Departamento de Tecnología Electrónica
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Laboratorio 2
Sistemas de Control de Versiones

Planificación y Gestión de Proyectos Informáticos

1. Introducción y objetivos

La duración estimada de esta sesión de laboratorio es de **6 horas**. El propósito general de esta sesión de laboratorio es comparar modo de operación de los Sistemas de Control de Versiones (SCV) centralizados frente a los distribuidos. Se tratarán los siguientes aspectos:

- Instalación de una infraestructura para servir repositorios Subversion con control de usuarios y permisos.
- Instalación de una infraestructura para servir repositorios Git con control de usuarios y permisos.

Para la realización de los laboratorios se facilitarán diversos ficheros, concretamente la tabla 1 resume el contenido y el objetivo .

Nombre del fichero	Descripción
preguntas.svndump	Volcado del repositorio subversión de preguntas

Tabla 1. Ficheros necesarios durante la sesión de laboratorio.

2. Subversion

Subversión es un sistema de control de versiones centralizado con licencia de software libre Apache. Aunque presenta deficiencias en comparación con los sistemas distribuidos como Git o Mercurial, es ampliamente utilizado en desarrollos tanto libres como corporativos, principalmente por la facilidad de aprendizaje y uso frente a los sistemas distribuidos. Por ello es posible su uso en multitud de forjas de software existentes por la red.

El objetivo principal de subversion, al igual que el resto de SCV, consiste en mantener un histórico de todos los cambios realizados en los ficheros que estén bajo control de versiones. El modo de operar de subversion es numerar cada cambio (commit) realizado en el repositorio de manera consecutiva y comenzando en 1. En un repositorio de este tipo, el número de revisión hace referencia al número de cambios realizados.

Subversion sólo se encarga de mantener las diferentes versiones pero no establece como deben organizarse los ficheros en un repositorio, por ello, existe una estructura habitual y considerada un convenio de buenas prácticas en el uso de subversion, consistente en crear siempre un repositorio inicialmente con tres carpetas: *trunk*, *tags* y *branches*. Estas carpetas se deben utilizar con el siguiente propósito:

- Trunk: Rama principal.
- Tags: Rama usada para congelar versiones, nunca se debe trabajar en esta rama, excepto para cerrar una nueva versión.
- Branches: Ramas paralelas a Trunk donde se realiza el desarrollo.

2.1. Trabajo en grupo con Subversion

Para comprender el funcionamiento centralizado de subversion se ha preparado un repositorio con un conjunto de ficheros de texto sobre los que se trabajará de manera colaborativa. Cada fichero de texto contiene una pregunta, a la que hay que aportar varias respuestas. Cada usuario solo aportará una respuesta, así varios usuarios deben compartir la edición del fichero de texto, generándose conflictos que se deberán resolver.

El primer objetivo es congelar una primera versión con todas las respuestas. Cada alumno tras contestar parcialmente una de las cuestiones, debe intentar subirla al repositorio solucionando todos los conflictos. Una vez contestadas todas las cuestiones se creara la primera versión 1.0 en la carpeta *tags*.

El ejercicio continúa partiendo de la versión 1.0, donde cada alumno debe crear una rama en *branches*, con su nombre, partiendo de la versión 1.0 congelada en *tags*. Cada uno en su rama, aportará una nueva pregunta y se irán mezclando en con la rama principal (*trunk*) hasta obtener la versión 2.0. Aquí el principal problema será solucionar los conflictos en los nombres de ficheros.

Tarea 1.- Para trabajar, debe instalar un cliente subversion. Existen varios clientes tanto para Windows como Linux, se recomienda no usar *rapidsvn*, ya que no se realiza mantenimiento sobre -el. Si está en Linux se recomienda utilizar la línea de comandos o instalar el cliente *kdesvn*. Puede consultar en la tabla 2 los comandos más comunes de subversion, y si utiliza Windows, puede descargar *tortoisesvn* desde http://www.tigris.org/.

T1.1.- Si utiliza Windows, inicie *tortoisesvn* mediante el botón derecho del ratón en alguna carpeta vacía. Use la opción SVN Checkout mostrado en la figura 1 y escriba la URL indicada por el profesor. Es importante en el directorio destino sea un nuevo directorio vacío, de lo contrario se mezclará el repositorio con los ficheros existentes.

