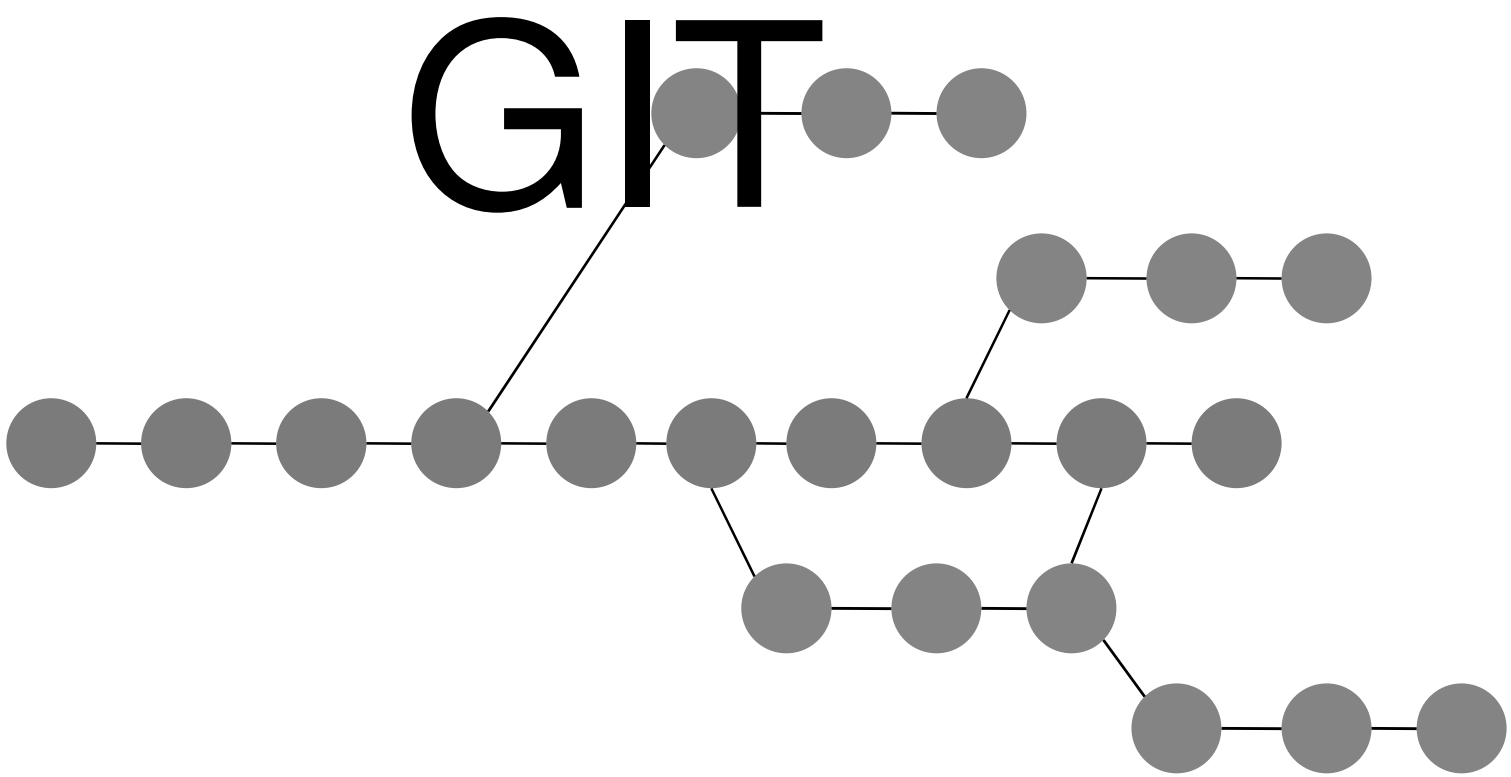
COMPIL - Versionning



COMPIL – Versionning GIT

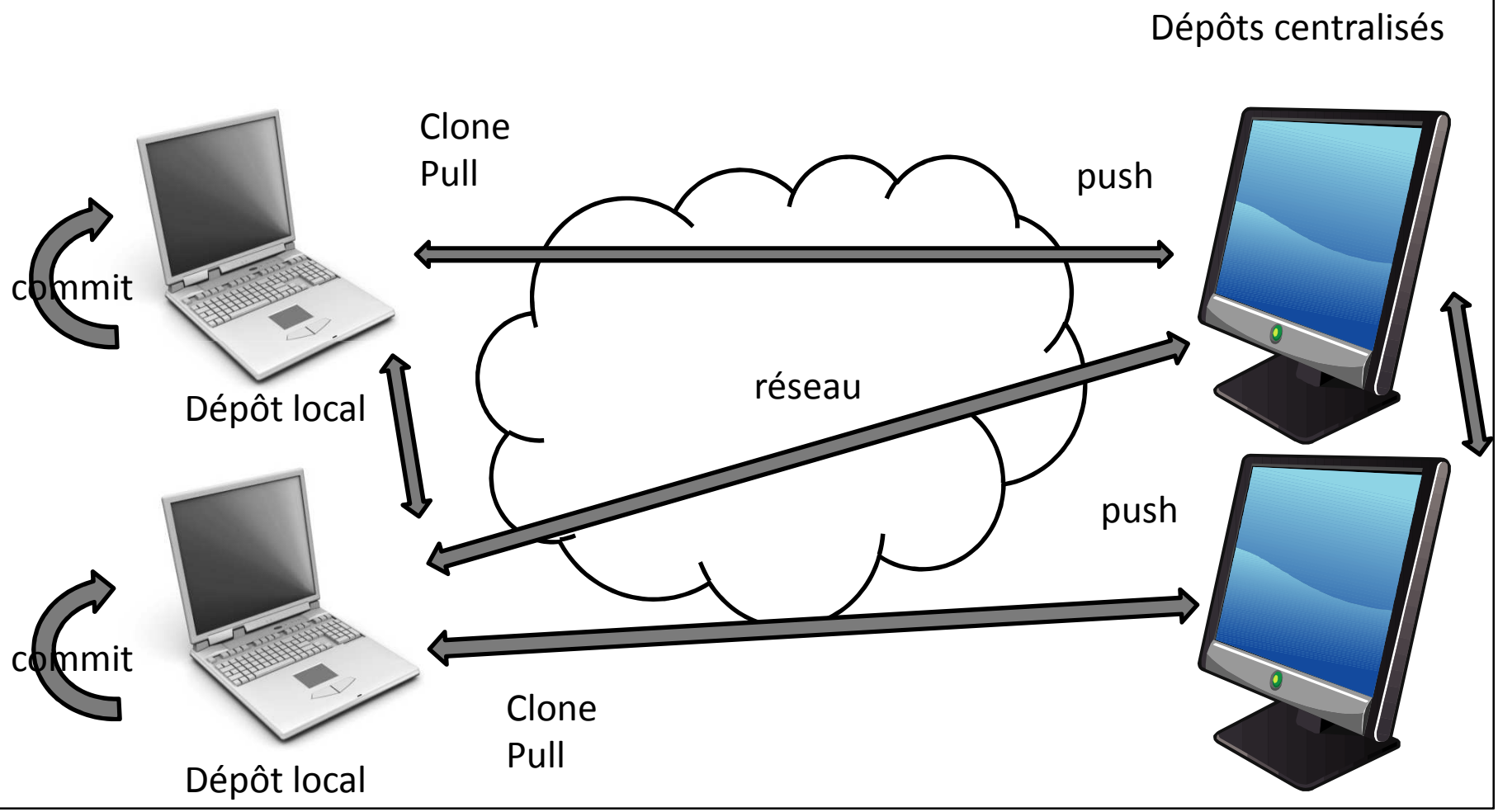
- GIT outil de gestion de version
  - Historique
    - 2001 – Linux est développé avec CVS
    - 2002 à 2005 – Linux est développé avec Bitkeeper, non libre
    - 2005 – Développement de GIT par Linus Thorvald
    - Git est inspiré de BitKeeper et Monotone
    - Git a été initialement conçu comme outil de bas niveau pour d'autres outils de plus haut niveau comme Cogito et StGIT
  - Quelques précisions
    - GIT est distribué sous licence GNU GPL2 (donc libre)
    - Portable sur la plupart des Unix et Linux, Windows (cygwin)

COMPIL – Versionning GIT

- Principe de base
  - Gestion de code source
  - Conserver toutes les versions de tous les fichiers
  - Récupérer toutes les versions de tous les fichiers
  - Comparer les versions des fichiers
  - Gérer une arborescence
  - Travail coopératif décentralisé (échange et fusion de données)

