Introduction à Mercurial et RhodeCode

Gary Verhaegen

14 novembre 2011

1 Introduction

Dans le cadre des projets d'informatique du cours INFO-H-100, nous vous demandons d'utiliser le logiciel de gestion de version Mercurial pour remettre votre code.

Le présent document a pour but de présenter les bases de Mercurial.

2 La gestion de version

Mercurial est un logiciel de gestion de version, c'est-à-dire un logiciel dont le but est d'aider l'utilisateur a conserver plusieurs versions successives de ses documents. Cela inclu la sauvegarde de ces différentes versions, la possibilité de passer relativement facilement d'une version à l'autre, et la possibilité de comparer deux versions différentes.

Il faut manuellement indiquer à Mercurial quels dossiers il doit gérer, et il faut ensuite lui dire quand vous voulez enregistrer une nouvelle version.

Mercurial enregistre toutes les informations dont il a besoin dans un *dépôt*. Quand vous demandez à Mercurial de gérer les versions d'un dossier, il créé par défaut un dépôt directement dans le dossier géré ; ce dépôt est caché par défaut.

Mercurial permet également d'envoyer cet historique de versions sur un serveur RhodeCode. Un tel serveur est mis à disposition des étudiants de BA1 à l'adresse `http://informa2.ulb.ac.be`.

3 Mise en place du dépôt sur `informa2`

Pour utiliser Mercurial dans le cadre des projets d'informatique, vous devez Commencer par vous connecter au serveur en tapant l'adresse `http://informa2.ulb.ac.be` dans votre navigateur internet. Vous arrivez sur une page de login ; vous pouvez vous connecter en utilisant votre netid [1]. Une fois

1. C'est-à-dire la même combinaison de login et mot de passe que pour votre compte mail ULB, monULB, etc.

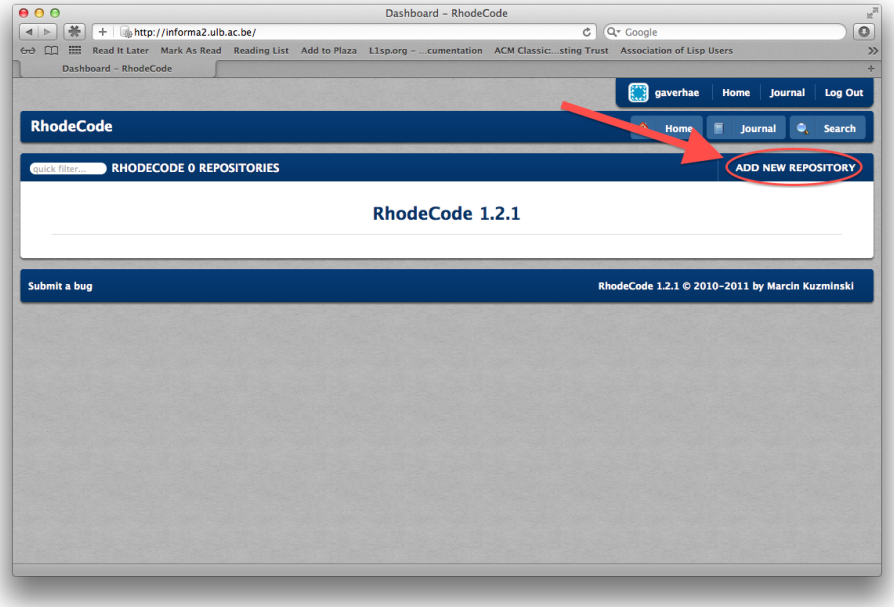

FIGURE 1 – Ajout d'un nouveau dépôt.

connecté, vous devez cliquer sur `ADD NEW REPOSITORY`, comme montré dans la figure 1.

A l'écran de création du dépôt, vous devez mettre "INFO-H-100-<netid>-1" comme nom de projet et "Projet INFO-H-100" comme description du projet. Vous ne devez pas modifier les autres options. Par exemple, dans mon cas, ça donnerait "INFO-H-100-gaverhae-1".

ATTENTION

Vous devez absolument respecter la convention de nommage décrite ci-dessus pour le dépôt de votre projet. Si vous donnez un autre nom à votre projet, il ne sera pas pris en compte et on considèrera que vous n'avez pas remis le projet.

Une fois le projet créé, vous revenez à la page d'accueil qui liste vos projets. Cliquez sur le projet que vous venez de créer ; vous devriez voir un résumé de votre projet similaire à la capture d'écran de la figure 3. A ce stade, la seule information qui vous intéresse est l'URL De votre projet, qui est entourée sur la figure 3.

Pour utiliser cette URL, vous allez avoir besoin de Mercurial. Voyons donc comment l'installer.

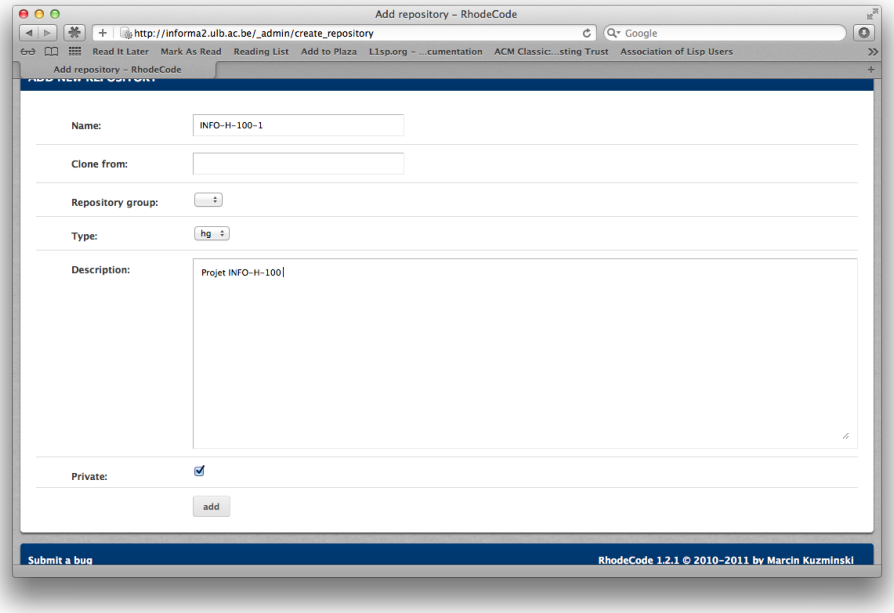

FIGURE 2 – Options à remplir pour créer le projet.

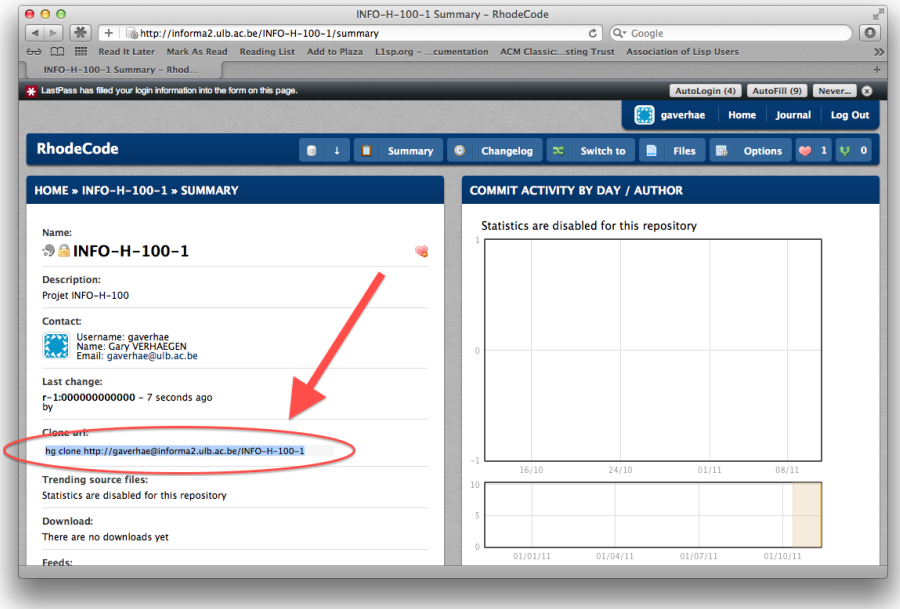

FIGURE 3 – Page de résumé d'un projet.

