CVS : Gestion de configuration client-serveur

Linux Gazette numéro 66

Kapil Sharma

kapil@linux4biz.net

Jérôme Fenal

jerome@fenal.org

1. Introduction

CVS est un système de gestion de configuration. En l'utilisant, vous pourrez enregistrer l'histoire de vos fichiers source. CVS aide si vous faites parte d'un groupe de personnes travaillant sur le même projet, partageant le même code. Plusieurs développeurs peuvent travailler sur le même projet à distance en utilisant le modèle client serveur de CVS, modèle dans lequel le code existe sur un serveur central, et où chaque programmeur récupère le source sur sa machine locale depuis le serveur CVS (checkout) et le re-sauvegarde sur le serveur CVS (checkin) après développement. Chaque fois qu'un programmeur insère son nouveau code sur le serveur CVS, seule la différence [avec la version précédente] est sauvegardée au lieu d'écraser la précédente

version. Cela autorise le serveur à recréer quelque version précédente que ce soit à la demande, bien que distribuant, par défaut, la dernière version.

2. Obtenir CVS

Vous trouverez CVS dans votre distribution ou pouvez obtenir les sources à http://www.cvshome.org/downloads.html.

La page d'accueil de CVS est http://www.cvshome.org/.

3. L'entrepôt de code

L'entrepôt de code CVS stocke une copie complète de tous les fichiers et répertoires qui sont soumis à la gestion de version. En temps normal, vous n'avez pas à accéder directement à ces fichiers de l'entrepôt. En lieu et place, vous utilisez les commandes CVS pour obtenir votre propre copie des fichiers dans un répertoire de travail, puis travaillez sur cette copie. Quand vous avez fini vos modifications, vous les validez dans l'entrepôt. L'entrepôt contient ainsi les modifications que vous avez faites, de même qu'il a enregistré ce que vous avez exactement modifié, quand vous l'avez modifié, et autres informations du genre.

3.1. Créer un dépôt

Pour créer un entrepôt, lancez la commande CVS **init**. Cela créera un dépôt vide dans la racine CVS spécifiée de la manière habituelle.

```
cvs -d /usr/local/cvsroot init
```

Ici, le répertoire `/usr/local/cvsroot` deviendra le dépôt.

3.2. Variable d'environnement CVSROOT

Positionnez la variable d'environnement CVSROOT dans les scripts de démarrage de votre interpréteur de commande. Par exemple, dans `~/.bashrc` :

```
$ export CVSROOT=:pserver:username@foo.com:/usr/local/cvsroot
```

3.3. Sauvegarde du dépôt

Il y a quelques points à considérer quand vous sauvegardez le dépôt :

- Personne ne doit utiliser le dépôt pendant la sauvegarde, ou le logiciel de sauvegarde doit verrouiller CVS pendant la sauvegarde ;
- Pour verrouiller CVS, vous devez créer un fichier « `#cvs.rfl` » dans chaque répertoire de dépôt.

3.4. Dépôt distant

La copie de travail de vos fichiers sources peut se situer sur une autre machine que celle hébergeant le dépôt. Cette utilisation de CVS est connue sous le terme de mode client/serveur.

Configuration du serveur : ajoutez l'entrée suivante dans le fichier `/etc/inetd.conf` du serveur :

```
2401 stream tcp nowait root /usr/local/bin/cvs cvs -f -allow-root=/usr/cvsro
```

Si votre super-daemon inetd veut un nom symbolique de service au lieu d'un simple numéro de port, alors ajoutez ceci dans le fichier `/etc/services`

```
cvspserver      2401/tcp
```

et mettez « cvspserver » au lieu de « 2401 » dans `inetd.conf`.

Après avoir effectué vos changements, envoyez un signal HUP à **inetd**.

3.5. Authentification par mot de passe pour les dépôts distants

Pour configurer l'authentification par mot de passe, créez un fichier `$CVSROOT/CVSROOT/passwd`. Le contenu du fichier ressemblera à :

```
anonymous:
kapil:1sOp854gDF3DY
melissa:tGX1fS8sun6rY:pubcvs
```

Les mots de passe sont chiffrés à la mode Unix. La première ligne de l'exemple autorisera l'accès à n'importe client CVS cherchant à s'authentifier avec l'utilisateur anonyme anonymous, quelque soit le mot de passe qu'il emploie. La deuxième ligne autorisera l'accès à kapil s'il fournit son mot de passe correspondant à la chaîne chiffrée.