COMPIL – Versionning GIT

- Gestion de version décentralisée

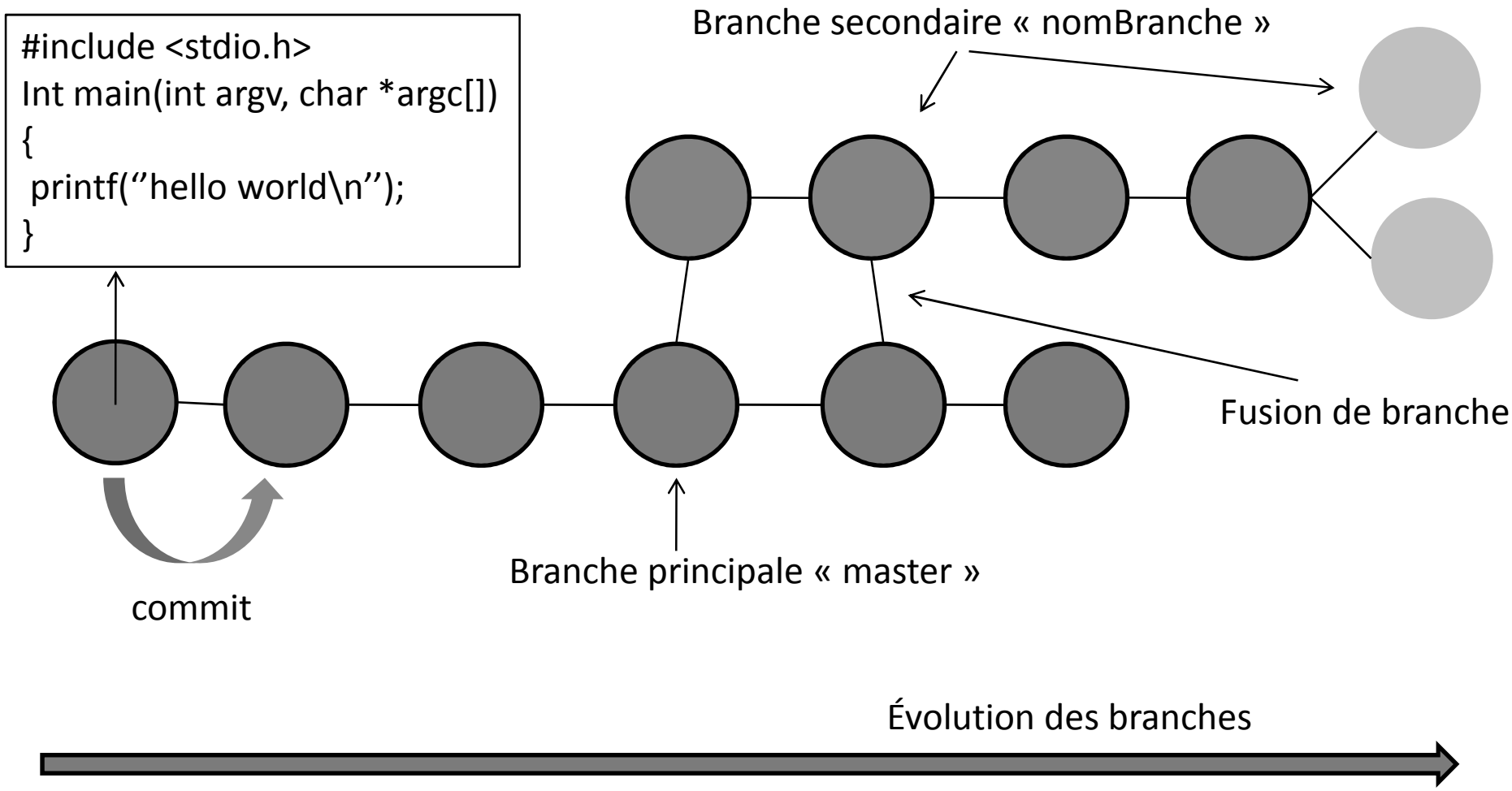


COMPIL – Versionning GIT

- Outils de gestion décentralisé
  - « Avec l'arrivée des logiciels libres et leur développement communautaire, une autre façon de voir la gestion de versions est apparue. Cette autre vision consiste à voir l'outil de gestion de versions comme un outil permettant à chacun de travailler à son rythme, de façon désynchronisée des autres, puis d'offrir un moyen à ces développeurs de s'échanger leur travaux respectifs. C'est ce que l'on nomme la gestion de versions décentralisée ».
  - Git
  - Bazaar
  - Darcs
  - Mercurial
  - Monotone

COMPIL – Versionning GIT

- Définition d'une branche

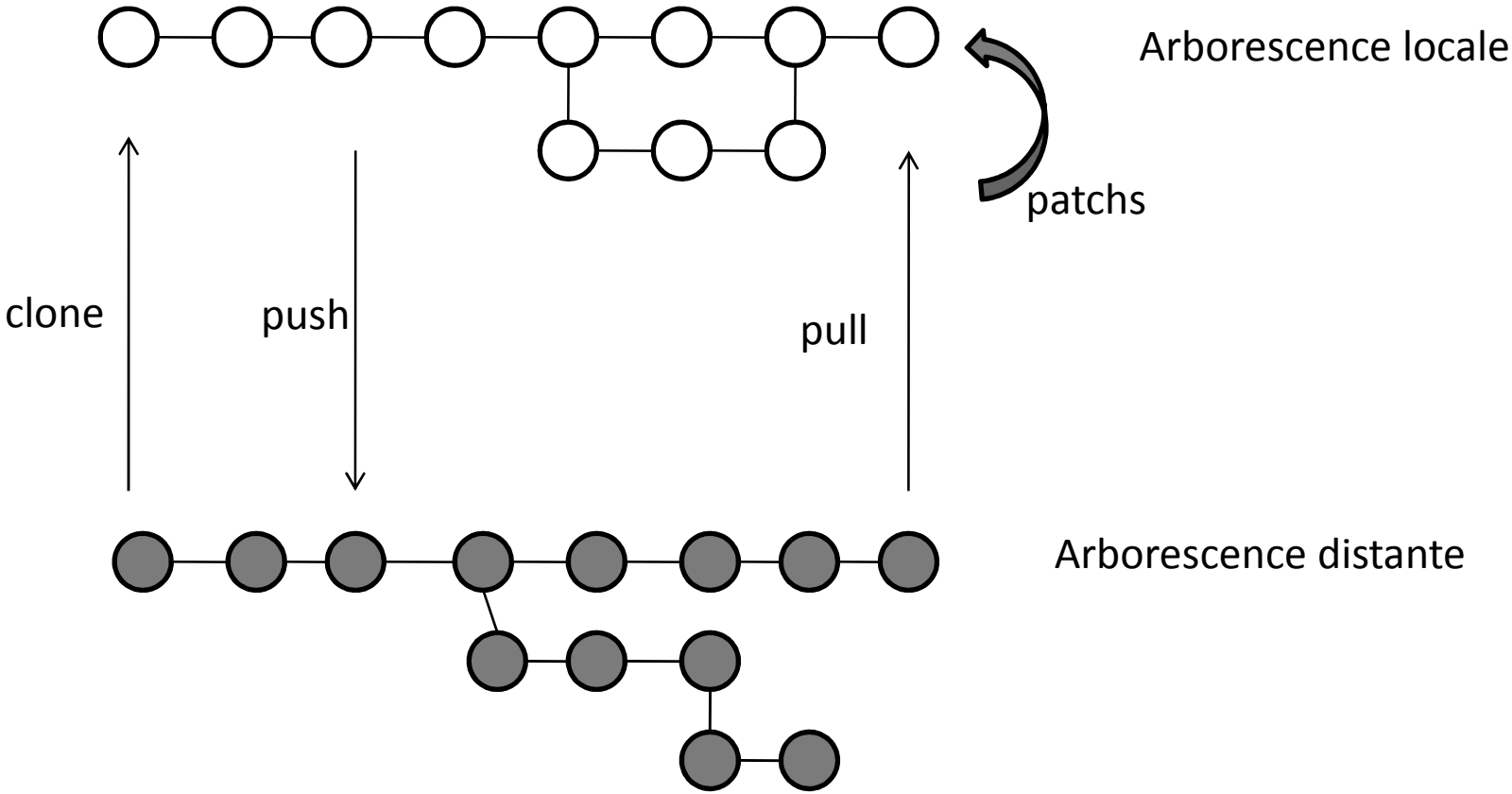


COMPIL – Versionning GIT

- Principe de base de GIT
  - Gestion d'un arborescence locale
    - Chaque client possède son dépôt local
    - Changement de branche et fusion de branche en local
    - Travail hors ligne sans connexion réseau
    - Accès au dépôt distant que pour la mise à jour du dépôt local et l'envoi d'information
  - Synchronisation avec une arborescence distribuée
    - Serveur http et deamon GIT
    - SSH, FTP

