EBook Gratis

APRENDIZAJE

svn

Free unaffiliated eBook created from
Stack Overflow contributors.

#svn

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: svn

It is an unofficial and free svn ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official svn.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con svn

Observaciones

Apache Subversion (SVN) es un sistema de control de versiones de código abierto, universal y centralizado. Subversion es actualmente un proyecto bajo Apache Software Foundation (ASF) y tiene licencia bajo la Licencia Apache, Versión 2.0 .

Subversion está diseñado para administrar y controlar archivos y directorios, y hacer un seguimiento de los cambios realizados en ellos; Actúa como una *máquina de tiempo* confiable y *una herramienta de gestión* para los proyectos desarrollados en colaboración. Puede responder fácilmente a las preguntas estándar que cualquier sistema de control de versiones debe responder de manera confiable. Por ejemplo,

- ¿Cómo fue el proyecto / archivo FOO el 12/12/2012?
- ¿Qué cambios introdujo USERNAME o el 20/12/2012?
- ¿Quién modificó la cadena en particular desde la última revisión?
- y mucho, mucho más.

Versiones

Versión	¿Qué hay de nuevo?	Fecha de lanzamiento
1.9.x	Numerosas mejoras de usabilidad y rendimiento.	2015-08-05
1.8.x	Rastreo mejorado de nombres, combinaciones de reintegración automática, propiedades heredadas de la versión, herramienta de resolución de conflictos incorporada	2013-06-18
1.7.x	Reescritura completa de la biblioteca de copia de trabajo, uso mejorado del protocolo HTTP	2011-10-11
1.6.x	Identificación de conflictos de árbol, resolución de conflictos interactiva mejorada, soporte de URL relativo a la repo	2009-03-20
1.5.x	Seguimiento de fusión y ramificación (`svn:mergeinfo`), resolución de conflictos de archivos interactivos, `svn:mergeinfo` dispersas, `svn:externals` sintaxis `svn:externals`	2008-06-19
1.4.x	Herramienta `svnsync` para la replicación del repositorio, biblioteca de copia de trabajo nueva y mejorada	2006-09-10
1.3.x	Registro de alto nivel de las operaciones del usuario en el lado del servidor, mejoras de rendimiento	2005-12-30

Versión	¿Qué hay de nuevo?	Fecha de lanzamiento
1.2.x	Soporte para modelo de bloqueo-modificación-desbloqueo (es decir, bloqueo), DAV autoversioning, FSFS se utiliza de forma predeterminada para los nuevos repositorios	2005-05-21
1.1.x	Nuevo backend del repositorio (FSFS), versiones de enlaces simbólicos	2004-09-29
1.0.x	Lanzamiento público inicial	2004-02-23

> Apache Subversion 1.9.x es actualmente la última y mejor versión de SVN que es totalmente compatible. Subversion 1.8.x es parcialmente compatible. Subversion 1.7.xy versiones anteriores ya no son compatibles.

Examples

Instalación y configuración inicial

Instale el cliente `svn` para comenzar a colaborar en el proyecto que utiliza Subversion como su sistema de control de versiones.

Para instalar Subversion, puede compilarlo usted mismo a partir de una versión de código fuente o descargar un paquete binario precompilado para su sistema operativo. La lista de sitios donde puede obtener un cliente compilado de Subversion (`svn`) para varios sistemas operativos está disponible en la página oficial de paquetes binarios . Si tienes ganas de compilar el software por ti mismo, toma la fuente en la página del Código fuente .

Con Subversion, no está limitado a usar solo el cliente de línea de comandos `svn` estándar. Hay algunos clientes gráficos importantes de Subversion para varios sistemas operativos y la mayoría de los IDE en la actualidad ofrecen una integración robusta con SVN de forma inmediata o mediante complementos. Para ver la lista de clientes gráficos, consulte la página de Wikipedia: https://en.wikipedia.org/wiki/Comparison_of_Subversion_clients .

