Tutorial GIT

Sébastien Valat 02/06/2010

Plan

- GIT dans les grandes lignes
- Les bases (svn vs git)
- Les tags (svn vs git)
- Les branches
- Effacer nos traces
- Collaboration
- Mise avec pratique : implémentation de malloc

GIT dans les grandes lignes

Une rapide description de GIT

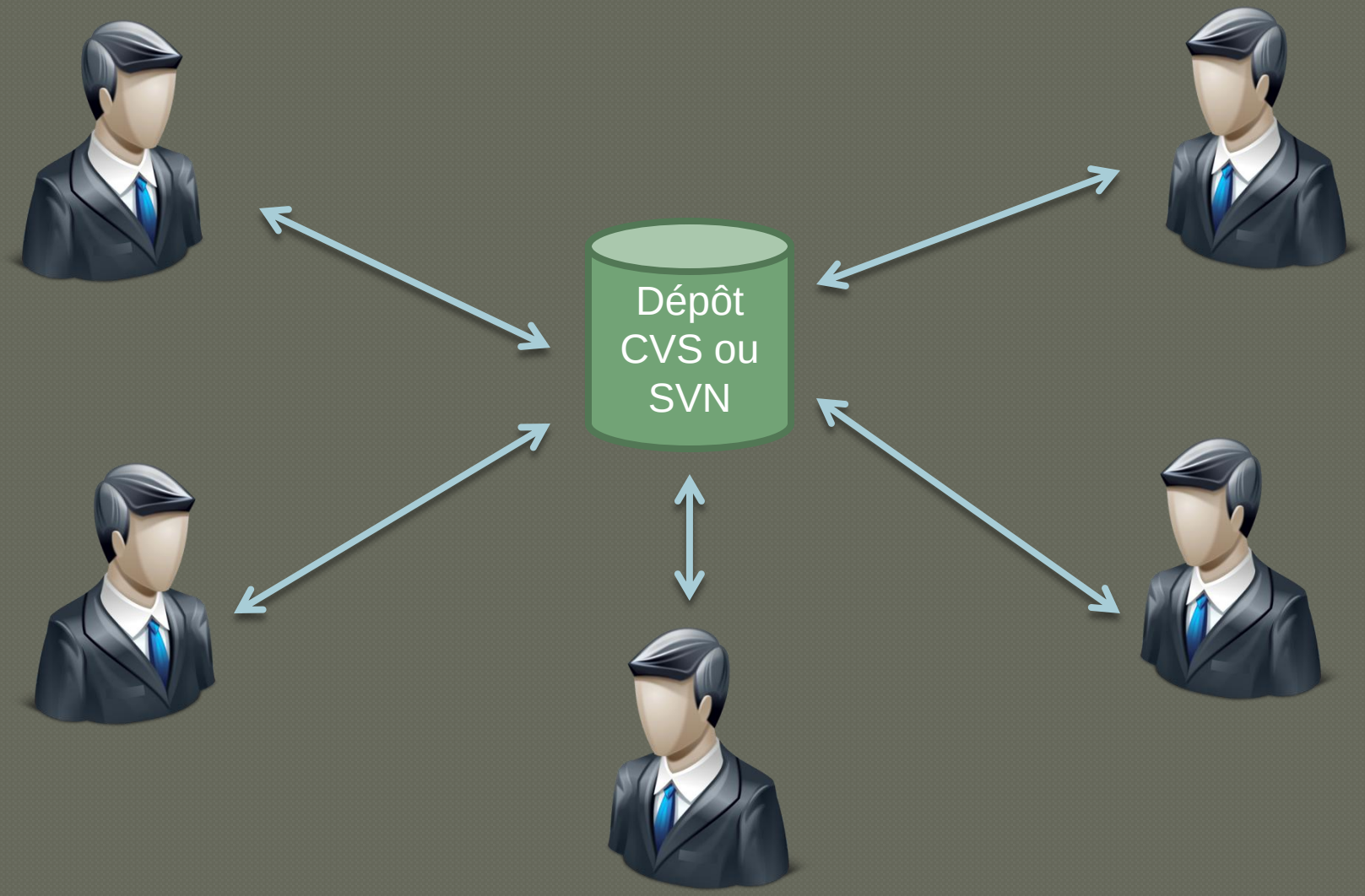
Histoire de Git

- **2002** : BitKeeper est utilisé pour le noyau Linux
- **Avril 2005** : BitKeeper change de licence
- En réponse Linus Torvald propose un prototype de remplacement : Git
- **Avril 2005** : Développement initial
- **Juin 2005** : Utilisation pour la version 2.6.12 du noyau

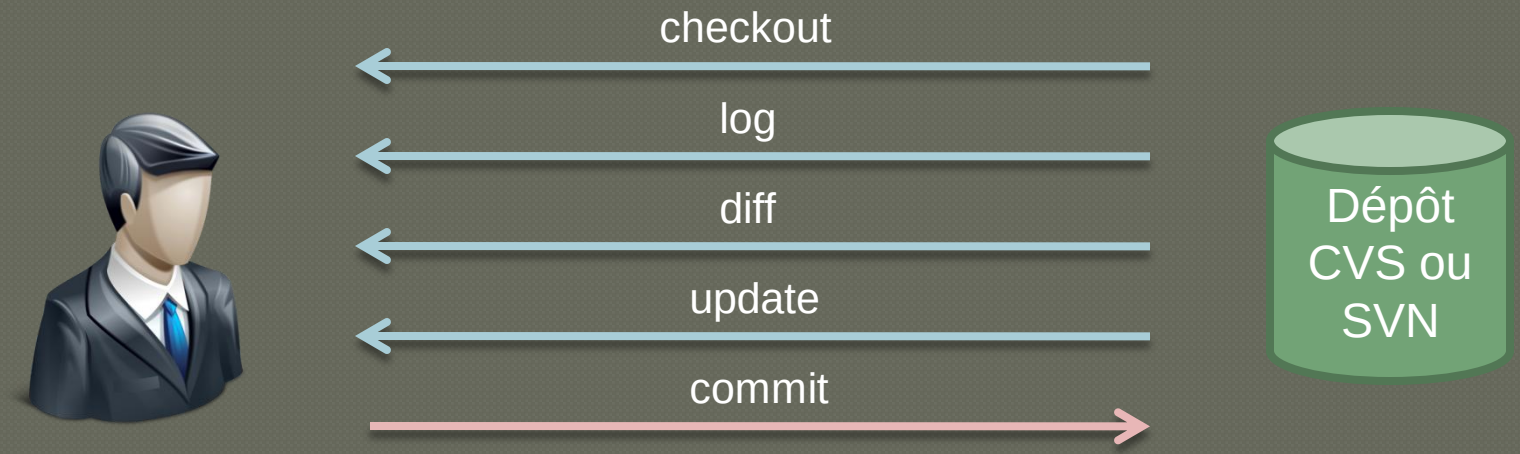
Les choix de conception de Git

- Pensé pour le développement du noyau
- Besoin de supporter de nombreux accès
- Système distribué
- Détection de corruption
- Développement hautement non linéaire
- Open Sources
- Performances (Linus réalise jusqu'à 100 merges par jours [2])

