Initiation DVCS

Table des matières

1 Les VCS

Les VCS (Version Control Systems) sont des outils utilisés par les développeurs pour garder un historique de leurs codes sources. Chaque modification peut être validée (changeset) et enregistrée pour que le développeur puisse revenir dans le temps s' il y a eu un problème. Il existe un grand nombre de systèmes VCS :

– CVS
– SVN

Ces systèmes se basent sur un dépôt de source centraliser où chaque développeur peut ajouter ses changements. On parle de commit. Les autres développeurs peuvent alors se mettre à jour. On parle d'update. Voir figure 1.

Plus récemment, des systèmes décentralisés ont vu le jour. On parle de DVCS (Decentralized Version Control Systems). Les plus connus sont :

– Mercurial

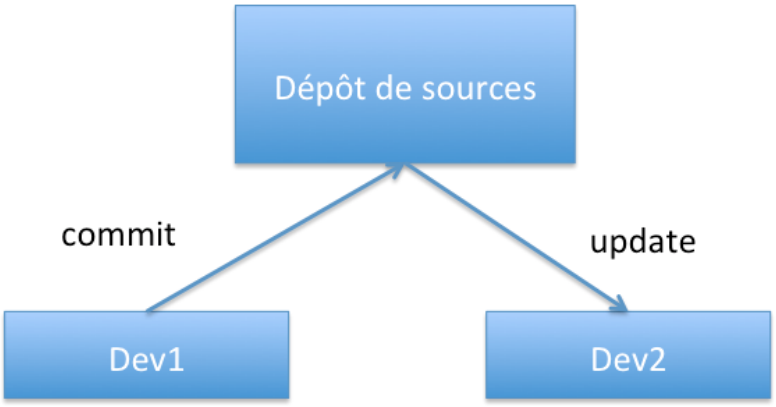

FIGURE 1 – Commit et Update sur VCS

– Git

Ici le principe est radicalement différent : chaque développeur possède un dépôt sur lequel il peut valider ses changements. Les développeurs se synchronisent ensuite avec un ou plusieurs autres dépôts via les opérations de push et de pull. Voir figure 2.

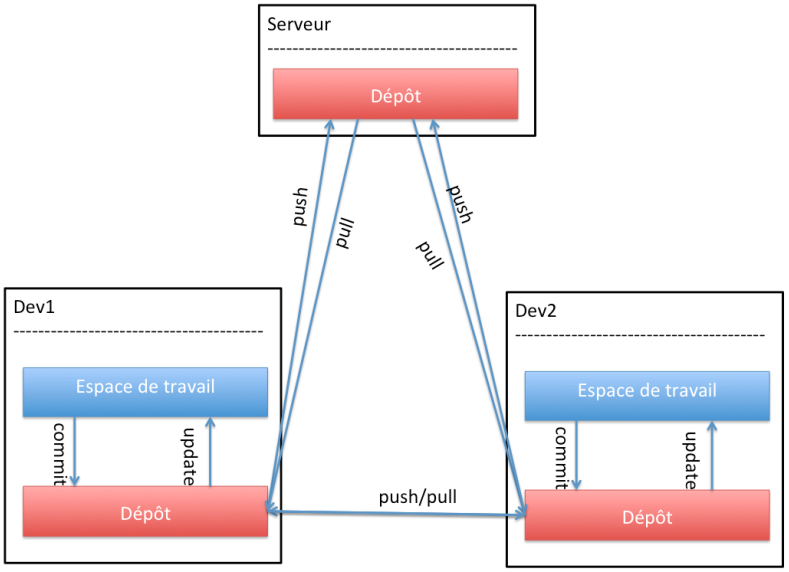

FIGURE 2 – Commit, Update, Push et Pull sur DVCS

Dans ce TP nous utiliserons Mercurial.

2 Les premiers pas avec Mercurial

2.1 Configuration

Dans le fichier $HOME/.hgrc ajoutez les lignes suivantes :

```
[ui]
username = Prenom Nom <votreadreesse@iut.lerest.fr>
editor = emacs
```

Ce fichier permet d'associer votre nom et adresse email à chaque validation de changement que vous réaliserez (*commit*). **Attention :** Comme vous travaillez à deux sur la même machine il est nécessaire de changer ce fichier à chaque fois que vous changerez d'utilisateurs. Utilisez des commentaires :

```
[ui]
#username = Prenom Nom <votreadreesse@iut.lerest.fr>
username = Super Noob <sonadreesse@iut.lerest.fr>
editor = emacs
```

2.2 Création d'un dépôt

Avant toute chose, il est nécessaire de créer un dépôt de source. Cela doit être fait par une seule et même personne avant le début du projet :

```
#Creation du projet :
mkdir MonProjet
cd MonProjet
#Initialisation du depot
hg init
#Creation d'un fichier readme
touch README.md
#Ajout du fichier README.md au depot
hg add README.md
#Validation des changements (ajout du fichier et init)
hg commit -m "Init"
```

2.3 Exercices

Créer un dépôt et ajouter des fichiers

1. Créez un nouveau dépôt (nommé Bidon ?). Avec un fichier README.md et un premier commit.
2. Ajoutez le texte "modification 1" dans le fichier README.md en respectant la syntaxe de type Markdown.

