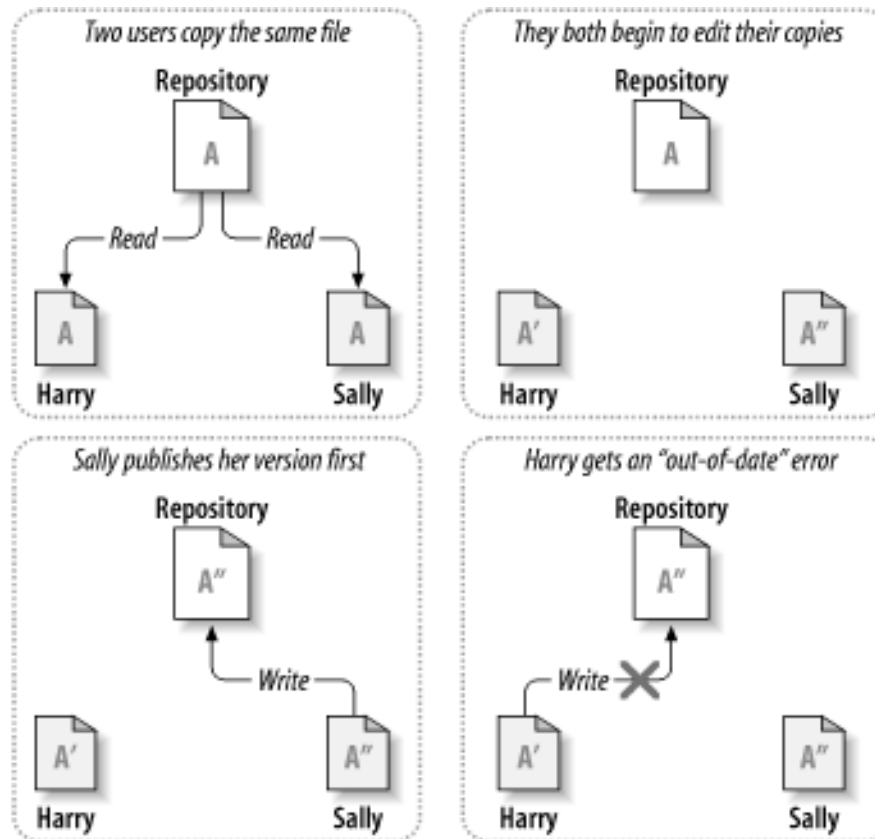
- Solution 1 : verrouiller pendant le travail (« lock-modify-unlock »)



Ca marche, mais impossible de travailler en parallèle sur un même fichier.

