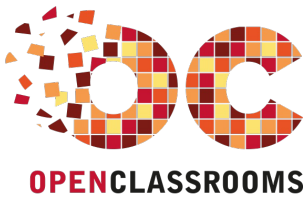
Gérer vos projets avec Mercurial

Par Meianki



www.openclassrooms.com

Dernière mise à jour le 29/01/2011

Sommaire

Gérer vos projets avec Mercurial

Par Meianki
Mise à jour : 29/01/2011
Difficulté : Facile

Vous avez déjà travaillé à plusieurs sur un même projet ?
Alors vous avez sûrement déjà été confronté aux échanges interminables de fichiers *via* ftp/mail/messagerie...
Vous avez aussi évidemment connu la joie de voir toutes vos modifications écrasées par un collègue qui mettait à jour juste après vous.

Vous n'avez jamais travaillé à plusieurs sur un projet et pourtant vous avez réussi par un hasard malchanceux à écraser vos modifications ?
Vous avez même perdu les sources de votre projet ?

Si vous vous reconnaissez dans un de ces cas, alors ce tutoriel est pour vous.
Je vais vous présenter Mercurial qui est un logiciel de gestion de version totalement libre et gratuit.
Il vous permettra de gérer efficacement vos projets et de revenir facilement sur vos pas !
Sommaire du tutoriel :

Gestion de version... Quésaco ?

On en parle peu et pourtant c'est quelque chose de magnifique.
Vous avez peut-être un jour vu ou entendu un des sigles suivant : CVS, SVN, GIT...
Tous sont des solutions de gestion de version.

Voyons ce qu'en dit Wikipédia :

Citation : Wikipedia

> La gestion de versions (en anglais *version control* ou *revision control*) est une activité qui consiste à maintenir l'ensemble des versions ou révisions d'un logiciel ou autre document. Essentiellement utilisée dans le domaine de la création de logiciels, elle est surtout concernée par le code source ; mais elle peut être utilisée pour tout type de document informatique.
>
> Cette activité étant fastidieuse et relativement complexe, un appui logiciel est presque indispensable.

Bon, expliquons un peu plus en détails ce « maintenir l'ensemble des versions ou révisions d'un logiciel ».

Quand vous avez un logiciel en construction et que vous travaillez dessus, vous faites régulièrement des changements sur la source. Un gestionnaire de version permet de répertorier ces changements et de revenir dessus si besoin est.
De plus, il facilite le travail d'équipe en vous avertissant s'il y a des conflits (si deux personnes ont édité un même fichier en même temps).

Voilà donc ce qu'est la gestion de version.

Quand utiliser Mercurial ?

Là vous vous dites probablement que ce tutoriel ne vous sert à rien, en effet, vous travaillez souvent seul et au pire une bonne

répartition du travail fait qu'on se marche pas sur les pattes.

Ah ! Si tout était si simple.

Voici quelques situations que j'ai expérimentées ou vues, et où Mercurial fut d'un grand secours :

- un développement en solo, et boum on a tout cassé ! Le problème : on a modifié des dizaines de fichiers avant d'arriver à ce magnifique résultat... Mercurial permet de récupérer n'importe quelle version de fichier que vous avez sauvegardé !
- un développement en solo encore mais vous naviguez souvent entre plusieurs machines (maison, boulot, portable...) avec un dépôt online vous pouvez synchroniser tout votre développement sans vous demander où se trouve la dernière version à jour de votre logiciel !
- un développement en équipe ? Alors là Mercurial vous montrera toute sa puissance! Vous pouvez vous organiser comme des patates que ça sera même pas grave ! Il vous faudra juste vous habituer à *merger* (ce qui arrive plus fréquemment en fin de projet quand il faut vite finir !).
- un développement en équipe, mais vous partez en vacances et vous voulez absolument travailler ? Faites vos versions en local et préparez-vous à faire un *merge* du tonnerre en rentrant de vacances !

Installation

Eh oui ! Comme tout logiciel, il est d'usage de l'installer avant de l'utiliser.
Heureusement, c'est une des choses les plus faciles à faire.

Sous Linux

Vous êtes sous Linux ?
Parfait ! Il vous suffit d'installer le paquet « mercurial » à partir de vos dépôts.

Pour les gens qui n'auraient pas encore jeté un œil au tutoriel de M@teo (ou pour les feignants) et qui sont sous Debian/Ubuntu ou dérivés, tapez ceci dans votre shell préféré.

Code : Console

```
sudo apt-get install mercurial
```

Si ce n'est pas présent dans vos dépôts préférés, allez voir les binaires par ici.

