There are no translations available.
Repasa en este artículo los conceptos más importantes de los sistemas de control de versiones en general, y descubre también las principales características particulares de subversion

# Sistema de control de versiones: SUBVERSION

## 1. Introducción

El artículo pretende, por una parte, repasar los conceptos más importantes de los sistemas de control de versiones en general, así como mostrar las principales características particulares de subversion. Por otro lado, y con un carácter mucho más práctico, aprenderemos como instalar y configurar un servidor svn en Ubuntu.

La práctica descrita se ha realizado concretamente con Ubuntu 7.04, pero el proceso sería muy similar con otras versiones de Ubuntu o Debian.

Dada la limitada extensión del artículo, no es posible describir con detalle tanto la instalación como el uso de subversion de manera que nos hemos centrado únicamente en la instalación y configuración. Sin duda, se necesitaría otro artículo completo para poder describir adecuadamente la operación de los distintos clientes gráficos y de línea de comandos disponibles tanto para GNU Linux como para Windows.

## 2. Sistemas de control de versiones

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal características es que mantiene la historia de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema es capaz de recordar las versiones antiguas de los datos, lo que nos permite examinar el histórico de cambios o recuperar versiones anteriores de un fichero, incluso aunque haya sido borrado.

Los sistemas de control de versiones son ampliamente utilizados en los proyectos de desarrollo de software, para mantener las versiones del código fuente. No obstante, su aplicación no está limitada a esta actividad, sino que permiten gestionar documentos, imágenes y ficheros de todo tipo.

Antes de entrar en detalles específicos de Subversion, es importante familiarizarse con las características y conceptos más importantes que son comunes a los sistemas de control de versiones. Además de la descripción o la traducción, en la mayoría de casos se han incluido los nombres en inglés, ya que suelen utilizarse así, tanto en la documentación como en muchos de los programas.

En primer lugar, los sistemas de control de versiones utilizan para su funcionamiento algún mecanismo de almacenamiento de los datos y la información asociada (metadatos). Esta base de datos o almacén, suele denominarse **repositorio**. Los sistemas de control de versiones más populares funcionan con un repositorio centralizado, es decir, aunque permiten el trabajo colaborativo entre varios puestos de trabajo, mantienen el repositorio centralizado en único ordenador, estando accesible para el resto de equipos a través de red local o Internet.

Cada cliente del sistema, dispone de su propia **copia local de trabajo**, que puede ser examinada y/o modificada a voluntad.

Un **cambio** es cada una de las modificaciones realizadas en alguno de los documentos bajo el control de versiones. Muchos sistemas permiten agrupar múltiples cambios en una sola operación de escritura o actualización del repositorio. Cada uno de esos conjuntos de cambios agrupados constituyen lo que se conoce como una **lista de cambios (changeset)**.

Una **revisión** sería cada una de las versiones disponibles en el repositorio. Este término se utiliza refiriéndose tanto a ficheros individuales como a conjuntos de ficheros o incluso al repositorio entero. Suelen identificarse mediante un número (**número de revisión**).

Muchos sistemas permiten definir **etiquetas (tags)** para referirse a una revisión determinada de cierto conjunto de ficheros. La etiqueta permite asignar un nombre fácil de recordar o significativo de manera colectiva a varios ficheros a la vez (por ejemplo, a todos los archivos correspondientes a una misma versión publicada de un programa).

Las **ramas (branches)** permiten que a partir de cierto punto, un mismo fichero o conjunto de

ficheros, sea desarrollado de dos formas diferentes, manteniendo sus respectivas revisiones independientemente.

Al proceso de actualización o escritura de los ficheros bajo control de versiones, se le denomina **integración**. Esta palabra se utiliza tanto para indicar la escritura en el repositorio de los cambios realizados en la copia de trabajo, como a la actualización de la copia local desde el repositorio para incluir los cambios realizados por otros clientes.

En un proceso de integración pueden producirse **conflictos**, en el caso de que dos o más clientes hayan realizado cambios de manera independiente sobre un mismo documento. El sistema puede ser capaz de manejar algunos de estos conflictos, según el caso, pero en general, el proceso de **resolución** (conciliación) de los mismos suele requerir la intervención del usuario.