COMPIL – Versionning GIT

- Création et mise à jour d'une branche



COMPIL – Versionning GIT

- Fonctionnement interne de GIT
  - Utilisation de 4 types d'objet :
    - Blob : le contenu d'un fichier.
    - Tree : liste d'objets de type blobs et des informations associées : nom du fichier et les permissions.
    - Commit : historique d'une arborescence de source.
    - Tag : contient des méta-informations associées à un autre objet.
  - Git indexe les fichiers d'après leur somme de contrôle calculée avec la fonction SHA-1

COMPIL – Versionning GIT

- Les profils de GIT ?
  - **Développeur Individuel (Standalone)**
  - **Développeur Individuel (Participant)**
  - **Intégrateur**
  - **Administrateur du dépôt**

COMPIL – Versionning GIT

- Classification des commandes
  - Intégrateur
  - Développeur
  - Commande pour le dépôt
  - HIGH LEVEL commands (porcelain)
  - LOW-LEVEL commands (plumbing)

- Interopérabilité avec les autres gestionnaires
  - CVS, SVN , Arch

- Serveur dédié : git-deamon

COMPIL – Versionning GIT

- Mode de développement
  - Dépôt centralisé comme CVS, SVN avec un dépôt local
  - Dépôt local pour chaque développeur, synchronisation chez les autres
- Gestion des conflits
  - Des conflits peuvent apparaître lors de la fusion de branches locale/locale, locale/distante, distante/locale
  - Des conflits peuvent aussi apparaître lors d'un « revert »

COMPIL – Versionning GIT

- Web interfaces
  - gitweb –Perl
  - wit – Python
  - gitarella –Ruby
  - git-php –PHP
  - cgit-C

- Visualisation historique
  - Gitk Tcl/Tk GUI distribué avec Git.
  - QGit - Qt GUI.
  - Giggle - Gtk+ GUI .
  - gitview - Python && Gtk+ GUI, distribué avec Git.
  - tig - ncurses-based .
  - git-browser.

