

Subversion personal

© Rafaël Garcia-Suarez (<http://rgarciasuarez.free.fr>), 31 de octubre de 2002
Publicado en *The O'Reilly Network* (<http://www.oreillynet.com/pub/a/onlamp/2002/10/31/subversion.html>)

Traducción: Quique (<http://sindominio.net/quique>), Enero de 2003.

Verbatim copying and distribution of this entire article is permitted in any medium, provided this notice is preserved.

Se permite la distribución y copia literal de este artículo en su totalidad por cualquier medio, siempre que se conserve esta nota.

Subversion es un sistema de control de versiones de código abierto, de propósitos similares al bien conocido, ampliamente extendido, y obsolecente CVS. Está diseñado para proporcionar un sofisticado sistema de control de versiones, desarrollado con tecnología moderna.

Subversion está todavía en desarrollo y no ha llegado aún a la versión 1.0. Sin embargo, es bastante estable y ya se puede usar. En este artículo cubriremos los aspectos básicos de Subversion, como instalarlo, y como usar Subversion para proyectos personales. En un futuro artículo veremos como instalar y usar un servidor Subversion multiusuario en red.

- ¿Para qué sirve Subversion?
- Instalación de Subversion
- Creación de un repositorio
- Obtención de una copia de trabajo
- Órdenes básicas
- Examinando sus cambios
- Etiquetas y ramas
- Distribución
- Cosas que recordar...
- Consejos y enlaces

¿Para qué sirve Subversion?

Brevemente, Subversion ayuda a que los desarrolladores lleven un seguimiento de los cambios en los ficheros de código fuente de su proyecto. Quizá se pregunte porqué iba a necesitar un sistema de control de revisiones para sus proyectos domésticos, donde usted es la única persona que decide cómo y cuándo realizar cambios. Hay varias razones: para obtener y comparar versiones anteriores, cazar errores regresivos, mantener ramas compatibles con las versiones anteriores, producir excelentes registros de cambios (*changelogs*), trabajar sobre dos arreglos o mejoras diferentes sin

confusiones. Además, conseguirá todo esto con poco esfuerzo, porque Subversion es sorprendentemente fácil de instalar.

Un *repositorio Subversion* se comporta como un sistema de ficheros que recuerda conjuntos de cambios que se le han hecho. Esto lo hace almacenando ficheros en una estructura de árbol, llevando un control de su evolución a lo largo del tiempo. El repositorio incrementa un número global de revisión con cada conjunto de cambios *enviados (committed)* al repositorio. Como la totalidad del árbol está versionada, actúa como un sistema de ficheros normal. Es posible copiar y renombrar ficheros; crear una rama del proyecto es tan fácil como copiar un directorio. También se le puede pedir a Subversion que produzca una salida con las *diferencias* entre dos revisiones arbitrarias, o que recupere algún sub-árbol de la revisión *N*.

Instalación de Subversion

Esta sección cubre la instalación de Subversion en sistemas de tipo Unix. (También es posible compilar e instalar Subversion en Windows. Lea el fichero *INSTALL* de la distribución de Subversion). Para instalar Subversion, quizá tenga que actualizar (o instalar) algunas de las herramientas de su sistema (*autoconf, libtool, python2*). Necesitará también la biblioteca de análisis sintáctico de XML *expat*. Para información detallada, lea la sección *BUILD REQUIREMENTS* del fichero *INSTALL*. Subversion está totalmente construido con componentes de código abierto.

Subversion requiere también una versión reciente de Berkeley DB. Asegúrese de echar un vistazo a los ficheros *README* e *INSTALL* para estar seguro de que tiene la versión correcta. (En el momento de escribir esto, servía Berkeley DB 4.0.14). Subversion usa esta base de datos como el almacenamiento subyacente de sus repositorios. La puede conseguir en **Sleepycat Software**.

Las *instantáneas (snapshots)* de Subversion, disponibles en el **sitio principal de subversion**, incluyen todas las otras bibliotecas necesarias para instalar un repositorio local. Montar un servidor Subversion accesible a través de la red requiere Apache 2, pero eso es otro artículo. El fichero *INSTALL* explica también como obtener un nuevo Subversion 'fresco' del repositorio (sí, los desarrolladores de Subversion usan su propio software), pero eso no es necesario en absoluto, pues Subversion se está volviendo cada vez más estable. Nosotros usaremos una instantánea. En el momento de escribir esto, el número de versión de la última instantánea es el 3578 (también conocida como Subversion 0.14.5), pero llamémoslo *XXXX*.