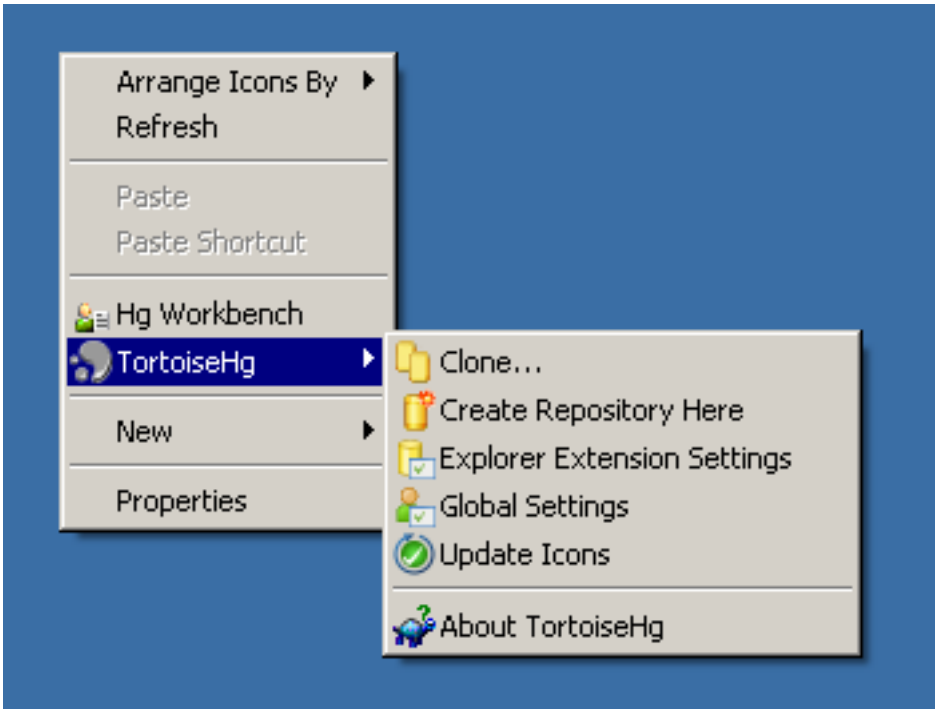

FIGURE 4 – Menu de TortoiseHG.

4 Installation et utilisation de Mercurial

4.1 Windows

Pour utiliser Mercurial sous Windows, le plus simple est de télécharger et d'installer TortoiseHg, que vous pouvez télécharger sur le site `http://tortoisehg.bitbucket.org`. Il s'agit d'un installateur classique, et il suffit de cliquer sur "Next" jusqu'à la fin — les options par défaut sont très bien. A la fin de l'installation, il faut redémarrer l'ordinateur.

Une fois l'ordinateur redémarré, il y a de nouvelles options dans le menu qui apparaît quand on fait un clic droit. Pour démarrer le projet, ouvrez d'abord le dossier dans lequel vous souhaitez mettre votre projet (dans les screenshots qui suivent, j'ai choisi de mettre le projet directement sur mon bureau). Ensuite, faites un clic droit dans la fenêtre de ce dossier, et choisissez `TortoiseHG` puis `Clone...` (voir Figure 4.

Une fenêtre similaire à la Figure 5 s'ouvre alors. Vous ne devez changer que le premier champ, pour y mettre l'adresse de votre projet, que vous aviez trouvé dans la page de résumé du projet à la Figure 3.

Une fois que Mercurial a terminé de télécharger le projet (cela devrait aller très vite), le dossier est marqué d'un point d'interrogation dans l'explorateur Windows, comme à la Figure 6.

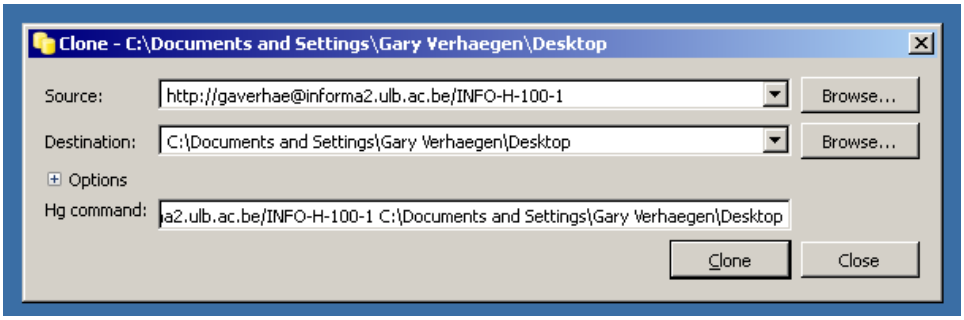

FIGURE 5 – Introduction de l'adresse du dépôt.

FIGURE 6 – Les fichiers et dossiers gérés par Mercurial indiquent leur statut dans l'explorateur Windows.

Le point d'interrogation signifie que Mercurial ne sait pas encore où le projet en est ; cela est dû au fait que le projet est pour le moment vide. En utilisant le vocabulaire de Mercurial, il n'y a pas encore de *révision* dans le projet, une révision étant le nom que Mercurial donne à une version enregistrée.

Si on ouvre maintenant ce dossier INFO-H-100-1 et qu'on y ajoute un fichiers contenant du code Python, par exemple avec Notepad.exe, on obtient la situation de la Figure 7.

Une fois que le projet a atteint un stade suffisant pour enregistrer une nouvelle version, il faut faire un clic droit dans la fenêtre du dossier du projet. On voit alors apparaître beaucoup plus d'options que précédemment, comme illustré à la Figure 8. La plupart de ces options supplémentaires sont dans le menu `TortoiseHG`, mais celle qui nous intéresse maintenant est l'option `Hg Commit...`. Dans le vocabulaire de Mercurial, *commit* est l'opération qui créée une nouvelle *révision*.

En cliquant sur `Hg Commit...`, vous obtenez une fenêtre (Figure 9 qui vous propose d'enregistrer une nouvelle version de votre projet. Pour ce faire, vous pouvez, dans la colonne de gauche, sélectionner les fichiers à ajouter à cette révision, et, dans le champ de texte en haut à droite, introduire un message. Ce message a pour but de décrire les changements entre la version précédente et la nouvelle version que vous êtes en train de créer. Enfin, dans

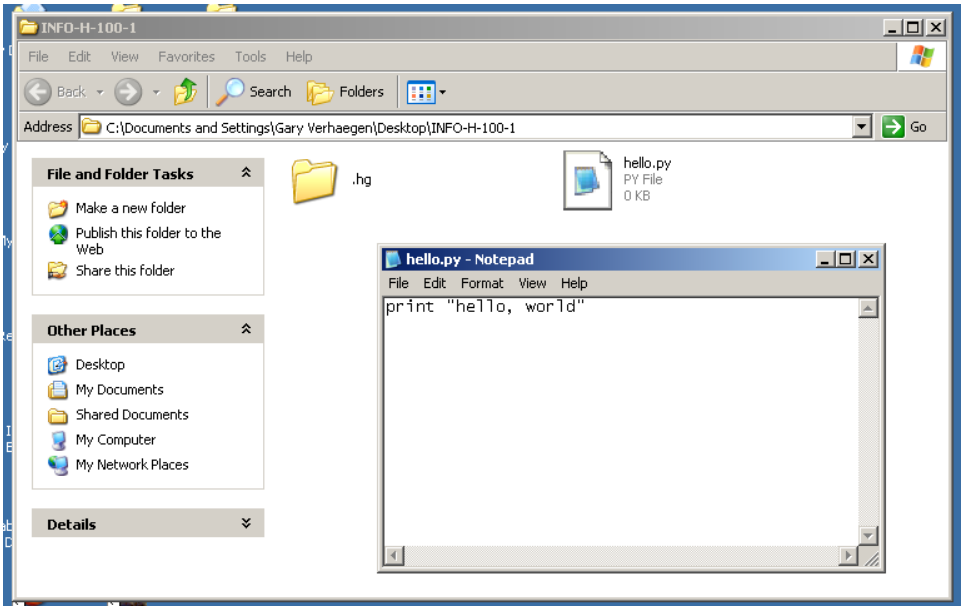

FIGURE 7 – Ajout d'un fichier.

