

Práctica 2. Clonar la información de un sitio web

Duración: 2 sesiones

1. Objetivos de la práctica

Los objetivos concretos de esta segunda práctica son:

- aprender a copiar archivos mediante ssh
- clonar contenido entre máquinas
- configurar el ssh para acceder a máquinas remotas sin contraseña
- establecer tareas en cron

2. Crear un tar con ficheros locales en un equipo remoto

En el caso que necesitemos crear un tar.gz de un equipo y dejarlo en otro pero no disponemos de espacio en disco local, podemos usar ssh para crearlo directamente en el equipo destino.

Para ello, deberemos indicar al comando tar que queremos que use stdout como destino y mandar con una pipe la salida al ssh. Éste debe coger la salida del tar y escribirla en un fichero. El comando quedaría:

```
tar czf - directorio | ssh equipodestino 'cat > ~/tar.tgz'
```

De esta forma en el equipo de destino tendremos creado el archivo tar.tgz

Sin embargo, esto que puede ser útil en un momento dado, no nos servirá para sincronizar grandes cantidades de información. En ese caso, va mejor la herramienta rsync.

3. Instalar la herramienta rsync

La página web de esta herramienta, con toda la documentación, descargas, etc, se encuentra en:

```
http://rsync.samba.org/
```

Disponemos de versiones compiladas para diferentes sistemas en:

```
http://rsync.samba.org/download.html
```

En nuestras máquinas Ubuntu podemos instalarlo (si no lo está) usando apt-get:

```
sudo apt-get install rsync
```

Es conveniente haber activado la cuenta de root para trabajar con todos los permisos sobre todos los ficheros. Para ello, como vimos, ejecutaremos:

```
sudo passwd root
```

Nos pedirá la clave, y ahora que la tenemos activa, conviene que nos salgamos de la sesión de usuario y entremos como root (siempre).

Ahora ya podemos probar el funcionamiento del rsync, clonando una carpeta cualquiera. Por ejemplo, para clonar la carpeta con el contenido del servidor web principal, en la máquina 2 (secundaria) ejecutaremos:

```
rsync -avz -e ssh root@maquina1:/var/www/ /var/www/
```

Nos pedirá la clave de root en la máquina principal, y tras unos segundos, podremos comprobar que el directorio /var/www queda clonado de forma idéntica en ambas máquinas (de la máquina-1 se copia en la máquina-2):

```
ls -la /var/www
```

Los directorios y ficheros copiados en la máquina destino mantienen los permisos y dueño que en la máquina origen.

Dependiendo de la versión del sistema operativo y su configuración, es posible que el acceso por SSH para root esté deshabilitado. En ese caso, en la máquina principal debemos editar el archivo de configuración /etc/ssh/sshd_config y allí buscar la línea correspondiente al parámetro PermitRootLogin y ponerlo tal y como sigue:

```
PermitRootLogin yes
```

Guardamos los cambios y reiniciamos el servicio SSH:

```
service ssh restart
```

Ahora sí debería dejarnos acceder al servidor principal de forma remota por SSH identificados como root. Si aún así, Linux no nos deja trabajar remotamente como root, entonces se recomienda trabajar siempre como el usuario sin privilegios de root.

Por otro lado, la herramienta rsync nos permite especificar qué directorios copiar y cuáles ignorar en el proceso de copia, de forma que no se sobrescriban los archivos que no queramos. Así, si hacemos:

```
rsync -avz --delete --exclude=**/stats --exclude=**/error --  
exclude=**/files/pictures -e "ssh -l root"  
root@maquina1:/var/www/ /var/www/
```

estaremos haciendo la copia completa del directorio /var/www pero excluyendo /var/www/error , /var/www/stats y /var/www/files/pictures

La opción --delete indica que aquellos ficheros que se hayan eliminado en la máquina origen, también se borren en la máquina destino (para que el clonado sea perfecto). La opción --exclude indica que ciertos directorios o ficheros no deben copiarse (p.ej., archivos de log). De esta forma, en la orden de ejemplo anterior, cuando indicamos --exclude=**/error significa "no copies lo que hay en /var/www/error" ya que corresponde a mensajes de la máquina original (y la segunda máquina ya generará sus propios mensajes).

