

0.1 Utilisation de CVS

0.1.1 1. Qu'est-ce que CVS, à quoi ça sert ?

CVS signifie Concurrent Versions System. C'est un utilitaire qui facilite le développement de projet en équipe, en conservant l'historique des modifications apportées à chaque fichier. Il travaille essentiellement sur des fichiers **textes**, c'est-à-dire qu'il ne sert à rien si vous travaillez sur des images ou sur des fichiers compressés. Cependant, il peut être très utile pour les projet de programmation (C, JAVA, HTML, ...), de rédaction (au format texte, donc pas de .doc), et tous les fichiers XML.

0.1.2 2. Installation du client cvs

CVS est surement inclu dans les packages de votre distribution. Pour les inconditionnels des sources, vous pouvez le télécharger sur www.cvshome.org¹.

0.1.3 3. CVS côté client

0.1.4 3.1 Exemple d'utilisation classique

Le mieux est d'étudier un cas concret (le nom du serveur est fictif).

Récupération du projet

Tout d'abord, configurez les variables de cvs (à mettre éventuellement dans le ~/.bashrc ou ~/.bash_profile). Par exemple :

```
export CVSROOT=morgoth@cvs.creatta.com :/var/cvs/cvsroot
export CVS_RSH=ssh
export CVSEEDITOR=vim
```

Fermez puis relancez votre terminal afin que ce dernier lise ces variables. Ensuite, récupérez une copie de travail du projet. Par exemple :

```
$ cvs checkout CreatTA_java
cvs checkout : Updating CreatTA_java
U CreatTA_java/TestCreation.java
...
```

Notez le "U" devant le nom du fichier. Cela signifie mis-à-jour (Update).

Modifications

Vous pouvez faire ensuite des modifications comme vous le souhaitez dans le dossier (ici CreatTA_java). Par contre, ne modifiez pas les fichiers inclus dans les répertoires CVS. Ils sont nécessaires au bon fonctionnement de CVS.

```
$ cd CreatTA_java
$ vim TestCreation.java
```

Il vous faut maintenant valider ces modifications sur le serveur.

¹<https://www.cvshome.org>

```
$ cvs commit nom du fichier
```

ou

```
$ cvs commit
```

CVS vérifie que personne n'a modifié le fichier depuis que vous l'avez récupéré, et ajoute vos modifications en incrémentant le numéro de version du fichier. En cas de conflit, voyez plus bas.

```
$ cvs commit TestCreation.java
```

CVS vous demande de remplir un petit message expliquant vos modifications. N'oubliez pas de sauver ce message.

```
Checking in TestCreation.java ;
/var/cvs/cvsroot/CreataTA_java/TestCreation.java,v <- TestCreation.java
new revision : 1.7 ; previous revision : 1.6
done
```

Ici, tout s'est bien passé. Si on veut ensuite récupérer toutes les dernières versions des fichiers :

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

Ici, tout est à jour, donc aucun fichier n'est récupéré.

0.1.5 3.2 Commandes usuelles

Récupérer les derniers fichiers du projet dans un dossier du même nom :

```
$ cvs checkout nom du module
```

Valider les changements effectués :

```
$ cvs commit [nom du fichier]
```

Récupérer la dernière version du fichier, en fusionnant éventuellement avec vos modifications (voir Les conflits) :

```
$ cvs update [nom du fichier]
```

Ajouter le fichier ou le répertoire au dépôt. Un "commit" est nécessaire pour valider l'ajout :

```
$ cvs add nom de fichier ou de répertoire
```

Retirer du dépôt un fichier ou un répertoire **vide** :

```
$ cvs remove nom de fichier ou de répertoire vide
```

Lister l'ensemble des modifications effectuées sur le fichier :

```
$ cvs log nom du fichier
```

0.1.6 3.3 Les conflits

En utilisant CVS à plusieurs développeurs, des conflits peuvent apparaître. Il faut pour cela comprendre un peu comment fonctionne CVS. Quand vous récupérez un fichier (`cvs checkout`), CVS mémorise le numéro de version du fichier à ce moment. Disons qu'il s'agit de la version 1.5. Un autre utilisateur récupère cette version, fait des modifications, et les valide (`cvs commit`). La version sur le serveur devient 1.6. De votre côté, vous faites des modifications sur ce même fichier, et vous tentez de les valider (`cvs commit`). CVS compare la version que vous aviez récupérée (1.5) et la version courante (1.6) : il sait que quelqu'un a modifié le fichier entre-temps ; il y a conflit, et donc il refuse de valider les changements. La solution consiste à fusionner les modifications. Pour cela, faites un `cvs update nom du fichier`. Vous devriez voir un M devant votre fichier (M pour merge = fusionné). Éditez maintenant le fichier et regardez si la fusion s'est bien passée. Si vous n'avez pas édité les mêmes parties du fichier, il ne devrait pas y avoir de problème. %RB% En revanche, si vous avez touché à la même ligne, CVS ne sait pas quoi faire, et vous demande de corriger manuellement. Ces conflits "graves" sont signalés dans le fichier par des lignes du style "CVS»»»»»»»»»»»»»»»»>". Réparez donc ces conflits, retirez les lignes ajoutées par CVS, sauvez et validez (`cvs commit`). Normalement, tout devrait bien se passer, et le fichier devrait obtenir le numéro de version 1.7. Sauf que quelqu'un a pu encore modifier le fichier pendant que vous faisiez la fusion... C'est pourquoi il faut éviter les conflits comme la peste, et éviter de travailler sur les mêmes parties d'un fichier. Il faut donc **communiquer** entre développeurs, ce que ne permet pas CVS.