El desarrollo de Subversion es rápido. Para hacer más fáciles las actualizaciones, instalaremos Subversion en su propio subdirectorio. También daremos por supuesto que usted necesita instalar la versión apropiada de Berkeley DB. Las órdenes precedidas por una almohadilla (#) deben ejecutarse como root. (Si no tiene acceso a root, instale Subversion en su directorio home en vez de usar */usr/local* como se hace en los ejemplos posteriores).

```
# mkdir /usr/local/subversion-rXXXX
# ln -s /usr/local/subversion-rXXXX /usr/local/subversion

$ gunzip -c db-4.0.14.tar.gz | tar xf -
$ cd db-4.0.14/build_unix
```

```
$ ../dist/configure --prefix=/usr/local/subversion-rXXXX
$ make
```

```
# make install
```

Asegúrese de que su sistema puede encontrar las bibliotecas en `/usr/local/subversion/lib`. Generalmente esto se consigue estableciendo el equivalente en su sistema a la variable de entorno `LD_LIBRARY_PATH`. En GNU/Linux, también puede añadir esta ruta a su fichero `/etc/ld.so.conf` y ejecutar `/sbin/ldconfig`. Después, compile Subversion:

```
$ gunzip -c subversion-rXXXX.tar.gz | tar xf -
$ cd subversion-rXXXX
$ ./configure --with-berkeley-db=/usr/local/subversion-rXXXX \
  --prefix=/usr/local/subversion-rXXXX
$ make
$ make check          # opcional: ejecuta las comprobaciones

# make install
```

Finalmente, añade `/usr/local/subversion/bin` a su `PATH`. ¡Eso es todo!

Creación de un repositorio

El siguiente paso es crear un repositorio para almacenar sus ficheros. Yo voy a poner mi repositorio en `/home/rafael/svn`, pues tengo mucho espacio libre en esta partición.

```
$ cd /home/rafael
$ svnadmin create svn
```

Ahora tengo un directorio, `/home/rafael/svn`, que contiene mi repositorio. No contiene ningún fichero y el número de revisión es 0. Solucionaré eso importando un árbol con ficheros fuente.

Supongamos que estoy trabajando en un *manipulador* (*frobnizer*). Para crear un directorio `frobnizer` en el nivel raíz del repositorio, e importar el contenido de mi árbol `/home/rafael/frobnizer`, la orden apropiada es:

```
$ svn import file:///home/rafael/svn /home/rafael/frobnizer
frobnizer
```

¿Por qué el URI en la orden anterior? Las pocas órdenes de `svn` que necesitan tratar directamente con un repositorio hacen referencia a él mediante un URL. Los URL `file://` hacen referencia a repositorios que se encuentran en un disco local. Subversion soporta también los URI `http://` y `https://` para servidores Subversion remotos, funcionando con Apache.

De hecho, para organizar mejor su repositorio, es preferible crear un directorio para su proyecto, importando sus ficheros a un subdirectorio *truncal*. Veremos más tarde las razones concretas para ello. En lugar de la orden anterior, utilice:

```
$ svn mkdir file:///home/rafael/svn/frobnizer -m 'Crear el proyecto
frobnizer'
$ svn import file:///home/rafael/svn /home/rafael/frobnizer \
    frobnizer/trunk -m 'Initial import of frobnizer project'
```

La primera orden, `mkdir`, crea un subdirectorio vacío, *frobnizer*. También etiqueta este cambio con el mensaje para el registro *Crear el proyecto frobnizer*. La segunda orden realiza la importación como tal, añadiendo su propio mensaje al registro.

Puede crear tantos repositorios como guste. Si trabaja en varios proyectos diferentes, no relacionados entre sí, puede querer repositorios independientes para cada uno de ellos. Esto le permitirá manejarlos y moverlos por separado más adelante.

Obtención de una copia de trabajo

No se trabaja directamente en el repositorio. Para hacer cambios en sus ficheros, debe hacer una copia de trabajo del repositorio completo o de uno de sus subdirectorios. Para hacer esto, use `checkout`, que se ejecuta aquí en un nuevo directorio *frobwork*:

```
$ svn checkout file:///home/rafael/svn/frobnizer
~/frobwork
```

Esta copia de trabajo contiene todos los ficheros que ha recuperado, y ahora puede editarlos con seguridad. Los subdirectorios ocultos `.svn` también contendrán datos sobre el estado en el lado del cliente. Estos ficheros permiten a Subversion realizar algunas operaciones sin tratar directamente con el repositorio. Esto nos permite trabajar desconectados incluso si el repositorio está normalmente accesible por la red.