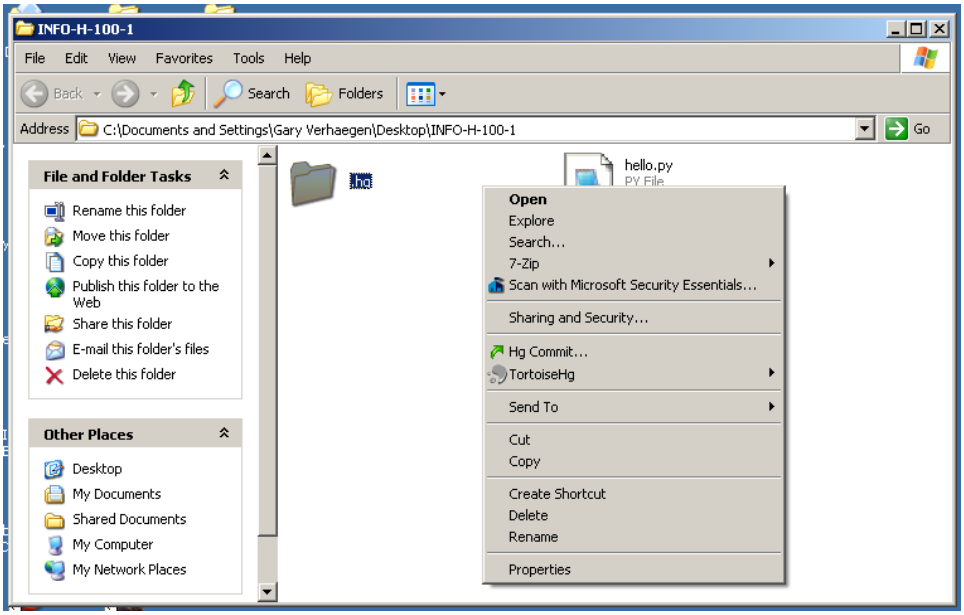

FIGURE 8 – Options de Mercurial dans un dépôt.

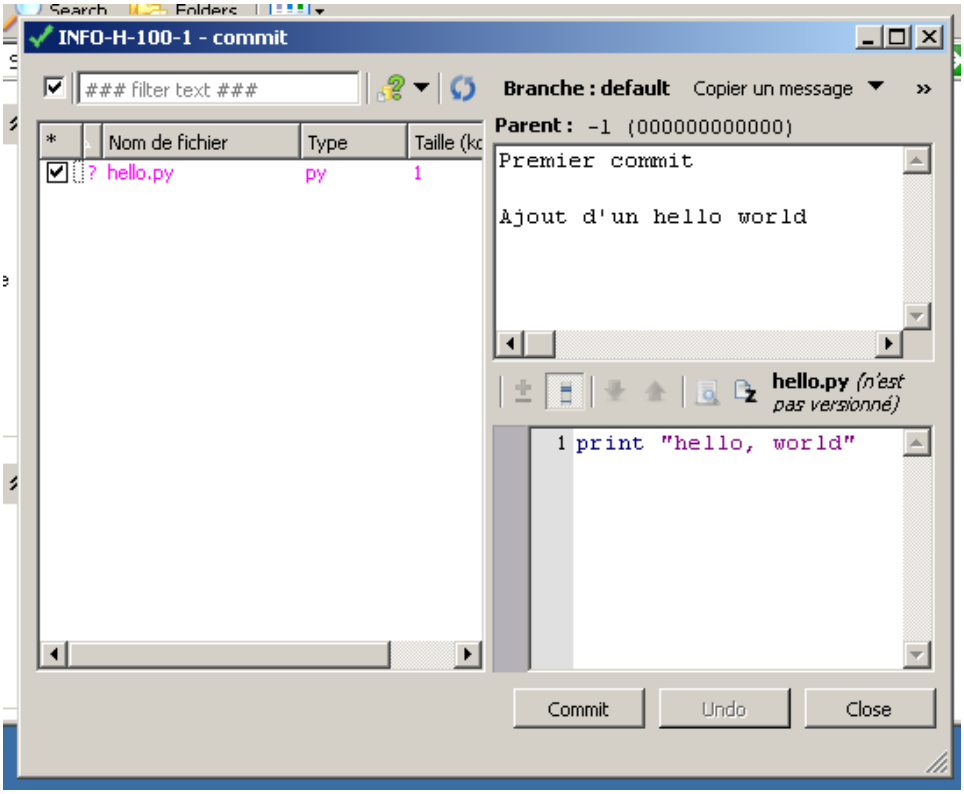

FIGURE 9 – Fenêtre de création de la première révision.

la case en base à droite, Tortoise Hg vous indique, pour le fichier sélectionné dans la colonne de gauche, les différences entre la nouvelle version que vous allez enregistrer et la dernière version précédente. (Dans le cas de la Figure 9, comme il n'y a pas de version précédente, l'affichage se limite au contenu complet du fichier.)

Quand vous cliquez sur `Commit`, TortoiseHg vous demande de confirmer l'ajout des fichiers (Figure 10). Cela n'arrive qu'une fois par fichier ; une fois qu'un fichier est géré par Mercurial, TortoiseHg ne demande plus de confirmer l'enregistrement de ses modifications.

Une fois le commit créé, TortoiseHg va indiquer que les fichiers sont suivis en ajoutant à leur icône un tag qui indique leur statut. Juste après le commit, quand le fichier n'a pas encore été modifié, il est marqué d'un V sur fond vert, comme à la Figure 11.

Cela signifie que le fichier est bien dans l'état de la dernière révision. Une fois le fichier modifié, par exemple pour ajouter de la ponctuation dans le code, Mercurial indiquera par un point d'exclamation rouge que le fichier n'est plus dans le même état que dans la dernière révision, comme vous pouvez le voir à la Figure 12.

FIGURE 10 – Confirmation d'ajout de nouveau fichiers.

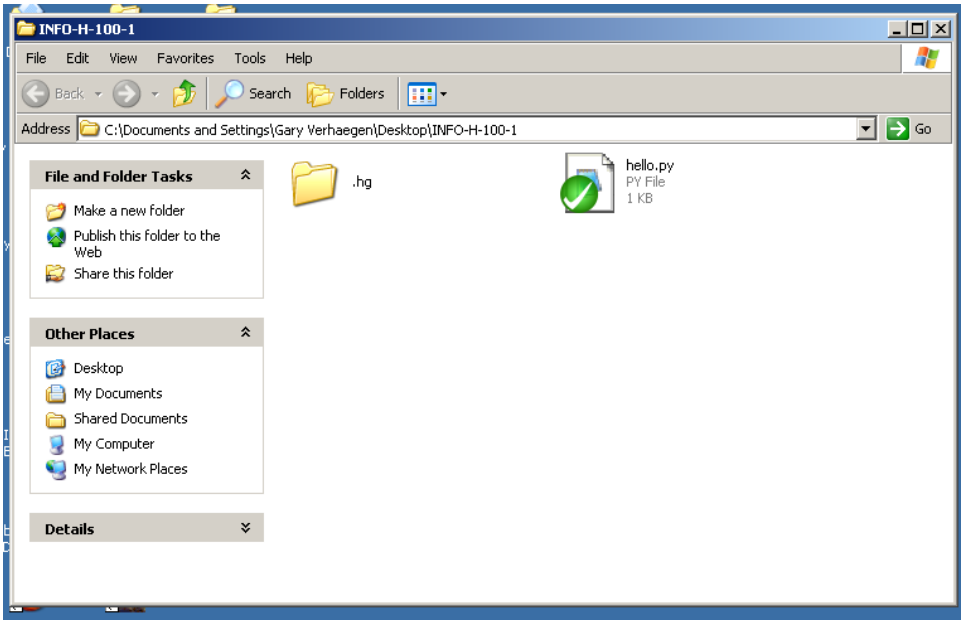

FIGURE 11 – Apparence d'un fichier à jour.

