Control de versiones con Subversion v1.2

Raúl García-Castro

rgarcia@fi.upm.es

Índice

Capítulo 1

¿Qué es Subversion?

Subversion es un sistema centralizado de compartimiento de información.

La información está almacenada en un repositorio (que no es más que una jerarquía de directorios y ficheros). Los clientes se conectan al repositorio y leen o escriben esos ficheros. Además de almacenar los ficheros, subversion almacena todos los cambios que se han producido sobre los ficheros y directorios de repositorio.

Subversion no utiliza el sistema de bloquear ficheros antes de hacer cambios sobre ellos, ya que impide que varios usuarios trabajen a la vez sobre el mismo fichero, da lugar a que los clientes se olviden de desbloquear ficheros, etc.

Subversion utiliza el método copiar-modificar-mezclar. En este método los clientes se crean localmente una copia del contenido del repositorio. Pueden trabajar en paralelo y, cuando han modificado los ficheros y quieren publicar los cambios, se realiza una mezcla de la copia local de los ficheros con el contenido del repositorio. El sistema de control de versiones asiste al usuario a la hora de hacer las mezclas entre ficheros y, en caso de que existan conflictos, el usuario puede elegir cómo resolverlos.

Capítulo 2
Repositorios

2.1. ¿Qué es un repositorio?

En Subversion, un repositorio es una secuencia de árboles de directorios.

Una revisión es una instantánea del repositorio en un punto determinado del tiempo. Cada vez que se realiza un cambio en un fichero o directorio de un repositorio, el número de revisión de éste se incrementa en uno.

2.2. Creación recomendada de un repositorio

A la hora de crear un nuevo repositorio (por ejemplo el repositorio "prueba") se recomienda hacerlo con la siguiente jerarquía de directorios:

/prueba/trunk Para la copia actual en desarrollo del proyecto.

/prueba/branches Para las ramas del proyecto.

/prueba/tags Para las etiquetas del proyecto.

Capítulo 3

Ramas y etiquetas

3.1. Ramas

Una rama es una línea de desarrollo que existe independiente de otra línea. Una rama siempre se crea a partir de una copia de un directorio del repositorio.

Subversion tiene distintos comandos para mantener ramas paralelas de los ficheros y directorios. Permite crear ramas y recuerda que éstas están relacionadas, permite detectar cambios entre ramas e incluso permite realizar cambios en múltiples ramas a la vez.

La forma en que subversion crea ramas es copiando un directorio en otro lugar del repositorio. De esta forma se puede seguir trabajando en cada uno de estos directorios por separado.

Ej. Para crear una rama llamada "mi-rama-prueba.ª partir del código actual del proyecto "prueba":

```
svn copy http://svn.example.com/repos/prueba/trunk
         http://svn.example.com/repos/prueba/branches/mi-rama-prueba
         -m "Creada una rama de /prueba/trunk."
```

3.2. Etiquetas

Una etiqueta es una instantánea de un proyecto en el tiempo. Para crear una etiqueta lo que se hace es realizar una copia del directorio deseado tal como se hacía al crear una nueva rama. Se puede decir que una etiqueta no es más que una rama sobre la que no se han realizado cambios.

Ej. Para etiquetar el código actual del proyecto "pruebaçomo la release -elease-1.0":

```
svn copy http://svn.example.com/repos/prueba/trunk
         http://svn.example.com/repos/prueba/tags/release-1.0
         -m "Etiquetando la release 1.0 del proyecto 'prueba'."
```

3.3. Cambiando la copia de trabajo local

El comando "svn switch"permite cambiar la copia de trabajo local de una rama a otra.

Ej. Si estamos trabajando en el código actual del proyecto y queremos cambiar a la rama "mi-rama-prueba":

```
svn info | grep URL
URL: http://svn.example.com/repos/prueba/trunk

svn switch http://svn.example.com/repos/prueba/branches/mi-rama-prueba

svn info | grep URL
URL: http://svn.example.com/repos/prueba/branches/mi-rama-prueba
```

Capítulo 4
Comandos

4.1. Crear un repositorio

Los pasos que hay que realizar para crear un repositorio en el servidor son los siguientes:

1. Conectarse al servidor utilizando ssh y con el usuario svn.
2. Crear el repositorio en el servidor:

   ```
   svnadmin create <directorio en el repositorio>
   ```

   Ej. Para crear el repositorio "prueba":

   ```
   # svnadmin create /srv/svn/repos/prueba
   ```

   Hay que comprobar que los ficheros y directorios del repositorio tengan como usuario y grupo los del Apache (wwwrun y www respectivamente).
3. Si se desea, se puede activar la notificación por email de los cambios realizados en el repositorio. Para ello, hay que configurar el fichero post-commit del directorio hooks del repositorio. Los pasos a seguir en el ejemplo del repositorio prueba son los siguientes:

   ```
   # cd /srv/svn/repos/prueba/hooks
   # cp post-commit.tmpl post-commit
   # chmod 755 post-commit
   # emacs post-commit
   ```

   Y dejar las líneas del final de la siguiente manera:

   ```
   /usr/share/subversion/tools/hook-scripts/commit-email.pl "$REPOS" "$REV" <email1> <e
   # log-commit.py --repository "$REPOS" --revision "$REV"
   ```

Ante cualquier duda, consultar el script post-commit de otro repositorio.

