Introduction à Subversion : principes et conseils

Georges-Etienne Legendre (http://legege.com)
Félix-Antoine Bourbonnais (http://rubico.info)

Mai 2006

1 Introduction

Subversion (SVN) est un outil performant de versionnement de fichiers, créé dans le but de remplacer et corriger plusieurs des lacunes de CVS.

Ce document est destiné aux nouveaux utilisateurs de Subversion. Le but du présent document n'est pas d'indiquer étape par étape comment réaliser des opérations particulières avec Subversion. Il regroupe plutôt un ensemble de conseils et de bonnes pratiques accumulés par nos expériences avec l'outil Subversion. Il ne remplace pas non plus le manuel officiel de Subversion, que vous êtes fortement invités à consulter.

2 Spécificités de Subversion

Un système optimiste

Subversion est un **système optimiste** d'archivage (versionnage) et de partage de fichiers entre utilisateurs. Subversion autorise plus d'une personne à modifier un même fichier en même temps. Ce système suppose que les conflits de modifications sont rares. Si deux utilisateurs du dépôt modifient des parties non-communes d'un fichier, les outils optimistes fusionnent tout simplement les modifications. Inversement, dans le cas de modifications conflictuelles, le conflit doit être résolu manuellement.

À l'opposé des systèmes optimistes, il existe le modèle de **réservation de fichiers**. Ce principe accorde l'exclusivité de modification à une unique personne. C'est la philosophie adoptée, par exemple, sous *Microsoft Visual Source Safe*. Cependant, depuis la version 1.2 de Subversion, un mode similaire est disponible. Il permet de réserver l'exclusivité des « commit ».

Pourquoi préférer les systèmes optimistes aux autres ? À première vue, la possibilité de fusionner des documents peut sembler périlleuse. Cependant, ce mode de fonctionnement offre beaucoup plus d'avantages que d'inconvénients. Le rythme de production au sein d'une équipe n'est pas brisé ; une personne n'est pas empêchée de faire une modification alors qu'une autre a réservé le fichier cible.

Un dépôt centralisé

Subversion est un système à dépôt centralisé. Un dépôt (ou référenciel) représente un point unique auquel les utilisateurs s'adressent pour obtenir, ajouter ou mettre à jour fichiers et dossiers. On peut appeler cet endroit de référence la **copie officielle**.

Chaque utilisateur du dépôt possède ensuite une **copie locale** du référentiel commun. Comme un utilisateur fait ses modifications sur sa copie locale, la copie officielle distante n'est pas changée automatiquement. Pour partager ses modifications, il doit les soumettre au dépôt. Par la suite, ses collègues, qui veulent obtenir la version plus récente, doivent, à leur tour, mettre à jour leur copie locale.

3 Terminologie

Il convient de définir quelques termes utilisés dans Subversion. Habituellement, les termes anglais sont employés.

« Revision » Une révision est un numéro de version pour l'ensemble du dépôt à un moment précis. Chaque « commit » augmente le numéro de révision d'une unité.

« Update » Un « update » ou mise-à-jour est l'action de mettre à jour sa copie locale par rapport au dépôt.

« Checkout » Un « checkout » est l'action de récupérer les fichiers et les dossiers du dépôt dans le but de construire une copie locale. Cette action est semblable à exécuter un « update », mais est effectuée la première fois seulement.

« Commit » Un « commit » est l'action de soumettre ses ajouts, suppressions ou modifications dans le dépôt.

4 Conseils et bonnes pratiques

4.1 Interactions avec le dépôt

Un « update » doit toujours précéder un « commit ». Effectuer un « update » avant de procéder à un « commit » n'est pas obligatoire avec Subversion, mais reste fortement conseillé. Il s'agit là d'une très bonne habitude à prendre. En fait, Subversion autorise les « commit » sans « update » préalable dans la mesure où les fichiers comportant des modifications locales n'ont pas été aussi modifiés par un autre utilisateur. Appliquer ce conseil premet d'éviter de travailler dans une copie locale périmée tout en pensant, à tort, qu'elle est à jour.