Approche SVN : centralisée

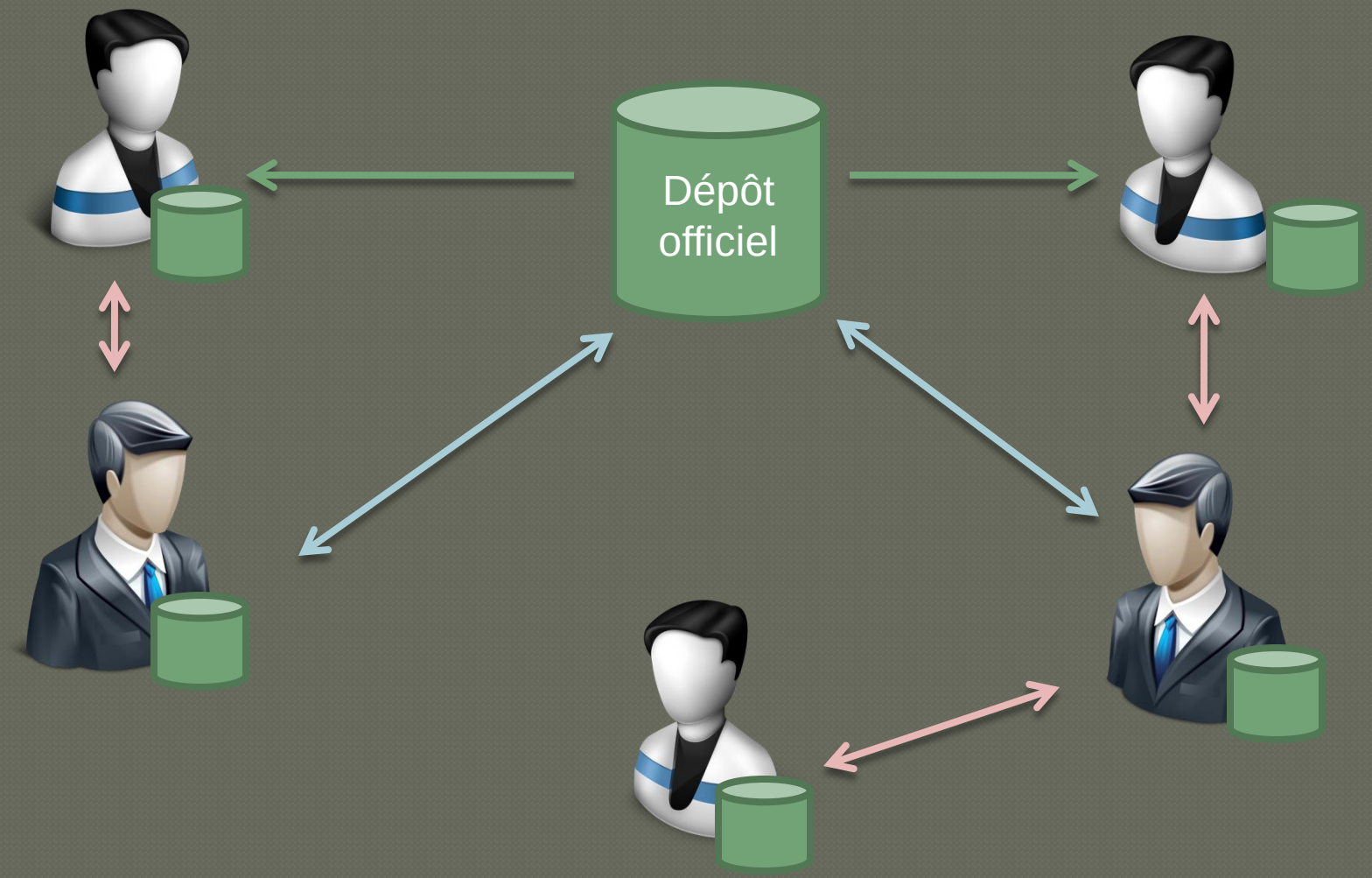


Échange avec le dépôt

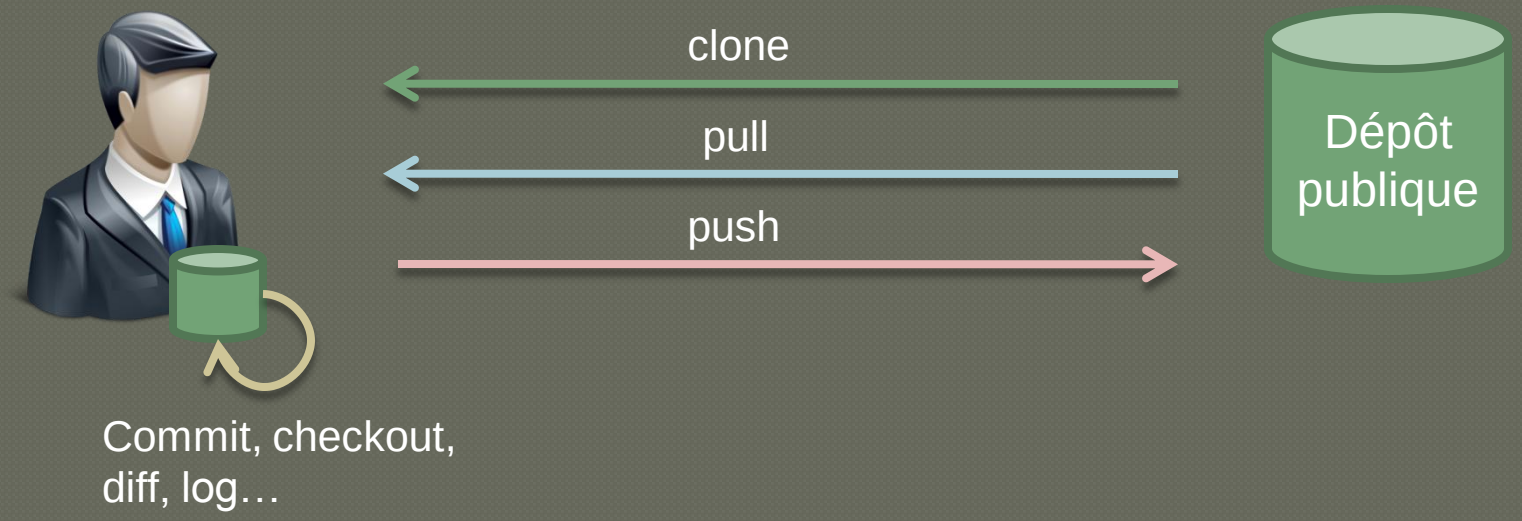


(Importé de [1])

Approche GIT : distribuée



Échange avec le dépôt



(Importé de [1])

Les bases

Manipulations simples avec SVN et GIT

Configuration du client

```
$ git config --global user.name "Paul Atréides"
$ git config --global user.email Paul.Atreides@dune.org

$ git config --global core.editor nano

$ git config --global color.branch auto
$ git config --global color.diff auto
$ git config --global color.interactive auto
$ git config --global color.status auto
```

Ces options sont écrites dans ~/.gitconfig au format INI.

Création du dépôt

SVN

```
$ svnadmin create myProject
$ su
# vi /etc/apache/httpd.conf
```

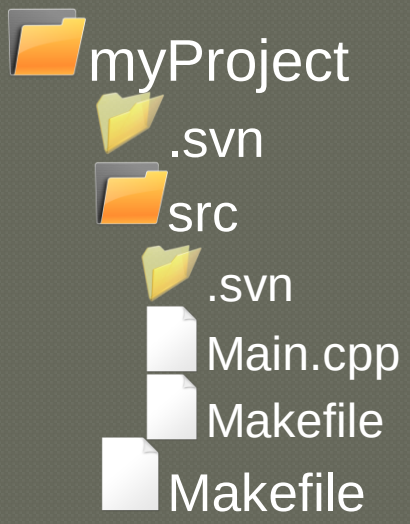
```
$ svn checkout http://....
```

GIT

```
$ mkdir myProject
$ cd myProject
$ git init
```

Structure de la copie de travail

SVN



GIT

Ajout de fichier au dépôt

SVN

```
$svn add main.cpp
```

GIT

```
$ git add main.cpp
```


- Sous git on ajoute des « modifications » pas des fichiers.
- Les modifications doivent être ajoutées à chaque commit.
- Git ne peux pas ajouter de dossier vide

diff, mv, rm …

SVN

```
$ svn diff main.cpp
$ svn diff
```

```
$ svn mv main.cpp src/main.cpp
$ svn rm old_config.cfg
```

GIT

```
$ git diff main.cpp
$ git diff
$ git diff -cached
```

```
$ git mv main.cpp src/main.cpp
$ git rm old_config.cfg
```

Commit

SVN

```
$ svn commit
 * introduction d'un bug !!!
$
```

GIT

```
$ git commit -a
 * introduction d'un bug !!!
$
```

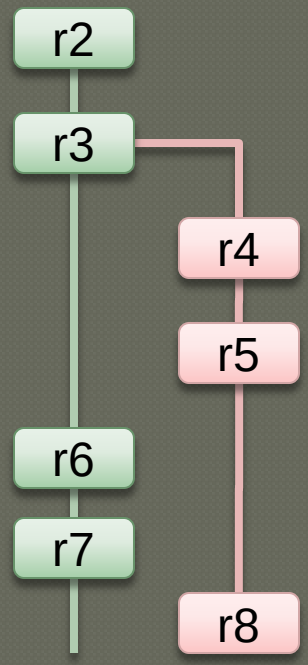
Le « *commit* » de git ne valide que les modifications qui ont été marquées avec « *add* ».

