Mundo real: cómo administrar un servidor subversion para un equipo pequeño

Estas son mis notas tras 4 años de administración de un servidor Subversion (SVN) en Linux, para un equipo pequeño de 4 o 5 personas que acceden desde la oficina, o desde casa, usando clientes Windows, Mac o Linux.

Si lo que te interesa es alojar tu servidor SVN en windows, puedes echar un vistazo a VisualSVN Server para Windows (con versión gratuita y de pago).

Lo habitual es que pensemos en Subversion como una manera de "poder trabajar en equipo al mismo tiempo en el mismo proyecto, controlar los cambios en el código", etc. Un concepto adicional que he aprendido para entender mejor cómo utilizar SVN en el día a día en un equipo es el siguiente: **usamos Subversión con el objetivo de propagar fácilmente los cambios realizados en los proyectos al resto del equipo y a los servidores de prueba, de producción, etc.** Esta idea ayuda a comprender mejor cómo debemos enviar (*commitar*) los cambios que realizamos en el código fuente y los cambios en las carpetas o directorios, especialmente en SVN.

Las ventajas de alojar Subversion/SVN en Linux:

1. Se puede hostear en servidores (o VMs) más pequeños, resultando más baratos que Windows.
2. Se pueden aprovechar herramientas (scripts) ya escritos en Bash, Perl, Python…
3. Dispondremos fácilmente del protocolo `svn+ssh` para tener transferencias seguras (encriptadas) de forma simple, sin tener que usar Apache con certificados SSL, etc.
4. La gestión de usuarios la realizaremos usando el propio sistema de cuentas de usuario de Linux.

Inconvenientes de alojar Subversion en Linux bajo el protocolo `svn+ssh`:

1. Estamos obligados a usar línea de comandos **para administrar el servidor**, no hay intergaz gráfica (GUI). En el lado del cliente (el usuario), podremos usar GUIs como TortoiseSVN para Windows o cualquier otra para otros sistemas, además de línea de comandos.
2. No tenemos la misma granularización de permisos en los repositorios y directorios. Todos los usuarios pueden acceder a todo y escribir en todas partes, si bien siempre hay alternativas o pequeños "hacks".
3. No tenemos "de entrada" un visor HTML de los repositorios como cuando instalamos una típica instalación SVN con el servidor web Apache, etc. aunque siempre se pueden instalar aparte.

Cosas que necesitaremos:

- Instalación y actualización de Subversion.
  - Instalar la versión de Subversion deseada, en Ubuntu o CentOS, cliente o servidor.
  - Poder actualizar el cliente/servidor SVN para disfrutar de la última versión.
- Interfaz: en Windows, la herramienta TortoiseSVN es la más completa. En línea de comandos, CMDer lee los metadatos de Subversion y nos muestra el estado/rama de cada path como en git.

- Gestión de usuarios
  - Añadir usuarios al servidor Subversion (un nuevo empleado, por ejemplo).
  - Dar de baja usuarios (expiración de usuarios, por ejemplo cuando un empleado deja la empresa). Evitar perder el histórico de commits de dicho usuario.
  - Si el servidor es solo para SVN, por seguridad los usuarios *solo* deben poder hacer Subversion una vez conectados por `ssh`, y **nada más**.
- Decidir nuestro “esquema de repositorios” (layout), por ejemplo el de “uno o más proyectos en cada repo”, más flexible que el de “un repo = un proyecto”.
- Aprender a usar los “externals” de SVN para tener carpetas/librerías comunes en varios proyectos a la vez, de forma centralizada.
- Scripts:
  - Script para crear un nuevo repositorio
  - Script para añadir un nuevo a un repositorio existente
  - Script para crear un esqueleto de aplicación en un proyecto nuevo (boilerplate/skeleton)
  - Script para listar los repositorios disponibles y los diferentes proyectos en los repositorios
  - Script para buscar cadenas en ciertos archivos de todos los repositorios
  - Script para “empaquetar” los repositorios (ahorro de espacio con el paso del tiempo)
  - Scripts para hacer copias de seguridad (backup) de los repositorios. Hay diversos modos de hacerlo.
- Extras:
  - Uso de una rama “trunk” estándar (algo como “master” en git), y como mucho una rama “release” para proyectos que lo requieren.
  - Svnnotify (al hacer commit, envío automático de un e-mail al equipo con los cambios/diferencias efectuados).
  - Utilizar un usuario de “actualizaciones/despliegue” para desplegar el proyecto (deploy)
  - Uso de “phinx” para cambios en Base de datos (*migrations* o migraciones de DB)