Mercurial

-

Travaux Pratiques

zenika
ARCHITECTURE INFORMATIQUE

0. Mise en route

Objectifs

Ce premier TP a pour simple objectif de mettre en place les divers éléments necessaires à l'utilisation de Mercurial sur les postes clients.

Mise en place des éléments

Il est entendu que les TPs se dérouleront sur l'environnement suivant :

- Windows XP ou supérieur
- Droit d'administration sur les postes
- Mercurial v1.8.0 ou supérieur

Client shell et TortoiseHG

- Téléchargez le client à cette adresse : http://tortoisehg.bitbucket.org/
- Version 2.0.x
- Suivez les instructions pour l'installation
- Installez la documentation ainsi que l'outil Kdiff3 (les ssh utils ne sont pas obligatoires pour cette formation)
- TortoiseHG fournit également les commandes Shell pour l'utilisation de Mercurial directement en ligne de commande (le Path utilisateur a été également modifié pour vous)

Vérifiez que l'installation s'est correctement déroulée en tapant cette commande dans un shell :

```
$ hg --version
```

Vous devriez également voir l'extension de l'explorateur Windows en invoquant le menu contextuel sur n'importe quel dossier.

Client MercurialEclipse

- Téléchargez Eclipse, la distribution Helios pour Java Developpers est suffisante : http://eclipse.org/downloads/packages/eclipse-ide-java-developers/heliossr2
- Installez Eclipse (n'oubliez pas de setter la JVM si nécessaire)
- Installation du plugin via l'update site à cette adresse http://cbes.javaforge.com/update
- Menu Help → Install New Software
- Cliquez sur Add pour ajouter l'update site fournit
- Sélectionnez MercurialEclipse pour l'installation (les autres éléments ne sont pas requis)
- Assurez vous que l'exécutable Mercurial a bien été trouvé par le plugin
  - Window → Preferences → Team → Mercurial

Configurer le client Mercurial

Pour finir l'installation de Mercurial, nous devons impérativement spécifier un identifiant pour l'utilisateur.

Editez directement le fichier Mercurial.ini qui se trouve dans la Home de l'utilisateur, ou utilisez les clients.

Examinez les autres configuration possibles.

Documentation sur la configuration : http://www.selenic.com/mercurial/hgrc.5.html

1. TP : Utilisation quotidienne (locale)

Objectifs

Création d'un repository Mercurial puis manipulation des sources en local.
Dans un premier temps, toutes les opérations se feront en ligne de commande puis le stagiaire sera ammené à répéter le TP en utilisant le client riche de son choix.

Création d'un Repository Mercurial

- Créez un répertoire de travail C:\work
- Créez un repository Mercurial dans ce dossier

```
$ hg init .
```

- Examinez le résultat de cette opération

A tout moment vous pouvez consulter l'aide de Mercurial

```
$ hg help
```

```
$ hg help <commande>
```

Ajout des sources

- Créez un fichier a.txt dans le dossier

Nous souhaitons versionner les sources de notre projet dans le repository nouvellement créé.

- Examinez l'état du fichier nouvellement créé

```
$ hg status
```

- Que remarquez-vous ?
- Ajouter ce fichier au système de gestion de source

```
$ hg add .
```

- Vérifiez le statut des fichiers ajoutés
- Qu'observez-vous ?

```
$ hg status
```

- Commitez votre première révision


```
$ hg commit -m « Premier import des sources »
```

N'oubliez pas d'écrire des messages de commits simples et clairs, ils sont essentiels à la bonne compréhension de l'objectif de la révision.
Mettez-vous à la place d'un collaborateur qui tente de comprendre le but des modifications que vous avec apportez aux sources.

- Examinez le statut de vos sources
- Puis l'historique

```
$ hg log
```

- Identifiez les différentes métadonnées que possède une révision (hash et numéro de révision, auteur, commentaire, ...)

Manipulation des sources

- Ajoutez du texte dans le fichier a.txt (n'oubliez pas le retour à la ligne à la fin du fichier)
- Vérifiez le statut du fichier, que remarquez-vous ?
- Commitez votre modification
- Examinez l'historique des révisions

Copie de source

- Copiez le fichier a.txt en a-copy.txt

```
$ hg help copy
```

- Examinez le statut de la copie de travail
- Commitez ce changement
- Observez l'historique du fichier a.txt puis du fichier a-copy.txt

```
$ hg log FILE
```

Mercurial est capable de tracer les modifications d'une ressource copiée

- Trouvez l'option permettant d'obtenir l'historique entier de a-copy.txt
- Testez cette option sur a.txt

Reprenez les étapes de cet exercice mais cette fois en copiant a.txt depuis l'explorateur ou à l'aide de la commande `cp`.