Inmediatamente después de instalar el cliente, debería poder ejecutarlo emitiendo el comando `svn` . Deberías ver lo siguiente:

```
$ svn
Type 'svn help' for usage.
```

Todo está casi listo. Ahora debe crear un espacio de trabajo local llamado *copia de trabajo* que se conectará al *repositorio* central remoto. En otras palabras, usted va a *retirar una copia de trabajo* . Con la ayuda de la copia de trabajo, operará con los datos versionados y podrá publicar sus cambios (llamados *confirmación* en SVN) para que otros que trabajen en el mismo proyecto puedan verlos y beneficiarse de sus cambios. Para recuperar posteriormente los cambios futuros realizados por otros usuarios desde el repositorio, debe *actualizar su copia de trabajo* . Estas

operaciones básicas están cubiertas en otros ejemplos.

Revisando una copia de trabajo

Para comenzar a realizar modificaciones en los datos del proyecto, debe obtener una copia local del proyecto versionado. Use el cliente de línea de comando `svn` o su cliente SVN favorito (TortoiseSVN, por ejemplo). Su copia local del proyecto se llama una *copia de trabajo* en Subversion y la obtiene al emitir el comando `svn checkout <URL>` donde `<URL>` es una URL del repositorio. p.ej

```
$ svn checkout https://svn.example.com/svn/MyRepo/MyProject/trunk
```

Alternativamente, puede usar `svn co <URL>` como una taquigrafía para sacar una copia local.

Como resultado, obtendrá una copia de trabajo de `/trunk` de un proyecto llamado MyProject que reside en el repositorio MyRepo. La copia de trabajo se ubicará en un directorio llamado `trunk` en su computadora en relación con el directorio en el que emitió el comando.

Si desea tener un nombre diferente para su copia de trabajo, puede agregarlo como parámetro al final del comando. p.ej

```
$ svn checkout https://svn.example.com/svn/MyRepo/MyProject/trunk MyProjectSource
```

Esto creará una copia de trabajo llamada `MyProjectSource` .

Tenga en cuenta que en lugar de revisar el tronco, puede revisar algunas sucursales, estanterías privadas o una etiqueta (asumiendo que ya existen en el repositorio); puede tener un número ilimitado de copias de trabajo locales en su máquina.

También puede obtener la copia de trabajo de todo el repositorio MyRepo. Pero debes abstenerte de hacerlo. En general, **no** necesita tener una copia de trabajo de todo el repositorio para su trabajo porque su copia de trabajo se puede cambiar instantáneamente a otra rama / etiqueta / lo que sea de desarrollo. Además, el repositorio de Subversion puede contener una serie de (des) proyectos relacionados y es mejor tener una copia de trabajo dedicada para cada uno de ellos, no una sola copia de trabajo para todos los proyectos.

Exportando los datos versionados (descarga plana)

Si desea obtener los datos del proyecto versionado, pero no necesita ninguna de las capacidades de control de versión que ofrece Subversion, puede ejecutar el comando `svn export <URL>` . Aquí hay un ejemplo:

```
$ svn export https://svn.example.com/svn/MyRepo/MyProject/trunk
```

Como resultado, obtendrá la exportación de datos del proyecto, pero a diferencia de una copia de trabajo, no podrá ejecutar comandos `svn` en él. La exportación es solo una simple descarga de los datos.

Si algún tiempo después desea convertir los datos descargados a una copia de trabajo completamente funcional, ejecute `svn checkout <URL>` al directorio donde ejecutó la exportación.

Actualizando una copia de trabajo

No eres la única persona que trabaja en el proyecto, ¿verdad? Esto significa que sus colegas también están haciendo modificaciones a los datos del proyecto. Para mantenerse actualizado y obtener las modificaciones confirmadas por otros, debe ejecutar el comando `svn update` en su copia de trabajo. Como resultado, su copia de trabajo se sincronizará con el repositorio y descargará los cambios realizados por sus colegas.

Shorthand para `svn update` is `svn up` .

Es una regla ejecutar la `svn update` antes de confirmar sus cambios.

Haciendo cambios en su copia de trabajo local

La *copia de trabajo (WC)* es su *espacio de trabajo local y privado* que utiliza para interactuar con el repositorio central de Subversion. Utiliza la copia de trabajo para modificar el contenido de su proyecto y obtener los cambios confirmados por otros.

La copia de trabajo contiene los datos de su proyecto y se ve y actúa como un directorio regular en su sistema de archivos local, pero con una gran diferencia: la copia de trabajo rastrea el estado y los cambios de los archivos y directorios. Puede pensar en la copia de trabajo como en un directorio normal con un sabor de control de versión agregado por un directorio de metadatos `.svn` oculto en su raíz.