Órdenes básicas

La mayoría de las órdenes de `svn` no usan el URI del repositorio, sino que actúan sobre la copia de trabajo local. La forma más general de una orden de `svn` es:

```
$ svn <orden> [<opciones>]
[<objetivos>]
```

donde `objetivos` es la lista de ficheros o directorios sobre los que operar, y `opciones` es una lista de parámetros, en el estilo habitual de la mayoría de los interfaces de línea de órdenes de Unix. En la mayoría de las órdenes de `svn`, `objetivos` es por defecto el directorio actual, y la orden opera recursivamente sobre cada directorio que procesa.

La primera orden que veremos es `svn commit` (también conocida como `svn ci`, que debería resultar familiar a los usuarios de CVS). Ésta envía las modificaciones de los ficheros locales al repositorio. Por ejemplo,

```
$ svn commit *.c include
```

comprobará recursivamente si han sido modificados todos los ficheros `.c` en el directorio actual y todos los ficheros en el subdirectorio *include*, y los incorporará al repositorio. Sólo se incluirán en el conjunto de cambios los ficheros que hayan sido modificados.

Como consecuencia de esto, un simple

```
$ svn commit
```

envía todos los ficheros modificados que haya en el directorio actual y en sus subdirectorios.

Tenga en cuenta que Subversion necesita un mensaje para el cuaderno de bitácora cuando se envían cambios. Puede indicar el mensaje mediante la opción `-m` en la línea de órdenes, como en los ejemplos anteriores de `import` y `mkdir`. Si falta esta opción, `svn` lanzará el editor indicado en una de las variables de entorno `SVN_EDITOR`, `VISUAL`, o `EDITOR`.

Los mensajes del cuaderno de bitácora se pueden obtener mediante la orden `svn log`, que muestra una lista formateada de los cambios en los objetivos indicados. Acepta la opción `-v` (*verbose*, prolijo), que incluye en el fichero de cambios la lista de ficheros que se modificaron en cada revisión. La usual opción `-r` limita la lista a una revisión o un intervalo de revisiones. Por ejemplo, actualizar un fichero *ChangeLog* con todos los cambios desde la revisión #42 es tan simple como:

```
$ cd ~/frobwork/tronco
$ svn log -r42:HEAD >> ChangeLog
$ svn commit ChangeLog -m 'Actualizar fichero de cambios'
```

Aquí, `HEAD` es una palabra clave que se puede usar en lugar de un número de revisión. Hace referencia al número de la última revisión que se envió al repositorio.

Si no está satisfecho con sus cambios, la orden `svn revert` deshará todos las ediciones locales que haya realizado a sus objetivos. A diferencia de otras órdenes de `svn`, `revert` no opera recursivamente sobre subdirectorios, a menos que le pase la opción `-R`.

Para añadir o eliminar ficheros, use las órdenes `svn add` y `svn remove`. Estas órdenes marcan los ficheros o directorios a añadir o eliminar en y desde el repositorio en el siguiente `commit` (envío). De manera similar, `svn mkdir` crea un nuevo directorio bajo control de versiones en la copia de trabajo.

Como regla general, el conjunto de cambios incluídos en un `commit` debería tratar un problema específico o implementar una funcionalidad bien definida. Pensar el mensaje para el cuaderno de bitácora antes de haer los cambios le puede ayudar a centrarse. Un `commit` con un propósito único se examina más fácilmente (como un `diff`), de modo que cualquier *bug* (gazapo o error de programación) que se haya podido introducir aparece más claramente. Controlar regresiones es más difícil si en un único `commit` hay varios cambios no relacionados entre sí.

Examinando sus cambios

En algún momento del desarrollo, probablemente se haya preguntado *¿qué ficheros he modificado, y qué cambios hice?* Las órdenes `svn status` y `svn diff` dan las respuestas.

`svn status` rastrea los objetivos indicados de su copia de trabajo buscando fichero que hayan cambiado desde la última descarga o `commit`. Se muestra el estado de cada fichero: modificado, añadido, borrado, o desconocido para el sistema de control de versiones.

`svn diff` produce un fichero `diff` aplicable por el programa `patch(1)`. Por defecto, usa el formato `diff` unificado. Es posible pasar opciones adicionales a la utilidad `diff(1)` subyacente por medio del modificador `-x`. Por defecto, compara la copia de trabajo actual con la última revisión descargada, pero también puede producir un parche entre dos revisiones arbitrarias del repositorio, o entre su copia de trabajo y una revisión anterior con el modificador `-r`:

```
$ svn diff -r14:18 *.c *.h > /tmp/r14-to-r18.patch
```

Etiquetas y ramas

Una manera más sencilla de recuperar cualquier situación anterior de un fichero presente en su copia de trabajo es usar `svn update`. Esta orden actualiza sus objetivos a su situación en la revisión indicada (por defecto a la revisión `HEAD`). Por ejemplo,

```
$ svn update -r42 include
```

devuelve el directorio *include* (y sus contenidos) a la revisión 42. (También se usa `update` cuando se quiere integrar cambios de otra copia de trabajo en la copia de trabajo propia, pero cubriremos esto en otro artículo). Subversion proporciona un mecanismo más potente para controlar los puntos de publicación de sus proyectos: *etiquetas y ramas*. Desde el punto de vista de Subversion, no hay diferencia entre etiquetas y ramas; la diferencia está en como las usaremos.