Mettre à jour l'ensemble de la copie locale. Subversion permet de ne mettre à jour qu'une partie de la copie locale. Par exemple, la commande « svn update » procède, par défaut, à la mise à jour du répertoire courant et de ses enfants. Il est toutefois important d'effectuer la mise à jour de l'ensemble de la copie locale

assez régulièrement pour s'assurer d'avoir une copie locale uniformément mise à jour. Une copie locale inégalement à jour peut causer des problèmes parfois difficiles à diagnostiquer particulièrement quand les fichiers font référence les uns aux autres.

Mettre à jour la copie locale périodiquement. Cette pratique permet d'éviter le plus possible les conflits. Elle va de pair avec des « commit » fréquents. Plus les « update » sont réguliers plus les conflits sont rares ou faciles à corriger.

Ajouter systématiquement les nouveaux fichiers. La création d'un nouveau fichier n'ajoute pas ce dernier automatiquement dans la copie locale et encore moins dans le dépôt. Le fait d'ajouter les fichiers à la copie locale dès leur création permet de ne pas les oublier lors du prochain « commit ». Un tel oubli serait susceptible de provoquer un problème de dépendance et empêcherait les autres de travailler.

Ne versionner que les fichiers nécessaires au projet. Afin de garder le dépôt propre et d'une taille raisonnable, il est important de ne pas versionner des fichiers qui sont inutiles aux autres membres de l'équipe. Il n'est pas nécessaire, par exemple, de versionner les fichiers de type objet (.o ou .obj) créés lors d'une compilation. Le versionnement de l'exécutable (le programme) produit ne devrait, très souvent, même pas se retrouver dans le dépôt. Il faut aussi ignorer les répertoires « build », « debug », « release », « .libs », etc., qui sont souvent générés par les environnements de développement.

Notez qu'il est possible de demander à Subversion d'ignorer systématiquement certains types de fichiers avec la propriété « svn:ignore » ou via une configuration globale de Subversion.

Un « commit » doit représenter un tout. Réaliser des « commit » pour chaque fichier modifié, ou encore, faire des « commit » représentant le travail de toute une journée (ou pire d'une semaine complète !) sont des comportements à éviter. Il est plutôt recommandé de réaliser des « commit » qui représentent une idée en soi. Par exemple, envoyez dans un seul bloc toutes les modifications qui concernent une anomalie particulière. De cette façon, il devient très facile de revenir en arrière (*rollback*) si nécessaire.

Chaque « commit » doit comporter un message représentatif des modifications envoyées. Les commentaires associés aux « commit » permettent aux autres utilisateurs de prendre rapidement connaissance des modifications soumises dans le dépôt. Ces messages aident grandement la compréhension des changements.

4.2 Gestion du projet

Créer initialement les dossiers « trunk », « branches », et « tags ». Bien que Subversion laisse pleine liberté concernant le choix de l'arborescence d'un dépôt, il existe toutefois des schémas recommandés. Il est suggéré de **toujours créer trois dossiers** à la racine du projet :

trunk/ C'est le répertoire de travail courant. Il contient généralement la version la plus récente et en cours de développement.

tags/ Contient des références à certaines révisions importantes du dépôt. Les différentes versions (« releases ») d'un logiciel par exemple. On ne devrait jamais travailler dans ce répertoire.

branches/ Contient des branches pour le projet (consultez le manuel à ce sujet).