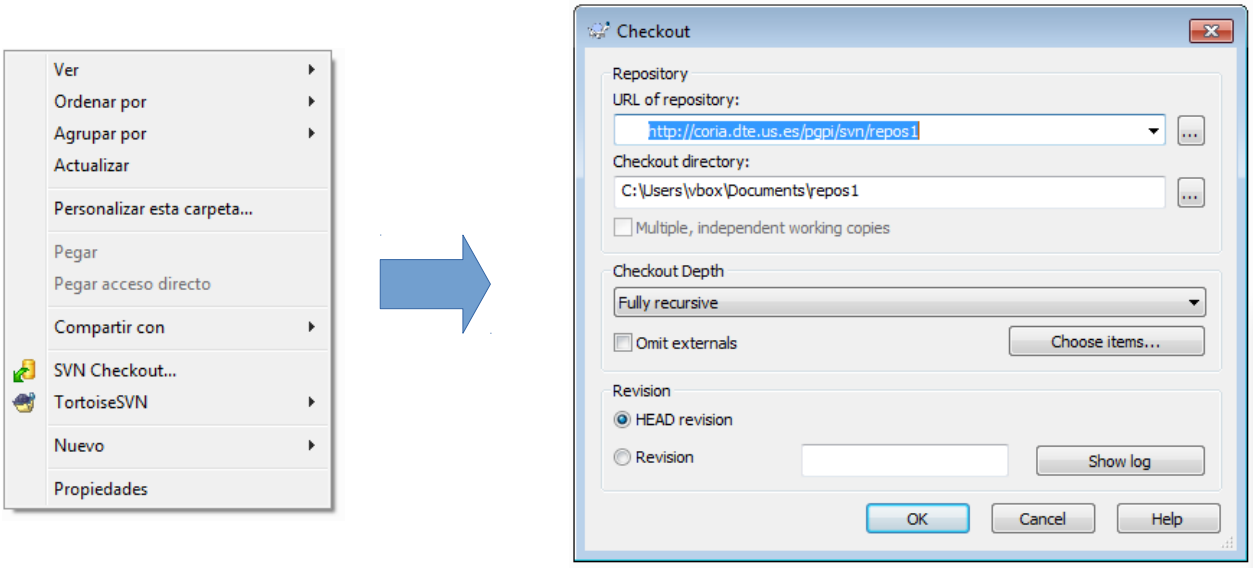

Figura 1. Obtención de copia local con svn.

T1.2.- Si utiliza Linux puede usar *kdesvn* y descargar el repositorio desde el menú Subversion → General → Obtener un repositorio.

Comando	Descripción
svn co <*repositorio*>	Descarga una copia local del repositorio remoto
svn status -u	Comprobación de cambios remotos, no descarga nada solo muestra los ficheros remotos modificados
svn update	Actualiza la copia local con los cambios existentes en el repositorio
svn commit	Envía los cambios locales al repositorio

Tabla 2. Comandos básicos de subversion en línea de comandos.

Tarea 2.- Una vez retirada una copia local del repositorio, observe la estructura del repositorio y entre en *trunk*. Encontrará 5 ficheros numerados del 0 al 4.

T2.1.- Debe editar un fichero siguiendo el siguiente criterio: divida entre 2 el último dígito de su DNI y escoja el fichero numerado con el resultado obtenido. Responda una única opción de la pregunta existente en el fichero, otros compañeros responderán otras opciones.

T2.2.- Intente subir su respuesta al repositorio, si falla el *commit*, entonces el repositorio ha sido modificado por otro usuario. Compruebe los cambios con `svn status -u` y actualice su copia con el comando `svn update` y si surgen conflictos debe solucionarlos. C1:¿Cómo marca subversion los conflictos en los ficheros?

Tarea 3.- Navegue a Google-code y descargue algún desarrollo que utilice subversion como SCV. Alternativamente puede descargar algunos de estos dos proyectos:

- http://ipcamera-for-android.googlecode.com/svn/trunk/
- http://sourceforge.net/p/doublecmd/code/HEAD/tree/

T3.1.- Intente ahora traerse el repositorio completo de alguno de los proyectos para ver las versiones congeladas disponibles. C2:¿Donde están las versiones congeladas?

2.2. Instalación de servidor Subversion

Ahora el objetivo consiste configurar un servidor propio de subversion con tres repositorios y diferentes niveles de acceso. Todos ellos residirán en un mismo directorio y con los mismos permisos UNIX[1]. Estos repositorios se servirán mediante un servidor WEB, usando el protocolo HTTP y/o HTTPS. La configuración del servidor será quien establezca los permisos correspondientes requiriendo usuario y contraseña si es necesario.

Tarea 4.- Sobre la máquina Linux, si todavía no lo tiene instalado, instale el paquete *subversion*.

T4.1.- Elija un directorio para almacenar los repositorios, por ejemplo, `/opt/svn`.

T4.2.- Se utilizará un directorio por proyecto, para crear un repositorio en `/opt/svn` use el comando `svnadmin create proyecto-secreto`. Repita el proceso creando dos repositorios más: `proyecto-publico` y `proyecto-anarquico`.

Tarea 5.- Ahora se añadirá al servidor WEB Apache la funcionalidad capaz de servir repositorios. Instale los paquetes para el servidor WEB Apache *libapache2-svn*.

T5.1.- Antes de configurar el servidor debe hacer accesibles los ficheros del directorio `/opt/svn` a Apache. Puesto que Apache se ejecuta con el usuario *www-data*, utilice el comando `chown -R www-data:www-data /opt/svn` para establecer los permisos adecuadamente.

T5.2.- Para activar el soporte subversion de Apache debe usar el comando `a2enmod`, este comando lista los módulos disponibles y debe activar `authz_svn` y `dav_svn`.