Para crear una etiqueta o una rama, use la orden `svn copy`. `svn copy` crea una copia del recurso versionado (un fichero o un directorio), al tiempo que conserva su historia. En otras palabras, no añade un fichero (o un árbol entero) - solamente marca una nueva entrada en el repositorio como idéntica a otra, *en una revisión indicada*. Así, las copias se realizan en tiempo constante, aunque lógicamente pueden crear enormes árboles paralelos.

Veamos un ejemplo. Supongamos que acabo de publicar un nuevo `frobnizer`, versión 2.0, y que quiero que Subversion recuerde por mí que `frobnizer v2` corresponde a la revisión 42. Suponiendo que el actual nivel de revisión del directorio *tronco* en mi copia de trabajo es 42, diré:

```
$ cd ~/frobwork
$ svn copy trunk frobnizer-2.0
$ svn commit -m 'Versión v2.0'
```

Mi repositorio tiene ahora dos subdirectorios principales: */frobnizer/tronco* y */frobnizer/frobnizer-2.0*. Parten de la misma revisión, pero ahora cada rama puede evolucionar por separado.

Si decido no enviar ningún cambio al subdirectorio */frobnizer/frobnizer-2.0*, se le llama una **etiqueta**. De lo contrario, es una **rama**. La ventaja de una etiqueta es obvia. Es

posible descargar `/frobner/frobner-2.0` para obtener el código fuente de `frobner v2`.

Las ramas son útiles principalmente cuando un equipo de desarrolladores tiene permisos para enviar cambios a un mismo repositorio. Si se tiene que hacer un gran desarrollo experimental, es mejor crear una rama, hacer los cambios en ella, y reintegrarlo al tronco cuando se haya acabado el desarrollo. Esto evita interferencias. Subversion proporciona un mecanismo (la orden `svn merge` para informar de los cambios hecho en una rama a otra. Puede que las ramas no sean muy útiles si usted es el único usuario de Subversion en su sistema, y esa es la razón por la que no me voy a extender más sobre este asunto. Sin embargo, los desarrolladores de Subversion recomiendan crear dos subdirectorios, *ramas/* y *etiquetas/*, junto a *tronco*, para almacenar ramas y etiquetas. (Así es como está organizado el **repositorio de Subversion**).

Distribución

Ahora que ya ha solucionado muchos errores de programación, ha decidido publicar otra versión de su proyecto como un *tarball* de código fuente.

No puede simplemente usar `tar+gzip` sobre su copia de trabajo: ésta contiene ficheros ocultos de Subversion y probablemente algunos otros ficheros temporales (copias de respaldo, ficheros objeto, etc). La orden `svn export` está aquí para ayudarle. Descarga una versión indicada sin incluir metadatos.

```
$ svn export file:///home/rafael/svn/frobner/frobner-2.0
$ tar cf frobner-2.0.tar frobner-2.0
$ gzip -9 frobner-2.0.tar
```

Cosas que recordar...

¡No olvide hacer copias de respaldo! Su repositorio guarda información importante. Es posible volcar el contenido del repositorio (incluyendo todas las revisiones) en fichero de volcado (*dumpfile*). Este formato es portable, y podría usarlo para reconstruir un repositorio si fuese necesario.

Por ejemplo, es útil tener, en un `crontab`, algo como:

```
svnadmin dump /home/rafael/svn | gzip -9 > dump.gz
```

De esta manera, se puede restaurar el repositorio mediante

```
gunzip -c dump.gz | svnadmin load /home/rafael/svn
```

Consejos y enlaces

La sub-orden más útil de `svn`, al menos para los principiantes, es `help`. `svn help` proporciona un sumario de todas las sub-órdenes disponibles, y `svn help foo` describe la sub-orden `foo`. Lea también las páginas de manual de `svn(1)` y

svnadmin(1). svnadmin es para órdenes administrativas.

La **página de Subversion** contiene una gran cantidad de información. El **li bro de Subversion**, en construcción, también es útil, y contiene muchísima más información de la que es poner en un pequeño artículo.

Rafaël Garcia-Suarez es un ingeniero de software y administrador de sistemas Unix francés.