Sous Windows

Là, c'est plus compliqué. (À quand le système de paquets pour Windows.)

Allez à l'adresse suivante : http://mercurial.selenic.com/downloads/ puis téléchargez Mercurial pour windows (dernier lien).

Un fois téléchargé, il suffit de double cliquer pour installer.

Cela fait, vous êtes maintenant en mesure d'utiliser Mercurial sur votre ordinateur !
C'est magnifique !

Sous MacOS X

Vous pouvez vous procurer des binaires à jour pour Mac par ici.

Intégration dans Netbeans

Il existe un plugin pour Mercurial qui est intégré dans les distributions officielles de Netbeans, donc normalement vous pouvez allez jeter un œil dans le menu "Versioning" puis dans le sous menu "Mercurial" et là vous aurez une liste de commandes qui correspondent à ce qui sera présenté dans le suite.

Intégration dans Eclipse

Pour Eclipse le plugin existe aussi et est utilisable à partir de la version 3.3 de Eclipse. Voici un lien (en Anglais) pour vous indiquer la démarche à suivre : Vertrace (Mercurial Eclipse)
Pour les anglophobes voici la procédure à suivre indiqué par ce site :
Tout d'abords allez dans le menu "Aide" puis dans "Mise à jours" puis dans "Chercher et installer".
Sélectionnez "Chercher de nouveau plugins" puis dans la fenêtre qui s'ouvre cliquez sur "Ajouter un site"
Puis ajoutez ceci :

Nom: Vectrace (Mercurial Eclipse plugin)
URL: http://www.vectrace.com/eclipse-update/

Toute la suite du tutoriel se fera en console pour des explications plus poussée et une meilleure transparence! Vous pourrez toutefois suivre le tutoriel avec les outils graphiques puisqu'ils intègrent toutes les commandes de base.

Utilisation

Opérations préalables

Toutes les opérations de ce chapitre ne sont à effectuer **qu'une seule fois par projet et par machine**.

Créer un dépôt online

Lorsqu'on travaille en équipe avec Mercurial, il faut avant tout un dépôt accessible tout le temps.

Si vous souhaitez juste faire de la sauvegarde pour votre projet en solo, cette étape n'est pas utile. Si vous vous synchronisez régulièrement en vous voyant, ce n'est pas nécessaire non plus.

Pour ce faire, on utilisera http://bitbucket.org qui fournit un dépôt Mercurial gratuitement.

Je passe assez rapidement ici car le site est extrêmement bien documenté et très simple d'utilisation.

La première chose à faire maintenant est de se créer un compte sur ce site.
Puis vous créez un projet et enfin vous rajoutez tous vos collègues (qui se seront inscrits aussi) aux utilisateurs/administrateurs.
Je vous recommande vivement ce site car il fournit aussi un bug tracker et un wiki par projet ce qui permet une gestion du projet de bout en bout !

Si bitbucket ne vous convient pas, je préconisais http://mercurial.intuxication.org/dashboard dans une version précédente du tuto.

Une fois cela fait, vous pouvez enfin faire du Mercurial !

À partir de maintenant, tout se passera en ligne de commande.

Pour les Linuxiens, pas de soucis : vous ouvrez un shell. Pour les Windowsiens par contre, il faut aller dans « Démarrer » puis « Exécuter » puis tapez « cmd » et validez avec la touche Entrée. Cela devrait vous ouvrir une console.

Il faut maintenant vous placer dans le dossier où vous voulez stocker vos projets à coup de CD (voir le tutoriel de M@teo sur Linux pour ça).

La commande CD fonctionne aussi dans la console Windows !

On va commencer par récupérer tous les fichiers du projet. On dit qu'on clone le dépôt (en fait, vous créez un dépôt identique à celui que vous clonez, simplement comme votre PC n'est pas tout le temps joignable, on ne l'utilisera pas comme dépôt principal).
Vous allez donc taper la commande :

Code : Console

```
hg clone http://mercurial.intuxication.org/hg/monprojet
```

Remplacez « monprojet » par le nom de votre projet.

Euh ?! Moi je veux juste faire de la sauvegarde en local, je fais comment ?

En effet, là, vous ne pouvez rien cloner. Il faut initialiser le dépôt.
Allez dans le répertoire de votre projet puis tapez :

Code : Console

```
hg init
```

Maintenant la travail commence !