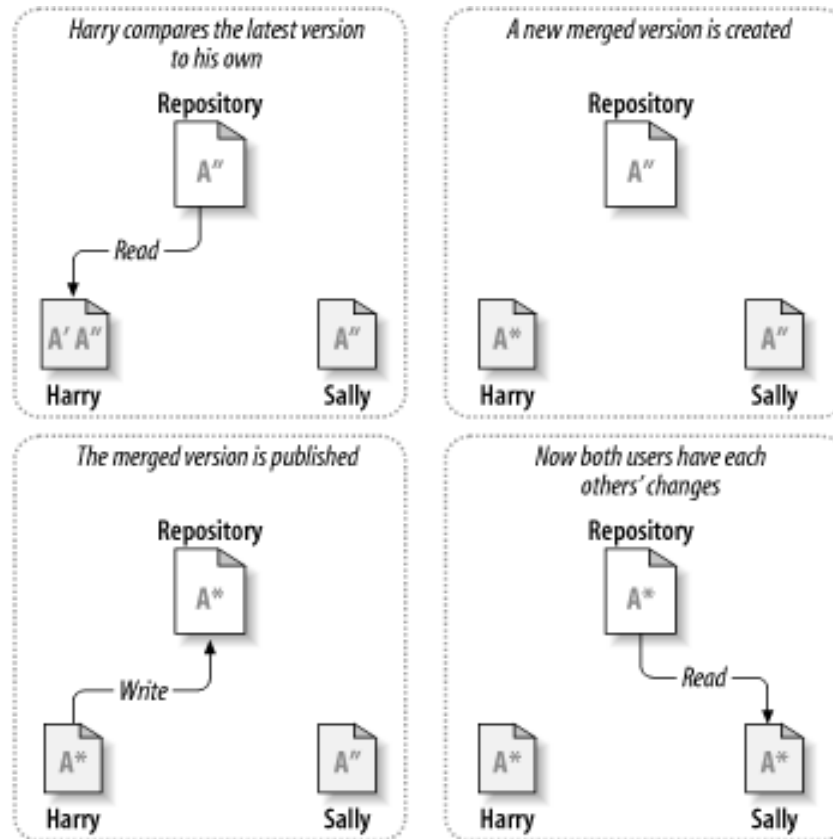
Gestionnaire de versions

- Solution 2 : travailler sur une copie (« copy-modify-merge ») (1/2)



Gestionnaire de versions

- Solution 2 : travailler sur une copie (« copy-modify-merge ») (2/2)



Ca marche, mais il faut gérer ensuite les conflits à la main.
C'est la solution retenue. Bonne nouvelle : en pratique les conflits se règlent automatiquement ou assez facilement à la main.

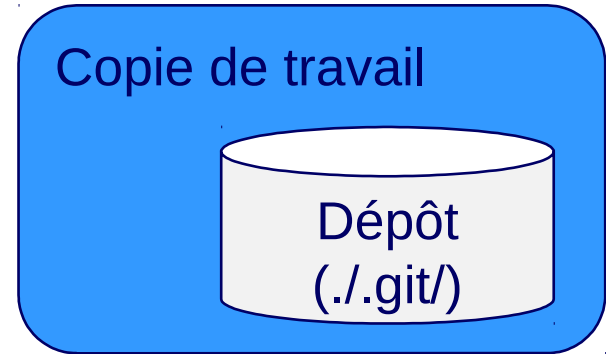
Légère introduction à Git

Git

- Développé par Linus Torvalds (créateur de Linux) en 2005.
- Décentralisé
 - Chaque machine possède un dépôt local (<Monprojet>/.git/) associé à sa copie locale
 - Permet d'interroger le dépôt même sans le net
 - Beaucoup plus rapide
 - Généralement il y a 1 dépôt central et des dépôts secondaires
 - Les manipulations de base sont plus complexes qu'avec svn...

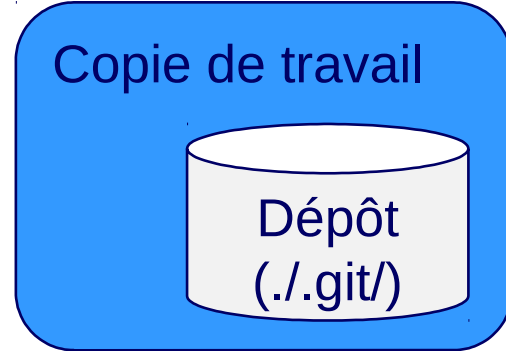
Création d'un dépôt (en local : 1 développeur)

- `cd monProjet`
- `git init`
- `git add *.c`
- `git commit -m «première version »`



« commiter » une nouvelle version

- def **Committer** : publier une nouvelle version dans un dépôt.
- Avec Git cela se fait en 2 temps
 - 1) mettre dans l'**index** (également appelé **stage**) la liste des fichiers à commiter. Cela se fait avec la commande **git add**
 - 2) Envoyez ces fichiers dans le dépôt pour en faire une nouvelle version. Cela se fait avec la commande **git commit**.