La troisième ligne autorisera l'accès à melissa si elle donne son mot de passe, mais ses opérations CVS se feront en tant que l'utilisateur pubcvs.

Note : CVS peut être configuré pour ne pas utiliser le vrai fichier de mot de passe d'Unix, à savoir `/etc/passwd`, pour l'authentification CVS en positionnant le paramètre `SystemAuth` à « no » dans le fichier de configuration de CVS `$CVSROOT/CVSROOT/config`.

3.6. Utilisation du client avec l’authentification par mot de passe

Vous devez vous connecter au serveur CVS pour la première fois :

```
cvs -d :pserver:kapil@foo.com:/usr/local/cvsroot login
```

Vous pouvez ensuite utiliser toutes les commandes de CVS sur la machine distante :

```
cvs -d :pserver:kapil@foo.com:/usr/local/cvsroot checkout someproj
```

3.7. Dépôts en accès lecture seulement

Il est possible de n’autoriser l’accès qu’en lecture seule à certaines personnes en utilisant le serveur authentifié par mot de passe. Il y a deux façons de spécifier l’accès en lecture seule pour un utilisateur : par inclusion ou par exclusion.

Par inclusion signifie spécifier un utilisateur dans le fichier `$CVSROOT/CVSROOT/readers`, qui est une simple liste d’utilisateurs sur des lignes séparées. En voici un exemple :

```
kapil
yogesh
john
```

N’oubliez pas le dernier retour à la ligne après le dernier utilisateur.

Par exclusion, on signifie que tous les utilisateurs doivent avoir un accès en lecture-écriture, et que tous les autres auront un accès en lecture seule. Le fichier `writers` a le même format que le fichier `readers`.

3.8. Mise en place des fichiers dans le dépôt

Si les fichiers que vous voulez installer dans CVS résident dans un répertoire `mon_projet`, et que vous voulez les voir apparaître dans le dépôt en tant que `$CVSROOT/mon_projet`, vous pouvez faire ceci :

```
$ cd mon_projet
$ cvs import -m "sources importés" mon_projet vendor rel1-1
```

Ici, la chaîne « vendor » est une balise pour l'éditeur, la chaîne « rel1-1 » est une balise concernant la version du produit.

3.9. Verrous CVS dans le dépôt

Tout fichier dans le dépôt qui possède un nom commançant par « #cvs.rfl. » est un verrou en lecture. Tout fichier dont le nom commence par « #cvs.wfl. » est un verrou en écriture. Le répertoire `#cvs.lock` sert de verrou général. Cela signifie qu'une personne doit d'abord obtenir ce verrou avant de pouvoir créer les autres verrous.

Pour obtenir un verrou en lecture, créez d'abord le répertoire `#cvs.lock`. Si la création échoue parce que le répertoire existe déjà, attendez un peu et ré-eesayez. Après avoir obtenu le verrou `#cvs.lock`, créez un fichier dont le nom est `#cvs.rfl.` suivi par l'information de votre choix (par exemple, un nom de machine ou un identifiant de processus). Puis retirez le verrou `#cvs.lock` pour relâcher le verrou général. Vous pouvez maintenant lire le dépôt. Quand vous avez fini, retirez le fichier `#cvs.rfl.` pour relâcher le verrou.

Pour obtenir un verrou en écriture, créez d'abord le répertoire verrou `#cvs.lock`, comme pour le verrou en lecture. Vérifiez ensuite qu'il existe pas de fichiers dont les noms commencent par « #cvs.rfl. ». S'il en existe, enlevez le verrou principal `#cvs.lock`, attendez un peu, puis recommencez. S'il n'y pas de lecteur, créez alors un fichier dont le nom sera `#cvs.wfl` suivi d'informations de votre choix (par exemple, nom de machine et identifiant de processus.) Suspendez-vous au verrou `#cvs.lock`. Procédez à vos écritures dans le dépôt. Quand vous avez fini, retirez d'abord le verrou `#cvs.wfl` puis le répertoire `#cvs.lock`.