La mayoría de las veces, va a realizar modificaciones a los datos del proyecto modificando el contenido de la copia de trabajo. Tan pronto como esté satisfecho con las modificaciones y las haya revisado a fondo, estará listo para publicarlas en el repositorio central.

Puede realizar cualquier acción con los datos de su proyecto dentro de la copia de trabajo, pero las operaciones que implican copiar, mover, renombrar y eliminar deben realizarse utilizando los comandos `svn` correspondientes:

- **Modificando archivos existentes** . Modifique los archivos como lo hace normalmente usando su procesador de texto favorito, editor de gráficos, software de edición de audio, IDE, etc. Tan pronto como guarde los cambios en el disco, Subversion los reconocerá automáticamente.
- **Añadiendo nuevos archivos** . Ponga los archivos nuevos en la copia de trabajo y Subversion los reconocerá como sin *versión* . No iniciará automáticamente el seguimiento de los nuevos archivos a menos que ejecute el comando `svn add` :

  ```
  svn add foo.cs
  ```

- **Mover archivos y directorios** . Mueve archivos y directorios usando el comando `svn move` :

```
svn move foo.cs bar.cs
```

- **Renombrando archivos y directorios** . Renombra archivos y directorios usando el comando `svn rename` :

```
svn rename foo.cs bar.cs
```

NOTA: el comando `svn rename` es un alias del comando `svn move` .

- **Copiando archivos y directorios** . Copie archivos y directorios usando el comando `svn copy` :

```
svn copy foo.cs bar.cs
```

- **Eliminando archivos y directorios** . Eliminar archivos y directorios usando el comando `svn delete` :

```
svn delete foo.cs
```

- **Comprobación del estado de los archivos y directorios en la copia de trabajo** . Revise sus cambios usando el comando `svn status` (o `svn st` para abreviar):

```
svn status
```

> **IMPORTANTE:** siempre revise sus cambios antes de cometerlos. Esto le ayudará a evitar cometer cambios innecesarios o irrelevantes.

- **Revertir los cambios** . Revierta sus cambios usando el comando `svn revert` :

```
svn revert foo.c
```

- **Revirtiendo todos los cambios** : Desde la raíz del repositorio:

```
svn revert -R .
```

> **IMPORTANTE: Los** cambios no comprometidos revertidos se perderán para siempre. No podrás recuperar los cambios revertidos. Utilice `svn revert` con precaución! Si desea mantener los cambios pero necesita revertir, guárdelos en un parche. Vea el ejemplo de cómo crear y aplicar un parche.

Confirmando tus cambios locales en el repositorio

Para publicar los cambios realizados en su copia de trabajo, ejecute el comando `svn commit` .

> **IMPORTANTE: ¡** Revisa tus cambios antes de cometerlos! Utilice `svn status` y `svn diff` para revisar los cambios. Además, asegúrese de que está en la ruta correcta antes de realizar una confirmación. Si actualizó muchos archivos en varios directorios, debería

estar en el nivel adecuado para incluirlos debajo de su ubicación.

Aquí hay un ejemplo del comando commit:

```
svn commit -m "My Descriptive Log Message"
```

Alternativamente, `svn ci` es la abreviatura de `svn commit`

Note la opción `-m (--message)` . Los buenos mensajes de confirmación ayudan a otros a comprender por qué se realizó una confirmación. Además, en el lado del servidor es posible imponer mensajes no vacíos , e incluso imponer que cada mensaje de confirmación menciona un ticket existente en su sistema de seguimiento de errores.

Revisando una copia de trabajo en una revisión específica

Para obtener la versión 5394 use:

```
svn co --revision r5394 https://svn.example.com/svn/MyRepo/MyProject/trunk
```

O la versión más corta:

```
svn co -r 5394 https://svn.example.com/svn/MyRepo/MyProject/trunk
```

O mediante el uso de revisiones vinculadas:

```
svn co https://svn.example.com/svn/MyRepo/MyProject/trunk@5394
```

Si ya está desprotegido, puede usar el comando de `update` para mover a una revisión en particular, haciendo lo siguiente:

```
svn up -rXXX
```

Usando un repositorio protegido por contraseña

Se puede configurar un repositorio de Subversion para que ciertos contenidos o comandos solo sean accesibles para ciertos usuarios. Para acceder a este contenido restringido, deberá especificar un nombre de usuario y una contraseña.