Exemple

- emacs main.c matrice2d.c
- git add main.c matrice2d.c
- git commit -m «correction du bug N° 23»

Historique et Nommage des révisions

- La commande **git log** permet de consulter l'historique des révisions.
- Chaque révision(commit) est identifié par une clé de hachage. Ex : `git log`
 - (...)
 - commit cba66fbca3c4751d0a056aba2adf99ba63cb7122
 - Author: niko
 - Date: Mon Feb 1 11:07:46 2012 +0100
 - (...)
- C'est pénible, mais il y a des raccourcis
 - Préfixe : cba66fb (OK, si c'est le seul commit avec ce préfixe)
 - HEAD : dernier commit
 - HEAD~2 : avant-avant dernier commit
 - Cba66fb~2 : 2 commit avant le commit cba66fb...
 - ...

gitk : pour visualiser l'historique

The screenshot shows the gitk application window titled "bulto2: All files - gitk". The interface includes a menu bar (Fichier, Éditer, Vue, Aide) and a commit history list on the left. The history list shows a series of commits with their messages and authors. The selected commit is c9ee5f5cd0643b933849bdaa16daec0aab8b7b, which is a merge of the master branch. The bottom pane displays the diff for the selected commit, showing changes to the file "ens/src/game_solver.c". The diff includes a new function "dead_end" and a comment in French.

Fichier Éditer Vue Aide

- ajout de commentaires dans game
Merge branch 'master' of https://ser
essai que ca marche au CREMI ET s
Merge branch 'master' of https://ser
input pour ca compile au CREMI.
TTF en minuscule pour que ca marc
quelques typos dans les .h
c'est pas fini mais on peut déjà joue
ajout croix sur le noeud quand il a t
Je suis passée au "render" qui per
ça affiche des ponts quand on cliqu
mis load game from filename dans

Nicolas Bonichon <bonichon@labri.fr>
Paul Dorbec <paul.dorbec@u-bordeaux.fr>
idurand <idurand@labri.fr>
Paul Dorbec <paul.dorbec@u-bordeaux.fr>
irdurand <irene.durand@u-bordeaux.fr>
irdurand <irene.durand@u-bordeaux.fr>
Paul Dorbec <paul@dorbec2.labri.fr>
idurand <idurand@labri.fr>
idurand <idurand@labri.fr>
idurand <idurand@labri.fr>
idurand <idurand@labri.fr>
idurand <idurand@labri.fr>
idurand <idurand@labri.fr>

2016-09-23 16:25:09
2016-09-23 15:32:52
2016-09-23 15:33:51
2016-09-23 15:30:49
2016-09-23 14:42:37
2016-09-23 14:35:22
2016-09-23 15:28:34
2016-09-22 18:39:54
2016-09-22 17:37:33
2016-09-21 18:03:05
2016-09-20 09:49:27
2016-09-19 15:19:53

Id SHA1 : c9ee5f5cd0643b933849bdaa16daec0aab8b7b ← → Colonne 3 / 238

Recherche ↓ ↑ commit contient : Exact Tous les champs

Rechercher

◆ Diff ◆ Ancienne version ◆ Nouvelle version Lignes de contexte: 3 Ignorer les modifications

Auteur: Nicolas Bonichon <bonichon@labri.fr> 2017-03-02 11:35:13
Auteur du commit: Nicolas Bonichon <bonichon@labri.fr> 2017-03-02 11:35:13
Parent: 7393f8b7b2929aaddbb643b9f7657bf7a87b0759 (Merge branch 'master' of https://services.emi
Branche: master
Suit: V1
Précède:

petit clean

----- ens/src/game_solver.c -----
index c685866..49f3e28 100644
@@ -59,41 +59,6 @@ static bool dead_end (cgame g) {
 return false;
}

-// // Niko: teste toutes combinaisons possibles de pont pour tous les noeuds à partir du num_m
// // et pour le num_m à partir de la direction du nœud

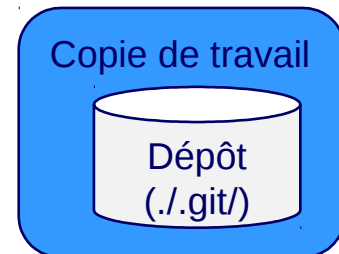
◆ Patch ◆ Arbre
Commentaires
ens/src/game_solver.
c

Oups !