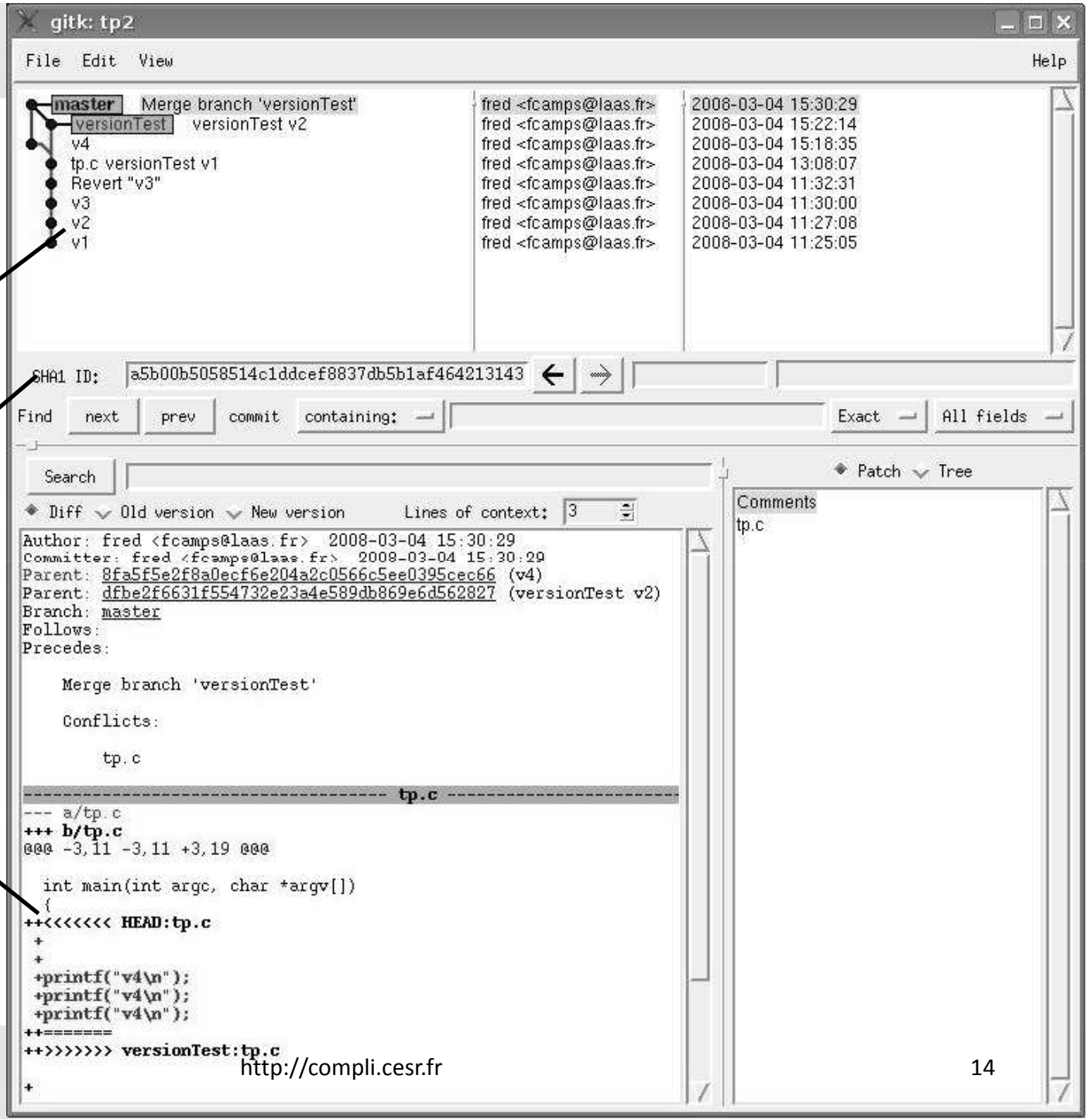
- Outil externe

  GITK

branche

Signature SHA1

État des modif



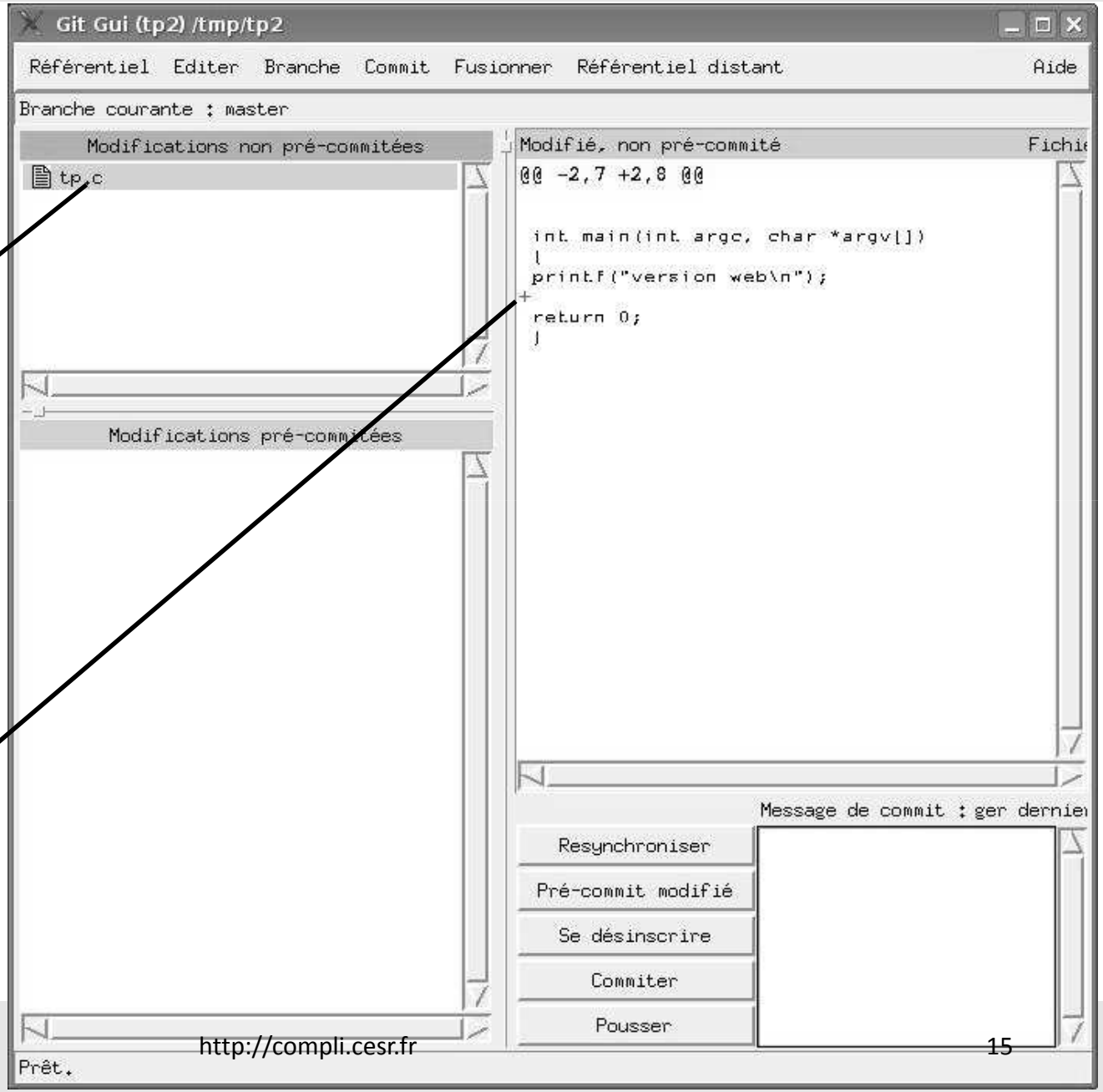
COMPIL – Versionning GIT

- Outil externe

  GIT-gui

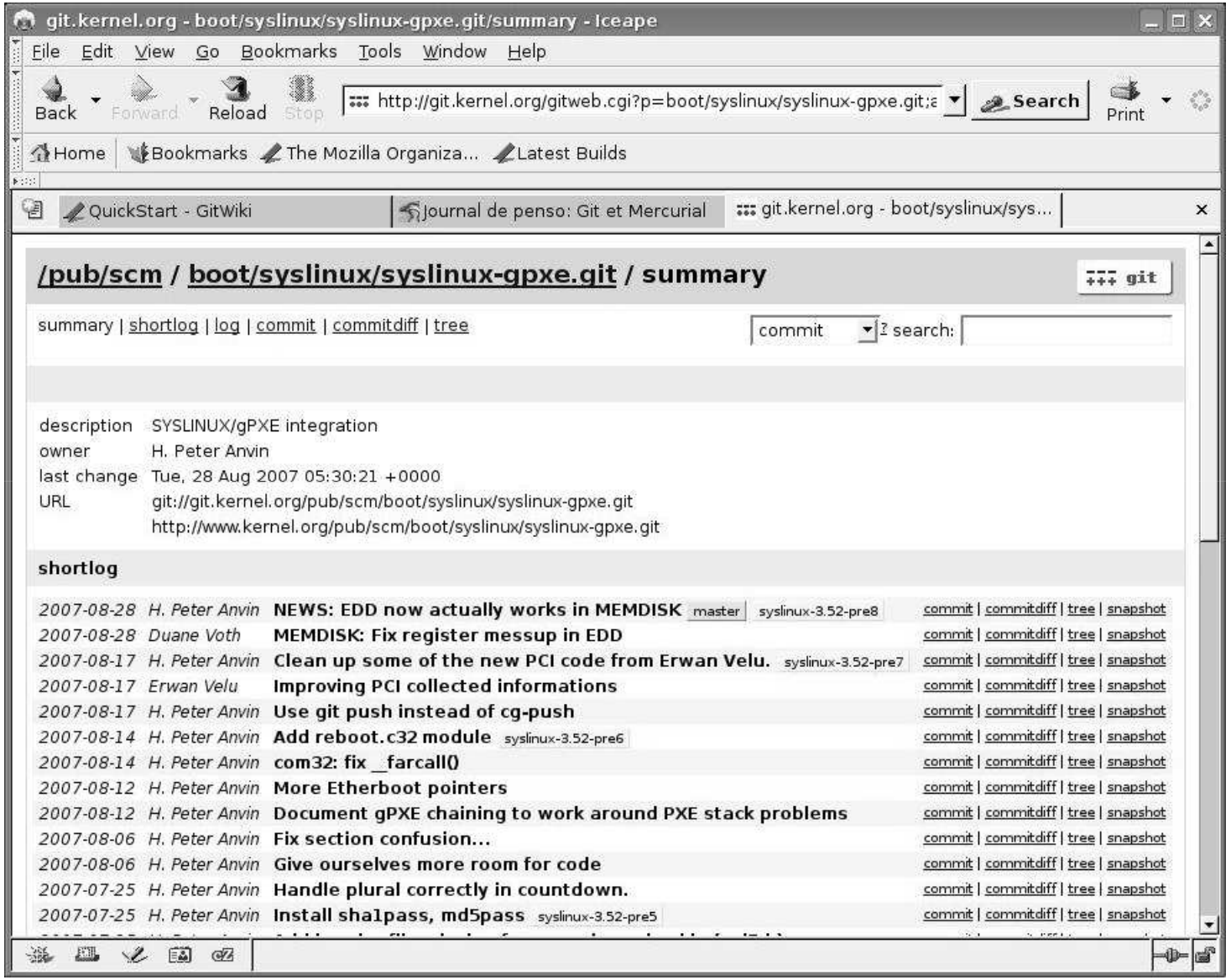
Fichier à commiter

État des modifs



COMPIL – Versionning GIT

- Interface gitweb



COMPIL – Versionning GIT

- Références officielles :
  - http://git.or.cz/
- Pour commencer :
  - http://www.kernel.org/pub/software/scm/git/docs/tutorial.html
  - http://www.kernel.org/pub/software/scm/git/docs/user-manual.html
  - http://www.kernel.org/pub/software/scm/git/docs/everyday.html
  - http://git.or.cz/gitwiki/GitBenchmarks *(comparatif)*
  - http://fr.wikipedia.org/wiki/Git
  - http://en.wikipedia.org/wiki/Git_(software)
  - http://git.or.cz/gitwiki/GitDocumentation
  - http://git.or.cz/gitwiki/QuickStart