3.10. Révisions symboliques en utilisant les balises CVS

Le numéro de version des produits en version finale sont différents des révisions dans CVS. Les numéros de révision peuvent changer plusieurs fois entre deux versions. Vous pouvez utilisez la commande **tag** pour donner un nom symbolique à une certaine révision d'un fichier.

Changez de répertoire pour le répertoire de travail, and passez la commande suivante pour baliser :

```
$ cvs tag rel1-1 fichier.c
```

Cette commande marquera le fichier `fichier.c` comme étant de révision 1.1.

```
$ cvs tag rel1-1 .
```

Cette commande marquera récursivement tous les fichiers du répertoire courant comme étant de révision 1.1.

Vous pouvez utilisez l'option `-v` de la commande **status** pour voir toutes les balises assignées au fichier, et quels numéros de révision ils représentent, en passant la commande suivante :

```
$ cvs status -v fichier.c
```

Vous pouvez maintenant extraire n'importe quelle révision d'un module en utilisant la commande suivante :

```
$ cvs checkout -r rel1-1 module1
```

où « module » est le nom du module. L'option `-r` associé à la commande **checkout** permettra à n'importe quel moment dans le futur d'extraire les sources constituant la révision 1.1 du module « module1 ».

4. Plusieurs développeurs

4.1. État des fichiers

La commande **cvs status** donne l'état des fichiers. Vous pouvez les obtenir en passant la commande :

```
$ cvs status [options] files
```

4.2. Mettre un fichier à jour

Quand vous voulez mettre à jour ou fusionner un fichier, utilisez la commande **cvs update**. Elle amène à votre copie de travail tous les changements que les autres développeurs auront récemment validés. Vos modifications à un fichier ne sont jamais perdues. Si aucune nouvelle révision n'est disponible, CVS fusionnera alors tous les changements dans votre copie de travail.

4.3. Résolution des conflits

Si deux personnes font simultanément des modifications sur différentes parties d'un fichier, CVS est suffisamment intelligent pour fusionner les changements tout seul. Mais si deux personnes font des changements sur la *même* partie d'un fichier, CVS ne peut savoir quel doit être le résultat final, et va donc abandonner en se plaignant : « Conflict ! » Les conflits surgissent quand un développeur valide un changement et qu'un second développeur, qui n'a pas utilisé la commande **cvs update** pour recevoir les modifications du premier développeur, essaie de valider ses propres changements incompatibles. La résolution des conflits peut prendre des heures voire des jours. Dans cette section, je vous expliquerai comment résoudre les conflits de source.

Quand vous utilisez la commande **cvs commit** pour envoyer automatiquement sur le serveur tous les fichiers que vous avez modifié ou ajouté à un projet, le serveur du dépôt CVS pourra vous informer que vos fichiers modifiés localement ne sont pas à jour avec le serveur, ou vous avez besoin de fusionner à la main un ou plusieurs fichiers avec les nouvelles versions déjà validées sur le serveur par un autre développeur. Voici un message d'avertissement typique survenant lors d'une session CVS :

```
$ cvs commit
cvs commit: Examining .
cvs commit: Up-to-date check failed for ‘andy.htm’
cvs commit: Up-to-date check failed for ‘sample.htm’
cvs commit: Up-to-date check failed for ‘index.htm’
...
cvs [commit aborted]: correct above errors first!
```

Vous pouvez utiliser la commande **cvs update** pour mettre à jour la copie locale de votre projet avec les derniers changements du dépôt CVS. Pour mettre à jour l'intégralité de votre copie locale, ouvrez un interpréteur de commandes, placez vous dans le répertoire contenant le projet que vous développez, et lancez la commande :

```
$ cvs update
```

Cela va mettre à jour et fusionner automatiquement tous les fichiers qui ont changé depuis la dernière extraction de nouveaux fichiers du dépôt CVS. Les mises à jour lignes à lignes sur des fichiers textes (comme des fichiers HTML) peuvent souvent être gérées automatiquement. CVS vous listera les fichiers qui requièreront votre attention en vue d'une modification et fusion manuelle.