Que remarquez-vous ?

Suppression des sources

- Supprimez le fichier a-copy.txt

```
$ hg remove [FILE]...
```

- Examinez le statut de la copie de travail ainsi que les fichiers présents dans le dossier work
- Commitez

Il est possible de supprimer une source du repo Mercurial tout en la gardant intacte dans le dossier.

```
$ hg forget [FILE]...
```

Distinguez les deux actions `remove` et `forget`.

Supprimez une source via l'explorateur (sans passer par Mercurial) et examinez le statut de la copie de travail (commande `status`).

Il est toujours possible de supprimer la ressource via la commande `remove`.

En cas d'erreur

Revert : supprimer les changements en cours sur la copie de travail

- Modifiez le fichier a.txt sans commiter
- Ajoutez une source b.txt
- Examinez le statut de la copie de travail

- Supprimez les modifications

```
$ hg help revert
```

Rollback : Le rollback permet d'annuler la dernière transaction

Ici nous souhaitons annuler l'effet du dernier commit

- Modifiez le fichier a.txt
- Commitez puis examinez l'historique
- Rollbackez ce commit

```
$ hg rollback
```

- Examinez l'historique du repo à nouveau

Backout : supprimer l'effet d'un changeset particulier en « ajoutant » au repo

- Modifiez le fichier a.txt
- Commitez
- Vérifiez l'historique
- Supprimez les effets de cette révision à l'aide de la commande

```
$ hg backout -m REV
```

- Analysez le résultat

[+] Testez le backout sur une autre révision que la dernière.

Mise à jour de la copie de travail

A tout moment il est possible d'extraire la copie de travail à partir d'une révision particulière pour examiner l'état des sources.

Changez l'état des sources en mettant à jour la copie de travail

```
$ hg update REV
```

Examinez la copie de travail

.hgignore

Documentation : http://www.selenic.com/mercurial/hgignore.5.html

- Créez un fichier .hgignore à la racine de la copie de travail
- Commitez
- Ajoutez un fichier nommée par exemple a.compile
- Examinez le statut du repo
- Ajoutez une ligne au fichier .hgignore de façon à ce que Mercurial ne se préoccupe pas de tous les fichiers d'extension .compile (syntaxe shell)
- Observez à nouveau le statut du repo
- Commitez

[+] Testez avec des fichiers marqués 'A'

Autres clients

Familiarisez-vous aux clients riches de votre choix en reprenant les éléments du TP1.

2. TP : Communiquer avec des repository distants

Objectifs

Comprendre les mecanismes de pull et push permettant de partager avec d'autres repositories les révisions qui concernent un même projet.
En ligne de commande tout d'abord, puis avec les vue dédiées dans les clients riches.

Cloner un repository

Nous avons besoin d'un nouveau repository avec lequel nous allons communiquer.

- Clonez le repository du TP 1 dans C:\work-clone

```
$ hg clone SOURCE DEST
```

- Examinez le repository nouvellement créé (structure, historique, ...)

Le Pull

Nous allons rappatrier les révisions à partir du repository work.

- Se positionner dans le repo work
- Faites une succession de modifications et commits sur le fichier a.txt
- Vérifiez l'historique

- Reprendre le repo work-clone
- Vérifiez le différentiel des révisions

```
$ hg incoming URL
```

- Répétez la commande incoming en omettant l'URL
- Pourquoi cela fonctionne-t'il ?
- Rappatriez les révisions du repo work

```
$ hg pull
```

- Examinez la copie de travail, les sources sont-elles à jour ?
- Mettre à jour les sources afin de voir les modifications apportées par les révisions rappatriées

Le push

Cette fois modifiez le repo work-clone et pousser vos modifications vers le repo work.


```
$ hg outgoing
```

```
$ hg push
```

- Vérifiez l'historique du repo work

Autres clients

En utilisant les vues synchronize des clients TortoiseHG et Eclipse, réalisez à nouveau le TP 2.