COMPIL – Versionning GIT

- Initialisation – compte user

jiminy[tp2] **git --version**
git version 1.5.4.1

jiminy[tp2] **git config --global user.name "fred "**
jiminy[tp2] **git config --global user.email fcamps@laas.fr**

jiminy[tp2] **cat ~/.gitconfig**

```
[user]
        name = fred
        email = fcamps@laas.fr
```

COMPIL – Versionning GIT

- Création d'un dépôt :

// Création d'un dépôt vierge :

jiminy[tp2] **git-init-db** // --shared pour dépôt centralisé mode CVS
Initialized empty Git repository in .git/

// le dépôt contient des informations git

jiminy[tp2] **ls -al**
total 16
drwxrwxr-x 3 fcamps ii 4096 mar 4 11:20 .
drwxrwxrwt 17 root root 4096 mar 4 11:19 ..
drwxrwxr-x 7 fcamps ii 4096 mar 4 11:20 .git

COMPIL – Versionning GIT

- Contenu du dépôt

```
jiminy[tp2] cd .git
jiminy[.git] ls -al
total 40
drwxrwxr-x 7 fcamps ii 4096 mar  4 11:20 .
drwxrwxr-x 3 fcamps ii 4096 mar  4 11:20 ..
drwxrwxr-x 2 fcamps ii 4096 mar  4 11:20 branches
-rw-rw-r-- 1 fcamps ii   92 mar  4 11:20 config
-rw-rw-r-- 1 fcamps ii   58 mar  4 11:20 description
-rw-rw-r-- 1 fcamps ii   23 mar  4 11:20 HEAD
drwxrwxr-x 2 fcamps ii 4096 mar  4 11:20 hooks
drwxrwxr-x 2 fcamps ii 4096 mar  4 11:20 info
drwxrwxr-x 4 fcamps ii 4096 mar  4 11:20 objects
drwxrwxr-x 4 fcamps ii 4096 mar  4 11:20 refs
```

COMPIL – Versionning GIT

- Ajout d'un fichier au dépôt

```
jiminy[tp2] cat tp.c
#include <stdio.h>
int main(int argc, char *argv[]) {
printf("hello world \n");
return 0; }

jiminy[tp2] git add tp.c //ajout du fichier au dépôt
jiminy[tp2] git status //affiche l'état du du dépôt
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#       new file: tp.c
```