L'option –a effectue un « *add* »automatique des fichier versionnés

Identification des commits

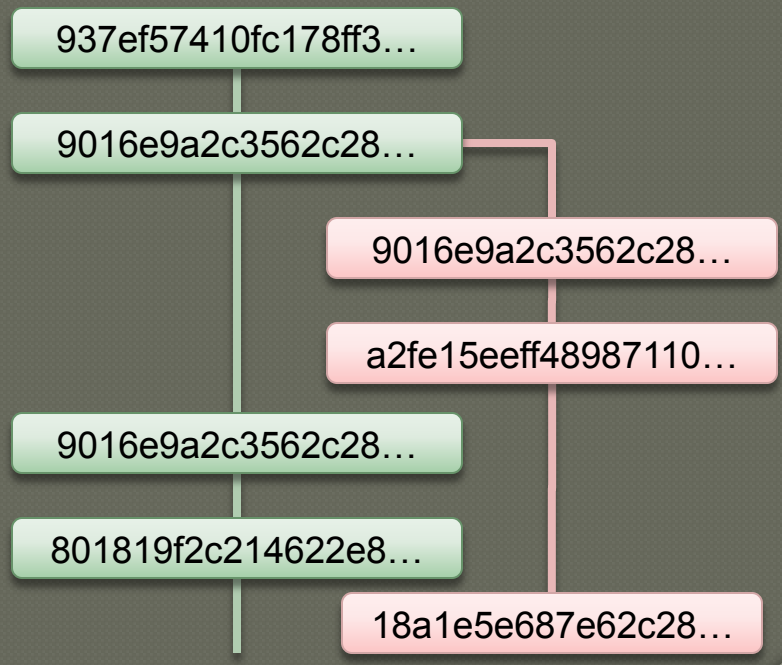
SVN

- Révision numérotés de manière croissante



GIT

- Le numéro de révision est un hash SHA1



Ooops commit ou --amend

SVN

```
$ svn commit
 * introduction d'un bug !!!
$ svn add Makefile
$ svn commit
 * ooops oubli du nouveau fichier
$
```

GIT

```
$ git commit -a
 * introduction d'un bug !!!
$ git add Makefile
$ git commit --amend
 * introduction d'un bug !!!
$
```

N'utilisez pas --amend après un push

Status

SVN

```
$> svn status
?       test_git
M       main.cpp
M       Makefile
$>
```

svn propedit svn:ignore test_git

GIT

```
$> git status
# On branch master
#
#       modified:   Makefile
#
# Changed but not updated:
#
#       modified:   main.cpp
#
# Untracked files:
#
#       main.o
#       test_git
```

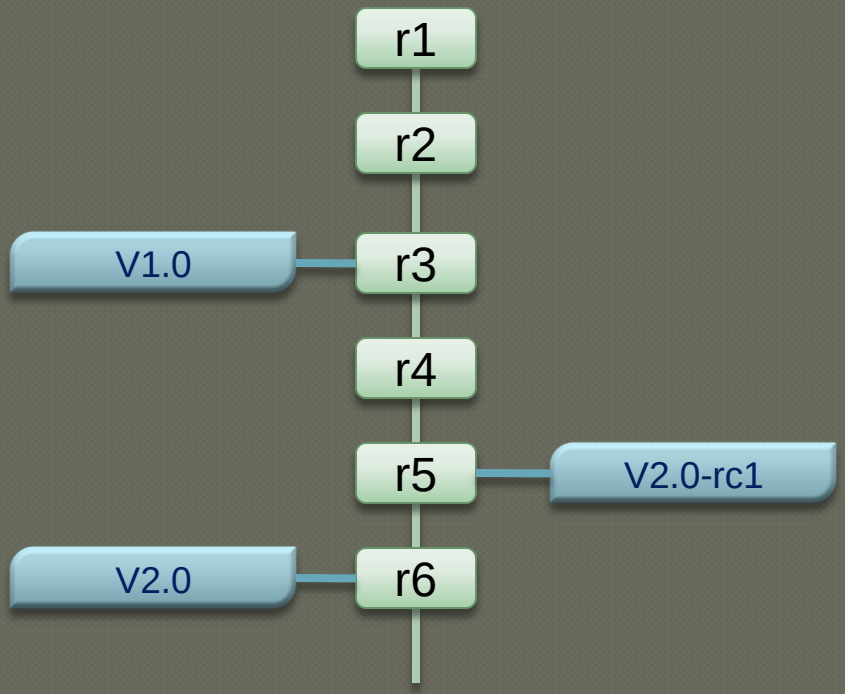
echo *.o >> .git/info/exclude

Les tags

Comment nommer ses révisions

Les tags

- Étiquette permettant de nommer une révision



Création de tags

SVN

- Tag⇔ dossier

```
$ svn cp tunk tags/v1.0
$ commit
$ svn cd tags/v1.0
```

- Voir les tags :

```
$ svn ls tags/
v1.0
```

GIT

- Instructions spécialisées
- Tag définit par défaut : HEAD

```
$ git tag v1.0 90933ef
$ git checkout v1.0
```

- Voir les tags:

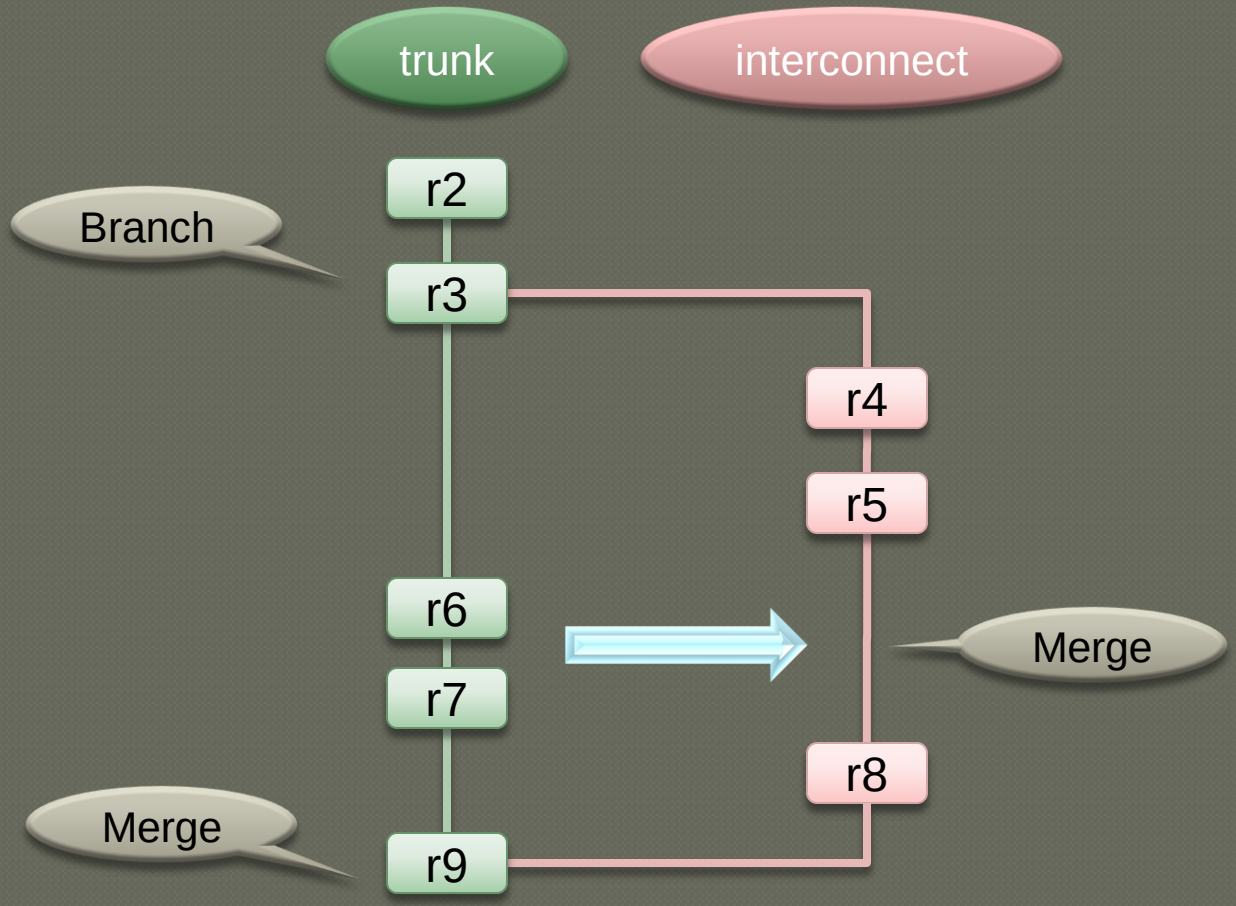
```
$ git tag
v1.0
```

Les banches

Comment gérer ses versions et les travaux expérimentaux

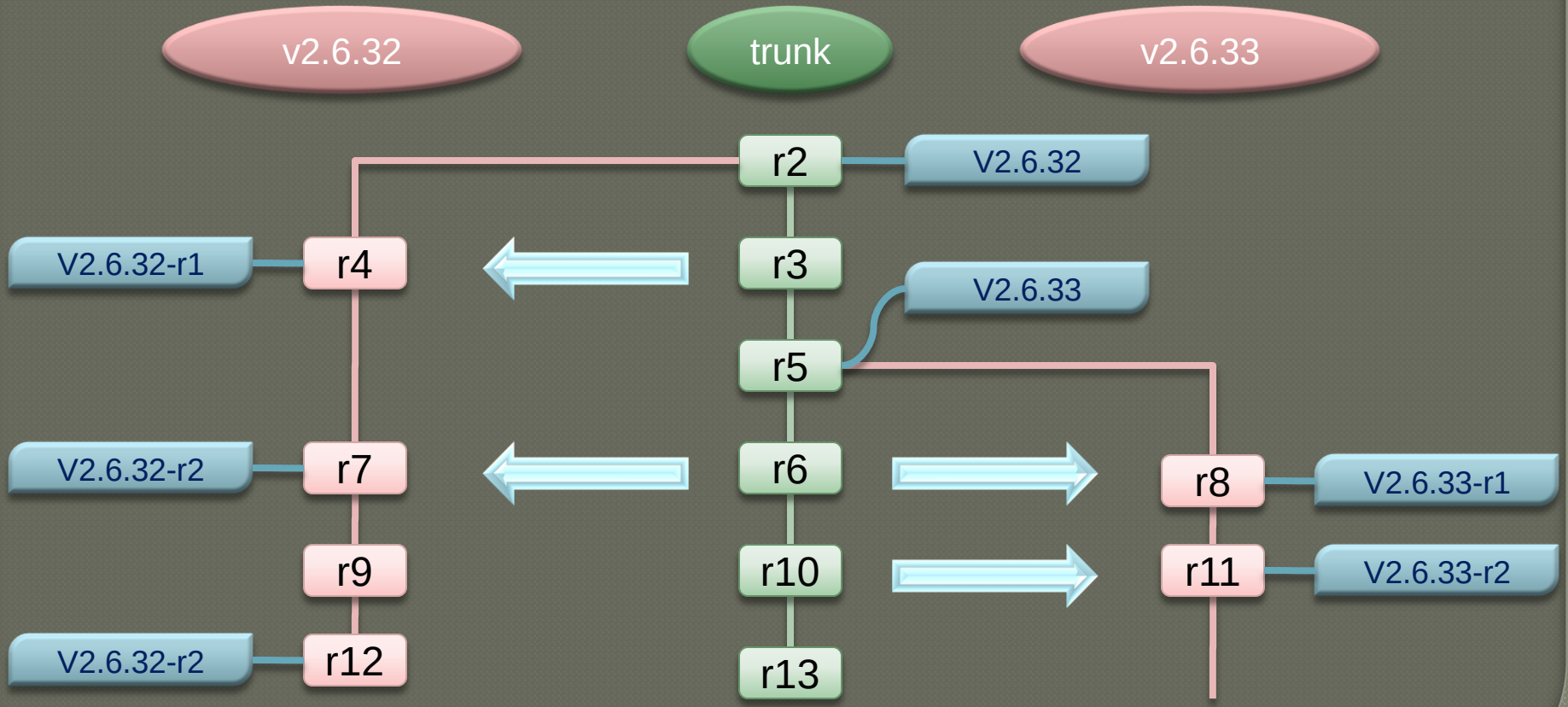
Les branches

- Isoler une portion de travail expérimental jusqu'à son terme



Les branches

- Gérer la maintenance de plusieurs versions



Création de branches

SVN

- Branche ⇔ dossier
- Branche par défaut : *trunk*

```
$ svn cp tunk branch/test
$ commit
$ svn cd branch/test
```

- Voir les branches :

```
$ svn ls branch/
test
```

GIT

- Instructions spécialisées
- Branche par défaut : *master*

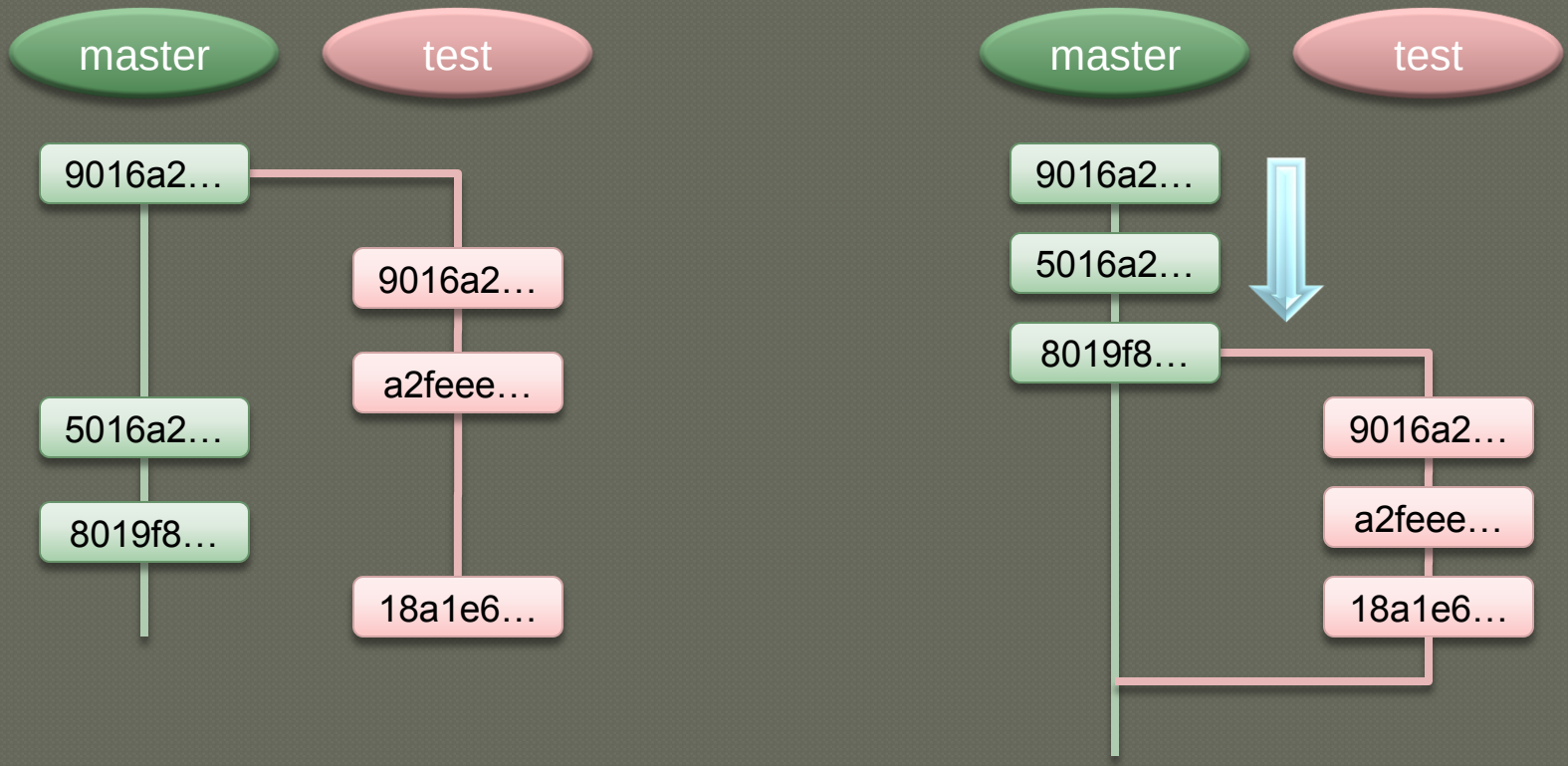
```
$ git branch test
$ git checkout test
```

