

# Subversion

## ¿Qué es?

Control de versiones, permite tener en un servidor centralizado un histórico de todo lo que hemos ido haciendo y quién lo ha hecho.

Mantiene una copia de todo en el servidor, sirve de "backup".

Sincronización entre varias personas. Todos pueden subir o bajar.

Automáticamente fusiona los ficheros, salvo que haya conflictos.

Cada revisión tiene un número. Cada vez que se sube algo se incrementa este número y se le pone a todos los archivos subidos. Los directorios tienen el número del último archivo subido.

Estructura de árbol jerárquico:

- trunk, la última versión de cada proyecto.
- tags, versiones anteriores, ej kirando\_1\_0.
- branches, cuando queremos empezar a desarrollar algo o probar sin cargarnos el trunk.

Escenario de uso:

- El proyecto está en el servidor, te lo descargas en una copia local (checkout). La copia local se llama working copy.
- Modificas y pruebas todo lo que quieras.
- Actualizas tu copia local para asegurarte que nadie ha subido nada (update). Si había cambios compruebas que funciona todo y solucionas posibles conflictos.
- Subes (commit).

## ¿Qué se puede hacer?

- Subir un proyecto, import.
  - `svn import dirlocal urlrepositorio`
- Descargarse un proyecto, checkout.
  - `svn checkout http://servidor/trunk/miproyecto (Opcionalmente --username )`
- Añadir un fichero a un proyecto, add. (Hay que hacer el add siempre, no vale con crear el fichero).
  - `svn add README`
- Cambiar el nombre o mover, move.
  - `svn move README README2`
- Borrar un fichero, del.
  - `svn del README`
- Actualizar la copia local con la última del servidor. update.
  - `svn update`
- Visualizar los cambios de nuestra versión local respecto a la última descargada. status.
  - `svn status`
    - A significa que se va a añadir el fichero.
    - D que se va a borrar.
    - C que hay algún tipo de conflicto
    - M que se ha modificado desde la última vez

- ? Subversion no tiene constancia de ello. Debemos hacer add para añadirlo o borrar el fichero.
- Comparar dos versiones del mismo archivo. compare/diff
  - `svn -r 15:17 README` (Compara las versiones 15 y 17 del fichero README, si solo se indica una revisión compara con la copia actual.)
- Descartar los cambios actuales y volver a lo último que haya en el servidor. revert
  - `svn revert README`
- Indicar que lo que vale es lo mio, y no lo que ha hecho otro. resolved.
  - `svn resolved README`
- Ver quién ha sido el último en modificar una determinada línea del código. blame
  - `svn blame README`
    - 1 mario Hola
    - 2 alej Probando
- Ver qué le ha ido ocurriendo a un fichero. log
  - `svn log README`
    - Revisión | Quién | fecha | Número de líneas que han cambiado | Comentario.
- Crear tags para etiquetar una versión determinada. Ej, kirando\_2\_0
- Crear ramas independientes, branch
- Unir ramas, merge.

## Gestión de conflictos

Si hemos modificado dos lo mismo como lo arreglo?

svn status lo marca con una C de conflicto y crea 3 ficheros adicionales:

- Readme (Intento de fusión, marca con <<<< y === donde están los conflictos)
- Readme.mine (El mio tal cual lo tenia, working).
- Readme.r4 (La versión que yo me descargué antes de tocar nada, base)
- Readme.r6 (La versión del repositorio que me ha fastidiado, theirs).

Ante ello hay varias opciones:

1. Lo mio era lo válido:
  1. `svn resolve --accept working Readme`
2. Lo válido era lo del servidor:
  1. `svn revert Readme`
3. Tengo que mezclarlo a mano.
  1. Edito el fichero, busco los <<<< === y dejo sólo lo que corresponda.
  2. `svn resolve --accept working`

## ¿Cómo lo aplico a mi trabajo? (Buenas prácticas)

- Subir frecuentemente lo que estoy haciendo (Para que los demás lo tengan disponible y haya menos conflictos).
- Probar que lo que voy a subir funciona antes de subirlo. Para que luego cuando se lo descargue otro siempre compile.
- Poner siempre un comentario al subir algo (En consola con -m "Mensaje"). No hace falta contar la vida, pero no dejarlo vacío. Por ejemplo, "Corregido bug en la internacionalización en firefox", "Primera versión del módulo de gestión de usuarios", "Añadida validación de noseque".

## Truquillos de la abuela.

- NO usar el explorador de windows o comandos del shell para mover o copiar ficheros. Siempre usar los comandos de SVN.
- svn add, move y del afectan solo a la copia local hasta que hagas el commit.
- Por mucho que nos lo propongamos, siempre nos encontraremos con conflictos.
- Podemos decir que no suba nunca un determinado fichero o directorio con svn ignore. Esto es interesante para ficheros de log, imágenes...
- Podemos añadir propiedades a los ficheros. Por ejemplo le podemos poner un content/type para que al navegar por el repositorio via web tenga ese formato.
- Evitar subir ficheros muy grandes, imágenes, índices,
- Evitar subir ficheros autogenerados, por ejemplo un war o un zip con la aplicación, o binarios compilados de C.

## IDEs.

- En eclipse, siempre hemos usado subclipse y funciona bastante bien.
- En windows, tortoissvn.
- En mac, ¿?
- En linux, ¿?

## Ejemplo

```
En el .bashrc
export SVN_EDITOR=vim
export REP=file:///Users/mck/svnrepository/
```

Creo un repositorio:

```
$ svnadmin create /Users/mck/svnrepository --fs-type fsfs
```

Creo directorios base

```
$ svn mkdir -m "Initial structure" $REP/trunk $REP/tags $REP/branches
```

Añado un proyecto

```
$ cd
$ mkdir miproj
$ cd miproj
$ touch a b c
$ svn import miproj/ $REP/trunk/miproj    (Atención que hay que poner el nombre de carpeta!!)
```

Me descargo un proyecto

```
$ cd
$ mkdir projs
$ svn co $REP/trunk/miproj
```

Hago cambios en mi copia

```
$ cd miproj
$ echo "hola" > a
$ svn del b
$ touch d
$ svn add d
```

Veo los cambios que he hecho

```
$ svn status
```

```
M    a  
D    b  
A    d  
?    e
```

Subo los cambios

```
$ svn commit -m "Reorganization"
```

```
Sending      a  
Deleting     b  
Adding       d  
Transmitting file data ..  
Committed revision 3.
```

## Referencias:

El todopoderoso SVN book: <http://svnbook.red-bean.com/nightly/en/svn-book.html>

Una especie de introducción: [http://cesnavarra.morfeo-formacion.org/file.php/16/moddata/scorm/17/7\\_gestin\\_de\\_cdigo\\_y\\_control\\_de\\_versiones.html](http://cesnavarra.morfeo-formacion.org/file.php/16/moddata/scorm/17/7_gestin_de_cdigo_y_control_de_versiones.html)

Como instalar subclipse: <http://subclipse.tigris.org/servlets/ProjectProcess?pageID=p4wYuA>

Mario Arias  
<http://lab216.com/>