COMPIL – Versionning GIT

- Commit du fichier en v1

// commit du fichier avec un message qui indique v1

jiminy[tp2] **git commit -m "v1 " tp.c** *// commit -a ou encore commit .*
Created initial commit bb17470: v1
1 files changed, 9 insertions(+), 0 deletions(-)
create mode 100644 tp.c

jiminy[tp2] **cat tp.c**

```
#include <stdio.h>
int main(int argc, char *argv[]){
printf("hello world \n");
printf("version 2\n");
return 0;}
```

COMPIL – Versionning GIT

- Modification du fichier en v2

jiminy[tp2] **git commit –a** *//ouverture de ecmacs pour spécifier v2*
Created commit 3be2008: v2
1 files changed, 2 insertions(+), 0 deletions(-)

jiminy[tp2] **git status**
On branch master
nothing to commit (working directory clean)

jiminy[tp2] **cat tp.c**
```
#include <stdio.h>
int main(int argc, char *argv[]) {
printf("hello world \n");
printf("version 3\n");
return 0;}
```

COMPIL – Versionning GIT

- Commit V3 et revert de la version V3 à V2

jiminy[tp2] **git commit –a** *//ouverture de ecmacs pour spécifier v3*
Created commit 463baa0: v3
1 files changed, 1 insertions(+), 1 deletions(-)

// pour le revert on peut utiliser gitk pour récuépérer l'ID du commit
jiminy[tp2] **git-revert 463baa023c55c** *// signature SHA1 possibilité de raccourcir eg :* 463baa
Finished one revert.
Created commit 92b7e78: Revert "v3"
1 files changed, 1 insertions(+), 1 deletions(-)

jiminy[tp2] **cat tp.c**
```
#include <stdio.h>
int main(int argc, char *argv[]){
printf("hello world \n");
printf("version 2\n");
return 0;}
```

COMPIL – Versionning GIT

- Log du dépôt

```
jiminy[tp2] git log
commit 92b7e789777985703f520b9b3af737b606a5ef61
Author: fred <fcamps@laas.fr>
Date:   Tue Mar 4 11:32:31 2008 +0100
    Revert "v3«
…
    v2
commit bb17470696d13dc80079c17f6a323a8c4af2aeb3
Author: fred <fcamps@laas.fr>
Date:   Tue Mar 4 11:25:05 2008 +0100
    v1
```

COMPIL – Versionning GIT

- Note :

Nous avons vu les commandes de base suivantes :

- Initialiser un dépôt : **git init**
- Ajouter un fichier pour un commit : **git add fichier**
- Faire un commit : **git commit**
- Vérifier l'état du dépôt : **git status**
- Revenir en arrière : **git-revert id_sha1 ou git checkout -f**
- Visualiser l'historique du dépôt : **git log**

COMPIL – Versionning GIT

- Création d'une branche

jiminy[tp2**] git branch** *// on vérifie sur quelle branche on se trouve !!*
* master

jiminy[tp2] **git branch versionTest** *// création d'une branche*
jiminy[tp2] **git branch**
* master
versionTest

jiminy[tp2] **git checkout versionTest** *// on se positionne sur le nouvelle branche*
Switched to branch "versionTest"

COMPIL – Versionning GIT

- Modification et commit sur la nouvelle branche

```
jiminy[tp2] cat tp.c
#include <stdio.h>
int main(int argc, char *argv[]){
printf("version versionTest v1\n");
return 0;}

 jiminy[tp2] git add tp.c
jiminy[tp2] git commit -m "tp.c versionTest v1"
Created commit 0eb1683: tp.c versionTest v1
 1 files changed, 1 insertions(+), 3 deletions(-)
```

COMPIL – Versionning GIT

- Différences entre branche

```
jiminy[tp2] git diff master versionTest
diff --git a/tp.c b/tp.c
index 718b670..1d28250 100644
--- a/tp.c
+++ b/tp.c
@@ -3,9 +3,7 @@

 int main(int argc, char *argv[]) {
-printf("hello world \n");
-
-printf("version 2\n");
+printf("version versionTest v1\n");
 return 0; }
```

COMPIL – Versionning GIT

- Merge entre branches

// retour à la branche principale pour ensuite faire un merge du code de la branche
// versionTest vers la branche principale master :

jiminy[tp2] **git checkout master**
Switched to branch "master«

jiminy[tp2] **git merge versionTest**
Updating 92b7e78..0eb1683
Fast forward
 tp.c | 4 +---
 1 files changed, 1 insertions(+), 3 deletions(-)

COMPIL – Versionning GIT

- Note:

Nous avons vu l'utilisation des commandes qui permettent de merger le code d'une branche à une autre :

•Création d'une branche : **git branch nom_branche**

•Changement de branche : **git checkout nom_branche**

•Merge de deux branches : **git merge nom_branche**

COMPIL – Versionning GIT

- Dépôt public et utilisation

```
jiminy[tp2] cd ..
jiminy[/tmp] git clone --bare –shared  tp2  tp2.git
Initialized empty Git repository in /tmp/tp2.git/
0 blocks

jiminy[/tmp] ll
total 713068
...
drwxrwxr-x 7 fcamps ii         4096 mar  4 17:23 tp2.git
...
jiminy[/tmp] cd tp2.git
jiminy[tp2.git] ll
total 32
drwxrwxr-x  2 fcamps ii 4096 mar  4 17:23 branches
-rw-rw-r--  1 fcamps ii   66 mar  4 17:23 config
-rw-rw-r--  1 fcamps ii   58 mar  4 17:23 description
-rw-rw-r--  1 fcamps ii   23 mar  4 17:23 HEAD
drwxrwxr-x  2 fcamps ii 4096 mar  4 17:23 hooks
drwxrwxr-x  2 fcamps ii 4096 mar  4 17:23 info
drwxrwxr-x 26 fcamps ii 4096 mar  4 15:30 objects
drwxrwxr-x  4 fcamps ii 4096 mar  4 17:23 refs
```