En fonction de l'ampleur et du nombre de projets contenus dans votre dépôt, l'emplacement de ces trois répertoires peut varier. Il existe en fait deux formes recommandées en fonction de vos besoins :

1. Commun à tous les projets

```
trunk/
  projetA/
  projetB/
tags/
  projetA/
    version1/
  projetB/
branches/
```

2. Propre à chaque projet

```
projetA/
  trunk/
  tags/
  branches/
projetB/
  trunk/
  tags/
  branches/
```

Annoncer vos actions importantes à toute l'équipe. La communication entre les membres d'une équipe est fondamentale. Bien que Subversion offre la fonctionnalité de fusionner un document édité par plus d'une personne, il vaut mieux essayer de réduire l'occurence de ces situations. Subversion est un outil de versionnement pour les sources et documents. Il ne remplace en aucun cas le courriel, les listes de diffusion ou tout autre moyen de communication au sein de l'équipe.

Les changements envoyés ne doivent pas empêcher les autres de travailler. Avant de faire un « commit » de vos changements, assurez-vous que vos modifications sont fonctionnelles et qu'elles compilent le cas échéant. Testez également l'ensemble du projet afin de vous assurer que vous n'avez pas introduit de nouvelles erreurs. À ce titre, des tests unitaires peuvent s'avérer un bon moyen de vérification.

Réduire les conflits au minimum. Les conflits devraient normalement être rares. Des conflits fréquents cachent peut-être un autre problème. Il faut alors s'interroger sur sa façon d'utiliser Subversion et sur la gestion du projet. Peut-être que les « update » ne sont pas assez fréquents, les « commit » trop gros ou peut-être qu'il existe un manque de communication au sein de l'équipe.

Une autre réaction possible depuis Subversion 1.2 est l'utilisation systématique des verrous (« lock »). Cette solution est discutable, mais n'est souvent pas une solution au problème. Vous risquez alors de diminuer encore davantage la productivité.

4.3 Projets avec utilisateurs multi-plateformes

Éviter d'utiliser des caractères spéciaux dans les noms d'entités. Les sytèmes d'exploitation possèdent diverses restrictions à respecter dans les noms de fichiers. Faire un « commit » d'un fichier comportant un « : » à partir de *GNU/Linux* empêche un client Subversion sous *Microsoft Windows* d'effectuer correctement son « update ».

Le mieux est donc de se restreindre aux caractères alpha-numériques de base (lettres et chiffres). Évitez particulièrement les caractères `\ / : * ? < > | "` qui ne seront pas acceptés sous *Windows* mais qui, pour la plupart, passeront sans problème sous *GNU/Linux*.

Adopter un type de retour de chariot pour les fichiers textes et l'appliquer systématiquement. Subversion offre la possibilité de forcer les fichiers textes à être entreposés dans le dépôt avec un certain type de retour de chariot (svn:eol-style). En effet, certains éditeurs transforment automatiquement à la sauvegarde les fichiers pour qu'ils utilisent le type de retour de chariot du système hôte. Ce choix est très important pour s'assurer que les fusions automatiques et les comparaisons fonctionnent correctement.

4.4 Généralité

Vérifier si la fusion automatique s'est bien déroulée. Subversion est en mesure de fusionner automatiquement les différentes versions d'un fichier texte si les modifications ne sont pas conflictuelles. Cependant, il est très important de vérifier que cette fusion est cohérente. Dans le cas d'un code source, il se pourrait bien que le code ne compile ou ne s'exécute plus suite à une fusion en raison, par exemple, de la duplication d'une déclaration de méthode.

Prendre au sérieux les conflits. Les débutants avec Subversion ont tendance à se sentir découragés lors d'un conflit et à adopter la solution qui leur paraît la plus simple : effacer et recommencer. Malheureusement, dans bien des cas, il serait bien moins long de résoudre correctement le conflit que de recommencer. Afin de faciliter cette tâche, il est important de bien comprendre la mécanique entourant la résolution de conflits (consultez la section 5.2). Les premières résolutions de conflits sont souvent longues et maladroites mais vous prendrez de l'expérience en persévérant. Essayez également de diminuer le risque de conflits en adoptant de bonnes pratiques adaptées à votre projet.