T5.3.- La configuración del servidor WEB reside en el directorio `/etc/apache2`, donde el directorio `/etc/apache2/sites-enabled` permite crear y configurar los sitios WEB existentes en el servidor. Edite el fichero que contiene la configuración del sitio por defecto `/etc/apache2/sites-enabled/000-default.conf` y añada dentro del entorno `<VirtualHost … > … </VirtualHost>` la siguiente configuración:

```
#Cuidado LOCATION debe tener la barra al final /SVN/ de lo contrario falla
<Location /svn/>
  DAV svn
  SVNParentPath /opt/svn
  SVNListParentPath On
</Location>
```

Código 1. Fichero /etc/apache/sites-enabled/000-default.conf.

T5.4.- Reinicie Apache con el comando `/etc/init.d/apache2 restart` y navegue a la dirección http://localhost/svn/, deberían mostrarse los repositorios que acaba de crear.

T5.5.- Usando un cliente subversion retire una copia local de uno de los repositorios, realice una modificación en su copia local e intente realizar *commit* con los cambios. Intente obtener la copia desde la máquina anfitrión, no desde la máquina virtual. C3:Con esta configuración, ¿puede realizar commits sobre los tres repositorios?.

Con la configuración realizada hasta el momento todos los repositorios son accesibles y modificables directamente desde la dirección WEB sin ningún tipo de control de usuarios. Se avanzará configurando el módulo de Apache encargado de servir repositorios, el cual, contempla el control de usuarios y permisos. Este control está basado en dos ficheros: fichero de usuarios y ficheros de permisos.

1 Posteriormente se detalla el funcionamiento básico de los permisos en UNIX.

Tarea 6.- Para crear un fichero de usuarios y contraseñas para autenticación básica con Apache se utiliza el comando `htpasswd` y para disponer de este comando debe instalar el paquete *apache2-utils*. Utilice el comando `touch` para crear un fichero vacío y después usando `htpasswd` añada tres usuarios, uno de ellos con nombre `admin`. Fíjese en el siguiente ejemplo: C4:¿Se pueden ver las contraseñas en el fichero que ha creado?

```
touch /etc/apache2/svn.passwd
htpasswd /etc/apache2/svn.passwd admin
htpasswd /etc/apache2/svn.passwd alumno1
```

Código 2. Comandos para creación ficheros de usuarios y contraseñas para Apache.

T6.1.- Una vez creados los tres usuarios, debe configurar Apache para que solicite autenticación en los repositorios subversion basado en el fichero que ha creado. Realice los cambios resaltados en el fichero correspondiente (vea T5.3.-):

```
<Location /svn/>
  DAV svn
  SVNParentPath      /opt/svn
  SVNListParentPath  On
  AuthType           Basic
  AuthName           "Respsitorio SVN - PGPI"
  AuthUserFile       /etc/apache2/svn.passwd
  AuthzSVNAccessFile /etc/apache2/svn.authz
  Require            valid-user
  <Limit GET PROPFIND OPTIONS REPORT>
    Satisfy Any
  </Limit>
</Location>
```

Código 3. Configuración de subversion con autenticación y permisos en Apache.

T6.2.- En la configuración anterior se añadió el fichero *svn.authz*, el cual todavía no ha sido creado. Este fichero es el encargado de controlar los permisos. Debe crearlo en la ubicación correcta, siendo el objetivo disponer de un repositorio público de solo lectura, otro público donde cualquiera pueda subir cambios y uno totalmente privado. Como ejemplo use código 4 mostrado cuyo significado se describe línea por línea en la tabla 3.

```
[/]
* = r
admin = rw

[proyecto-secreto:/]

* =
paulino = rw
jorge = r

[proyecto-publico:/]
paulino = rw
```

Código 4. Fichero svn.authz

Línea	Significado
`[/]`	Reglas para todos los repositorios
`* = r`	Todos los usuarios pueden hacer checkout (comprobaciones)
`admin = rw`	El usuario admin puede hacer checkout y commit
`[proyecto-secreto:/]`	Reglas aplicables al repositorio proyecto-secreto
`* =`	Ningún usuario tiene permisos
jorge = r	El usuario jorge puede sólo hacer checkout

Tabla 3. Significado de reglas svn.authz.

T6.3.- Para poder comprobar si hay errores en la configuración que está realizando es recomendable mantener abierto un terminal adicional donde se muestre en tiempo real la bitácora de Apache. El comando es `tail -f /var/log/apache2/*`.

T6.4.- Navegue a la dirección http://localhost/svn/ y vea el listado de repositorios. Intente entrar con

el navegador en los diferentes proyectos. C5:¿Que repositorios se listan y qué ocurre cuando se intenta acceder al proyecto-secreto?.

T6.5.- Desde un cliente subversion externo obtenga una copia de cada uno de los repositorios, realice algunos cambios e intente subir las modificaciones. C6:¿Qué ocurre al subir los cambios en cada uno de los repositorios?.

Tarea 7.- Busque documentación sobre la creación de grupos en el fichero de permisos de subversion y cree un grupo de desarrolladores con permisos de escritura en todos los repositorios. C7: Indique un ejemplo de creación de un grupo de 3 usuarios.