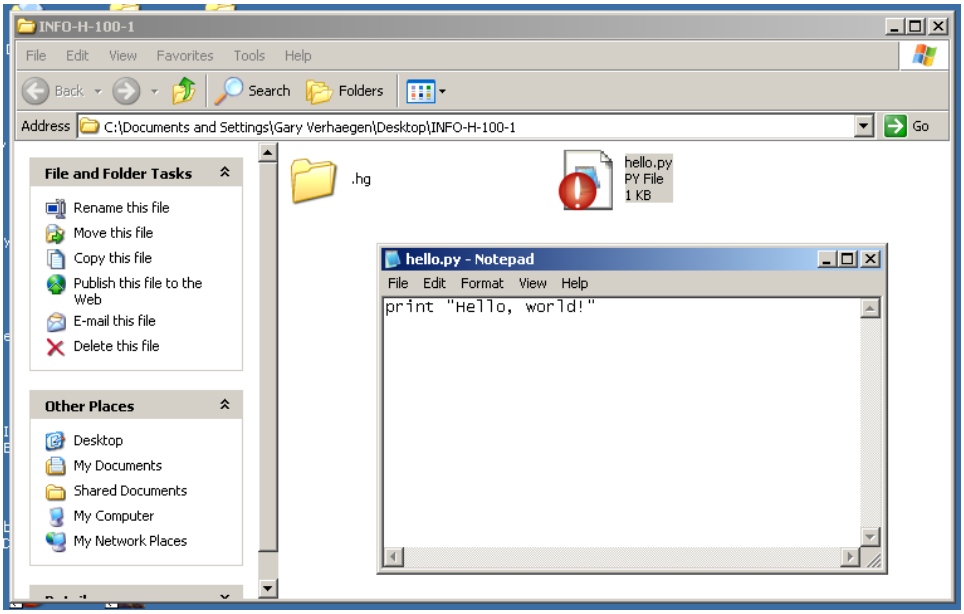

FIGURE 12 – Apparence d'un fichier modifié.

On peut à nouveau faire un commit pour créer une seconde révision qui inclu les modifications, en repassant par l'interface de commit (Figure 13), et on a de nouveau un dossier à jour (Figure 14).

Jusqu'à présent, nous n'avons qu'enregistré des versions locales. Pour sauvegarder ces versions sur le serveur, pour se prémunir par exemple d'une mort subite de disque dur, il faut aller dans le menu `TortoiseHg` dans le dépôt et choisir `Synchronise`, comme indiqué à la Figure 15.

L'écran qui apparaît alors (Figure 16) vous montre les différences entre le contenu du serveur et ce que vous avez sur votre ordinateur. Vous ne devez à ce stade pas vraiment vous en préoccuper. Dans cette fenêtre, vous devez simplement cliquer sur le bouton qui représente l'envoi vers le serveur, comme indiqué sur la Figure 16.

Après confirmation de votre part (Figure 17), Mercurial va envoyer les révisions sur le serveur. Si tout se passe bien, il devrait l'indiquer dans la fenêtre de `log` qui apparaît alors (Figure 18. Enfin, si vous retournez maintenant dans votre navigateur internet, vous pouvez explorer les différentes versions successives de votre projet sur le site `http://informa2.ulb.ac.be` (Figure 19).

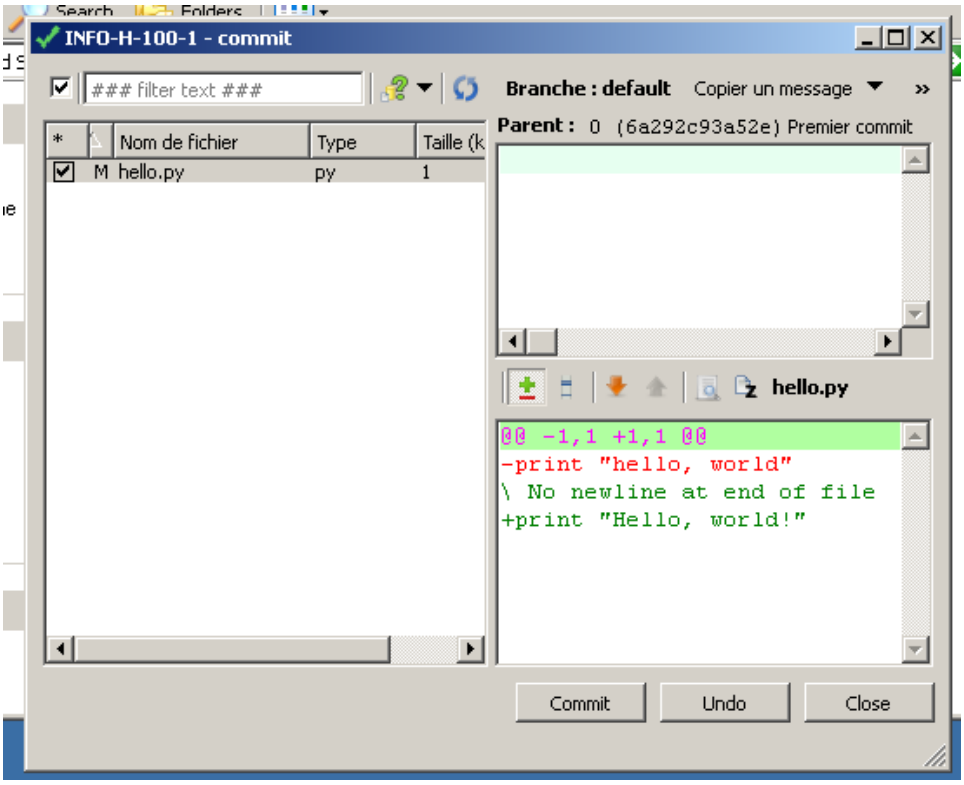

FIGURE 13 – Commit d'une modification.

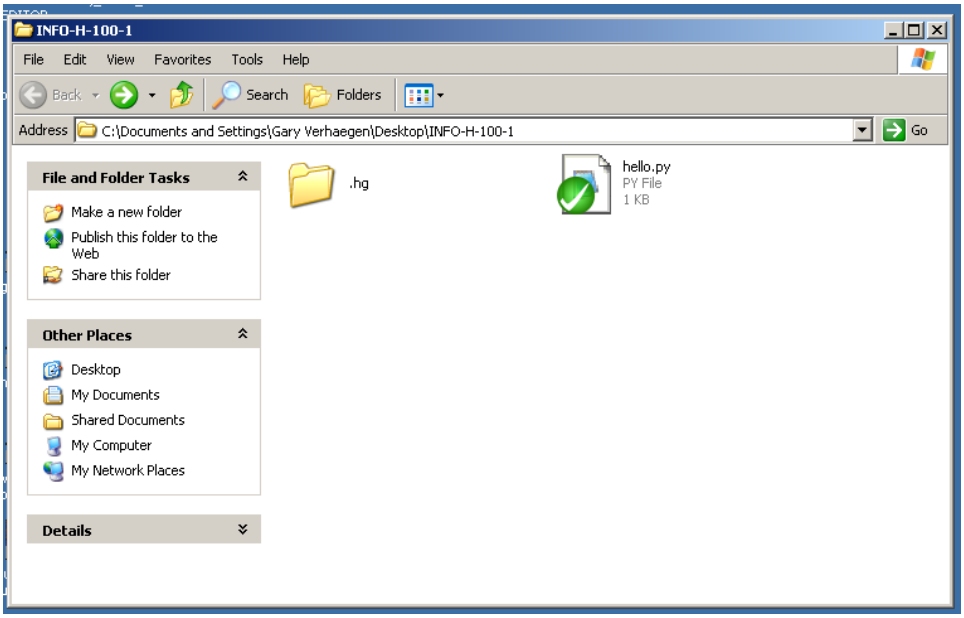

FIGURE 14 – Dossier à jour après le second commit.