Respecto de las operaciones más comunes que un cliente puede realizar dentro del sistema de control de versiones, tenemos:

-

Posibilidad de añadir, borrar, crear, modificar, mover, etc ... cada uno de los elementos (ficheros y directorios) bajo control de versiones.

-

 **check-out"**: Permite obtener una copia local de trabajo (correspondiente a la última revisión o bien a otra anterior) que puede ser examinada y/o modificada por el cliente.

-

 **check-in** (o commit): es la operación mediante la que se integran en el repositorio los cambios realizados en la copia local.

-

 **update** (o sync ): mediante esta opción, integramos en nuestra copia local actual los cambios que otros usuarios han consolidado en el repositorio.

## 2. Características de subversion

Subversion, también conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red, y se distribuye bajo una licencia libre de tipo Apache.

SVN surge con la intención de sustituir y mejorar al conocido CVS (Concurrent Versions System). CVS, pese a sus limitaciones, constituyó el estándar de facto de los sistemas de gestión de versiones en el ámbito del software libre (aunque  sólo por que no había otra cosa ;-), según algunos). SVN mantiene las ideas fundamentales de CVS pero suple sus carencias y evita sus errores. Al parecer, incluso el nombre ( subversion ) fue fruto de la frustación que producía a sus creadores el trabajo diario con CVS.

Las principales características de SVN y sus mejoras frente a CVS son:

-

mantiene versiones no sólo de archivos, sino también de directorios

-

también se mantienen versiones de los metadatos asociados a los directorios.

-

además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.

-

atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.

-

posibilidad de elegir el protocolo de red. Además de un protocolo propio (svn), puede trabajar sobre http (o https) mediante las extensiones WebDAV. WebDAV (más conocido como DAV) es un protocolo que amplia las posibilidades del HTTP/1.1 añadiendo nuevos métodos y cabeceras. La capacidad de funcionar con un protocolo tan universal como el http simplifica la implantación (cualquier infraestructura de red actual soporta dicho protocolo) y universaliza las posibilidades de acceso (si se quiere, puede utilizarse a través de Internet).

-

soporte tanto de ficheros de texto como de binarios.

-

mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.

-

mayor eficiencia en la creación de ramas y etiquetas que en CVS.

## 3. Instalación de un servidor svn

En este apartado, mostraremos paso a paso como instalar y configurar un servidor svn funcionando sobre http con Apache2. Esta configuración nos proporciona la ventaja de poder hojear el repositorio desde cualquier navegador web y, además, nos brinda la posibilidad de abrir el acceso al repositorio incluso a través de Internet.

Necesitamos instalar los siguientes paquetes:

```
# apt-get install apache2 subversion subversion-tools libapache2-svn
```

Mantendremos la ubicación por defecto para el directorio principal del svn (/var/lib/svn). Como ejemplo, dentro de este directorio crearemos un repositorio denominado 'programas'. Para ello, creamos primero el subdirectorio, y luego generamos el repositorio indicando dicha ruta:

```
# mkdir -p /var/lib/svn/programas
```

```
# svnadmin create /var/lib/svn/programas
```

Para que Apache pueda acceder al directorio, cambiaremos el propietario para que sea el usuario www-data y asignaremos permisos al directorio:

```
# chown -R www-data.www-data /var/lib/svn
```

```
# chmod -R g+ws /var/lib/svn
```

También es necesario activar y configurar el módulo 'dav'. Para ello, como superusuario, tendremos que editar el fichero:

```
/etc/apache2/mods-enabled/dav_svn.conf
```

Para activar el repositorio en el directorio /var/lib/svn, tenemos que descomentar las siguientes líneas:

```
DAV svn
```

SVNParentPath /var/lib/svn

Tenemos varias posibilidades para los permisos de acceso. No obstante, en todos los casos, para poder establecer distintas restricciones según el usuario necesitamos primero activar la autenticación en Apache, que se consigue descomentando las opciones:

AuthType Basic

AuthName "Repositorio Programas"

AuthUserFile /etc/apache2/dav_svn.passwd

(la directiva AuthName es informativa y puede usarse cualquier texto, pero debe tenerse en cuenta que algunos navegadores la muestran cuando piden las credenciales al usuario)

La configuración actual establece el mecanismo mediante el que Apache2 recuperará usuarios y contraseñas cuando se necesite autenticación, pero todavía no hemos especificado cuando debe pedirse esta autorización. El ajuste más sencillo consiste en proteger todas las peticiones. Para ello descomentamos la opción:

Require valid-user

Con esto, cualquier acceso al repositorio, incluidas las operaciones de lectura, exigen la autenticación previa mediante usuario y contraseña.

En ocasiones, puede resultar útil permitir el acceso anónimo (sin autenticación) para ciertas operaciones proteger tan sólo otras. Esta configuración se consigue mediante las directivas  Limit y  LimitExcept . Los parámetro presentes en estas directivas son las peticiones HTTP que se ven afectadas por dicho bloque (en el caso de Limit) o bien todas excepto las indicadas (LimitExcept). Su uso requiere conocimientos sobre el protocolo DAV, pero expondremos un ejemplo de considerable interés práctico.

Supongamos que queremos permitir el acceso anónimo para las operaciones de lectura (GET, PROPFIND, OPTIONS y REPORT) pero permitir la escritura sólo a usuarios autenticados. Para conseguirlo, escribiremos:

Require valid-user

Es importante observar que la directiva Require valid-user queda DENTRO del bloque  LimitExcept .

Por último, a nuestro fichero sólo le falta cerrar el bloque  Location descomentando (al final del mismo) la línea:

Para evitar confusiones, incluimos aquí el contenido completo del fichero /etc/apache2/mods-enabled/dav_svn.conf:

```
# dav_svn.conf - Example Subversion/Apache configuration

#

# For details and further options see the Apache user manual and

# the Subversion book.

#

# NOTE: for a setup with multiple vhosts, you will want to do this

# configuration in /etc/apache2/sites-available/*, not here.

# ...

# URL controls how the repository appears to the outside world.
```

# In this example clients access the repository as http://hostname/svn/

# Note, a literal /svn should NOT exist in your document root.

# Uncomment this to enable the repository

DAV svn

# Set this to the path to your repository

#SVNPath /var/lib/svn

# Alternatively, use SVNParentPath if you have multiple repositories under

# under a single directory (/var/lib/svn/repo1, /var/lib/svn/repo2, ...).

# You need either SVNPath and SVNParentPath, but not both.

SVNParentPath /var/lib/svn

# Access control is done at 3 levels: (1) Apache authentication, via

# any of several methods. A "Basic Auth" section is commented out

# below. (2) Apache  and  , also commented out

# below. (3) mod_authz_svn is a svn-specific authorization module

# which offers fine-grained read/write access control for paths

# within a repository. (The first two layers are coarse-grained; you

# can o-nly enable/disable access to an entire repository.) Note that

# mod_authz_svn is noticeably slower than the other two layers, so if

# you don't need the fine-grained control, don't configure it.

```
# Basic Authentication is repository-wide. It is not secure unless

# you are using https. See the 'htpasswd' command to create and

# manage the password file - and the documentation for the

# 'auth_basic' and 'authn_file' modules, which you will need for this

# (enable them with 'a2enmod').

AuthType Basic

AuthName "Repositorio Programas"

AuthUserFile /etc/apache2/dav_svn.passwd

# To enable authorization via mod_authz_svn
```

#AuthzSVNAccessFile /etc/apache2/dav_svn.authz

# The following three lines allow anonymous read, but make

# committers authenticate themselves. It requires the 'authz_user'

# module (enable it with 'a2enmod').

Require valid-user

NOTA: Existe la posibilidad de realizar una configuración más avanzada mediante el modulo mod_authz_svn de Apache2. Este módulo permite especificar permisos individuales directorio a directorio, pero afecta muy negativamente a la velocidad de acceso. Dado que se recomienda no usarlo a menos que sea imprescindible, no hemos incluido ningún ejemplo con él.