Les commandes principales

Voir le statut de tous les fichiers

Mercurial tient un statut de tous les fichiers qui sont dans le dossier de votre projet.
Pour voir le statut de tous les fichiers, il faut taper la commande suivante :

Code : Console

```
hg st
```

Cela va vous afficher tous les fichiers qui ont un statut particulier.
Voici un exemple de ce que peut donner cette commande :

Code : Console

```
M teletubies/lala
M listePersonnages
A pokemon/pikachu
R telerama/zoidberg
! onepiece/pipo
? HNG/touyaAkira
```

Explications !
Vous pouvez voir que sur chaque ligne, vous avez un statut qui tient sur un caractère et un fichier.
Les statuts sont les suivants :

- M : le fichier a été modifié depuis la dernière version ;
- A : le fichier a été ajouté à Mercurial ;
- R : le fichier a été supprimé de Mercurial ;
- ! : le fichier a été supprimé physiquement mais pas de Mercurial ;
- ? : le fichier a été ajouté physiquement mais n'est pas pris en charge par Mercurial.

Vous remarquerez que les fichiers qui sont pris en charge par Mercurial et qui ne sont pas modifiés depuis la dernière version n'apparaissent pas.

Qu'est-ce que c'est que cette histoire de prise en charge, là ?

C'est un point important pour comprendre Mercurial.
Vous pouvez vouloir faire des fichiers qui ne seront pour autant pas mis sur le dépôt. Prenons l'exemple d'un fichier de test qui récupère les sorties du programme pour que vous fassiez des vérifications. Un tel fichier n'a pas de réel intérêt pour le projet. Il est juste là temporairement pour des besoins de débogage. Du coup, si l'on ne veut pas le mettre dans sa version de Mercurial, il ne faudra pas l'y ajouter. Il aura le statut « ? ».
Et si par malheur vous avez ajouté un fichier qui ne sert à rien au Mercurial, vous pouvez toujours le supprimer du Mercurial ; il prendra alors le statut « R ».

Ajouter/supprimer un fichier à Mercurial

Comme on vient de le voir, créer un fichier ne suffit pas pour que Mercurial prenne en charge ses versions. Il faut lui dire de le faire.
Rien de bien compliqué, il suffit de taper :

Code : Console

```
hg add monfichier
```

Et pouf !  Mercurial le prendra en charge dans la prochaine version !
Si vous faites `hg st`, vous verrez que « monfichier » est passé du statut « ? » au statut « A ».

Pour supprimer un fichier, c'est tout aussi simple que pour l'ajout :

Code : Console

```
hg remove monfichier
```

Si vous faites `hg st`, vous verrez que « monfichier » est passé du statut « normal » (il n'apparaît pas) au statut « R ».
S'il était en « ! » (donc supprimé physiquement), vous en perdrez définitivement trace dans les versions suivantes (mais vous pourrez toujours le récupérer d'une ancienne).

Fatigué de faire ça à chaque fichier ?
Si vous avez beaucoup de fichiers à ajouter et à supprimer, il y a une commande bien utile :

Code : Console

```
hg addr
```

Cette commande va passer tous les fichiers dont le statut est « ? » en « A », et tous les « ! » en « R ».
En résumé, ça applique tous les ajouts/suppressions que vous avez faits physiquement.

Création de version, mise à jour du serveur et synchronisation

Vous avez modifié/supprimé/ajouté des fichiers jusqu'à plus soif ? Une fois la série de modifications effectuée, il faut maintenant les « pousser » sur le serveur.

 Pousser ?!

Oui oui, je sais, c'est bizarre. Mais vous comprendrez quand vous verrez la commande.

Mais avant de pousser, il faut dire à Mercurial : « J'ai bien travaillé, regarde ma nouvelle version ».
On tape donc cette commande :

Code : Console

```
hg ci
```

Elle va ouvrir un document texte un peu pré-rempli (Vim sous Linux et Bloc-notes sous Windows).
Il faut laisser un court message (une ligne) qui explique où commence ce que vous avez fait (ex. : correction bug #124596).
Ne modifiez pas ce qui est en dessous, ça décrit les fichiers modifiés.
Pour les allergiques à Vim, il est possible de changer et d'ouvrir Nano par défaut (voir la section « Configurer Mercurial »).

Certains auront peut-être une erreur ici.

En effet il se peut que Mercurial vous dise :

Code : Console

```
abandon : no username supplied (see "hg help config")
```

Il y a deux façon de corriger ce problème. Soit vous modifiez le fichier **hgrc** dans le dossier **.hg**. Soit vous créez/modifiez le fichier **.hgrc** dans votre dossier utilisateur. Dans les deux cas, il vous faut mettre les lignes suivantes :

Code : Console

```
[ui]
username=mon nom  <mon_adresse@mail.pipo>
```

Vous pouvez mettre ce que vous voulez, c'est juste ce qui apparaîtra comme nom d'utilisateur dans les commits.