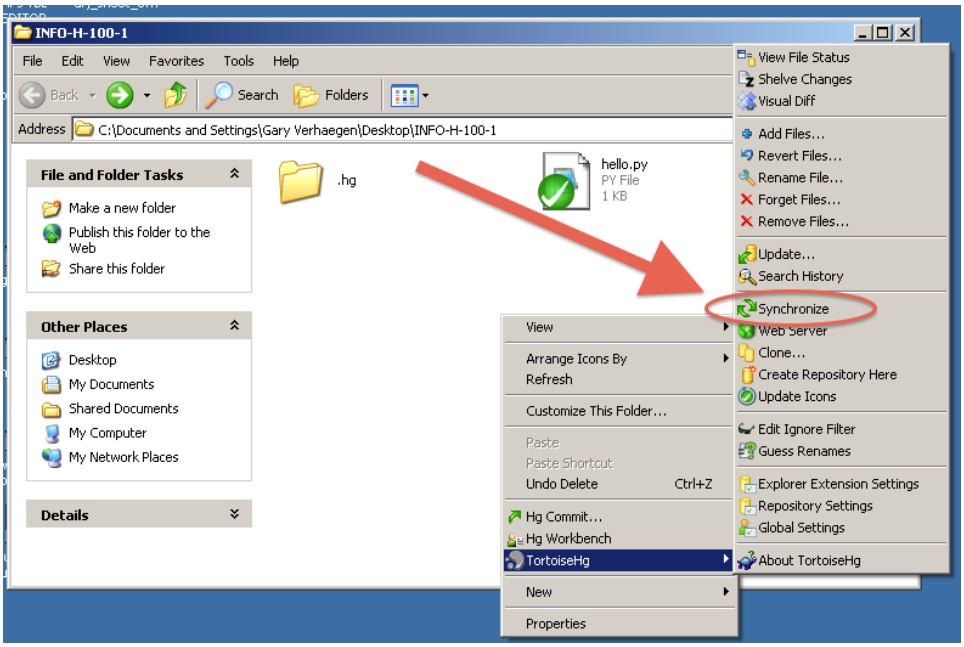

FIGURE 15 – Sauvegarde sur le serveur.

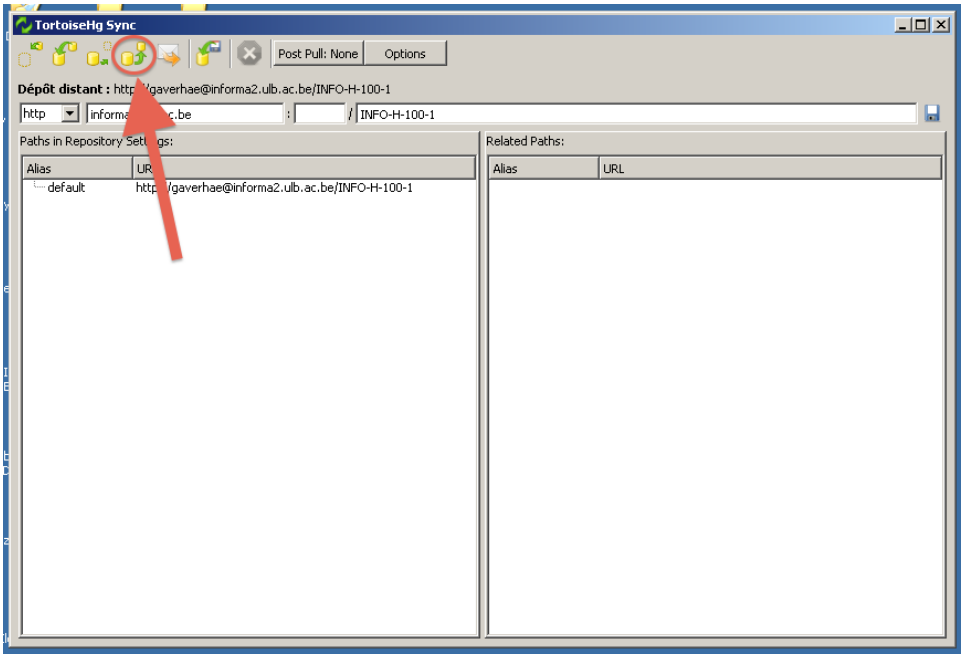

FIGURE 16 – Sauvegarde des révisions sur le serveur.

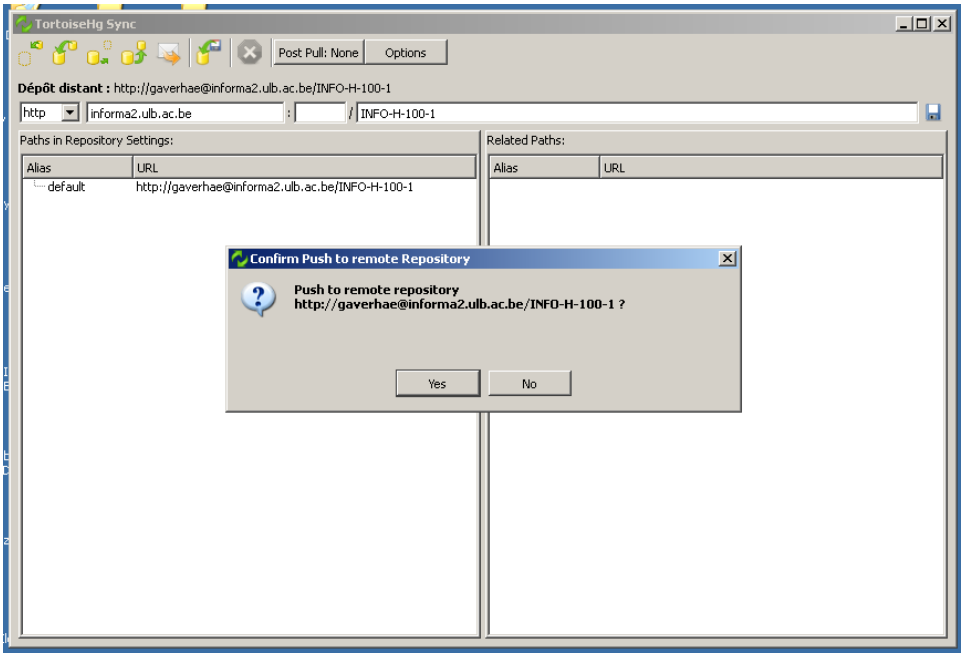

FIGURE 17 – Confirmation de la sauvegarde des révisions sur le serveur.

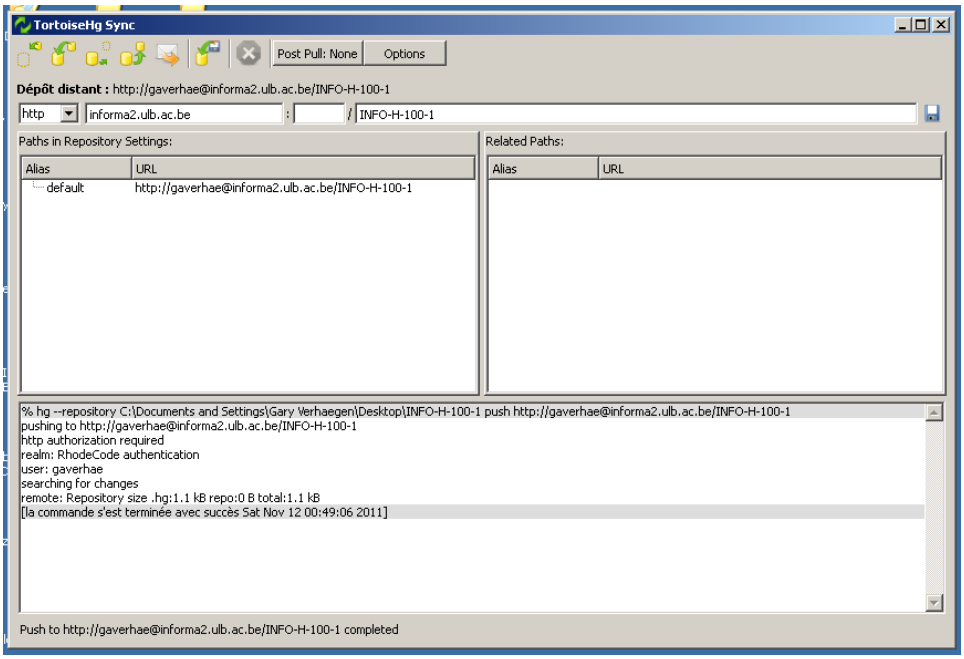

FIGURE 18 – Message indiquant que tout s'est bien passé.