Utiliser le client Subversion pour les supressions, déplacements et copies plutôt que les fonctions du système d'exploitation. Bien que vous soyez habitués de renommer, supprimer, copier et déplacer des fichiers et dossiers à partir des fonctions du système d'exploitation, il est dangereux de faire ces manipulations quand vous vous situez dans votre copie locale. Les clients Subversion possèdent des fonctionnalités permettant de faire ce genre de manipulation sécuritairement.

Une autre conséquence de ceci est que si vous essayez de supprimer un fichier directement avec les fonctions de votre système d'exploitation ou via un programme tiers, il reviendra automatiquement au prochain

« update » puisqu'il sera toujours marqué comme versionné dans le dépôt. Il faut donc passer par le client Subversion afin que le fichier cesse d'être versionné.

Supprimer pour réparer. En cas de problème majeur, il peut être plus simple de supprimer une section complète de la copie locale. Avant tout, faites cependant une copie de votre copie locale pour conserver vos modifications. Ensuite, supprimez la branche problématique, puis exécutez un « update » sur le parent de la branche supprimée. Cela aura pour effet de reprendre, à partir du dépôt, la partie manquante de la copie locale. Ensuite, vous pourrez réintégrer manuellement vos modifications à partir de votre copie. Notez qu'un « revert » peut également être pratique dans de telles situations.

Dans bien des cas, vous éviterez de tels problèmes en suivant les conseils du présent document.

5 Conflits et fusions

5.1 Introduction

En cas de modifications simultanées sur un même fichier, deux situations sont possibles pour un fichier texte :

1. Les modifications ne sont pas conflictuelles (les deux développeurs n'ont pas modifié les mêmes lignes)

   Dans ce cas, Subversion se charge de fusionner automatiquement les deux versions sans aucune intervention lors de la mise à jour (« update »). Le résultat de la fusion est placé dans la copie locale et Subversion avise l'utilisateur qu'une fusion a eu lieu. Il est important de noter que **le dépôt ne possède toujours pas le résultat de cette fusion**. Cela laisse la possibilité à l'utilisateur de **vérifier que la fusion a du sens**, ce qui est fortement conseillé. Après quoi, ce dernier effectuera un « commit » afin d'envoyer le fichier fusionné dans le dépôt. La figure 1 illustre cette situation.

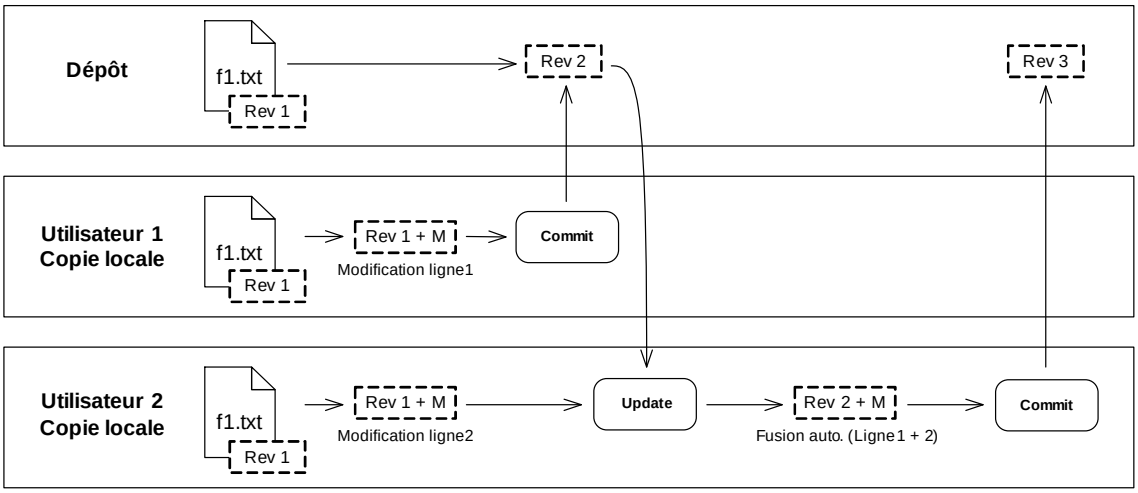

Figure 1 – Fusion automatisée

2. Le modifications sont conflictuelles (une ou plusieurs lignes ont été modifiées par les deux développeurs)

   Une intervention est alors nécessaire afin de procéder à une fusion manuelle. Consultez la section 5.2.