COMPIL – Versionning GIT

- Copie du dépôt dans un rép. Public HTTP

```
jiminy[/tmp] scp -r tp2.git/    fcamps@shuttle:~/public_html/
jiminy[/tmp] cd ~/public_html/
jiminy[~/public_html] ll
total 3768
...
drwxrwxr-x 7 fcamps ii    4096 mar  4 17:26 tp2.git
...

jiminy[~/public_html] cd tp2.git/
jiminy[tp2.git] git --bare update-server-info

jiminy[tp2.git] chmod a+x hooks/post-update
```

COMPIL – Versionning GIT

- Clonage d'un dépôt

jiminy[/tmp] **mkdir test**

jiminy[/tmp] **cd test**

jiminy[test] **git clone http://www.laas.fr/~fcamps/tp2.git**
Initialized empty Git repository in /tmp/test/tp2/.git/
got a5b00b5058514c1ddcef8837db5b1af464213143
walk a5b00b5058514c1ddcef8837db5b1af464213143
got 0ce73467e03cbb947456fab473ac46df81e43438
got 8fa5f5e2f8a0ecf6e204a2c0566c5ee0395cec66
got dfbe2f6631f554732e23a4e589db869e6d562827
walk 8fa5f5e2f8a0ecf6e204a2c0566c5ee0395cec66
…

COMPIL – Versionning GIT

- Mise à jour du dépôt locale

jiminy[tp2] **git repack** *// compression*
Counting objects: 22, done.
Compressing objects: 100% (15/15), done.
Writing objects: 100% (22/22), done.
Total 22 (delta 5), reused 0 (delta 0)
jiminy[tp2] **git prune** *// suppression des objets inutiles*

jiminy[tp2] **git pull ssh://fcamps@shuttle/~fcamps/public_html/tp2.git**
remote: Generating pack...
…
Updating a5b00b5..b5048a1
Fast forward
 tp.c | 15 +----------
 1 files changed, 1 insertions(+), 14 deletions(-)

COMPIL – Versionning GIT

- Mise à jour d'un dépôt distant

```
jiminy[tp2] git push ssh://fcamps@shuttle/~fcamps/public_html/tp2.git master
Counting objects: 5, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 316 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking 3 objects
refs/heads/master: a5b00b5058514c1ddcef8837db5b1af464213143 ->
b5048a11050c0b617b6034ca14bd8a3e88591571
To ssh://fcamps@shuttle/~fcamps/public_html/tp2.git
   a5b00b5..b5048a1  master -> master
```

COMPIL – Versionning GIT

- Note :

Nous avons vu les commandes qui permettent un travail collaboratif :

Création d'un dépôt : **git clone --bare --shared tp2 tp2.git**

Copie du dépôt dans un répertoire d'un serveur HTTP

Nettoyage du dépôt : **git repack && git prune (git-gc)**

Export vers le dépôt : **git push**

Import vers le dépôt local : **git pull**

COMPIL – Versionning GIT

- Note :

Une fois que l'on a récupéré une branche distante on peut se demander sur quelle branche travailler dans le dépôt locale, sur la branche master (branche clonée) ou sur une branche créée à partir de celle-ci ?

A partir du dépôt local, il est possible de créer une branche et de faire des pull vers la branche master. Il est aussi possible de travailler directement sur la branche master…

Comment faire ? :
http://www.kernel.org/pub/software/scm/git/docs/everyday.html

COMPIL – Versionning GIT

- Gestion par patch

  • Pour remettre à jour un dépôt on peut utiliser :
    - la fonction pull (déjà vu)
    - les patchs (les patchs peuvent être envoyés par mail pour un filtrage de l'intégrateur qui possèdent un dépôt sur lequel il applique les patchs puis fait des push sur le dépôt public. Il est préférable de créer une branche et d'appliquer le patch.)

  jiminy[tp5] **git format-patch origin** *// on peut aussi utiliser diff*
  0001-aaaa.patch

  jiminy[tp2] **git apply 0001-aaaa.patch**
  jiminy[tp2] **git commit -a**
  Created commit b74e5a5: application du patch
  1 files changed, 0 insertions(+), 1 deletions(-)

COMPIL – Versionning GIT

- Git-daemon

Le dépôt Git peut utiliser un serveur HTTP standard (exemple précédent)
Le dépôt peut utiliser git-daemon (réf. Linux Mag. 103 p50) (attention gère que la lecture pour l'écriture utiliser ssh)

jiminy[/tmp] **git clone --bare tp2 tp2.git** *// création d'un dépôt*
Initialized empty Git repository in /tmp/tp2.git/
0 blocks
cd tp2.git **; touch git-daemon-export-ok** *//rendre le dépôt accessible ou –export-all*
mkdir depot ; cp –r tp2.git depot *// copie dans le dépôt*

// écoute sur le port 9418, peut être modifié avec -port
shuttle[depot] **git-daemon --base-path=/tmp/depot --listen=shuttle --verbose**
[14056] Connection from 127.0.0.1:23721
[14056] Extended attributes (14 bytes) exist <host=shuttle>
[14056] Request upload-pack for '/tp2'
…

COMPIL – Versionning GIT

- GIT comme service réseau

shuttle[test] git clone git://shuttle/tp2 *// clonnage avec protocole git*
remote: Generating pack...
remote: Done counting 22 objects.
remote: Deltifying 22 objects.
 100% (22/22) done22) done
remote: Total 22, written 22 (delta 5), reused 0 (delta 0)
Indexing 22 objects.
 100% (22/22) done
Resolving 5 deltas.
 100% (5/5) done

Inetd (voir man page) : (pas de changement de port possible dans inetd)
git stream tcp nowait nobody /usr/bin/git-daemon
git-daemon --inetd --verbose --export-all
/pub/foo /pub/bar