Bon maintenant que nous avons réglé ce petit problème, vous voici l'heureux créateur de ce qu'on appelle une révision ou version. Maintenant, vous êtes prêt à mettre à jour le serveur avec vos changements.
Il faut taper la commande suivante :

Code : Console

```
hg push
```

(Vous comprenez maintenant ? On pousse. )
Et hop, c'est magique ! On a mis à jour le serveur avec nos fichiers où l'on a corrigé toutes les fautes (pour en ajouter d'autres mais ça, il ne faut pas le dire ).
Normalement, ça va vous demander un login et un mot de passe. Puis à la fin, ça va vous dire les changements effectués sur le serveur.
Ras-le-bol d'entrer le login et le mot de passe à chaque *push* ?
Je vous dirige encore une fois vers la partie « Configurer Mercurial ».

Maintenant, imaginons que vous êtes parti quelques jours mais que vos copains développeurs ont bossé (plus ou moins bien...).
Il faut donc télécharger leurs modifications. En d'autres termes, se synchroniser !
Tapez donc la commande suivante pour voir...

Code : Console

```
hg pull -u
```

Han, tiens ! ça marche. Chouette. 
Notez que *pull* veut dire « tirer », le contraire de *push* (« pousser ») donc.
Le -u veut dire qu'on met ses fichiers à jour (car on peut en effet récupérer les fichiers sans se mettre à jour mais c'est déjà de l'utilisation un peu plus évoluée que je n'aborderai pas...).
Bon : on va s'amuser un peu maintenant.

Mergeons !

Imaginons quelque chose comme ça :
Vous faites un `hg pull -u` pour vous mettre à jour. Dans une contrée lointaine, quelqu'un fait pareil. O.K. pour l'instant, tout va bien. Vous faites vos modifications. Notre protagoniste étrange fait de même mais il est plus rapide (rah ! le fourbe !). Il met le serveur à jour.
Quelque temps plus tard, vous avez enfin fini. Vous faites :

Code : Console

```
hg ci
hg push
```

Et là : PAF !
Mercurial ne veut rien savoir, vous n'êtes pas synchro avec le serveur. En effet, vous avez modifié une version antérieure à celle qu'il possède.
Il vous faut donc faire ce qu'on appelle un « merge », c'est-à-dire faire en sorte que vos modifications n'explosent pas tout ce qu'ont fait vos collègues.
On va commencer par récupérer ce qui a été fait sur le serveur : `hg pull -u` (pas d'inquiétudes, ça ne va pas écraser vos modifications).
Ensuite, il faut vous synchroniser avec ce que vous avez téléchargé. Autrement dit, il faut intégrer vos modifications dans cette nouvelle version.
On commence par faire `hg merge`, ce qui va régler tous les conflits mineurs, c'est-à-dire les fichiers que vous avez modifiés mais auxquels vos collègues n'ont pas touché ; c'est simple, il suffit de prendre vos modifications. Comme c'est facile, Mercurial le fait tout seul.
Par contre, si vous avez modifié un fichier que votre ami a aussi modifié, il faut en général régler le conflit à la main. Au sortir de la commande *merge* et s'il y a des conflits un peu compliqués, Mercurial va donc vous ouvrir un éditeur de texte avec trois versions du fichier : la vôtre, celle du serveur et le fichier résultat. À vous de vous débrouiller avec ça pour en tirer quelque chose. Il vous faudra donc analyser le fichier et tout et tout pour voir ce que vous pouvez garder/changer/supprimer.
Une fois le merge effectué, faites :

Code : Console

```
hg ci
hg push
```

Les modifications seront uploadées sur le serveur. Dans ce dernier `ci`, on indique en général qu'on a fait un *merge*.

Lire le log

Le *log* ? C'est quoi ça encore ?!

Ce qu'on appelle le *log*, c'est la liste des modifications faites. En gros, on va lire les messages laissés quand on fait un `hg ci`.
Deux commandes pour ça.
Lire le dernier changement :

Code : Console

```
hg tip
```

Lire tous les changements :

Code : Console

```
hg log
```

Tagger une version

Une dernière chose.
Il n'est pas beau d'avoir des numéros de versions qui ressemblent à af3f049e9b41.

Pour avoir des beaux numéros de versions, vous pouvez utiliser la commande `hg tag v0.5rc`.
Cette commande va *tagger* la version actuelle pour qu'elle ait le nom v0.5rc.
Ainsi, vous pouvez faire comme les vrais pros et avoir des versions qui ressemblent à quelque chose.