3. Commitez vos changements (*hg commit*) avec un message approprié.
4. Créez un nouveau dossier (nommé NouveauDossier ?) et contenant deux fichiers : fic1.md et fic2.md.
5. Ajouter l'ensemble du dossier, avec la commande *hg add*, au dépôt de source.
6. Commitez.
7. À la racine du projet, lancer la commande *ls -al*. À quoi sert le dossier *.hg* ?

Revenir en arrière

1. Ajoutez le texte "modification 2" dans le fichier README.md.
2. Commitez.
3. Lister l'ensemble des changements avec la commande *hg log*
4. Revenez à la version 1 avec la commande *hg update -c NUM_ VERSION*.
5. Revenez à la version courante avec la commande *hg update*.

3 Travaillez à plusieurs

3.1 Clone, Push et Pull

Pour travailler à plusieurs sur un même dépôt, il est tout d'abord nécessaire de *cloner* un dépôt existant. Pour l'instant, nous supposons que les deux développeurs sont sur la même machine :

```
#Developpeur 1:
hg clone Bidon Bidon-dev1

#Developpeur 2:
hg clone Bidon Bidon-dev2
```

L'opération de clone créer un dossier Bidon-dev1 pour le premier développeur, et un dossier Bidon-dev2 pour le second.

3.2 Exercices :

Pour Dev1 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. Cloner le dossier Bidon vers Bidon-votrenom

Pour Dev2 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. Cloner le dossier Bidon vers Bidon-votrenom

Pour Dev1 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.

2. Ajoutez le texte “modification 3” dans le fichier Bidon-votrenom/README.md.
3. Commitez.
4. Pusher avec la commande *hg push*.

Pour Dev2 :

1. Assurez-vous que votre nome est bien “activé” dans le fichier $HOME/.hgrc.
2. Puller les changements de Dev1 avec la commande *hg pull*.
3. Les changements de Dev1 sont alors dans le fichier .hg mais pas dans votre espace de travail. Comment mettre à jour l’espace de travail ? Lancer la bonne commande.
4. Ajoutez le texte “modification 4” dans le fichier Bidon-votrenom/README.md.
5. Commitez.
6. Pusher vos changements.

Pour Dev1 :

1. Mettez-vous à jour.

3.3 Modifications concurrentes

Comme mercurial est un DVCS il est par nature décentralisé. Ainsi, plusieurs développeurs peuvent commiter des changements en même temps. À vrai dire la règle est de toujours commiter avant de se mettre à jour (cela permet de revenir à sa version si l’on découvre des problèmes après une mise à jour). Le fait de commiter deux versions en parallèle va rendre l’historique non linéaire. Il faudra merger (fusionner) les deux versions avec la commande *hg merge*.

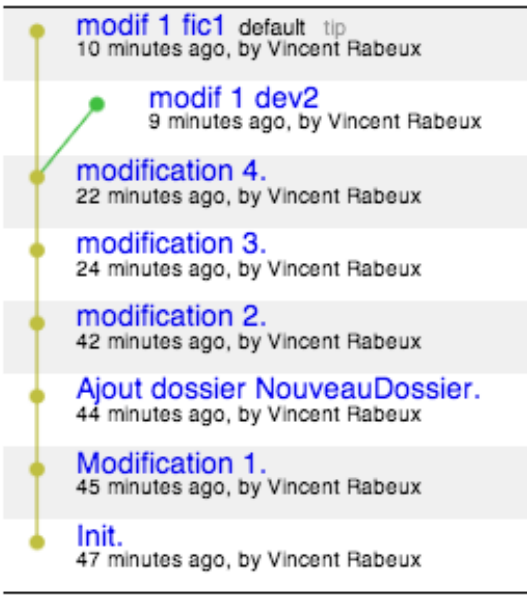

FIGURE 3 – Versions en parallèle

Pour Dev1 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. cd Bindon-votrenom
3. Ajoutez le texte "modification 1" dans le fichier Bidon-votrenom/NouveauDossier/fic1.md.
4. Commitez.