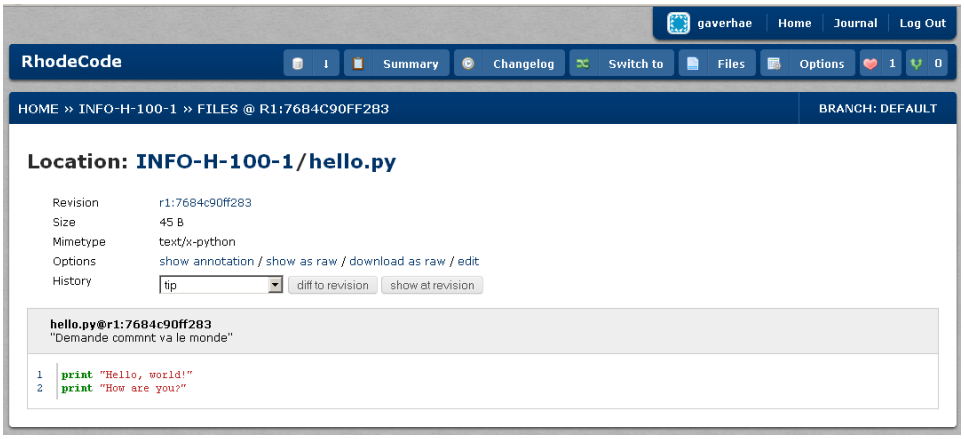

FIGURE 19 – Affichage d'un fichier dans RhodeCode.

4.2 Mac OS X

Pour utiliser Mercurial sur Mac OS X, si vous n'êtes pas familier avec la ligne de commande [2], nous vous recommandons d'utiliser MacHg, disponible à l'adresse `http://jasonfharris.com/machg/`. Une fois l'archive dézippée, MacHg s'installe comme la plupart des applications Mac, c'est-à-dire qu'il suffit de le déplacer vers le dossier Applications.

Quand MacHg s'ouvre pour la première fois, vous devez lui dire de cloner le répertoire que vous avez créé sur RhodeCode, comme indiqué sur les Figures 20, 21 et 22. Vous pouvez choisir le "short name" que MacHg vous demande ; ce nom ne sera utilisé que localement par MacHg pour désigner le répertoire du projet. Enfin, à la figure 22, vous pouvez choisir également le répertoire local dans lequel sera placé le projet. Comme vous pouvez le voir, j'ai décidé de mettre le répertoire sur mon bureau.

Une fois la copie local terminée, vous pouvez ouvrir le dossier et modifier les fichiers comme vous le souhaitez. Quand vous arrivez à un stade du projet qui vous semble mériter d'être enregistré, vous pouvez rouvrir MacHg (si vous l'aviez fermé). MacHg devrait se souvenir de votre dépôt et l'indiquer dans la colonne de gauche ; votre dépôt à vous est celui qui a une icône de dossier ; celui avec l'icône en forme de sphère représente le serveur (`http://informa2.ulb.ac.be`). Par défaut, MacHg vous indique sa légende ; comme vous pouvez le voir à la Figure 23, j'ai modifié le fichier `hello.py`. Pour voir le détail des modifications, je peux double cliquer sur le fichier, et MacHg va afficher les différences entre la nouvelle version et la dernière version enregistrée (Figure 24).

2. Si vous êtes familier avec la ligne de commande, l'installation de Mercurial peut se faire via MacPorts, qui peut se télécharger à l'adresse `http://www.macports.org/`. Une fois MacPorts installé, Mercurial s'installe en tapant `port install mercurial`. Pour l'utilisation de Mercurial en ligne de commande, voir plus loin.

FIGURE 20 – Première ouverture de MacHg.

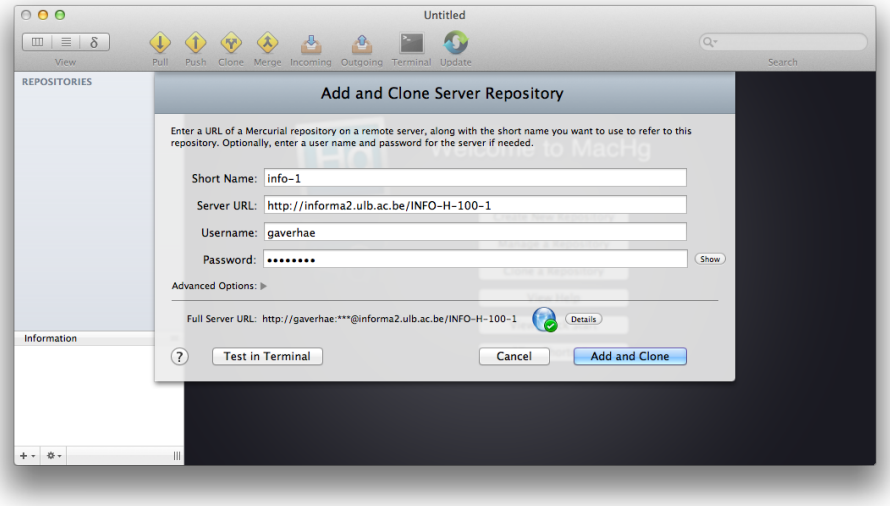

FIGURE 21 – Copie d'un dépôt — informations sur le serveur.

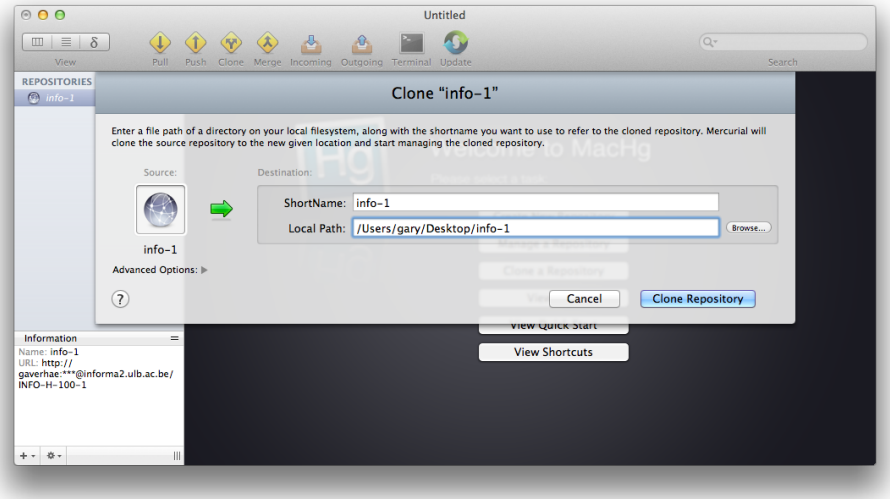

FIGURE 22 – Copie d'un dépôt — informations locales.