Comme toute opération dans Subversion, les fusions automatiques sont effectuées dans la copie locale lors d'un « update ». D'ailleurs, Subversion refuse d'effectuer un « commit » si l'un des fichiers modifiés localement n'est pas à jour par rapport au dépôt.

5.2 Résolution manuelle de conflits

En cas de modifications conflictuelles ou pour des fichiers binaires, il est nécessaire de procéder à la fusion de manière manuelle. Dans le but de faciliter la fusion, Subversion ajoute plusieurs fichiers dans la copie locale. Pour un conflit sur le fichier nommé fichier1, voici les fichiers ajoutés ou modifiés :

– fichier1.mine :

  Fichier texte Le fichier modifié dans la copie locale tel qu'il était avant la mise à jour (r100 + modifications).

  Fichier binaire Ce fichier n'est pas créé si le conflit a lieu sur un fichier binaire. Dans ce cas, la version modifiée localement reste inchangée (fichier1).

– fichier1.r102 : Dernière révision disponible sur le dépôt (r102)
– fichier1.r100 : Révision à partir de laquelle les modifications locales ont été effectuées (r100)
– fichier1 :

  Fichier texte Contient une fusion des modifications locales et de la dernière révision du dépôt. Étant donné que des conflits sont présents, Subversion insèrera des marqueurs afin de montrer visuellement la différence entre les deux versions. Ceci peut permettre à l'utilisateur ou à des outils de comparaison de repérer rapidement les différences entre les versions.

```
Ligne 1
<<<<<<< .mine
Ligne 2 : Modification par FA
=======
Ligne 2 : Modification par GE
>>>>>>> .r102
Ligne 3
```

  Fichier binaire Le fichier contenant les modifications locales (il reste donc inchangé).

Pour résoudre le conflit, l'utilisateur possède plusieurs choix. S'il juge ses modifications comme étant mineures ou qu'il préfère la version du dépôt, il peut simplement écraser ses modifications en renommant le fichier1.r102 pour fichier1. Cependant, dans la majorité des cas, l'utilisateur souhaite plutôt fusionner les deux versions à l'aide d'un outil approprié (selon le client utilisé) ou encore directement dans le fichier à l'aide des marqueurs insérés par Subversion.

Une fois le conflit résolu, il est nécessaire d'indiquer à la copie locale la résolution du conflit (svn resolved) et d'envoyer (« commit ») le résultat de la fusion[1].

Références

[1] COLLINS-SUSSMAN, Ben et W. FITZPATRICK Brian et PILATO C. Michael. *Version Control with Subversion*, http://svnbook.red-bean.com/

[2] BOURBONNAIS, Félix-Antoine. *Programmer avec Subversion*, http://www.rubico.info/docs/prog-subversion/

[3] *Version Control System Comparison*, http://better-scm.berlios.de/comparison/comparison.html

[4] HANSELMAN, Scott. *CVS and Subversion vs. VSS/SourceSafe*, http://www.hanselman.com/blog/CVSAndSubversionVsVSSSourceSafe.aspx

[5] *How to Organize a Subversion Repository*, http://docs.codehaus.org/display/HAUS/How+to+Organize+a+Subversion+Repository

[6] *KDE SVN Commit Policy*, http://developer.kde.org/policies/commitpolicy.html

[1] Cette procédure peut varier d'un client à l'autre.