- Récupérer la dernière version (HEAD) d'un ou plusieurs fichiers
 - `git checkout -- <monfichier>`
 - `git checkout -- «*.c»`
- Annuler le/les dernier(s) commit(s) :
 - `git revert HEAD` (Crée un nouveau commit qui annule le dernier commit)
 - `git revert HEAD~n` (crée un nouveau commit qui annule les n+1 derniers commits)

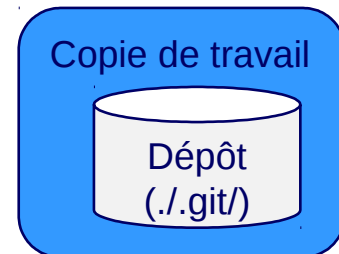
git : développer à plusieurs

- Création du dépôt principal



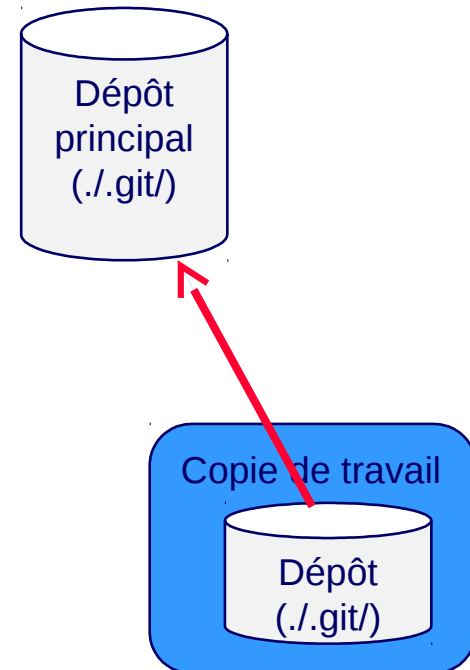
git : développer à plusieurs

- Création du dépôt principal
 - `ssh monServeur`
 - `cd monProjet`
 - `git init --bare`



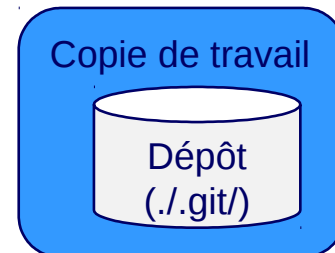
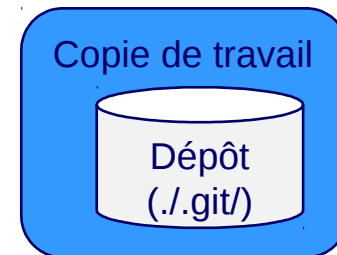
git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD



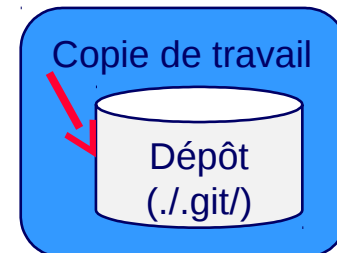
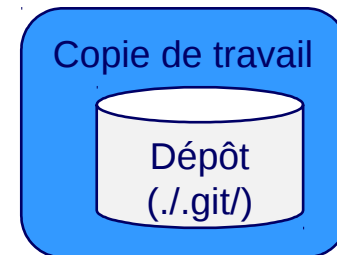
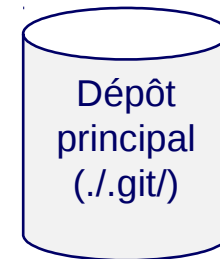
git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD
- Création d'un deuxième dépôt local
 - git clone ssh://monServeur/monProjet