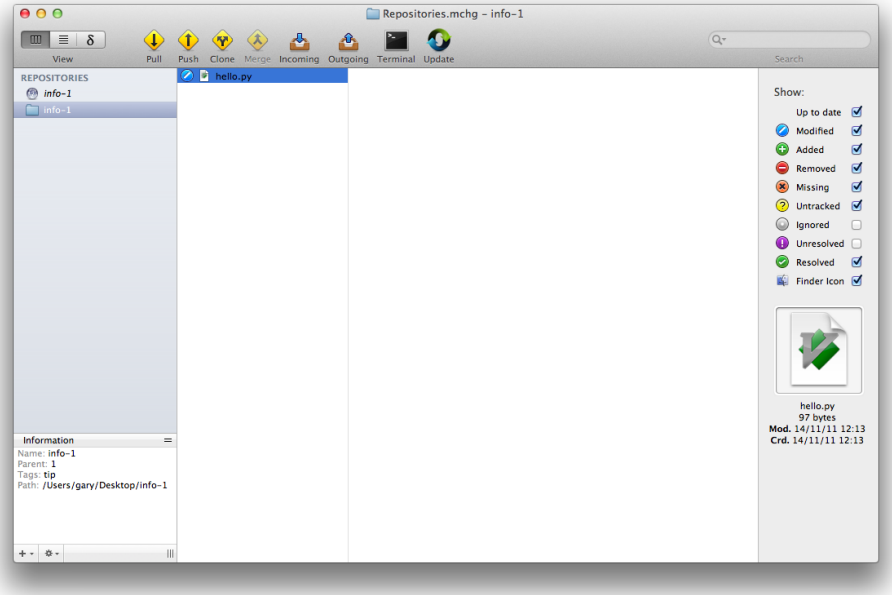

FIGURE 23 – Etat général d'un projet.

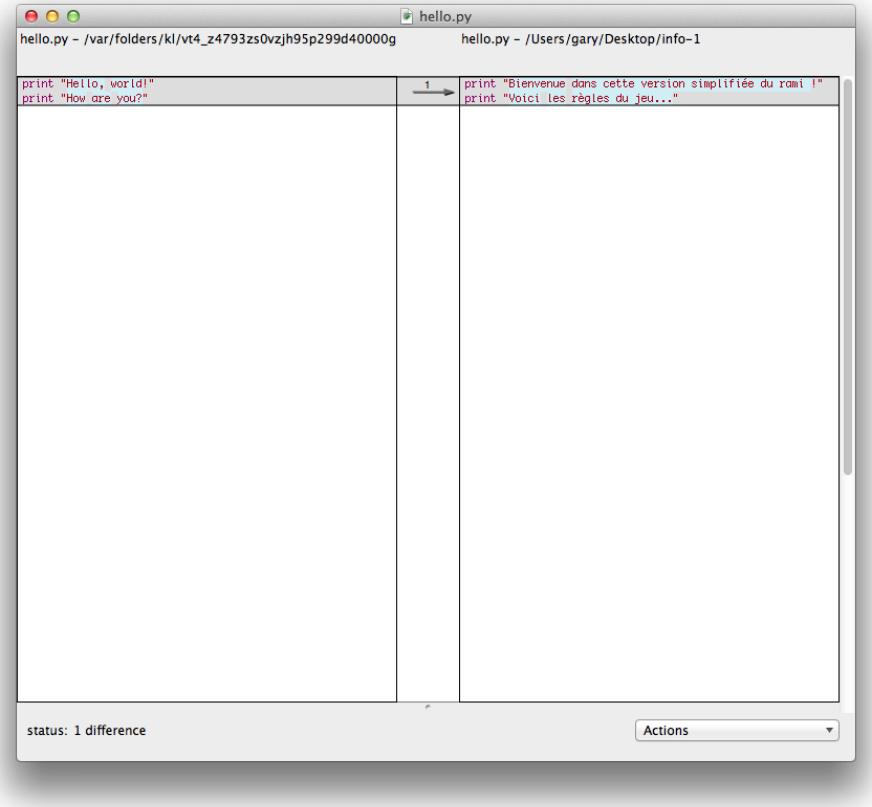

FIGURE 24 – Différences d'un fichier entre deux versions.

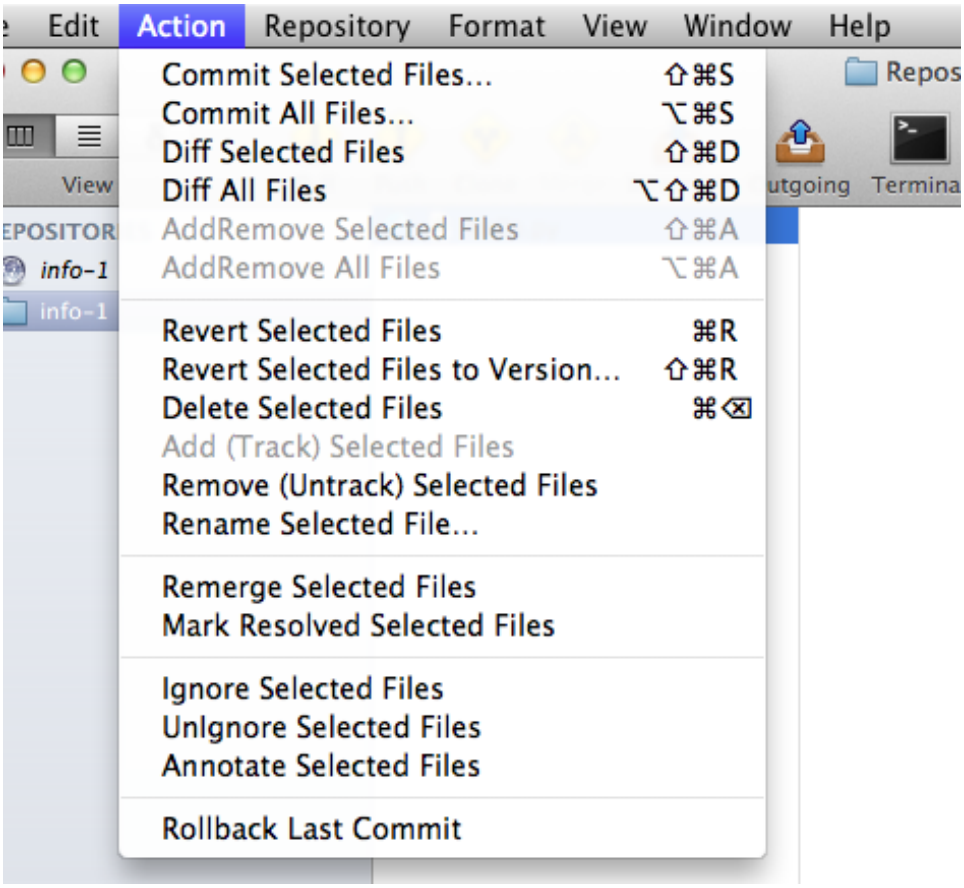

FIGURE 25 – Menu pour créer une nouvelle version.

Pour créer une nouvelle révision, il faut aller dans le menu `Actions` comme indiqué à la Figure 25. Vous pouvez alors choisir de créer une nouvelle révision avec tous vos changements (“Commit All Files”) ou seulement les changements apportés aux fichiers que vous avez sélectionnés (“Commit Selected Files”). L’écran qui apparaît alors vous permet de vérifier que la liste des fichiers est bien la bonne et d’introduire un message pour ce commit (Figure 26).

Enfin, pour sauvegarder l’état du projet sur le serveur, vous pouvez utiliser le bouton `Push` de l’écran d’état général de MacHg. Vous pouvez ensuite vérifier que tout s’est bien passé en allant voir l’adresse `informa2.ulb.ac.be` avec votre navigateur. Dans mon cas, en demandant le `changelog` de mon projet (c’est-à-dire la liste des versions), j’obtiens la Figure 27.

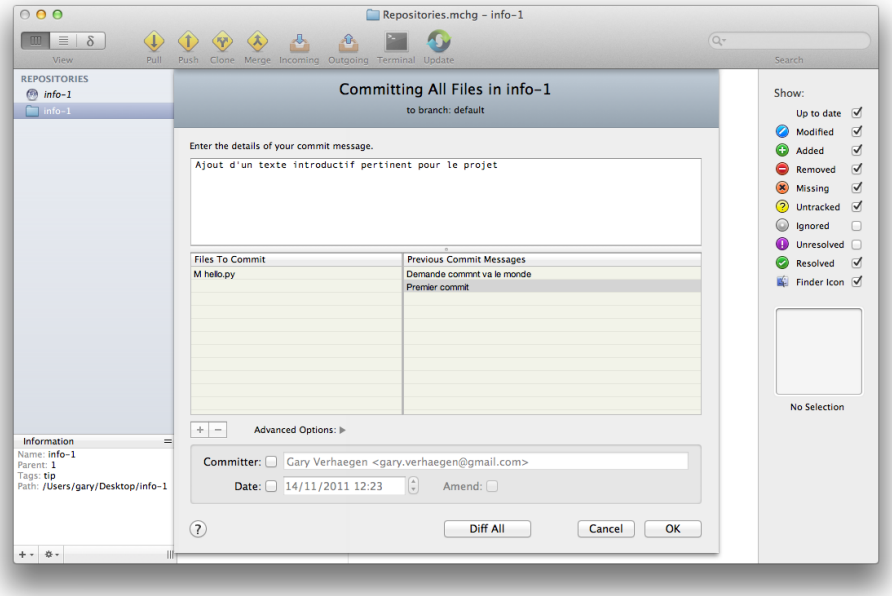

FIGURE 26 – Ecran de création d'une nouvelle version.