Esta orden más compleja será más útil en entornos reales, aunque para la práctica será suficiente con usar la que hay más arriba, al inicio de esta página.

4. Acceso sin contraseña para ssh

El uso de rsync nos será de gran ayuda para mantener el contenido de varias máquinas actualizado e idéntico en todas ellas. Sin embargo, esa actualización debería hacerse automáticamente, a través de scripts que no requieran la intervención del administrador que vaya tecleando las contraseñas.

Es más, al realizar scripts que se conectan mediante ssh a equipos remotos para ejecutar alguna acción (p.ej. copias de seguridad o clonado) nos podemos encontrar que se queden esperando la contraseña.

Veamos cómo ejecutar comandos en equipos remotos mediante ssh sin contraseña. Para ello, normalmente se usa autenticación con un par de claves pública-privada.

Mediante ssh-keygen podemos generar la clave, con la opción -t para el tipo de clave. Así, en la máquina secundaria ejecutaremos:

```
ssh-keygen -t dsa

Generating public/private dsa key pair.
Enter file in which to save the key (/home/pedro/.ssh/id_dsa):
Created directory '~/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ~/.ssh/id_dsa.
Your public key has been saved in ~/.ssh/id_dsa.pub.
The key fingerprint is:
57:2c:00:be:ef:00:0f:00:ba:be:00:19:c4:8e:be:73 pedro@maquina1
```

Según el tipo de claves a usar, se generan dos ficheros:

- **rsa1**: Generará, por defecto, el fichero ~/.ssh/identity para la clave privada y el fichero ~/.ssh/identity.pub para la clave pública. Este formato es válido para el protocolo 1 de SSH
- **rsa**: Generará, por defecto, el fichero ~/.ssh/id_rsa para la clave privada y el fichero ~/.ssh/id_rsa.pub para la clave pública. Este formato es válido para el protocolo 2 de SSH
- **dsa**: Generará, por defecto, el fichero ~/.ssh/id_dsa para la clave privada y el fichero ~/.ssh/id_dsa.pub para la clave pública. Este formato es válido para el protocolo 2 de SSH

La *passphrase* que nos pide es para añadir seguridad a la clave privada; se trata de una contraseña. En el caso de querer conectar a equipos sin contraseña debemos dejarla en blanco.

A continuación deberemos copiar la clave pública al equipo remoto (máquina principal) añadiéndola al fichero ~/.ssh/authorized_keys, que deberá tener permisos 600 (por defecto esto estará bien configurado; sólo si nos da algún error debemos hacerlo):

```
chmod 600 ~/.ssh/authorized_keys
```

Para hacer la copia de la clave existe una forma sencilla y elegante. Se trata de usar el comando ssh-copy-id, que viene integrado con el comando ssh. Lo que haremos es copiarla a la máquina principal (a la que queremos acceder luego desde la secundaria):

```
ssh-copy-id -i .ssh/id_dsa.pub root@maquina1
```

A continuación ya podremos conectarnos a dicho equipo sin contraseña:

```
[root@maquina2 ~]# ssh maquina1 -l root
Last login: Sun May 1 19:15:16 2011 from maquina2
CentOS release 5.5 (Final)
Linux maquina1 2.6.34.6-xxxx-grs-ipv6-64 #3 SMP Fri Sep 17
16:06:38 UTC 2010 x86_64 x86_64 x86_64 GNU/Linux
[root@maquina1 ~]#
```

Para ejecutar comandos simplemente deberemos añadirlo al final del comando ssh para conectarnos:

```
[root@maquina2 ~]# ssh maquina1 -l root uname -a
Linux maquina1 2.6.34.6-xxxx-grs-ipv6-64 #3 SMP Fri Sep 17
16:06:38 UTC 2010 x86_64 x86_64 x86_64 GNU/Linux
```

En el caso que por algún motivo se perdiese la clave del ~/.ssh/authorized_keys nos pediría contraseña de nuevo.