3. TP : Les branches

Objectifs

Identifier les différents types de branches techniques Mercurial et savoir les mettre en oeuvre. Comprendre le mécanisme de merge entre deux branches.

Les branches par clonage

Pour ce TP, nous allons reprendre un repository propre.

- Créez un nouveau repository dans C:\work
- Ajoutez-y une source a.txt
- Réalisez quelques modifications et commits afin de créer un historique
- Clonez ce nouveau repo dans C:\work-clone
  - vous venez de créer une branche !
- Commitez des modifications sur les deux repositories en prenant garde à ne pas créer de conflits.

  Exemple : créer une source b.txt dans le repo work et une source c.txt dans le repo work-clone

L'historique des deux repos est désormais divergent.
Notez les numéros de révisions des changesets divergents

- Réintégrez la branche du repo work dans celle du repo work-clone à l'aide d'un pull, et remarquez la dernière ligne d'output de la commande.
- Pourquoi ne pas faire un push plutôt qu'un pull ?

```
$ hg help push
```

- Combien de HEADs le repo work-clone possède-t'il ?

```
$ hg heads
```

- Mergez ces deux HEADs

```
$ hg merge
```

Les modifications portant sur deux fichiers différents, ce merge ne devrait créer aucun conflit

- Combien de parents possède la révision de merge ?

```
$ hg parents
```

Nous allons maintenant nous mettre dans une situation de résolution de conflit.

- Dans le repo work, ajoutez une ligne au fichier a.txt puis commitez.
- Faites de même dans le repo work-clone
- Réintégrez la branche work dans work-clone puis mergez, que se passe-t'il ?
- Résolvez le conflit

Les branches nommées

Dans le repo work :

- Listez les branche nommées

```
$ hg branches
```

- Qu'observez-vous ?
- Dans le repo work, créez une branche nommée stable

```
$ hg branch stable
```

- Vérifiez que la copie de travail se situe dans la branche stable et lister les branches

```
$ hg branch
$ hg branches
```

- Que remarquez-vous ?
- Commitez cette nouvelle branche et listez les branches

- Commitez des modifications dans la branche stable
- Observez l'historique de vos révisions, que remarque-t-on sur les métadonnées des changesets ?

Réintégrez la branche stable dans la branche default :

- Mettre à jour la copie de travail sur la dernière révision de default

```
$ hg update default
```

- Mergez avec la dernière révision de stable

```
$ hg merge stable
```

Reprenez le clone work-clone et intégrez les nouvelles révisions avec un pull. Observez que la branche stable a bien été propagée.

Bookmarks

- Dans le repo work, mettez à jour la copie de travail à la dernière révision de la branche default.
- Créez un bookmark

```
$ hg bookmark <nom>
```

- Affichez la log de cette dernière révision, que remarquez-vous ?
- Modifiez vos sources puis commitez
- Qu'observe-t'on à propos du bookmark ?

- Réintégrez ces révisions dans work-clone à l'aide d'un pull
- Qu'observe-t'on à propos du bookmark ?

Il est possible de propager les bookmarks entre repositories, comment ?

```
$ hg help pull
```

[+] Supprimez le bookmark dans le repo work

Branches Anonymes

Dans le repo work :

- Ajoutez du contenu à la source b.txt puis commitez
- Mettez à jour votre copie de travail afin qu'elle corresponde à l'état des sources de l'avant dernière révision
- Modifiez à nouveau la source b.txt puis commitez

Vous venez de créer une branche annonyme, observez les deux HEADs dans le repo.

- Continuez vos modifications sur ces deux branches puis mergez.

Qu'observe-t'on sur les métadonnées des révisions concernant ces deux branches ?

Réintégrer les révisions dans work-clone à l'aide d'un pull.

Tags

- Taguez la dernière révision du repo work avec le label v1.0.0

```
$ hg tag v1.0.0
```

- Observez l'historique, que remarquez-vous ?
- Continuez de commiter dans le repo work.
- Qu'observe-t'on à propos du tag ?

- Ouvrez le fichier .hgtags et analysez sa structure
- Mettez à jour votre copie de travail afin qu'elle corresponde à la révision taguée
- Réintégrez ce tag dans le repo work-clone

Autres clients

Créez chaque type de branches techniques à l'aide des clients TortoiseHG et Eclipse.