Una vez configurado DAV, ejecutaremos el siguiente comando para verificar que los siguientes módulos están activados (en Ubuntu lo están por defecto):

```
# a2enmod auth_basic authn_file authz_user
```

Por último, reiniciaremos apache2 con:

```
# /etc/init.d/apache2 restart
```

Si no hemos cometido errores, podremos acceder ya al repositorio usando un navegador web mediante la url:

http://localhost/svn/programas/

Para gestionar usuarios, disponemos de la utilidad htpasswd. Debemos indicarle la ruta del fichero que hemos puesto en la directiva AuthUserFile y el nombre de usuario. El programa es interactivo, y se encargará de pedirnos la contraseña a utilizar.

Por ejemplo, para asignar password al usuario USUARIO, escribiríamos:

```
# htpasswd -c /etc/apache2/dav_svn.passwd USUARIO
```

Aunque no es imprescindible, resulta útil utilizar en svn el mismo nombre de usuario que usamos para acceder al sistema, ya que de esta forma, para validarnos desde la línea de comandos, sólo tendremos que indicar el password.

Por último, y para que la estructura del repositorio esté completa, debemos crear los subdirectorios 'trunk' 'tags' y 'branches'. Para ello ejecutaremos los siguientes comandos (desde nuestra propia cuenta, sin necesidad de convertirnos en root):

```
$ svn mkdir http://localhost/svn/programas/trunk -m "trunk directory"
```

```
$ svn mkdir http://localhost/svn/programas/tags -m "tags directory"
```

```
$ svn mkdir http://localhost/svn/programas/branches -m "branches directory"
```

NOTA: En el primer comando, se nos pedirá la contraseña que hemos asignado con htpasswd. Es **muy importante** conocer que después quedará almacenada (en claro) dentro de nuestro home directory, bajo el directorio

```
~/.subversion/auth
```

Con estos pasos, el repositorio ya es operativo, y podría descargarse una copia de trabajo local desde cualquier equipo de la red con un comando similar a:

```
$ svn co http://DIRECCION_IP_SERVIDOR/svn/programas
```

Sin embargo, no podemos dar por terminada la configuración en este punto, dado que la autenticación a través del protocolo http es altamente insegura (se envía la contraseña sin ningún tipo de encriptación). Si pretendemos acceder desde otros ordenadores de la red, deberíamos configurar nuestro servidor Apache2 para usar SSL (https).

NOTA: Esta es una cuestión exclusivamente de configuración de Apache que no está estrictamente relacionada con los sistemas de control de versiones. No obstante, dada la importancia que tiene sobre la seguridad del nuestro sistema, y pensando en aquellos lectores que no estén familiarizados con SSL, indicaremos los pasos a seguir (casi a modo de receta) para configurar conexiones seguras en Apache2.

El protocolo SSL (Secure Sockets Layer) se desarrolló para proporcionar tanto autenticación como seguridad de datos. Encapsula la conexión TCP/IP por lo que puede ser usado prácticamente por cualquier aplicación para establecer conexiones seguras de forma sencilla y transparente.

La descripción detallada de las características de SSL queda evidentemente fuera del alcance de este artículo, por lo que nos centraremos en la aplicación concreta sobre el protocolo http.

El protocolo https consiste en utilizar SSL para cifrar conexiones http, y debe utilizarse siempre que tengamos que transmitir algún tipo de información sensible (como las contraseñas) por http.

Para configurar Apache2 con SSL es imprescindible disponer de un certificado digital X.509 para nuestro servidor. Estos certificados son emitidos por una Autoridad de certificación (CA) y cumplen una doble función:

-

garantizan nuestra identidad, ya que incluyen los datos del titular del certificado junto a la firma digital de la CA.

-

nos proporcionan la infraestructura de clave pública (PKI) necesaria para SSL y otras operaciones de cifrado y criptografía asimétrica (por ejemplo, firmar digitalmente nuestros mensajes).