Pour Dev2 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. cd Bindon-votrenom
3. Ajoutez le texte "modification 1" dans le fichier Bidon-votrenom/NouveauDossier/fic2.md. (ATTENTION fic2.md).
4. Commitez.

Pour Dev1 :

1. Pushez vos changements.

Pour Dev2 :

1. Puller les changements de Dev1.
2. Que vous dit la commande *hg update* ? Pourquoi ?
3. Lancer la commande *hg heads* pour voir les deux changeset concurrents.
4. Pour visualiser l'arbre des versions, nous allons utiliser une extension (graphlog). Éditez le fichier $HOME/.hgrc et ajoutez les lignes suivantes :

```
[extensions]
graphlog =
```

   Lancer la commande *hg log -G* pour voir l'arbre des changements.
5. Nous avons donc deux versions en parallèle, qu'il faut fusionner. Pour cela lancer la commande *hg merge*.
6. Vérifiez que tous les changements sont bien fusionnés.
7. Commitez la fusion avec *hg commit* et le message *Merged*.
8. Pushez !

Pour Dev1 :

1. Mettez-vous à jour.

3.4 Les conflits

Il peut arriver que la fusion ne puisse pas se faire automatiquement. On parle de conflits. Les conflits arrivent en général quand deux développeurs modifient les mêmes lignes dans le même fichier.

Pour Dev2 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. cd Bidon-votrenom
3. Ajoutez le texte "conflit dev2" dans le fichier README.md.
4. Commitez, Pushez

Pour Dev1 :

1. Assurez-vous que votre nome est bien "activé" dans le fichier $HOME/.hgrc.
2. cd Bidon-votrenom
3. Ajoutez le texte "conflit dev1" dans le fichier README.md
4. Commitez.
5. Essayer de vous mettre à jour. Quel est le message d'erreur ?
6. Fusionner le conflit à a la main en modifiant README.md (vous pouvez garder à la fois vos modifications et celle de Dev2).
7. Une fois le conflit résolut il faut le "dire" à mercurial avec la commande *hg resolve –mark README.md*.
8. Commitez la fusion avec le message *Merged*.
9. Pushez

4 Bitbucket

Bitbucket est un site web permettant d'héberger des dépôts Mercurial sur internet de façon privée ou publique. C'est par ce site web que vous pourrez travailler sur les projets de l'IUT.

1. créez-vous un compte `https://bitbucket.org/account/signup/`
   (a) utilisez un nom d'utilisateur du type *prenom_nom* ou *pnom* et pas un pseudo genre TitiLeLapin !
   (b) ainsi que l'adresse email de l'IUT.

Pour Dev1 :

1. Créez un nouveau projet appelé 2013-ASR2-Mercurial-votrenom. Sélectionnez Mercurial comme type de dépôt et activez les options Wiki et Issue Tracking.
2. Clickez sur le bouton clone du projet (en haut à gauche), et copier-coller la ligne de commande dans votre terminal.
3. Toujours dans le terminal, déplacer dans le dossier créé.
4. Créez un fichier README.md contenant une petite description de ce cours et le nom de votre binôme (Dev2).
5. Ajoutez le fichier, commitez et pushez.
6. Dans la configuration du projet (sur bitbucket) ajoutez, en tant que membre, moi (vrabeux) et votre binôme (dev2).

Pour Dev2 :

1. Loggez-vous sur bitbucket, allez sur le projet de Dev1, clonez-le.
2. Créez un dossier MercurialTutorial contenant un fichier tuto.md.
3. Dans le fichier tuto.md, ajoutez sous forme de liste de type markdown, les sections de ce même PDF.
4. Commitez et pushez.
5. Dans Bitbucket, allez dans la partie Issues, et clicher sur *Create your first issue*.

6. Titre : Écrire partie 1 - Les VCS.
7. Description : à vous de voir, mais cela soit être détaillé.
8. Assignee : Dev1 ! !
9. Validez.

Pour Dev1 :

1. Allez voir la demande de Dev2. Accomplissez-la, commitez,
2. Tagguer la version courante avec un petit nom (part1) avec la commande *hg tag part1*.
3. Pushez.
4. Fermer la demande sur bitbucket en précisant le numéro de changeset qui corrige la demande.

5 TM

Pour le 1er Avril, créez un projet (imaginaire) de votre choix :
- Celui-ci devra contenir plusieurs fichiers et dossiers.
- Chaque fichier doit être modifié (dans l'historique) par l'ensemble des membres.
- Attention : plus l'historique des versions est linéaire moins le travail sera bon.
- Il doit y avoir plusieurs demandes sur bitbucket. Chaque demande doit être fermé et préciser le numéro du changeset qui valide la demande.
- Envoyez-moi l'URL du projet par mail.