- Voir les branches :

```
$ git branch
 * test
   master
```

Rebase

```
$ git checkout test
$ git rebase master
```



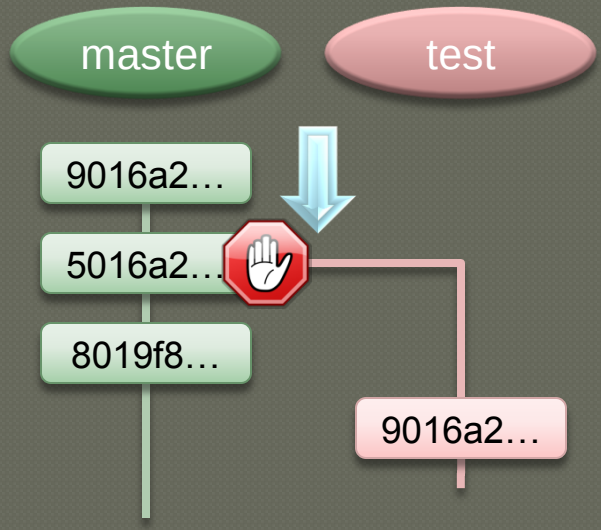
Rebase & conflits

```
int main(void)
{
-       printf(«Hello World !!»);
        reutrn 0;
}
```

master

```
int main(void)
{
-       printf(«Hello World !!»);
+       printf(«Cordialement»);
        reutrn 0;
}
```

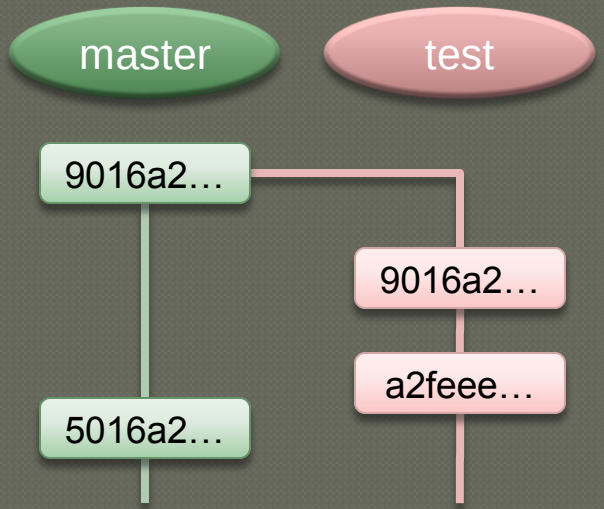
test

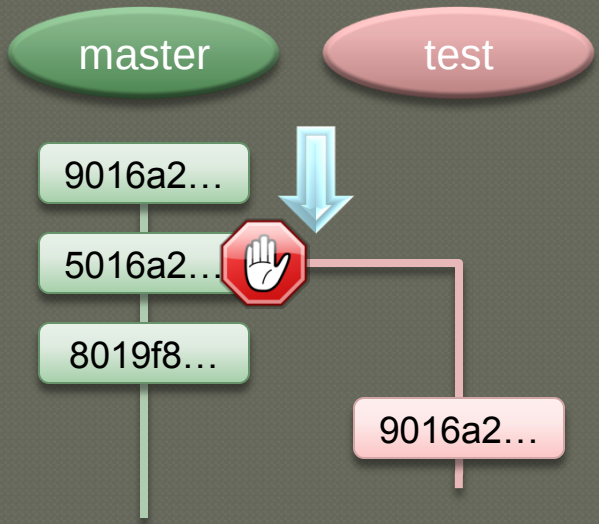


Rebase & conflits

```
$> git checkout test
$> git rebase master

$> git rebase --abord
```

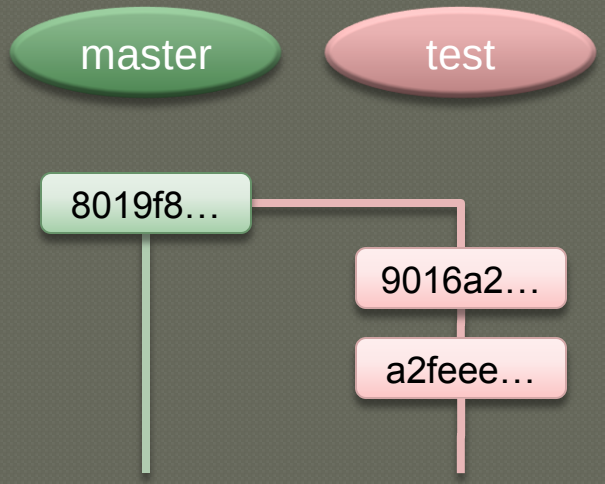
```
$> git checkout test
$> git rebase master

$> git mergetool -t kdiff3

$> git add main.cpp
$> git rebase --continue
```

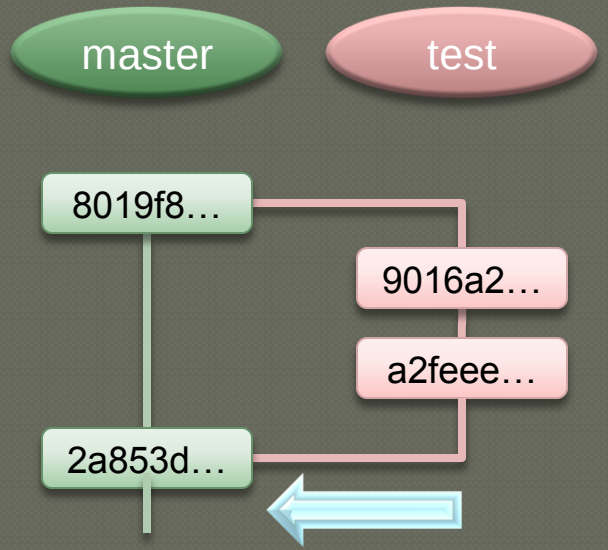




Merge

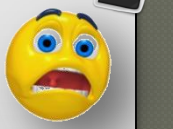
```
$> git checkout master
$> git merge test
$> git -d test
```



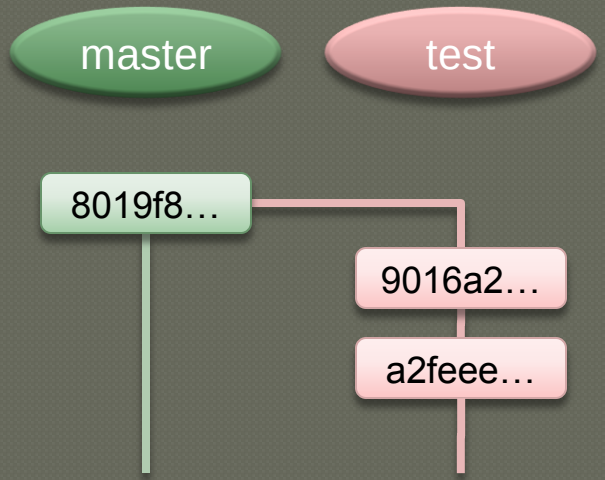


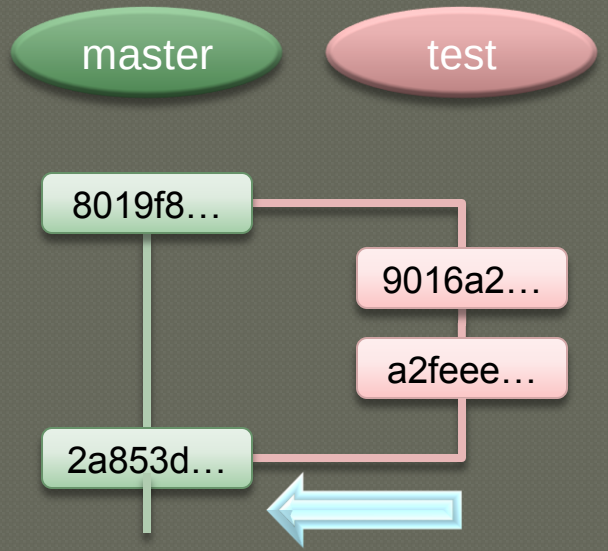
Merge & conflits

```
$> git checkout master
$> git merge test

$> git reset --hard HEAD
```



```
$> git checkout matser
$> git merge test

$> git mergetool -t kdiff3

$> git add main.cpp
$> git commit
```