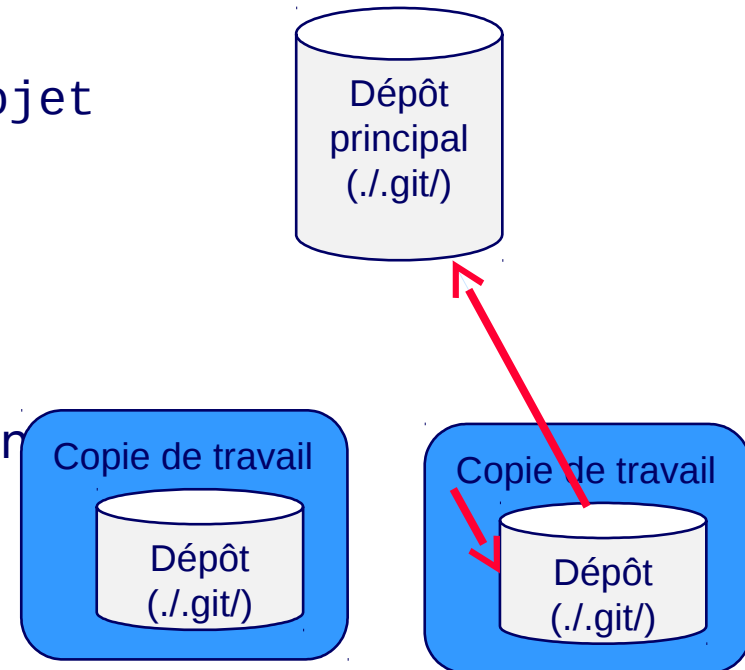
git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD
- Création d'un deuxième dépôt local
 - git clone ssh://monServeur/monProjet
- Publication de ses modifications
 - git add <les fichiers modifiés>
 - git commit -m « ... »
(-> dépôt local)



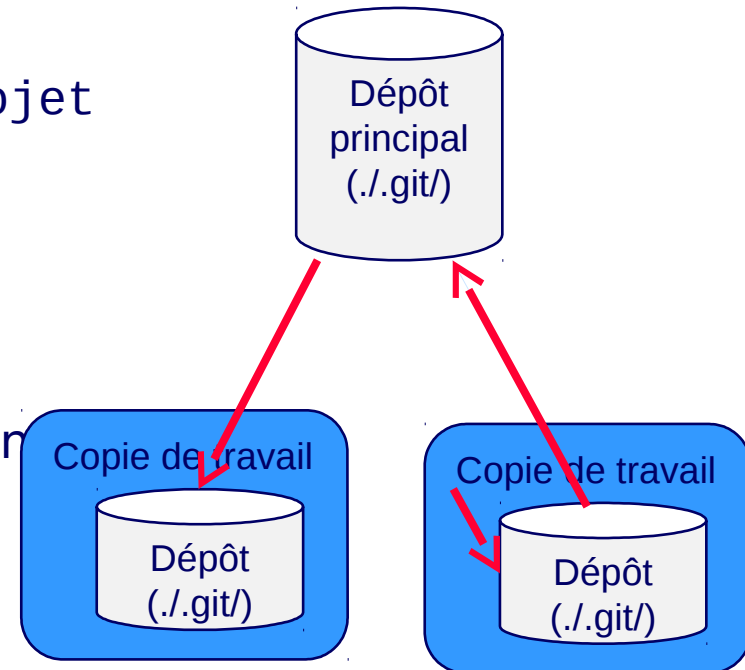
git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD
- Création d'un deuxième dépôt local
 - git clone ssh://monServeur/monProjet
- Publication de ses modifications
 - git add <les fichiers modifiés>
 - git commit -m « ... »
(-> dépôt local)
 - git push (-> recopie dans le dépôt central)



git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD
- Création d'un deuxième dépôt local
 - git clone ssh://monServeur/monProjet
- Publication de ses modifications
 - git add <les fichiers modifiés>
 - git commit -m « ... »
(-> dépôt local)
 - git push (-> recopie dans le dépôt central)
- Récupération des commits des autres
 - git pull



git : développer à plusieurs

- Création du dépôt principal
 - ssh monServeur
 - cd monProjet
 - git init --bare
- Copie du dépôt local
 - git push ssh://monServeur/monProjet HEAD
- Création d'un deuxième dépôt local
 - git clone ssh://monServeur/monProjet
- Publication de ses modifications
 - git add <les fichiers modifiés>
 - git commit -m « ... »
(-> dépôt local)
 - git push (-> recopie dans le dépôt central)
- Récupération des commits des autres
 - git pull
 - <résolution des conflits...>
 - git commit -m « ... »
 - git push