Tarea 8.- Cree un nuevo repositorio vacío llamado preguntas y restaure el volcado existente en el fichero `preguntas.svndump` del profesor mediante el comando `svnadmin load preguntas < preguntas.svndump`.

T8.1.- Establezca los permisos adecuadamente (ver T5.1.-), y retire una copia local del repositorio directamente desde el sistema de ficheros para comprobar si se ha realizado correctamente la importación. C8: ¿Cual es la URL que debe indicar el comando svn checkout para que opere sobre el sistema de ficheros?.

T8.2.- Intente retirar una copia del mismo repositorio pero ahora usando el protocolo SSH desde una máquina externa o la máquina anfitrión. C9: ¿Cual es la URL que debe indicar el comando svn checkout para que opere sobre SSH?.

T8.3.- Intente volcar de nuevo el repositorio usando el comando `svnadmin dump …`

T8.4.- Discusión: C10: Indique las ventajas, desventajas, limitaciones de trabajar sobre repositorios SVN en modo HTTP, ficheros locales y SSH. Plantee diferentes situaciones de desarrollo de proyectos donde pueda convenir cada uno de los modos estudiados.

2.3. Subversion avanzado (opcional)

Por no ser subversion un SCV descentralizado no facilita el clonado de repositorios para disponer del mismo en diferentes ubicaciones. Existen algunas soluciones que permite realizar copias como se ha visto anteriormente mediante el volcado de un repositorio (dump). También existen herramientas desarrolladas para sincronizar repositorios subversion, aunque muchas no son parte del sistema subversion.

Para mostrar esta dificultad opcionalmente se propone que intente realizar las siguientes operaciones buscando en la red información al respecto.

Tarea 9.- Intente clonar o volcar un repositorio existente en la red, al cual solo tiene acceso por HTTP. Comprobará la dificultad de clonar un repositorio subversion.

Tarea 10.- Intente mantener sincronizado un repositorio local con uno remoto alojado en Google Code o SourceForge. De entre la soluciones existente por la red, se recomienda el uso de *svnsync*.

3. Git

Git ha sido ampliamente estudiado previamente en la asignatura, así, el objetivo ahora no es estudiar el uso de git, sino la configuración de un servicio de alojamiento de repositorios git. Tras mostrar la

dificultad de mantenimiento de copias sincronizadas al usar SCV centralizados, se propone usar git para salvar estas dificultades. El nuevo objetivo consiste en servir varios repositorios con diferentes niveles de acceso, desde el acceso público a repositorios totalmente restringidos.

A diferencia de subversion, cuya solución estándar es el uso de un servidor WEB con el protocolo WEBDAV, con git existen muchas soluciones con diferentes niveles de complejidad. La filosofía de git como SCV descentralizado hace que aparezcan multitud de soluciones para resolver este problema, las más utilizadas y ampliamente documentadas en la red, son las siguientes:

- Uso del protocolo SSH con diferentes usuarios y permisos correctamente configurados en un servidor UNIX.
- Uso del protocolo SSH con un único usuario compartido por todos los usuarios y usando autenticación con clave pública/privada.
- Uso de un servidor WEB con el protocolo WEBDAV pero se requiere una configuración individualizada para cada repositorio si se desea usar autenticación.
- Uso el propio protocolo git, incluido con la aplicación git operando en modo servidor.
- Usar *gitolite*, el cual es una solución avanzada para control de usuarios y múltiples repositorios, pero es difícil de configurar y mantener.

Para el propósito de este laboratorio se estudiarán sólo tres soluciones: una basada en usuarios y permisos UNIX, otra usando el servidor WEB Apache y finalmente usando el protocolo Git. Antes de probar diferentes configuraciones, se descargarán tres repositorios que se usarán posteriormente. Siga los siguientes pasos:

Tarea 11.- Instale los paquetes *git* y *gitg*. Cree un nuevo directorio para alojar los proyectos en /opt/git

T11.1.- Permanezca como usuario root y clone tres proyectos *git* en modo *bare* en /opt/git. Puede buscar algunos proyectos que le interese en *Google Code*, en *GitHub*, o utilizar los indicados en la tabla 4.

T11.2.- Utilizando `gitg` sobre el proyecto *Node Beginners Book*: C11:¿Cuantas ramas hay? ¿Cuantas etiquetas? ¿Cuando fue el último cambio?

T11.3.- Utilizando de nuevo `gitg` compruebe la cantidad de ramas existentes en el proyecto *K-9 Mail*. Use el cuadro de diálogo Rama para seleccionar en la vista Todas las ramas. C12:¿Qué comando hay que usar para recuperar la versión 4.4 de este programa?

T11.4.- Pruebe clonar un repositorio con el comando `git clone --mirror URL` C13:¿Cual es la diferencia entre --mirror y --bare?