La obtención de dicho certificado supone su tramitación con la correspondiente CA y, en la mayoría de casos, el pago a la misma de los servicios prestados.

Por suerte, si nuestra única necesidad es la segunda de estas dos funciones, es decir, la encriptación de las comunicaciones mediante SSL, existe también la posibilidad de generar nosotros mismos el certificado. Es algo así como convertirnos en nuestra propia CA. Desde el punto de vista técnico, estos certificados autofirmados constituyen una solución para uso privado perfectamente válida para asegurar la confidencialidad de las transmisiones. Sin embargo, desde el punto de vista de los clientes de red, no representa ninguna garantía sobre la verdadera identidad de nuestro servidor, ya que no cuentan con el respaldo y la credibilidad de una verdadera CA. Veamos como generar no de estos certificados.

La implementación de referencia de SSL en Linux es OpenSSL, y es la que utilizaremos para nuestro certificado. Primero comprobaremos que la tenemos instalada con:

```
# apt-get install openssl
```

Si buscamos en Internet, encontraremos que la mayoría de guías para configurar SSL en Apache, utilizan un script denominado 'apache2-ssl-certificate' que suele distribuirse con Apache2 y que simplifica el proceso. Por error, en Ubuntu feisty no se ha incluido este script. No es un problema grave, simplemente tendremos que usar un comando notablemente más largo para generar el certificado:

# openssl req -x509 -nodes -days 365 -newkey rsa:1024 -out /etc/apache2/server.crt -keyout /etc/apache2/server.key

Los parámetros más importantes de este comado son:

-days [número _días]: Validez del certificado (en el ejemplo, un año)

-keyout [ruta]: ruta del fichero donde se guardará la clave privada

-out: ruta donde se generará el certificado

Se nos formularán diversas preguntas sobre nuestra ubicación (país, ciudad, empresa, ...). Podemos contestar lo que queramos, especialmente para un servidor de uso interno o doméstico. La pregunta más importante es:

Common Name (eg, YOUR name) [ ]:

Aquí deberíamos indicar el nombre correcto del servidor, (o en su defecto la dirección IP).

Una vez generado el certificado, protegeremos nuestra clave privada con:

# sudo chmod 440 /etc/apache2/server.key

A continuación, crearemos un nuevo host virtual en Apache para el acceso seguro al svn. Primero tendremos que crear un fichero de configuración nuevo (como root) en:

/etc/apache2/sites-available/svn

Podemos configurarlo de distintas formas, pero una posibilidad sencilla podría ser:

NameVirtualHost *:443

ServerName localhost

DocumentRoot /var/www

SSLEngine o-n

SSLCertificateFile /etc/apache2/server.crt

SSLCertificateKeyFile /etc/apache2/server.key

Ya casi hemos terminado. Sólo nos queda activar el módulo SSL con el comando:

```
# a2enmod ssl
```

```
# echo 'Listen 443' >> /etc/apache2/ports.conf
```

Y por último, activaremos el sitio virtual y reiniciaremos Apache2:

```
# a2ensite svn
```

```
# /etc/init.d/apache2 restart
```

Podemos comprobar que podemos acceder directamente al directorio raíz de nuestro svn mediante cualquiera de las urls:

https://localhost/svn/programas/

https://DIRECCION_IP_DEL_SERVIDOR/svn/programas/

(En ambos casos tendremos que aceptar el certificado)

No obstante, todavía está accesible directamente por http (sin SSL), lo que supone un enorme

riesgo. Para eliminar esta posibilidad, editaremos de nuevo el fichero:

/etc/apache2/mods-enabled/dav_svn.conf

y añadiremos la línea:

SSLRequireSSL

Un buen sitio para ponerla es justo debajo de la línea:

AuthUserFile /etc/apache2/dav_svn.passwd

Con esta configuración, ya no es posible acceder mediante http, sólo por https, lo que garantiza que las contraseñas circulan encriptadas por la red (en realidad no sólo las contraseñas, sino todas las comunicaciones con nuestro servidor svn se realizarán de manera segura).