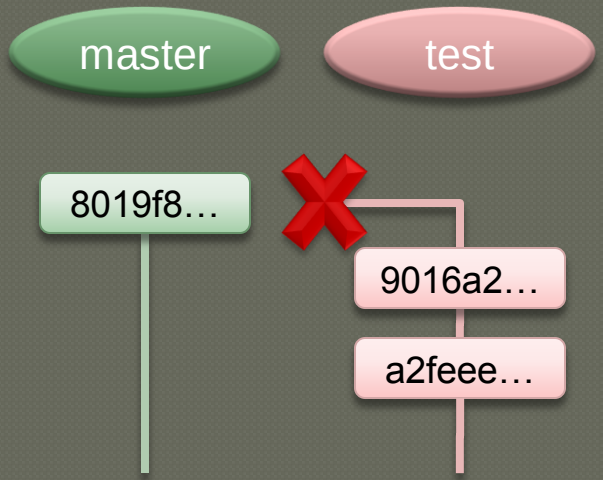


Effacer nos traces

Personne n’est parfait…

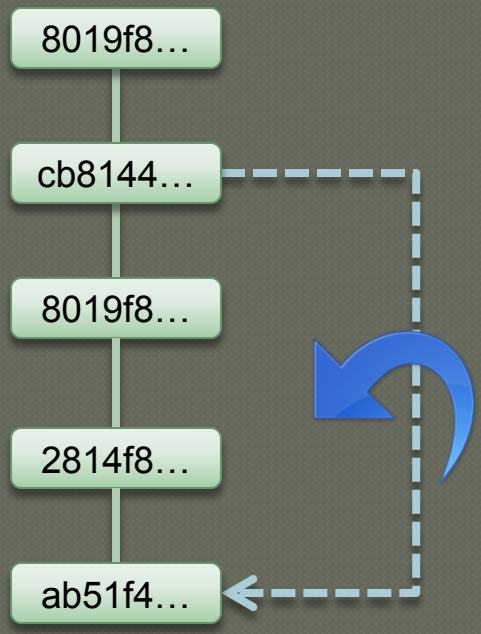
Supprimer une branche

```
$> git -D test
```



Revert

```
$ git revert cb8144f5c6198149c4cf97918438191a80401c03
```



Reset

```
$> git reset --hard 8019f8
```

N’utilisez pas reset après un push

Annuler les changements locaux

- Remettre le fichier comme à l'origine

```
$> git checkout main.cpp
```

- Annuler tous les changements (revenir à HEAD)

```
$> git reset --hard
$>
```

```
$> git checkout -f HEAD
$>
```

Tester un ancienne révision

- Si on ne souhaite pas faire de modifications :

```
$> git checkout v0.2.1
```

- Pour faire des modifications que l'on devra *merger* on doit créer une branche :

```
$> git checkout v0.2.1 -b bugfix
```

Collaboration avec GIT

Et si on travaillait ensemble ?

Première publication

- On clone le dépôt sans copie de travail :
-

```
$> cd ..
$> git clone --bare myProject
$> ls
myProject
myProject.git
```

- On envoie le dossier sur un serveur publique pour le rendre accesible par http, ssh, rsync ou git-deamon :

```
$> scp -r myProject.git paul@dune:~/public_html/repos/
```

Récupérer la version publique

- On clone le dépôt publique sur notre machine :

```
$ git clone http://server/~user/repos/myProject.git
$ cd myProject
```

- Par défaut git clone uniquement la branche *master*

```
$ git clone http://server/~user/repos/myProject.git experimental
$ cd myProject
```

Mettre à jour sa version : pull

- Si on synchronise avec la source :

```
$ git pull
```

- Si on synchronise avec une autre version :

```
$ git pull http://server/~bob/myProject.git
```

- Eviter de réécrire l'adresse :

$> git remote add bob http://server/~bob/myProject.git
$> git pull bob

Du point de vue de git

- Un dépôt distant est une branche

```
$ git branch -r
  origin/HEAD
  origin/master
  bob/HEAD
  bob/master
  bob/dev
  bob/test
```

- Lors d'un *pull* git effectue un *fetch* + *merge*

```
$ git fetch bob/test
$ git checkout FETCH_HEAD
$ ……
$ git checkout master
$ git merge FETCH_HEAD
```

- Vous devez régler d'éventuels conflits

Publier ses modifications :push

- Si on synchronise avec la source :

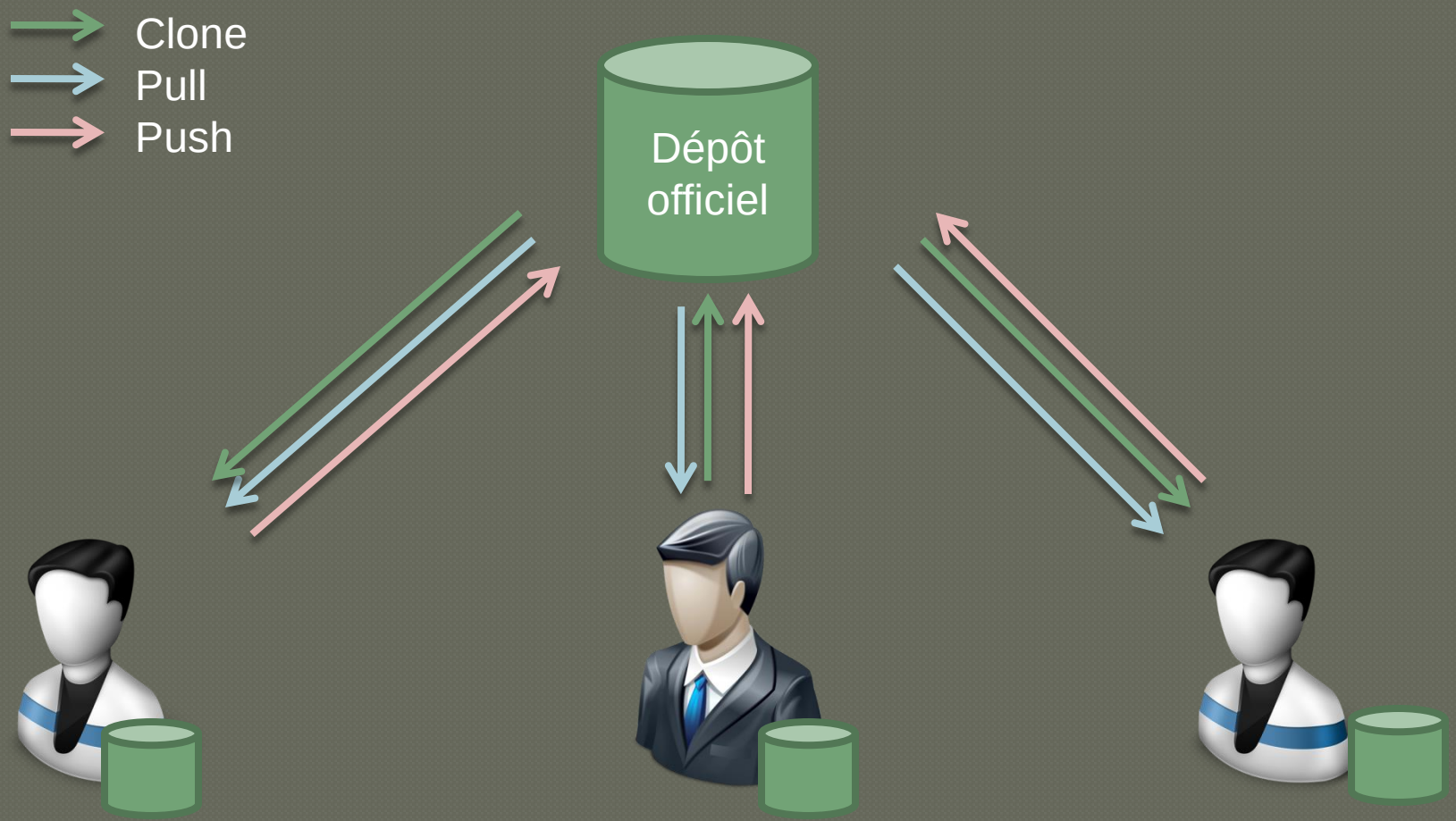
```
$ git push
```

- Avec un autre dépôt :