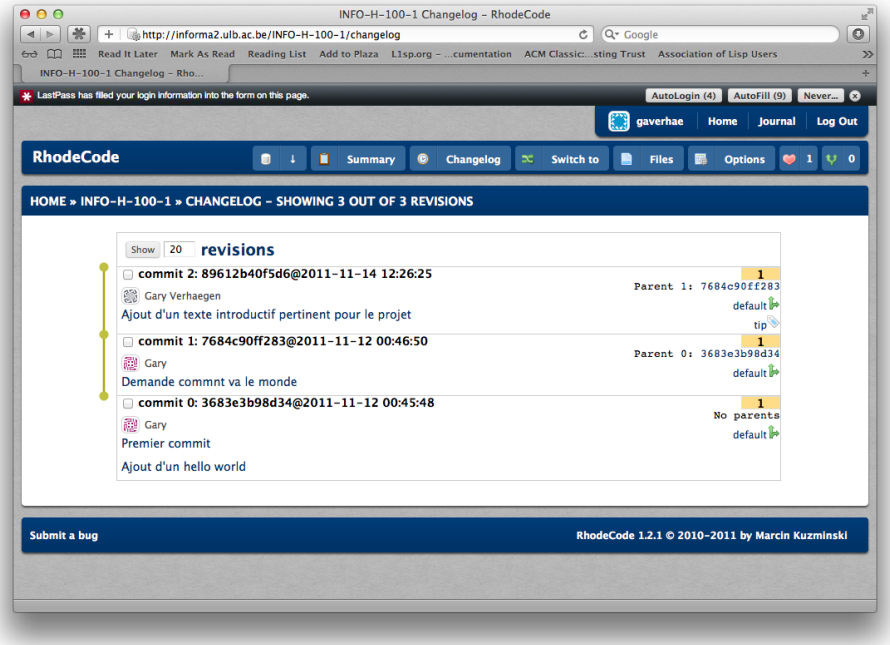

FIGURE 27 – Résultat du dépôt après ces manipulations.

4.3 Linux

Les plateformes Linux sont très différentes les unes des autres, ce qui rend peu pratique l'écriture d'un tutoriel général. Pour la grande majorité des distributions, il y a un paquet mercurial dans les paquets officiels. Par exemple, sous Debian et Ubuntu, mercurial s'installe en tapant `aptitude install mercurial`.

Consultez la documentation de votre distribution pour trouve comment installer mercurial. Une fois mercurial installé, consultez la section suivante pour son utilisation.

4.4 Ligne de commande

L'interface en ligne de commande est généralement plus utilisée sous Linux, mais elle est également disponible sous Windows et Mac OS X.

Une fois mercurial installé et le dépôt créé sur le serveur, vous pouvez créer une copie locale en tapant `hg clone <adresse> [<nom local>]`, par exemple dans mon cas :

```
hg clone http://gaverhae@informa2.ulb.ac.be/INFO-H-100-1 projet1
```

va copier le contenu de mon proejt sur le serveur RhodeCode dans un dossier "projet1". Une fois que vous êtes dans un dossier géré par mercurial, vous pouvez utiliser la commande `hg status` pour voir l'état des différents fichiers (égal à la dernière révision, modifié depuis la dernière révision, nouveau fichier pas encore géré par mercurial). Pour voir les modifications pas encore enregistrées, la commande est `hg diff`. Pour ajouter un nouveau fichier à la prochaine révision, la commande est `hg add <nom du fichier>`. Pour faire un nouveau commit, la commande est `hg commit -m "<message de commit>"`. Pour envoyer l'historique sur le serveur, la commande est simplement `hg push`. Exemple complet (le texte tapé par l'utilisateur est uniquement le texte qui suit un $) :

```
gary:/tmp
$ hg clone http://gaverhae@informa2.ulb.ac.be/INFO-H-100-1 projet1
http authorization required
realm: RhodeCode authentication
user: gaverhae
password:
requesting all changes
adding changesets
adding manifests
adding file changes
added 3 changesets with 3 changes to 1 files
updating to branch default
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

```
gary:/tmp
$ cd projet1
gary:/tmp/projet1
$ ls
hello.py
gary:/tmp/projet1
$ hg status
gary:/tmp/projet1
$ vim hello.py
gary:/tmp/projet1
$ hg status
M hello.py
gary:/tmp/projet1
$ hg diff
diff -r 89612b40f5d6 hello.py
--- a/hello.py  Mon Nov 14 12:26:25 2011 +0100
+++ b/hello.py  Mon Nov 14 12:48:22 2011 +0100
@@ -1,2 +1,3 @@
 print "Bienvenue dans cette version simplifiée du rami !"
 print "Voici les règles du jeu..."
+print "Vous allez recevoir des cartes ..."
gary:/tmp/projet1
$ hg commit -m "Ajout d'une ligne de texte à l'introduction"
gary:/tmp/projet1
$ hg status
gary:/tmp/projet1
$ hg push
pushing to http://gaverhae@informa2.ulb.ac.be/INFO-H-100-1
http authorization required
realm: RhodeCode authentication
user: gaverhae
password:
searching for changes
remote: Repository size .hg:2.0 kB repo:0 B total:2.0 kB
gary:/tmp/projet1
$
```

Et finalement, on peut aller vérifier que cette dernière version a bien été envoyée sur le serveur en s'y rendant avec un navigateur internet, comme vous pouvez le voir sur la Figure 28.

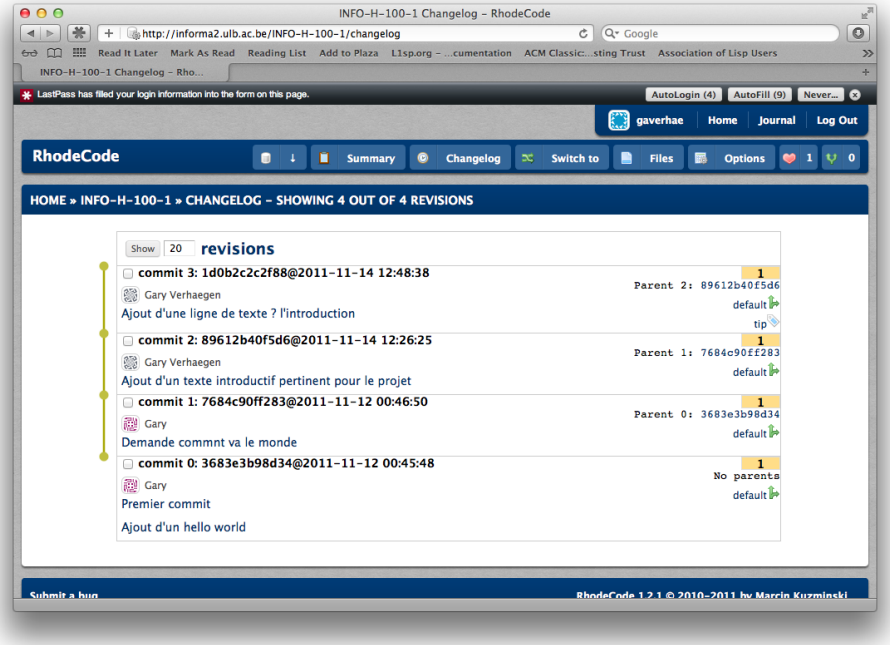

FIGURE 28 – Etat final du dépôt sur le serveur.