Le retour sur version

Vous avez fait une grosse boulette et vous avez mis tout ça sur le serveur ?
Pas de panique : on peut réparer ça.

Pour retourner à une version précédente d'un fichier, il faut utiliser la commande `revert`.

On l'utilise comme ceci :

Code : Console

```
hg revert -r XX monfichier
```

Cela aura pour effet de récupérer la version numéro XX du fichier monfichier.

Mais comment connaît-on la version qu'on veut récupérer ?

Ah ! Bonne question !
Il faut utiliser la commande `log` (ou `tip` si vous voulez ne reculer que d'une version).
Quand vous faites `hg log`, vous avez plusieurs blocs qui ressemblent à ça :

Code : Console

```
changeset:   18:af3f049e9b41
user:        meianki@Hyperion
date:        Tue Aug 11 00:37:49 2009 +0200
summary:     Plein de trucs
```

Bonne explication en partant du bas :

- summary : c'est ce qu'on a écrit quand on a fait `ci` pour créer cette version ;
- date : c'est relativement explicite ; 
- user : c'est pas mal explicite aussi...
- changeset : là, c'est intéressant ! Vous voyez un truc un peu exotique, mais pas du tout. Si vous faites `hg log` et que vous avez plein de révisions, regardez les changeset et en particulier ce qui se trouve avant les « : ». Vu ? Ce numéro (ici dans l'exemple : 18) s'incrémente à chaque fois qu'on fait `ci`. C'est ce numéro que vous devez récupérer pour faire le `revert`.

On aura donc cette commande :

Code : Console

```
hg revert -r 18 monfichier
```

Pour information, le numéro que l'on trouve après les « : » est l'identifiant unique de la modification.

En effet, si vous et votre ami Bob êtes tous les deux à la même version 17 et que vous faites des modifications différentes, lorsque vous ferez un `hg ci`, vous aurez alors tous les deux une version 18 mais elle sera différente (et différenciée par cet identifiant unique).

Configurer Mercurial

Authentification automatique

Vous en avez assez de taper votre *login* et votre mot de passe pour faire un `ci` ?

Allez donc dans le dossier de votre projet, puis dans le dossier `.hg/`. Enfin, éditez hgrc et ajoutez ceci :

Code : Console

```
[paths]
default = http://USER:PASS@adresse.de.mon.depot/mercurial/
```

Évidemment, il faut changer USER par votre nom d'utilisateur, PASS par votre mot de passe, et mettre la bonne adresse pour le dépôt.
Ainsi, vous n'aurez plus besoin d'écrire vos identifiants à chaque `ci`, Mercurial ira les chercher ici.

Ignorer des fichiers

Vous avez des milliers de fichiers qui sont des fichiers cache générés par votre programme, ce qui pourrit l'affichage à chaque `hg st` ?
Vous allez pouvoir faire du ménage.
Allez à la racine de votre projet, puis créez un fichier .hgignore. À l'intérieur, mettez les noms des fichiers qu'il ne faut jamais prendre en compte.
Si vous souhaitez faire quelque chose de global, vous pouvez mettre quelque chose comme `*.cache` ou `cache/*`.
Ça aura pour effet d'ignorer tous les fichiers qui finissent par .cache dans le premier cas, et tous les fichiers dans le dossier `cache` dans le second.

Attention ! Ce fichier .hgignore doit être ajouté au Mercurial avec `hg add` ! Il est en effet partagé par tout le monde !

Changer l'éditeur par défaut de Mercurial

Pour faire ça, il faut changer la variable d'environnement HGEDITOR (ou EDITOR).
La commande suivante le fait bien :

Code : Console

```
setenv HGEDITOR "nano"
```

Eh bien, je pense que je n'ai plus rien à vous apprendre sur Mercurial.
Le plus intéressant maintenant pour vous est d'aller voir les documentations sur Mercurial online (voici la Bible : http://mercurial.selenic.com/wiki/) ou encore en local :

Code : Console

```
hg --help
```

À vous de coder maintenant ! 
J'espère avoir amélioré coopération et collaboration dans le monde, voire amené la paix (sait-on jamais ! La puissance de

Mercurial est illimitée !).

Le mot de la fin : n'hésitez pas à utiliser Mercurial ! Aucun projet n'est trop petit ou trop gros pour lui.

Si vous avez besoin d'informations sur Mercurial :

- le site officiel ;
- la *man page* de `hg`.