Teniendo preparado el acceso por ssh sin contraseña, podemos hacer uso del rsync desde scripts que se ejecuten automáticamente con el cron (p.ej. cada noche de madrugada).

5. Programar tareas con crontab

cron es un administrador procesos en segundo plano que ejecuta procesos en el instante indicado en el fichero crontab.

cron se ejecuta en background y revisa cada minuto la tabla del fichero /etc/crontab en búsqueda de tareas que se deban ejecutar (si ha llegado su momento). Podemos agregar nuevas tareas a cron para automatizar algunos procesos (p.ej. copias de seguridad).

Para ello, debemos editar el archivo /etc/crontab que normalmente tendrá un aspecto similar al siguiente:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin

01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

En ese ejemplo vemos tareas que se ejecutan como usuario root, cada mes, cada semana, cada día o cada hora.

No hay límites de cuantas tareas podemos tener. Lo importante es cuidar la sintaxis y número de campos de cada línea del fichero. Los 7 campos que forman estas líneas están organizados de la siguiente manera:

Minuto	Hora	DíaDelMes	Mes	DíaDeLaSemana	Usuario	Comando
--------	------	-----------	-----	---------------	---------	---------

- Minuto: indica el minuto de la hora en que el comando será ejecutado, este valor debe de estar entre 0 y 59.
- Hora: indica la hora en que el comando será ejecutado, se especifica en un formato de 24 horas, los valores deben estar entre 0 y 23, 0 es medianoche.
- DíaDelMes: indica el día del mes en que se quiere ejecutar el comando. Por ejemplo se indicaría 20, para ejecutar el comando el día 20 de cada mes.
- Mes: indica el mes en que el comando se ejecutará (1-12).
- DíaDeLaSemana: indica el día de la semana en que se ejecutará el comando (1=lunes y hasta 7=domingo).
- Usuario: indica el usuario que ejecuta el comando.

- Comando: indica el comando que se desea ejecutar. Este campo puede contener múltiples palabras y espacios.

Un asterisco * como valor en los primeros cinco campos indicará el valor "todo". Así, un * en el campo de minuto indicará todos los minutos de la hora.

Ejemplos:

La siguiente tarea apagará el ordenador cada día a las 00:30h :

```
30 0 * * * root /sbin/shutdown -h now
```

La siguiente tarea eliminará todos los archivos de la carpeta /tmp todos los jueves (día 4) a la 15:30h :

```
30 15 * * 4 root rm /tmp/*
```

Por último, la siguiente tarea se ejecutará cada tres minutos:

```
*/3 * * * * root rm /tmp/*
```

Cuestiones a resolver

En esta práctica el objetivo es configurar las máquinas virtuales para trabajar en modo espejo, consiguiendo que una máquina secundaria mantenga siempre actualizada la información que hay en la máquina servidora principal.

Hay que llevar a cabo las siguientes tareas:

1. probar el funcionamiento de la copia de archivos por ssh
2. clonado de una carpeta entre las dos máquinas
3. configuración de ssh para acceder sin que solicite contraseña
4. establecer una tarea en cron que se ejecute cada hora para mantener actualizado el contenido del directorio /var/www entre las dos máquinas

Como resultado de la práctica 2 se mostrará al profesor el funcionamiento del proceso de clonado de la información. En el documento a entregar se describirá cómo se ha realizado la configuración de ambas máquinas y del software.

Normas de entrega

La práctica podrá realizarse de manera individual o por grupos de hasta 2 personas.

Se entregará como un archivo de texto en el que se muestre la información requerida. También se puede utilizar la sintaxis de Markdown para conseguir una mejor presentación e incluso integrar imágenes o capturas de pantalla. La entrega se realizará subiendo los archivos necesarios al repositorio SWAP en la cuenta de github del alumno, a una carpeta llamada "practica2".

Toda la documentación y material exigidos se entregarán en la fecha indicada por el profesor. No se recogerá ni admitirá la entrega posterior de las prácticas ni de parte de las mismas.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto del autor del original como de quien las copió).

Las faltas de ortografía se penalizarán con hasta 1 punto de la nota de la práctica.