Su nombre de usuario y contraseña se pueden especificar directamente como parte del comando:

```
$ svn checkout https://svn.example.com/MyRepo/trunk --username JoeUser --password topsecret
```

Desafortunadamente, esto hace que su contraseña aparezca en texto sin formato en la consola. Para evitar este posible problema de seguridad, especifique un nombre de usuario pero no una contraseña. Al hacer esto aparecerá un mensaje de contraseña, permitiéndole ingresar su contraseña sin exponerla:

```
$ svn checkout https://svn.example.com/MyRepo/trunk --username JoeUser
Password for 'JoeUser':
```

Al no proporcionar información de autenticación, Subversion le solicita el nombre de usuario y la contraseña:

```
$ svn checkout https://svn.example.com/MyRepo/trunk
Username:  JoeUser
Password for 'JoeUser':
```

Si bien el primer método es menos seguro, se lo ve con frecuencia en scripts automatizados, ya que es difícil para muchos tipos de scripts proporcionar información a un mensaje interactivo.

Nota : los comandos que solo operan en su copia de trabajo (como `revert` o `status`) nunca requerirán una contraseña, solo los comandos que requieren comunicación con el servidor de repositorio.

Creación y aplicación de parches.

Un parche es un archivo que muestra las diferencias entre dos revisiones o entre su repositorio local y la última revisión a la que apunta su repositorio.

Para compartir o guardar un parche de sus cambios locales no confirmados para revisión por pares o para aplicar más tarde, haga lo siguiente:

```
svn diff > new-feature.patch
```

Para obtener un parche de las diferencias entre dos revisiones:

```
svn diff -r NEWER_REVISION:OLDER_REVISION > feature.patch
```

Para aplicar un parche, ejecute:

```
svn patch new-feature.patch
```

> Para aplicar el parche correctamente, debe ejecutar el comando desde la misma ruta donde se creó el parche.

Revisando los logs

Ejecutar `svn log` le mostrará todos los mensajes de confirmación, probablemente desee revisar solo ciertas revisiones.

- Ver las `n` revisiones más recientes:

  ```
  svn log -n
  ```

- Ver una revisión específica:

```
svn log -c rXXX
```

- Ver los caminos afectados:

```
svn log -v -c rXXX
```

Revertir o deshacer un archivo

Para restaurar un archivo a la última versión de svn actualizada, es decir, deshacer los cambios locales, puede usar `revert` :

```
svn revert file
```

Para restaurar un archivo a una versión anterior (revisión XXX) use la `update` :

```
svn update -r XXX file
```

Advertencia : en ambos casos perderá cualquier cambio local en el archivo porque se sobrescribirá.

Para ver solo la versión anterior de un archivo usa `cat` :

```
svn cat -r XXX file
```

Y para ver las diferencias con su versión local del archivo:

```
svn diff -r XXX file
```

Lea Empezando con svn en línea: https://riptutorial.com/es/svn/topic/638/empezando-con-svn

Capítulo 2: Administrando SVN

Examples

Creando un nuevo repositorio

Se puede crear un nuevo repositorio con dos opciones diferentes:

1. Usando la línea de comando

Ejecutar el siguiente comando. Creará un directorio para el repositorio, pero la ruta principal debe estar presente. es decir, en el siguiente ejemplo, `/var/svn` ya debería estar allí, mientras se creará el directorio `my_repository` .

```
svnadmin create /var/svn/my_repository
```

Si está utilizando TortoiseSVN , puede usar la GUI para crear repo.

1. Abra el directorio donde desea crear un nuevo repositorio.
2. Haga clic derecho en la carpeta y seleccione `TortoiseSVN -> Create Repository here...`
3. Luego se crea un repositorio dentro de la carpeta seleccionada. ¡No edites esos archivos tú mismo! Si recibe algún error, asegúrese de que la carpeta esté vacía y no protegida contra escritura.