4.3.1. Exemple de fusion automatique :

Vous modifiez localement un fichier nommé `index.html` et au moment où vous le validez dans l'entrepôt CVS, CVS vous répond avec l'erreur suivante :

```
$ cvs commit index.html
cvs commit: Up-to-date check failed for ‘index.html’
cvs [commit aborted]: correct above errors first!
```

Cela survient car il y a une nouvelleversion de de ce fichier dans l'entrepôt CVS. Vous devez utiliser la commande **cvs update** pour en obtenir la dernière version sur votre machine :

```
$ cvs update index.html
RCS file: /usr/local/cvsroot/index.html,v
retrieving revision 1.4
retrieving revision 1.5
Merging differences between 1.4 and 1.5 into index.html
M index.htm
```

Après la fusion automatique, vous devez vérifier la copie fusionnée afin qu'elle fonctionne correctement. Une fois satisfait de votre copie locale de `index.html`, vous la validez dans l'entrepôt CVS :

```
$ cvs commit index.htm
Checking in index.htm;
/usr/local/cvsroot/index.htm,v <- index.htm
new revision: 1.6; previous revision: 1.5
done
```

4.3.2. Exemple de fusion manuelle

Dans certains cas, votre récent travail sur certains fichiers peut être si important que le CVS aura besoin de votre intervention manuelle de façon à réintégrer les modifications de tout le monde dans l'entrepôt.

```
$ cvs commit index.html cvs commit: Up-to-date check failed for
‘index.html’ cvs [commit aborted]: correct above errors first!
```

Utilisez la commande update pour mettre à jour votre copie locale :

```
$ cvs update
cvs update: Updating .
```

```
RCS file: /usr/local/cvsroot/index.html,v
retrieving revision 1.5
retrieving revision 1.6
Merging differences between 1.5 and 1.6 into index.htm
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in activity.htm
C index.htm
```

Cette fois-ci, CVS a été incapable de fusionner les fichiers automatiquement. Il a donc créé une copie spéciale du fichier provoquant le conflit en lieu et place du fichier originel. Ce fichier contient des lignes de marqueurs pour indiquer le début et la fin des régions conflictuelles, par exemple :

```
«««« filename
```

Pour résoudre les conflits, modifiez simplement le fichier `index.html` et remplacez le texte entre les marqueurs et testez le résultat jusqu'à ce que ça marche. Vous devez aussi enlever les marqueurs :

```
«««««========»»»»»
```

du fichier. Quand vous avez fini de corriger le fichier et l'avez testé, utilisez la commande **cvs commit** pour envoyer dans l'entrepôt cette dernière version de votre fichier :

```
$ cvs commit
Checking in index.html;
/usr/local/cvsroot/index.html,v <- index.html
new revision: 1.7; previous revision: 1.6
done
```

4.3.3. Les yeux (communication CVS)

CVS peut fonctionner en tant que moyen de communication de même que main courante. La fonctionnalité « garder un oeil » permet à plusieurs développeurs travaillant sur le même projet de notifier aux autres qui travaille sur quoi à un instant donné. En positionnant un oeil sur un fichier ou répertoire, un développeur peut demander à CVS de lui dire si quelqu'un d'autre commence à travailler sur ce fichier ou répertoire en lui envoyant un mél ou par un autre moyen.

Pour utiliser les yeux, vous devez modifier deux fichiers dans la partie administrative de l'entrepôt de code. Vous devez modifier le fichier `$CVSROOT/CVSROOT/notify` (qui spécifie à CVS comment doivent se faire les notifications) et le fichier `$CVSROOT/CVSROOT/users` (qui spécifie les adresses mél externes.) Le meilleur moyen de modifier

5. Revenir à une précédente version

6. Quelques commandes CVS

7. Autres outils et ajouts à CVS

8. Pour plus d’information

- Manuel de CVS : http://www.cvshome.org/docs/manual/cvs.html
- Listes de distribution CVS : http://www.cvshome.org/communication.html

Paru dans le numéro 66 de la Linux Gazette, Mai 2001.

Traduction française par Jérôme Fenal <jerome@fenal.org>.