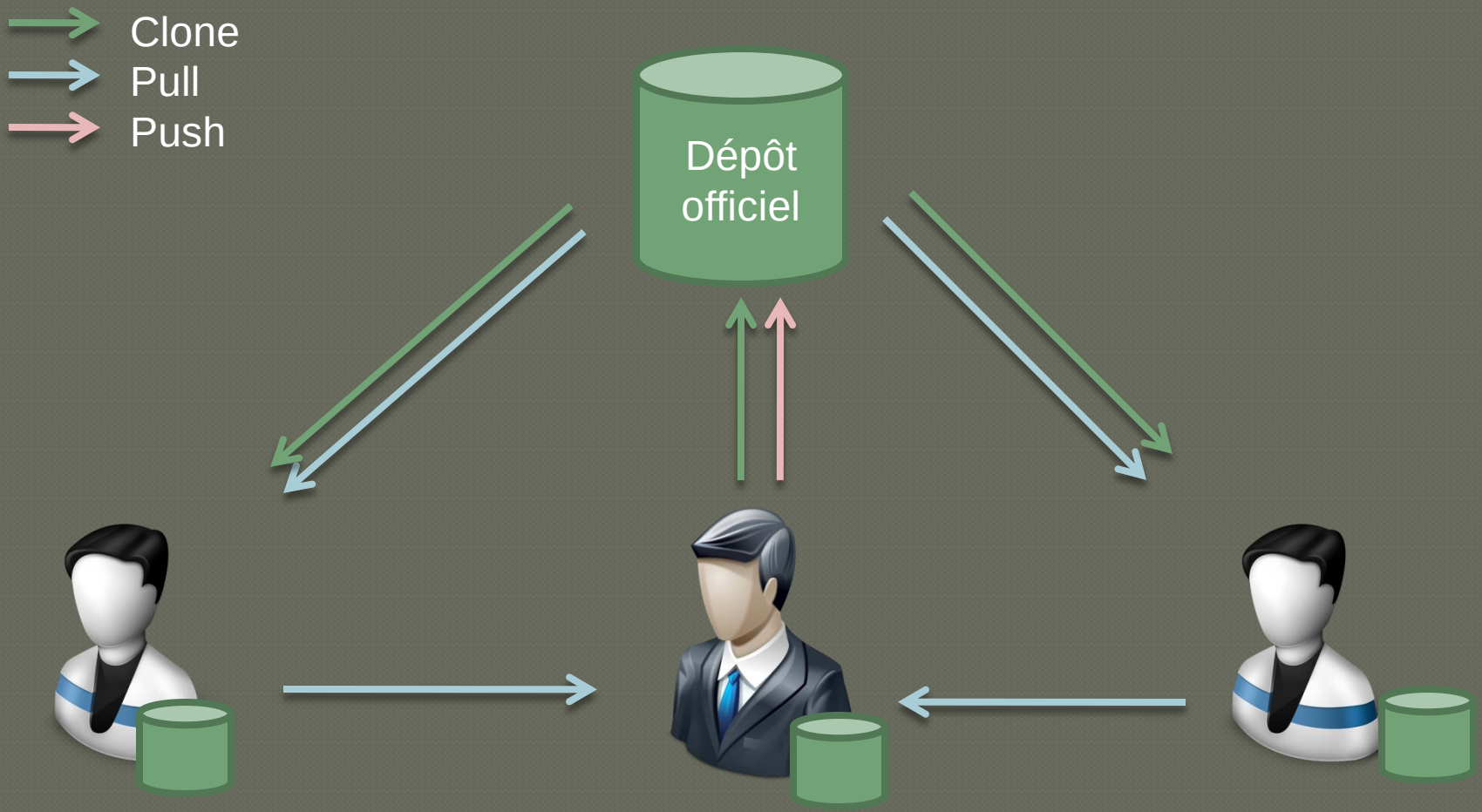
```
$ git push ssh://bob@server/~public_html/repos/myProject.git
```

- Contrairement à SVN on utilise plutôt SSH pour le push
- Il est possible d'utiliser HTTP via DAV pour l'écriture