0.1.7 3.4 CVS avancé

Les branches et les révisions.

0.1.8 4. CVS côté serveur

Il existe plusieurs façon de créer et d'accéder à un dépôt CVS. Je n'en aborderai que trois : le dépôt local, CVS avec SSH et CVS avec :pserver.

Le dépôt local

Un dépôt local peut être utile si vous travaillez seul sur un projet, mais que vous souhaitez utiliser CVS pour conserver l'historique de vos modifications. Création du dépôt :

```
# mkdir /var/cvs/cvsroot
```

Configuration des variables locales :

```
# export CVSROOT=/var/cvs/cvsroot
```

Initialisation du dépôt :

```
# cd /var/cvs/cvsroot ; cvs init
```

Il faut ensuite ajuster les droits afin que les utilisateurs aient le droit de lire **ET** d'écrire dans le dépôt. Pour accéder à votre dépôt, définissez CVSROOT :

```
$ export CVSROOT=/var/cvs/cvsroot
```

Accès SSH

L'accès à CVS par SSH est le seul moyen sécurisé pour autoriser des modifications à distance sur votre dépôt. Il suffit de créer le dépôt comme indiqué ci-dessus, puis de renseigner les variables CVSROOT et CVS_RSH :

```
$ export CVSROOT=morgoth@Angband.com :/var/cvs/cvsroot
$ export CVS_RSH=ssh
```

En remplaçant "morgoth" par votre login ssh, et "Angband.com" par votre serveur.

Accès :pserver

L'accès par :pserver permet d'autoriser des personnes à se connecter de manière anonyme en lecture seule à votre dépôt. Cette méthode est utilisée par un grand nombre de projets libres. Tout d'abord, vous devez vous assurer d'avoir un compte anonyme (par exemple) et de lui autoriser l'accès en lecture à votre dépôt CVS. Pour ce faire, tapez dans un terminal :

```
# (grep anonymous /etc/passwd || useradd anonymous -s /bin/false)
# echo anonymous : > /var/cvs/cvsroot/CVSROOT/passwd
# echo anonymous > /var/cvs/cvsroot/CVSROOT/readers
```

Maintenant, il ne vous reste plus qu'à activer le service :pserver et à le configurer.

- **Pour Mandriva :** Vous n'avez rien à faire, ou presque. Tout d'abord, vérifiez que le fichier `/etc/cvs/cvs.conf` contient bien : `CVS_REPOS="/var/cvs/cvsroot"` (ou autre si vous n'avez pas choisi ce répertoire). Ensuite, activez :pserver en éditant le fichier `/etc/xinetd.d/cvs` et en mettant `disable = no`. Ensuite, un petit redémarrage de xinetd :

```
# service xinetd restart
```

- **Pour les autres :** Ajoutez le service à xinetd (si il n'existe pas déjà) :

```
# cat » /etc/xinetd.conf « "EOF"
service cvspserver
{
port = 2401
socket_type = stream
protocol = tcp
wait = no
user = root
passenv = PATH
server = /usr/bin/cvs
server_args = -f --allow-root=/var/cvs/cvsroot pserver
}
EOF
```

Puis forcez xinetd à relire sa configuration par : `# killall -HUP xinetd`

Maintenant, pour accéder de manière anonyme à votre dépôt, il suffit de taper la commande suivante (ajustée à votre configuration) :

```
$ cvs -d :pserver :anonymous@nom_serveur :/var/cvs/cvsroot checkout nom_projet
```

Ajouter un nouveau projet

Pour ajouter un nouveau projet au dépôt CVS, allez dans le répertoire qui contient les sources de votre projet :

```
$ cd mon-projet
$ cvs import projet-XYZ Morgoth initial
```

Où "projet-XYZ" sera le nom de votre module, et "initial" le tag.

0.1.9 5. Interfaces graphiques

Il existe de nombreuses interfaces graphiques pour cvs. Citons TkCvs (que je n'ai jamais essayé), Cervisia (l'interface CVS de KDE), et enfin WinCVS (sous Windows). CVS s'intègre dans de nombreux outils de développement, comme Eclipse ou Netbeans.

Documentations complémentaires

Le tutoriel de Gentoo sur CVS², qui indique énormément d'exemples pratiques ainsi que des liens à des documentations officielles.

²<http://www.gentoo.org/doc/fr/cvs-tutorial.xml>