Proyecto	URL	Descripción
Node Beginners Book	https://github.com/manuelkiessling/NodeBeginnerBook.git	Documentación sobre Node.js
K-9 Mail	https://github.com/k9mail/k-9.git	Gestor de correo electrónico para Android
FDroid	https://gitorious.org/f-droid/fdroidclient.git	Repositorio de aplicaciones libres para Android
Android Calendar Widget	https://github.com/plusonelabs/calendar-widget.git	Widget de calendario

Tabla 4. Repositorios de ejemplo seleccionados en la red.

3.1. Permisos en sistemas UNIX

Los sistemas UNIX son en esencia sistemas multiusuario, por ello cualquier fichero existente tiene asignado un usuario propietario y un grupo. Los grupos están formados por usuarios existentes en el sistema. Sobre los ficheros existentes de pueden establecer tres permisos: lectura, escritura y ejecución. Y estos permisos se otorgan de manera individual al usuario propietario, al grupo y al resto de usuarios.

Los listados de ficheros muestran los permisos agrupados en tripletes en el formato mostrado en la figura 2. El primer indicador es el tipo de fichero y los demás indican los permisos habilitados. Cuando un permiso no está habilitado, aparece un "-" en lugar de la letra correspondiente. La letra que aparece tiene el siguiente significado:

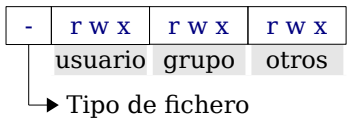

Figura 2. Permisos UNIX.

- r: Permiso de lectura, si es un directorio permite únicamente ver los nombres de los ficheros.
- w: Permiso de escritura, en caso de ser un directorio con este permiso se pueden crear nuevos ficheros dentro del directorio.
- x: Permiso de ejecución, en caso de ser un directorio este permiso indica si es posible entrar en el directorio para listar su contenido en su totalidad y acceder a sus ficheros/directorios.

El código 5 muestra la salida del comando `ls -l` (listado extendido) en un terminal. Para interpretar correctamente los permisos de cada uno de los ficheros se deben considerar las tres primeras columnas: siendo la primera columna los permisos, la segunda el usuario al que pertenece el fichero y la tercera el grupo al que está asignado el fichero. En la tabla 5 se interpretan los permisos fichero por fichero para facilitar la comprensión.

```
> ls -l
-rw-r-----  1 root       fuse        216 oct 18  2011 fuse.conf
drwxr-xr-x  4 root       lp         4096 nov 15 22:08 cups
srw-rw-rw-  1 root       acpi          0 nov 15 22:03 acpid.socket
-rw-r--r--  1 statd      root          5 nov 15 22:03 rpc.statd.pid
```

Código 5. Salida del comando ls -l en un terminal.

Fichero	Propietario grupo y permisos
fuse.conf	Fichero perteneciente al usuario *root* y al grupo *fuse*. ▪ El usuario *root* puede escribir y leer. ▪ Los usuarios pertenecientes al grupo *fuse* pueden leerlo. ▪ El resto de usuarios del sistema no tienen acceso.
cups	Directorio perteneciente al usuario *root* y asignado al grupo *lp*. ▪ El usuario *root* puede entrar, leer el directorio y añadir al directorio entradas. ▪ El grupo *lp* y el resto de usuarios del sistema pueden entrar en el directorio, ver los ficheros pero no puede añadir nuevas entradas en el directorio.
acpid.socket	Fichero de tipo socket perteneciente al usuario *root* y al grupo *acpi*. Todos los usuarios pueden leer y escribir en el fichero
rpc.statd.pid	Solo el usuario *statd* puede escribir en el fichero, el resto de usuarios pueden leerlo.

Tabla 5. Interpretación de los permisos UNIX del código 5 de salida del terminal

Comando	Descripción
chown	Cambia el usuario y/o el grupo de los ficheros
chmod	Cambia los permisos de los ficheros
chgrp	Cambia el grupo al que está asignado un fichero
id	Muestra el identificador numérico de usuario y los grupos a los que pertenece un usuario
adduser	Añade un nuevo usuario al sistema o añade un usuario existente a un grupo.
addgroup	Añade un nuevo grupo en el sistema

Tabla 6. Comandos básicos de administración usuarios grupos y permisos en UNIX.

Los comandos básicos para la gestión de los permisos usuarios y grupos son los mostrados en la tabla 6. De todos ellos el más usado es *chown*, y la manera de familiarizarse con ellos será realizar las diferentes tareas propuestas en los sucesivos epígrafes.

3.2. Repositorios privados usando usuarios y permisos UNIX

La primera configuración a realizar consiste en usar la característica de UNIX multiusuario para crear usuarios en el sistema con acceso al mismo. Estableciendo adecuadamente los permisos y los grupos a los que pertenece cada usuario y cada fichero es posible realizar diferentes tipos de configuraciones de repositorios con determinados niveles de acceso.

Concretamente se propone configurar el sistema, inicialmente, con repositorios de acceso restringido donde cada usuario pueda trabajar personalmente. Después, se compartirán algunos repositorios para trabajo en grupos, creando en el sistema dichos grupos de trabajo. También se crearán repositorios compartidos en modo solo lectura y con acceso restringido.