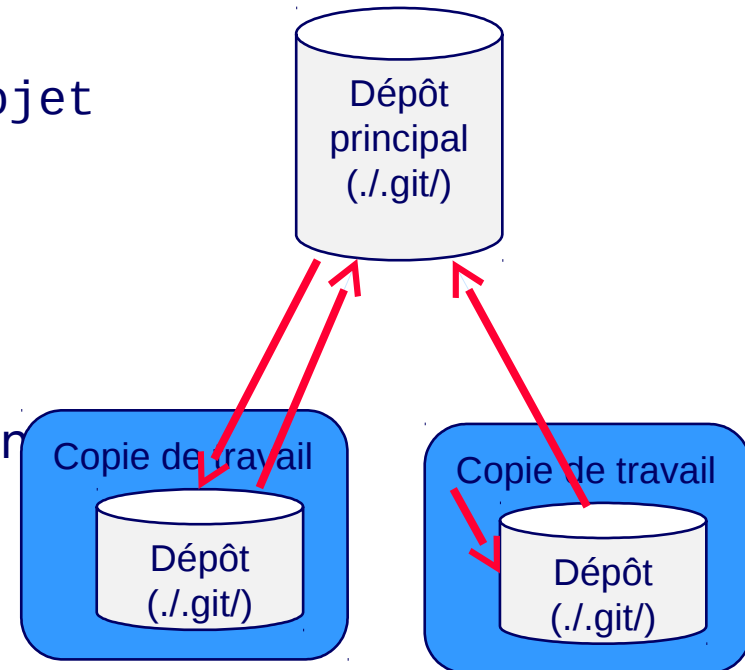
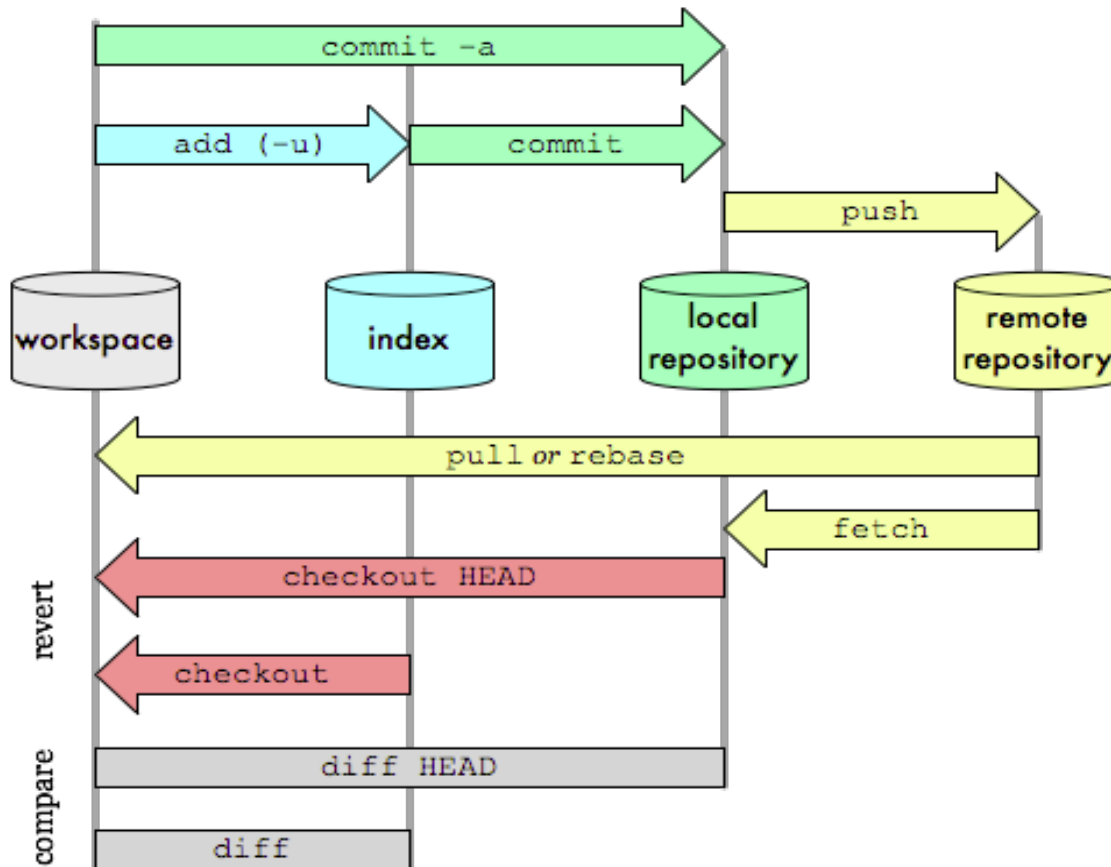