4. Editar el fichero de configuración del subversion:

```
# cd /etc/apache2/conf.d
# emacs subversion.conf
```

Añadir las siguientes líneas al fichero de configuración del subversion:

```
<Location /repos/prueba>
   DAV svn
   SVNPath /srv/svn/repos/prueba
      AuthType Basic
      AuthName "Authorization Realm"
      AuthUserFile /srv/svn/user_access/prueba_passwdfile
      Require valid-user
</Location>
```

5. Reiniciar el Apache:

```
# rcapache2 reload
```

6. Añadir los usuarios a los que se desee permitir el acceso al repositorio

```
# cd /srv/svn/user_access
# htpasswd2 -c prueba_passwdfile <usuario>
```

7. Añadir el repositorio creado al script de copias de seguridad: *backup.sh*.

8. Para introducir en el repositorio los ficheros que queremos que lo compongan, nos colocamos (en nuestra máquina) en el nivel superior al directorio que queremos incluir y ejecutamos:

```
# svn import prueba http://delicias.dia.fi.upm.es/repos/prueba/ -m "Creación inicial
```

A partir de aquí ya está listo el repositorio para poder trabajar con él. Para referirnos a él a través de la línea de comandos, escribiremos "http://svn.example.com/repos/prueba/τ si en un navegador escribimos esa misma URL, podremos acceder a la visualización de los contenidos del mismo.

4.2. Realizar una copia local del contenido del repositorio

```
svn checkout <directorio en el repositorio>
```

Ej. Para obtener una copia local del código del proyecto "prueba":

```
svn checkout http://svn.example.com/repos/prueba/
```

4.3. Ciclo de trabajo típico con Subversion

1. Actualizar la copia local con el contenido del repositorio.
   - svn update
2. Realizar cambios sobre los ficheros de la copia local.
   - svn add
   - svn delete
   - svn copy
   - svn move
3. Examinar los cambios realizados y comparar la copia local con el repositorio.
   - svn status
   - svn diff
   - svn revert
4. En caso de que el repositorio haya cambiado, mezclar los cambios del mismo en la copia local.
   - svn merge
   - svn resolved
5. Actualizar los cambios realizados localmente en el repositorio.
   - svn commit

4.4. Descripción básica de los comandos del ciclo de trabajo típico

Esto es una descripción MUY básica de los comandos anteriores. Normalmente se requerirá usar alguna que otra opción de los comandos. La referencia completa de los comandos con todas las opciones se puede encontrar en: http://svnbook.red-bean.com/svnbook/book.html#svn-ch-9.

4.4.1. svn update

Actualiza la copia local.

```
svn update [<PATH>]
```

Ej. Para actualizar la copia local del código del proyecto "prueba."antes de realizar modificaciones sobre el mismo[1]:

```
svn update http://svn.example.com/repos/prueba/
```

4.4.2. svn add

Añade ficheros y directorios.

```
svn add <PATH>
```

Ej. Para añadir el fichero "fich.java."l código del proyecto prueba:

```
svn add http://svn.example.com/repos/prueba/fich.java
```

4.4.3. svn delete

Elimina un fichero o directorio de la copia local o del repositorio.

```
svn delete <PATH>
svn delete <URL>
```

Ej. Para eliminar el fichero "fich.java"del código del proyecto prueba:

```
svn delete http://svn.example.com/repos/prueba/fich.java
```

4.4.4. svn copy

Copia un fichero o directorio en la copia local o en el repositorio.

```
svn copy <SRC> <DST>
```

Ej. Para copiar el fichero "fich.java."l fichero "fich2.java":

```
svn copy http://svn.example.com/repos/prueba/fich.java
         http://svn.example.com/repos/prueba/fich2.java
```

4.4.5. svn move

Mueve un fichero o directorio en la copia local o en el repositorio.

```
svn move <SRC> <DST>
```

Ej. Para mover el fichero "fich.java."l directorio ."ld":

```
svn move http://svn.example.com/repos/prueba/fich.java
         http://svn.example.com/repos/prueba/old/
```

[1]En este ejemplo y en todos los siguientes, si nos encontramos dentro del directorio "prueba", se puede omitir el path completo: "http://svn.example.com/repos/prueba/".