Tarea 12.- Para crear cuentas de usuarios en Linux use el comando `adduser`. Añada tres usuarios al sistema con sus correspondientes contraseñas, concretamente un jefe de proyecto y dos desarrolladores.

T12.1.- Conviértase en el usuario jefe de proyecto usando el comando `su <nombre_usuario>`. Ejecute el comando `id` para ver que usuario es y a que grupos pertenece. Con este usuario cree un repositorio git vacío en modo *bare* dentro de su directorio personal. C14:¿A qué grupos pertenece un usuario recién añadido?

T12.2.- Liste el directorio con `ls -l` para ver los permisos establecidos para el nuevo repositorio. Pruebe con un usuario desarrollador clonar el repositorio indicando la ruta completa del repositorio en el comando *clone*, e intente subir modificaciones. C15:¿Puede otro usuario modificar el repositorio? ¿Por qué?

T12.3.- Cambie los permisos para conseguir que el repositorio solo sea accesible por el usuario propietario y nadie pueda clonarlo. C16:¿Cuáles son los permisos correctos?

Cuando se utiliza git mediante SSH operan en el sistema de ficheros local con el usuario del sistema con el que se ha accedido, por tanto el sistema de permisos UNIX es aplicable a un servidor GIT+SSH. Los servidores SSH aceptan varios métodos de acceso a la máquina remota, se van a probar estos médotos a

continuación.

Tarea 13.- Desde la máquina anfitrión clone el repositorio que creo como jefe de proyecto, para ello utilice git con el protocolo SSH y con el usuario jefe de proyecto y la clave correspondiente. C17:¿Qué dirección completa ha indicado a git para clonar el repositorio?

T13.1.- Una vez clonado el repositorio habrá recibido un aviso indicando ha clonado un repositorio vacío. Cree la rama *master* y ejecute *commit* y *push* para subir la rama *master*.

T13.2.- Ahora el objetivo es que no solicite clave el servidor SSH para el usuario jefe desde la máquina anfitrión. Para ello cree el par de claves RSA pública y privada usando el comando `ssh-keygen` en la máquina anfitrión. C18:¿Dónde hay que guardar la clave pública? ¿Y la clave privada?

T13.3.- Ahora debe configurar la cuenta del jefe de proyecto en la máquina virtual para que acepte conexiones sin solicitar la clave. Para ello busque información sobre el uso de claves RSA para autenticación SSH. C19:¿Qué modificaciones ha tenido que hacer en la cuenta de jefe de proyecto?

Tarea 14.- El objetivo ahora es crear un repositorio sólo accesible a unos determinados usuarios del sistema. Cree un grupo llamado *grupo-secreto* usando el comando `addgroup`.

T14.1.- Añada a uno de los desarrolladores existentes al grupo *grupo-secreto* usando el comando `adduser`.

T14.2.- En /opt/git cree un nuevo repositorio en modo *bare* usando la opción `--shared=group` llamado *proyecto-secreto.git* y establezca para este repositorio el grupo *grupo-secreto* de manera recursiva, use la opción -R en este comando: `chgrp -R <grupo> <fichero>`. C20:¿Cuál es la función de la opción --shared en git?

T14.3.- Establezca para este directorio los permisos de grupo usando el comando `chmod -R g+srw /opt/git/proyecto-secreto.git`. El efecto de este comando es es equivalente a la opción --shared=group de git, por ello no habrá observado ningún cambio en los permisos. C21:¿Cuál es la función del permiso 's' entonces?

T14.4.- Del mismo modo, quite todos los permisos para el resto de usuarios del sistema usando el comando `chmod`., así conseguirá que solo os usuarios del *grupo-secreto* puedan acceder a este repositorio. C22:¿Cuáles son los permisos definitivos para este directorio y todo su contenido?

T14.5.- Compruebe que sólo los usuarios pertenecientes al grupo *grupo-secreto* tiene acceso de lectura y escritura al repositorio mediante *git* y SSH. Desde clonar el repositorio desde la máquina anfitrión intentando acceder con diferentes usuarios de la máquina virtual.

3.3. Repositorio público solo lectura

En muchas situaciones se desea poner un repositorio accesible de forma pública sin utilizar el sistema de usuarios y permisos UNIX. En primer lugar se propone realizar una infraestructura con repositorios públicos con un servidor y evitando el protocolo SSH. Este tipo de soluciones permiten utilizarla metodología *pull-request*, consistente en tener un repositorio público solo lectura e ir aceptando la peticiones de mezcla de diferentes desarrolladores que colaboran en el proyecto, también disponibles en repositorios públicos.

Una primera solución es utilizar *gitweb*, se realizará la instalación usando el servidor Web Apache y

comprobaremos algunas limitaciones. Se debe considerar que el servidor Web Apache se ejecuta en el sistema como el usuario *www-data* y pertenece al grupo *www-data*, este aspecto es clave para controlar los proyectos que deseamos que sean visibles mediante HTTP.

Tarea 15.- Antes de continuar se creará otro repositorio para realizar pruebas sobre él. Cree un nuevo repositorio en /opt/git/proyecto-limitado.git que sea accesible en modo solo lectura por todos los usuarios del sistema.