Crear nuevo usuario

Para agregar usuario, use el siguiente comando

```
htpasswd /etc/subversion/passwd user_name
```

Especifique `user_name` con el nombre de usuario que desea agregar en el comando anterior. Se le pedirá que proporcione la contraseña para el usuario.

Si está creando el primer usuario, debe agregar el comando `-c` switch in above, que creará el archivo.

```
htpasswd -c /etc/subversion/passwd user_name
```

Puede verificar la existencia del archivo o la lista de usuarios configurados utilizando el siguiente comando `cat /etc/subversion/passwd`

Es posible que deba ejecutar los comandos anteriores como superusuario.

Crear grupos de usuarios

Los grupos se pueden definir en el `/etc/subversion/svn_access_control` .

Crea / edita el archivo usando el siguiente comando

```
nano /etc/subversion/svn_access_control
```

Use la sintaxis especificada a continuación para definir grupos y asignar miembros.

```
[groups]
groupname = <list of users, comma separated>
```

p.ej

```
[groups]
myproject-dev = john, peter
myproject-support = maria, cristine
```

El ejemplo anterior creará dos grupos llamados `myproject-dev` y `myproject-support` . Agregará a los usuarios `john` y `peter` al grupo `myproject-dev` y los usuarios `maria` y `cristine` al grupo `myproject-support` .

Los grupos se pueden usar para administrar el acceso al repositorio

Gestionando permisos de repositorio

Las especificaciones de acceso para los repositorios de subversion están especificadas como el `etc/subversion/svn_access_control`

Crea / edita el archivo usando el siguiente comando

```
nano /etc/subversion/svn_access_control
```

Use la siguiente sintaxis para configurar los permisos de acceso para los repositorios a grupos / miembros

```
[Repository:<Path>]
@groupname = r/rw
User = r
```

p.ej

```
[myproject:/]
@myproject-dev = rw
@myproject-support = r
jack = r

[myproject:/branches/support]
@myproject-support = rw
patrick = r
```

La configuración de ejemplo anterior otorgará acceso de lectura y escritura a todo el repositorio de `myproject` a los usuarios que pertenecen al grupo `myproject-dev` , mientras que el acceso de solo lectura se otorga a los usuarios que pertenecen al grupo `myproject-support` y al `jack` usuario específico. **Tenga en cuenta que, los nombres de los grupos están precedidos por `@` .**

Del mismo modo, asignará acceso de lectura y escritura para `support` rama del repositorio `myproject` a todos los usuarios que pertenezcan a `myproject-support` y acceso de solo lectura a `patrick` .

Lea Administrando SVN en línea: https://riptutorial.com/es/svn/topic/7935/administrando-svn

Capítulo 3: Ramificación, estantería y etiquetado en Subversion de Apache.

Sintaxis

- svn copia [BRANCH-FROM-URL] [BRANCH-TO-URL] -m `<COMMIT-LOG-MESSAGE>`
- svn copy [^ / PATH-TO-BRANCH-FROM] [^ / PATH-TO-BRANCH-TO] -m `<COMMIT-LOG-MESSAGE>`

Observaciones

Como habrá notado, usamos el comando `svn copy` para crear sucursales, etiquetas y estantes (omitiremos las etiquetas y estantes en los párrafos siguientes). Este es el mismo comando utilizado para copiar elementos en su copia de trabajo y en el repositorio.

`svn copy` se utiliza para la bifurcación porque, bifurcando es técnicamente una copia de la fuente desde la que se copia. Sin embargo, esto no es una copia normal con la que están familiarizados cuando copian archivos en su sistema de archivos local. Las sucursales en Subversion son llamadas "Copias baratas" que son similares a los enlaces simbólicos. Por lo tanto, crear una nueva rama lleva un tiempo mínimo para completarse y prácticamente no ocupa espacio en el repositorio de Subversion. Cree sucursales y utilícelas para cualquier cambio que desee, independientemente del tamaño y el alcance del cambio.

`svn copy` se puede reducir a `svn cp` ya que Subversion tiene alias para la mayoría de los comandos.