4.4.6. svn status

Muestra el estado de los ficheros y directorios de la copia local.

```
svn status <PATH>
```

Ej. Para mostrar el estado del fichero "fich.java":

```
svn status http://svn.example.com/repos/prueba/fich.java
```

4.4.7. svn diff

Muestra las diferencias entre dos localizaciones.

```
svn diff <PATH>
```

Ej. Para mostrar las diferencias entre fichero "fich.java"de la copia local y el del repositorio:

```
svn diff http://svn.example.com/repos/prueba/fich.java
```

4.4.8. svn revert

Deshace todas las modificaciones locales al estado después del último checkout o update.

```
svn revert <PATH>
```

Ej. Para deshacer todas las modificaciones realizadas en el fichero "fich.java":

```
svn revert http://svn.example.com/repos/prueba/fich.java
```

4.4.9. svn merge

Aplica las diferencias entre dos fuentes a la copia local.

```
svn merge sourceURL1 sourceURL2 <PATH>
```

Ej. Para mezclar el contenido del fichero "fich.java"del repositorio con el fichero local:

```
svn merge http://svn.example.com/repos/prueba/fich.java
```

4.4.10. svn resolved

Elimina el estado de conflicto en los ficheros o directorios de la copia local.

```
svn resolved <PATH>
```

Ej. Para eliminar el estado de conflicto en el fichero "fich.java":

```
svn resolved http://svn.example.com/repos/prueba/fich.java
```

4.4.11. svn commit

Envía los cambios de la copia local al repositorio.

```
svn commit <PATH>
```

Ej. Para enviar al repositorio los cambios realizados en el fichero "fich.java":

```
svn commit http://svn.example.com/repos/prueba/fich.java
```

Capítulo 5

Uso de Subversion

- La estructura de directorios de un repositorio para un proyecto será la siguiente:

```
/proyecto/trunk    (Para el código actual del proyecto)
/proyecto/branches (Para las ramas del proyecto)
/proyecto/tags     (Para las etiquetas del proyecto)
```

- Hay que intentar mantener las mínimas ramas posibles (lo mejor sería mantener una única rama principal) ya que cuantas mas ramas haya mas mezclas habrá que hacer y mas código hay que mantener.
- La versión del código actual del proyecto (trunk) es una versión inestable.
- Todas las versiones estables (releases) deberán ser etiquetadas realizando una copia del directorio "trunk"dentro del directorio "branchesτ dándole un nombre apropiado.
- Las correcciones de fallos y actualizaciones en versiones estables se realizarán en la nueva rama creada. En caso de querer realizar nuevas releases de una versión estable modificada, se creará una nueva etiqueta.
- La copia de trabajo local ha de mantenerse sincronizada con el repositorio a través de frecuentes actualizaciones.
- Con el objetivo de evitar problemas e inconsistencias en el repositorio, hay que seguir el ciclo de trabajo recomendado:

  1. Actualizar la copia local con el contenido del repositorio.
  2. Realizar cambios sobre los ficheros de la copia local.
  3. Examinar los cambios realizados y comparar la copia local con el repositorio.
  4. En caso de que el repositorio haya cambiado, mezclar los cambios del mismo en la copia local.

5. Comprobar que el código que se va a enviar al repositorio compila y hacer pruebas sobre él.
6. Actualizar los cambios realizados localmente en el repositorio.

- TODAS las actualizaciones realizadas en el repositorio deben incluir una explicación de los cambios realizados.
- A la hora de realizar cambios en el código de un proyecto, es altamente recomendable que dichos cambios traten sobre una misma cosa. Es preferible hacer muchos cambios pequeños que un cambio grande.

  Por ejemplo, si sobre una versión estable del proyecto (release-1.0) se corrige un bug, si la actualización incluye únicamente la corrección de dicho bug se puede realizar instantáneamente en todas las ramas del proyecto. Si la corrección del bug se hubiera realizado dentro de una mega-actualización, sería muy difícil identificar qué cambios son los que corrigen el bug.

Capítulo 6

Mantenimiento del Subversion

Por motivos desconocidos, los repositorios del Subversion se corrompen de vez en cuando dando mensajes de error al intentar manejar un repositorio.

La solución es simple, pero hay que seguir los siguientes pasos para no estropear nada:

1. Apagar el servicio del apache.

   ```
   rcapache2 stop
   ```

2. Recuperar el repositorio.

   ```
   svnadmin recover <path_to_repository>
   ```

3. Encender el servicio del apache.

   ```
   rcapache2 start
   ```

Depende de con qué usuario se haga esto, los permisos del repositorio pueden cambiar. En este caso, hay que volver a asignar los permisos correctos al repositorio.

```
chown -R wwwrun:www <path_to_repository>
```

Capítulo 7

Utilidades para Subversion

Subversion tiene interfaces de usuario para todos los gustos y colores: desde extensiones del explorador de windows a plug-ins para IDEs. Un listado completo de herramientas para Subversion se encuentra en:

http://subversion.tigris.org/project_links.html.

Yo he probado en Windows el TortoiseSVN (http://tortoisesvn.tigris.org/), que es una extensión del Explorer y va muy bien.

Capítulo 8

Fuentes de información

- Página oficial:
  http://subversion.tigris.org/
- Libro online sobre Subversion:
  http://svnbook.red-bean.com/
- Comparativa entre sistemas de control de versiones:
  http://better-scm.berlios.de/comparison/