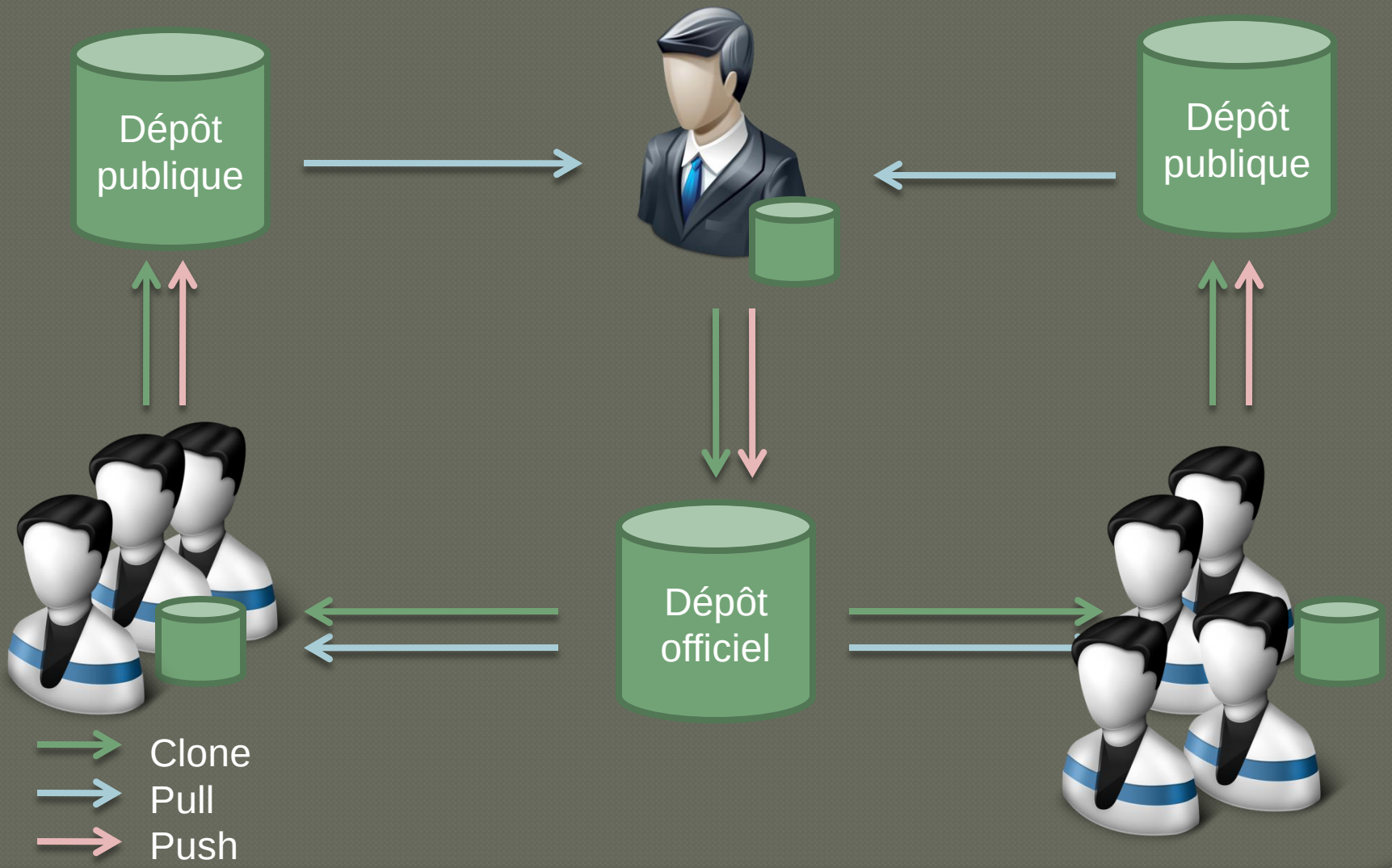
Exemple de travail commun 1



Exemple de travail commun 2



Exemple de travail commun 3



Archivage

SVN

```
$ svnadmin dump myProject > out.svn
$
```

GIT

```
$ tar -cvf out.tar myProject.git
$
```

```
$ cd ..
$ tar -cvf out.tar myProject
```

```
$ cd ..
$ git clone --bare myProject
$ tar -cvf out.tar myProject.git
```

Git-svn

- Il est possible d'utiliser GIT comme client SVN
- Checkout :

```
$ git svn clone http://server/depot_svn/
```

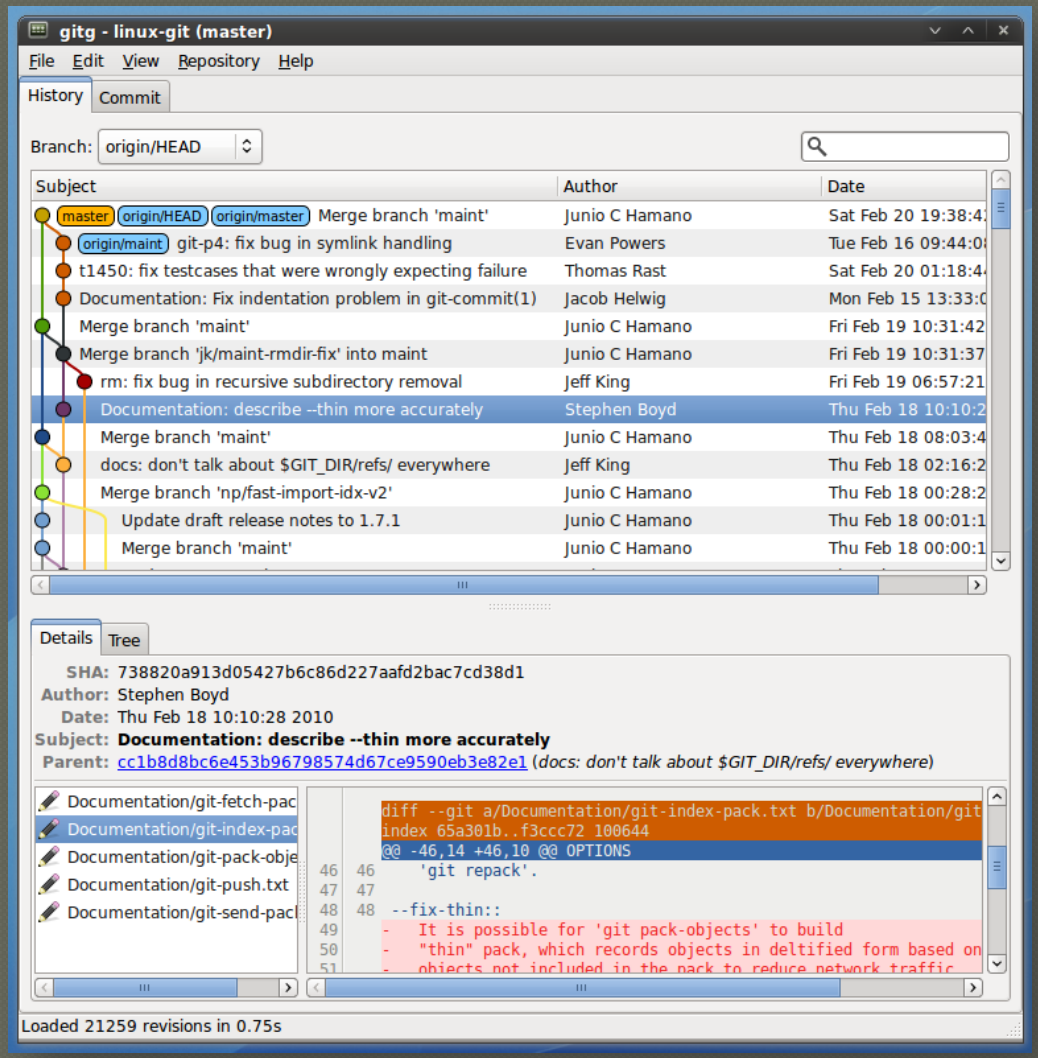
- Update :

```
$ git svn rebase http://server/depot_svn/
```

- Commit :

```
$ git svn dcommit
```

GUI : gitg


gitg - linux-git (master)
File Edit View Repository Help
History Commit
Branch: origin/HEAD
Subject Author Date
master origin/HEAD origin/master Merge branch 'maint' Junio C Hamano Sat Feb 20 19:38:4
origin/maint git-p4: fix bug in symlink handling Evan Powers Tue Feb 16 09:44:0
t1450: fix testcases that were wrongly expecting failure Thomas Rast Sat Feb 20 01:18:4
Documentation: Fix indentation problem in git-commit(1) Jacob Helwig Mon Feb 15 13:33:0
Merge branch 'maint' Junio C Hamano Fri Feb 19 10:31:42
Merge branch 'jk/maint-rmdir-fix' into maint Junio C Hamano Fri Feb 19 10:31:37
rm: fix bug in recursive subdirectory removal Jeff King Fri Feb 19 06:57:21
Documentation: describe --thin more accurately Stephen Boyd Thu Feb 18 10:10:2
Merge branch 'maint' Junio C Hamano Thu Feb 18 08:03:4
docs: don't talk about $GIT_DIR/refs/ everywhere Jeff King Thu Feb 18 02:16:2
Merge branch 'np/fast-import-idx-v2' Junio C Hamano Thu Feb 18 00:28:2
Update draft release notes to 1.7.1 Junio C Hamano Thu Feb 18 00:01:1
Merge branch 'maint' Junio C Hamano Thu Feb 18 00:00:1
Details Tree
SHA: 738820a913d05427b6c86d227aafd2bac7cd38d1
Author: Stephen Boyd
Date: Thu Feb 18 10:10:28 2010
Subject: Documentation: describe --thin more accurately
Parent: cc1b8d8bc6e453b96798574d67ce9590eb3e82e1 (docs: don't talk about $GIT_DIR/refs/ everywhere)
Documentation/git-fetch-pac
Documentation/git-index-pac
Documentation/git-pack-obje
Documentation/git-push.txt
Documentation/git-send-pacl
diff --git a/Documentation/git-index-pack.txt b/Documentation/git
index 65a301b..f3ccc72 100644
@@ -46,14 +46,10 @@ OPTIONS
'git repack'.
--fix-thin::
- It is possible for 'git pack-objects' to build
- "thin" pack, which records objects in deltified form based on
- objects not included in the pack to reduce network traffic
Loaded 21259 revisions in 0.75s

En résumé

- Utile pour travailler en mode déconnecté
- Facilite de travail intra-groupe
- Simple à mettre en place
- Stockage fiable
- Nombreuses possibilités (137 commandes)
- Quelques manipulations dangereuses

Quelques références

- Tutoriels officiel sur le site de git : http://git-scm.com/
- Linux-Mag 119 (Septembre 2009) http://www.unixgarden.com/index.php/administration-systeme/git-it
- http://svnbook.red-bean.com/
- [1] : http://excess.org/article/2008/07/ogre-git-tutorial/
- [2] : http://www.youtube.com/watch?v=4XpnKHJAok8

Pratique : démystifier malloc

Total : ~150 lignes + 9 fonctions