T15.1.- Establezca permisos los permisos adecuados para que sólo los usuarios pertenecientes a un nuevo grupo llamado *desarrolladores* puedan hacer cambios.

T15.2.- Retire una copia local del nuevo repositorio y cree dos ramas, *master* y *dev* con al menos 2 commits en cada una y envíe los cambios al repositorio mediante *push*.

T15.3.- Clone un tercer repositorio desde algún sitio de la red para tener otro repositorio más de pruebas.

Tarea 16.- Instale el paquete *gitweb* y navegue a http://localhost/gitweb/ para ver si funciona. Debería mostrarse una página de listado de proyectos vacío inicialmente. Si no funciona, puede deberse un error aún no corregido en la versión *Ubuntu 14.04*, consulte el anexo 1 de este documento.

T16.1.- Debe editar el fichero /etc/gitweb.conf y establecer raíz de proyecto `/opt/git`. Recargue la página anterior.

T16.2.- Considere hacer públicos por HTTP solo dos proyectos, el proyecto-limitado y el obtenido en T15.3.-. Para ello, establezca los permisos adecuadamente, sabiendo que, como se indicó anteriormente el usuario y/o grupo *www-data* son accesibles por el servidor Web Apache. Haga las pruebas que considere necesarias. C23: ¿Qué diferentes combinaciones de permisos ha encontrado para controlar la visibilidad de los proyectos mediando HTTP? ¿Cuál cree que es mejor y por qué?

T16.3.- Compare la vista obtenida con *gitweb* con la del programa *gitg* y navegue por diferentes ramas.

T16.4.- Consiga establecer una descripción adecuada para cada proyecto en la vista Web, ya que se está mostrando el texto "*named repository...*".

T16.5.- Intente clonar con *git* desde su máquina anfitrión usando la dirección HTTP de uno de los proyectos. C24: ¿Es posible?

Dadas las limitaciones de GitWeb para clonar y trabajar directamente con git sobre HTTP se probará otra configuración usando el servidor Web Apache para servir un repositorio público en modo sólo lectura.

Tarea 17.- Git soporta el protocolo WEBDAV el cual ya fue utilizado anteriormente para servir repositorios subversion. Compruebe si tiene activado el soporte WEBDAV en Apache usando el comando `a2enmod dav_fs`.

T17.1.- En los sistemas Ubuntu/Debian se deben ubicar las páginas y sitios Web servidos en el directorio /var/www del sistema. Cree un nuevo directorio /var/www/git para que sea visible desde la dirección http://localhost/git.

T17.2.- El siguiente paso es configurar este directorio del servidor Web para que sea visible y admita el protocolo WEBDAV permitiendo que se listen las capetas y archivos. Debe editar el fichero /etc/apache2/sites-enabled/000-default.conf y añadir dentro de la sección `<VirtualHost> …`

`</VirtualHost>` las siguientes directivas:

```
        Alias /git /var/www/git
        <Directory "/var/www/git" >
           DAV on
           Options Indexes MultiViews FollowSymLinks
        </directory>
```

Código 6. Configuración de Apache para servir un repositorio Git con WEBDAV.

T17.3.- Reinicie el servidor Web y compruebe que no ha cometido errores de configuración navegando a la dirección http://localhost/git.

T17.4.- El procedimiento para publicar los repositorios deseados consistirá en realizar enlaces simbólicos hacia los repositorios que se deseen publicar, recuerde que están en /opt/git. Además, hay que establecer correctamente los permisos UNIX de cada repositorio. En la Tarea 15.- creó un repositorio limitado, ahora este repositorio debe estar disponible por HTTP, para ello realice un enlace simbólico desde /var/www/git/proyecto-limitado.git a /opt/git/proyecto-limitado.git. Use los siguientes comandos:

```
cd /var/www/git/
ln -s /opt/git/proyecto-limitado.git
```

Código 7. Comando para crear enlaces simbólicos.

T17.5.- Navegue a http://localhost/git para ver el repositorio que acaba de enlazar. Intente clonar el repositorio con git mediante `git clone http://localhost/git/proyecto-limitado.git`. C25: Si ha obtenido un error al clonar ¿como se soluciona?

Tarea 18.- En la tarea anterior se mostró un procedimiento para publicar un repositorio con Apache, pero operaba en modo solo lectura. Para comprobarlo intente subir cambios con git.

T18.1.- Como Apache se ejecuta con el usuario *www-data*, cree un nuevo repositorio en /opt/git/proyecto-anarquico.git, establezca como usuario propietario de forma recursiva a *www-data* y enlácelo desde `/var/www/git`.

T18.2.- Pruebe clonar el repositorio desde la máquina anfitrión y realizar cambios en el repositorio C26: ¿Cuales son las desventajas de tener un repositorio con esta configuración?

Tarea 19.- Del mismo modo que se hizo con Subversion se limitará el acceso HTTP solicitando autenticación. Del mismo modo que se hizo en la Tarea 6.- cree un archivo llamado git.passwd en la misma ubicación y añada algún usuario con clave mediante el comando htpass.

