Table des matières

1 Préambule

Git permet de gérer un projet (de programmation généralement) et de garder en mémoire l'historique de toutes les versions d'un ensemble de fichiers. Il permet de gérer un projet à plusieurs, de programmer afin de pouvoir revenir en arrière, comparer avec d'anciennes versions et cie.

Le principe est d'avoir un serveur git (un seul possible) qui va garder en mémoire l'historique de toutes les versions et un client git (plusieurs possibles) qui vont se connecter au serveur pour mettre à jour la version des fichiers ou en récupérer les dernières versions.

Remarque : Il est possible que le serveur soit lui aussi client, dans le cas où il n'y aurait qu'un seul développeur et qu'on ne souhaite pas passer par internet.

1.1 Configuration

Afin d'avoir la coloration syntaxique, il faut faire :

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

2 Commandes universelles git

Ici, je note les commandes qui sont valables à la fois pour svn installé sur un serveur internet, ou sur une machine locale pour un usage personnel

2.1 Lister les modifications locales

La commande git status vous indique les fichiers que vous avez modifiés récemment :

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

2.2 Ajouter des fichiers ou dossiers au projet : add

Pour ajouter des fichiers il faut faire :

```
git add latex/ vim/
```

où **latex**/ et **vim**/ sont deux dossiers existant dans le dossier local de référence

Remarque : Au cas où ça serait pas clair. J'ai créé un dossier /**home**/**autiwa**/**Formulaires** grâce à [§ 3.1 on the facing page]. Dans ce dossier, j'ai créé et rempli à la main les sous-dossiers **latex**/ et **vim**/. Maintenant, grâce à la commande ci-dessus, je définis ces sous-dossiers comme étant rattachés au projet. En faisant ainsi le contenu est rajouté récursivement.

 Cette commande n'agit que sur le répertoire local (la *working copy*). Il faut ensuite appliquer ces changements au dépot (voir [§ **??** on page ??]) pour les valider.

2.3 Voir les différences entre la version du serveur et la version locale

```
git diff
```

Remarque : Il est possible de regarder les différences sur un fichier en particulier.

3 Subversion sur internet

Je vais prendre l'exemple de google code, qui est celui que j'ai choisi et que je suis en train d'apprendre.

3.1 Récupérer le contenu du projet

Une fois le projet créé (sur la page http://code.google.com/hosting/createProject), il faut faire :

```
cd Formulaire
git clone https://autiwa@code.google.com/p/autiwa-tutorials/
```

Cette commande permet de récupérer le contenu du projet et de le copier dans un dossier **Formulaires** qui sera créé dans le dossier courant.

Définition 1 (Clone)
Opération d'extraction d'une version d'un projet du repository vers un répertoire de travail local.

4 Subversion localement

Le serveur ET le client seront alors sur la même machine.

4.1 Création du projet

J'ai créé un dossier **mercury** dans mon **$HOME**, puis je fais, dans mon répertoire utilisateur :

```
cd mercury
git init
```

pour un projet que j'appelle **mercury**.

5 Gérer un conflit