Examples

Creando una rama usando URL directa para copiar URL

La ramificación en Subversion es *muy* simple. En la forma más simple, crear una nueva rama requiere que ejecute el comando contra las URL del repositorio remoto. Por ejemplo, vamos a crear una nueva rama a partir de la línea principal:

```
svn copy https://svn.example.com/svn/MyRepo/MyProject/trunk
https://svn.example.com/svn/MyRepo/MyProject/branches/MyNewBranch -m "Creating a new branch"
```

La nueva sucursal está lista y puedes comenzar a trabajar con ella. Verifique una nueva copia de trabajo con la nueva sucursal o cambie su copia de trabajo existente con el comando `svn switch` .

Creando una rama a través de una copia de trabajo.

Cuando interactúa con el repositorio central remoto utilizando su área de trabajo local privada, la

copia de trabajo, puede usar la URL relativa al repositorio en lugar de la URL directa a la copia de URL para crear una nueva rama:

```
svn copy "^/MyProject/trunk" "^/MyProject/branches/MyNewBranch" -m "Creating a new branch"
```

Cambio de una copia de trabajo a una rama diferente

Una copia de trabajo existente se puede transformar rápidamente para reflejar el contenido de una rama diferente en el mismo repositorio. Por ejemplo, puede tener una copia de trabajo del troncal y ahora necesita trabajar en una rama de desarrollo. En lugar de retirar una copia de trabajo completamente nueva (que puede desperdiciar mucho tiempo y espacio en el disco), puede usar el comando `svn switch` para modificar de manera eficiente su copia de trabajo existente:

```
svn switch ^/MyProject/branches/MyNewBranch
```

Su copia de trabajo ahora reflejará el contenido de la rama en lugar del tronco.

Usando etiquetas

Las "etiquetas" son un tipo de etiqueta que se puede aplicar a un repositorio en un determinado momento. Con frecuencia se usan para dar nombres legibles por humanos a hitos importantes para que se pueda acceder fácilmente más adelante (por ejemplo, "versión 1.2").

Crear una etiqueta es exactamente lo mismo que crear una rama:

```
svn copy -r 1234 ^/MyProject/trunk ^/MyProject/tags/version-1.2
```

En este caso específico, el argumento `-r 1234` se usó para indicar que la etiqueta debería crearse a partir de la revisión 1234 del tronco.

Subversion no hace ninguna distinción entre una etiqueta y una rama ordinaria. La única diferencia está en cómo decides usarlos. Tradicionalmente, no se realizan confirmaciones en una etiqueta una vez que se ha creado (para garantizar que siga siendo una "instantánea" precisa de un estado anterior del repositorio). Subversion no impone ninguna regla especial relacionada con etiquetas de manera predeterminada, ya que diferentes personas pueden usar etiquetas de manera diferente. Sin embargo, un administrador de repositorio puede configurar scripts de control de acceso para hacer cumplir las reglas que su equipo ha decidido usar.

> En Windows, necesitas usar una careta doble `^^` .

Borrando una rama

Solo corre:

```
svn delete https://svn.example.com/svn/MyRepo/MyProject/branches/MyNewBranch -m "Deleting no longer needed MyNewBranch"
```

O, usando la URL corta:

```
svn delete ^/branches/MyNewBranch -m "Deleting no longer needed MyNewBranch"
```

- En Windows, necesitas usar ^^
- Siempre puede recuperar una rama eliminada al crearla nuevamente especificando la revisión deseada cuando la rama estaba activa (una rama eliminada es solo una rama que no está disponible en la revisión HEAD). Por ejemplo, si se eliminó la rama en la revisión 101: `svn copy https://svn.example.com/svn/MyRepo/branches/MyNewBranch@r100 https://svn.example.com/svn/MyRepo/MyProject/branches/resurrected-branch -m "Resurrected MyNewBranch from revision 100"` . Ver la resurrección de elementos eliminados

Lea Ramificación, estantería y etiquetado en Subversion de Apache. en línea: https://riptutorial.com/es/svn/topic/668/ramificacion--estanteria-y-etiquetado-en-subversion-de-apache-

Creditos

S. No	Capítulos	Contributors
1	Empezando con svn	agold, bahrep, bta, Chad Nouis, Community, Eiren Smith, opticyclic, ronnyfm, V-R, wrothe
2	Administrando SVN	opticyclic, Rumit Parakhiya
3	Ramificación, estantería y etiquetado en Subversion de Apache.	bahrep, bta, opticyclic, ronnyfm