T19.1.- Cambie la configuración del sitio Web como se indica en rojo en el código 8 y reinicie el servidor. Pruebe a clonar repositorios para comprobar si el servidor solicita autenticación.

```
        <Directory "/var/www/git" >
           DAV on
           Options Indexes MultiViews FollowSymLinks
           AuthType Basic
           AuthName "Repositorio GIT"
           AuthUserFile /etc/apache2/git.passwd
           Require valid-user
        </directory>
```

Código 8. Configuración WEBDAV para un directorio en Apache.

T19.2.- Clone de nuevo el repositorio y compruebe que se le solicita el usuario y contraseña para

acceder al repositorio.

T19.3.- Intente subir cambios al repositorio, si recibe un error debe cambiar la URL del repositorio de forma que incluya el nombre de usuario (pero no la contraseña). C27: ¿Cómo ha cambiado la URL del repositorio sin volver a clonarlo? ¿Cuál es la nueva URL?

Tarea 20.- La última configuración consistirá en conseguir disponer de repositorios en modo solo lectura públicos (sin solicitar contraseña), pero solo disponibles en escrituras para usuarios autenticados.

T20.1.- En el archivo de configuración de apache se debe cambiar para el directorio git la opción que solicita un usuario solo para ciertos comando WEBDAV, por simplicidad se limitará solo el comando PUT, aunque no es la forma correcta de hacerlo. Realice los cambios indicados en el código 9.

```
     <Location "/git" >
       DAV on
       Options Indexes MultiViews FollowSymLinks
       AuthType Basic
       AuthName "Repositorio GIT"
       AuthUserFile /etc/apache2/git.passwd
       Require valid-user
       <limit PUT>
           Require valid-user
       </limit>
    </location>
```

Código 9. Configuración WEBDAV con limitaciones.

T20.2.- Reinicie Apache y clone de nuevo un repositorio para verificar que no le ha solicitado contraseña. Realice cambios e intente subirlos, en caso de error, considere el cambio de URL tratado en T19.3.-.

Tarea 21.- Opcional: Al no existir como en subversion un fichero de permisos para otorgar diferentes niveles de acceso a los usuarios incluidos en en fichero *git.passwd* es necesario controlar el acceso a diferentes repositorio mediante directivas de apache. Busque en la red y plantee una posible solución combinando directivas de apache: *Location*, *Limit*, *Limit Except*, *Require user*, etc.

3.4. Protocolo GIT

Otra posible configuración, aunque menos habitual, consiste en utilizar el propio protocolo git para servir un repositorio. Git por sí solo es capaz de servir un repositorio mediante la ejecución del comando `git daemon` dentro de un repositorio, quedando el proceso escuchando en el puerto TCP 9418. Usando este procedimiento necesitaríamos un puerto diferente para cada repositorio a servir, por ello se propone utilizar una configuración mejorada e integrada en un paquete *deb* preconfigurado.

Tarea 22.- Instale el paquete *git-daemon-sysvinit* y lea la documentación existente en /usr/share/doc/git-git-daemon-sysvinit. Siguiendo esa documentación debe conseguir lo siguiente:

T22.1.- Al reiniciarse el sistema el *git-daemon* debe iniciarse automáticamente, se deben servir dos de los proyectos de la tabla 4 mediante este servicio. C28: ¿Qué fichero de configuración hay que cambiar?.

T22.2.- Gitdaemon se ejecuta con el usuario del sistema *gitdaemon*, debe establecer los permisos adecuados en los repositorios para que este programa pueda leer los ficheros. C29: ¿Que permisos

ha establecido? ¿Existe otra posible solución para los permisos?

T22.3.- Intente clonar un repositorio mediante el protocolo *git* para verificar si su configuración es correcta. Por ejemplo, use el comando `git clone git://localhost/git/opensudoku.git`, si obtiene un error debe hacer el repositorio exportable. Para ello ejecute `touch git-daemon-export-ok` dentro del directorio del repositorio.

T22.4.- Si intenta subir cambios al repositorio por el protocolo git no podrá, de forma predeterminada *git-daemon* exporta los repositorios en solo lectura. Para activar la escritura ejecute `git config daemon.receivepack true` dentro del repositorio. C30: ¿Soporta el protocolo git autenticación de usuarios? Indique de donde ha obtenido la respuesta.

Anexo 1: Solución al bug #1329542 en en paquete gitweb

Copiar la configuración de gitweb manualmente a apache2

```
sudo cp /etc/apache2/conf.d/gitweb /etc/apache2/conf-available/gitweb.conf
cd /etc/apache2/conf-enabled
sudo ln -s ../conf-available/gitweb.conf
```

Añadir "+" en `FollowSymLinksal` editando el fichero copiado anteriormente (el lugar indicado en negrita+rojo)

```
Alias /gitweb /usr/share/gitweb

<Directory /usr/share/gitweb>
  Options +FollowSymLinks +ExecCGI
  AddHandler cgi-script .cgi
</Directory>
```

Activar los módulos cgi y perl de apache2 mediante los comandos

```
sudo a2enmod cgi
sudo service apache2 restart
```