Schéma global

Git Data Transport Commands

<http://osteele.com>



Bonnes pratiques

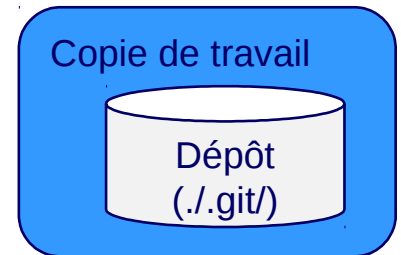
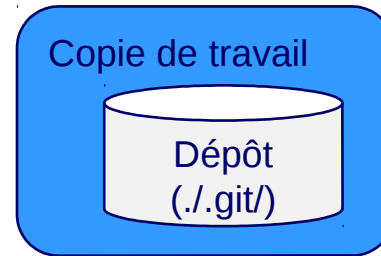
- Ne commitez que des versions qui compilent correctement (sans erreur et sans warning).
 - sinon vous risquez de bloquer les autres...
- Commentez chaque commit !
 - C'est important pour le suivi des modifications. (quel commit un corrigé quelle erreur / quel commit a introduit telle ou telle fonctionnalité / ...)
- « Commit Early, Commit Often » Commitez rapidement, commitez souvent.
 - Cela permet de mieux tracer les modifications
 - Cela permet de diminuer les conflits
 - Mais ne commitez que des choses qui compilent !

git : développer à plusieurs avec Savane

- Le dépôt principal est déjà créé sous savane.
- Il suffit que chaque développeur clone le dépôt
 - `git clone ssh://...`
- Publication de ses modifications
 - `git commit -m « ... »`
(-> dépôt local)
- Récupération des commits des autres
 - `git pull`
 - <résolution des conflits...>
 - `git commit -m « ... »`
 - `git push`
 -
- Pour ne plus taper son mot de passe à chaque fois
 - `git config credential.helper store`

git : développer à plusieurs (suite)

- Généralement on utilise des branches.
- La branche principale est « master »
- Généralement on crée une branche (`git branch maBranche`) et on travaille dessus (plusieurs commits).
- Ensuite on fusionne la branche avec « master »
 - `git checkout master`
 - `git merge mabranche`



Références

- Subversion :
 - Livre "Version Control with Subversion" :
disponible en ligne <http://svnbook.red-bean.com/>
- git :
 - <http://git-scm.com/>
 - Exposé de Linus Torval sur Git :
<https://www.youtube.com/watch?v=4XpnKHJAok8>
 - Présentation du sed (Inria) :
http://sed.bordeaux.inria.fr/seminars/GIT_20100511.pdf
 - Cours de Julien Sopena :
<http://julien.sopena.fr/enseignements/M2-SAR-Git/cours/01-Git/01-Git.pdf>