

Table des matières

1	Préambule	2
1.1	Configuration	2
2	Commandes universelles git	2
2.1	Mettre à jour sa copie locale : pull	2
2.2	Lister les modifications locales	3
2.3	Ajouter des fichiers ou dossiers au projet : add	3
2.4	Voir les différences entre la version du serveur et la version locale	3
2.5	Enregistrer les modifications localement : commit	3
2.6	Annuler un commit effectué par erreur	3
3	Git sur internet	3
3.1	Récupérer le contenu du projet	4
4	Git localement	4
4.1	Création du projet	4
5	Gérer un conflit	4

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
```

De même, il faut configurer votre nom (ou pseudo) :

```
git config --global user.name "votre_pseudo"
```

Puis votre e-mail :

```
git config --global user.email moi@email.com
```

Vous pouvez aussi éditer votre fichier de configuration .gitconfig situé dans votre répertoire personnel pour y ajouter une section alias à la fin :

```
vim ~/.gitconfig
```

```
[color]
    diff = auto
    status = auto
    branch = auto
[user]
    name = votre_pseudo
    email = moi@email.com
[alias]
    ci = commit
    co = checkout
    st = status
    br = branch
```

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 Mettre à jour sa copie locale : pull

```
git pull
```

Pour cela, il faut que le dossier dans lequel on se trouve ait déjà été défini comme un dossier git via un *clone* (voir [§ 3.1 on page 4])

2.2 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)
```

2.3 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 next 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 dépôt (voir [§ 2.5]) pour les valider.

2.4 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.

2.5 Enregistrer les modifications localement : commit

Pour mettre à jour les versions sur serveur à partir des modifications effectuées localement, il faut :

```
git commit -m "initialisation"
```

où **"initialisation"** est le commentaire qui décrit la mise à jour et les modifications effectuées.



Un commit avec git est local : à moins d'envoyer ce commit sur le serveur comme on apprendra à le faire plus loin, personne ne sait que vous avez fait ce commit pour le moment. Cela a un avantage : si vous vous rendez compte que vous avez fait une erreur dans votre dernier commit, vous avez la possibilité de l'annuler (ce qui n'est pas le cas avec SVN!).

On peut modifier ces commits là, tant qu'ils sont en local. Puis une fois vérifié. Il ne reste plus qu'à les envoyer sur le serveur. Pour cela on vérifie tout d'abord ce qu'on s'apprête à envoyer, et on envoie sur le serveur :

```
git log -p
git push
```

2.6 Annuler un commit effectué par erreur

3 Git 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 Git 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

Index

Ajouter des fichiers, **3**

commande git

add, **3**

clone, **2, 4**

commit, **3**

pull, **2**

git

status, **3**

Mise à jour du dépôt, **2**

mise à jour local -> serveur, **3**