Utilisation d’un logiciel de gestion de version

« GIT »

# 1. Introduction :

Le VCS appelé également Logiciel de Gestion de Révision (RCS), est un outil qui permet de :

* Suivre l’évolution d’un code source, pour retenir les modifications effectuées sur chaque fichier et être ainsi capable de revenir en arrière en cas de problème ;
* Retenir qui a effectué chaque modification de chaque fichier et pourquoi. Ils sont par conséquent capables de dire qui a écrit chaque ligne de code de chaque fichier et dans quel but ;
* Travailler à plusieurs, sans risquer de se marcher sur les pieds. Si deux personnes modifient un même fichier en même temps, leurs modifications doivent pouvoir être fusionnées sans perte d’information.
* Fusionner les modifications de deux personnes travaillant simultanément sur un même fichier, et d’éviter que le travail d’une de ces personnes ne soit écrasé.

Il existe deux familles :

* Logiciels de gestion de version centralisés : un serveur conserve les anciennes versions des fichiers et les développeurs s’y connectent pour prendre connaissance des fichiers qui ont été modifiés par d’autres personnes et pour y envoyer leurs modifications. Le plus connu : SVN (subversion).
* Logiciels de gestion de version distribués (DVCS) : le serveur est optionnel, chacun possède l’historique de l’évolution de chacun des fichiers. Les développeurs se transmettent directement entre eux les modifications, à la façon du peer-to-peer. Le plus connu : git.

Un gros avantage des DVCS est que, l'historique étant stockée également en local, un développeur unique et sans accès réseau peut bénéficier de la gestion des révisions. En pratique, les DVCS utilisent néanmoins généralement un serveur qui connaît l'historique des modifications et dont les développeurs se servent pour synchroniser leurs modifications. Les projets open-source utilisant git se servent généralement des serveurs github.com. (jQuery, PHP, NumPy/SciPy, ...) ou sourceforge.net (EMule, FileZilla, phpMyAdmin, ...).

# 2. gestion des correctifs :

Récupérez les fichiers Test1.c et Test2.c.

* 1. Exécutez la commande suivante : diff -u Test1.cTest2.c(ou diff -u Test1.cTest2.c¦ more) Elle vous permet de visualiser les différences entre ces deux fichiers.
* 2. Sauvegardez le résultat dans un fichier : diff -u Test1.cTest2.c> test.diff Vous venez de créer un fichier correctif (un patch) qui indique les modifications à effectuer pour passer de Test1.cà Test2.c
* 3. Pour appliquer le patch à Test1.c patch -i test.diff Test1.cVérifiez le contenu de Test1.c(commande more Test1.java) et constatez qu'il est le même que Test2.java. Pour effectuer cette comparaison directement : diff -u Test1.cTest2.c
* 4. Si vous réalisez que le patch n'était pas correct, vous pouvez revenir en arrière. Tapez : patch -R -i test.diff Test1.c Test1.c est à nouveau dans son état original.
* 5. Pour visualiser et fusionner : meld Test1.cTest2.c Entre parenthèses, remarquez que c'est un logiciel d'alignement de séquences :) (si meld n'est pas installé, kdiff3 ou tkdiff devraient avoir un rôle équivalent). Jouez avec pour comprendre comment cet outil fonctionne. En particulier mettre sous forme de correctif...

# 3 Utilisation de GIT :

Dans cette section il vous sera nécessaire de faire des recherches sur internet pour répondre à certaines questions.

Attention avant de poursuivre vous devez avoir préalablement créer un compte personnel sur github : <http://www.github.com>, si vous en avez déjà un utilisez le.

## 1 . Installation

sudo apt-get install git

ou http://git-scm.com/downloads

## 2 . Configuration

* git config --global user.name “Prénom Nom”
* git config --global user.email votre@adresse.email //cette adresse est votre adresse utilisé sur github
* git config --global core.editor votre\_editeur //optionnelle
* git config --global merge.tools meld // optionnelle
* git config –list

# 3- Git en local

Créer un répertoire de travail pour ce TP (commande mkdir)

Aller dans ce dossier (cd)

Initialiser un dépôt Git dans ce dossier : git init

Vous venez de créer deux zones : une zone modifiable appelée zone d'attente (staging area) où vous pouvez sauvegarder des modifications temporaires, et une zone immuable (de sauvegarde) vers laquelle vous enverrez vos versions dont vous êtes satisfait pour archivage.

Créer un fichier contenu.txt

* Quel est son statut ? (git status)

git add contenu.txt

* Que fait cette commande ?

Et maintenant, quel est le statut du fichier ?

git commit

* Que fait cette commande ?

Et maintenant, quel est le statut du fichier ?

Modifier votre fichier avec votre éditeur de texte préféré.

* Quel est le statut de contenu.txt ?

Placer la nouvelle version de contenu.txt dans la zone immuable.

Modifier à nouveau le fichier contenu.txt.

L’ajouter dans la zone d'attente (stage).

Le modifier une nouvelle fois.

* Expliquer le résultat de la commande git status dans ce cas précis.
* Comment afficher les dernières modifications apportées à contenu.txt (git help -a et git help -g)?
* Comment afficher les modifications (différences) entre la version de la zone modifiable (staged) et la version actuellement en zone immuable ?

Créer un programme en C qui affiche Hello World, puis le compiler.

Exclure l’extension liée au fichier exécutable que vous venez de créer (il faut créer un fichier caché .gitignore).

* Quels fichiers la commande git add . Va-t-elle indexer (stage) ?
* Quelle option de git add peut vous l’indiquer ?

Ajouter manuellement un nouveau fichier Java à la zone indexée (staged).

Le replacer en mode non suivi (réinitialiser pour qu'il soit untracked).

Placer tout votre projet, à l’exclusion des fichiers exécutables, en zone immuable (commit).

* Quelle option ajouter à la commande git commit pour indexer (stage) automatiquement tous les fichiers déjà suivis (tracked), ie. Tous ceux déjà connus par git ?
* Quelle option pour spécifier le message du commit dans la ligne de commande ?

Faire une copie du fichier contenu.txt dans le même répertoire sous un nouveau nom (cp), puis le supprimer du dépôt git avec la commande git rm.

* Que ce serait-il passé si vous aviez supprimé le fichier contenu.txt simplement avec la commande système classique rm ?

Remettre la copie de contenu.txt à la place du fichier que vous venez d’effacer, en utilisant la commande git mv.

* Que se passe-t'il ? L'indexer dans le dépôt Enregistrer dans le dépôt avec un nouveau message.

Afficher la liste des commit effectués (log).

Voyons des options très utiles de la commande précédente : -p et -3.

Que font chacune de ces deux options ?

Modifier le message du dernier commit (il faut utiliser une option de git commit).

* Mais au fait, où se trouvent les fichiers de la zone d'attente ? De la zone d'archivage ?