

ADMINISTRACIÓN REMOTA

Existen muchas formas de administrar remotamente un servidor Linux pero nosotros veremos algunas de las herramientas más usadas en la actualidad por su potencia y versatilidad como son **SSH, SCP o SFTP incluidas en OpenSSH**.

OpenSSH Server

OpenSSH es una aplicación sobre el protocolo Secure Shell (SSH) que nos ofrece un conjunto de herramientas para el control remoto de equipos o la transferencia de datos entre equipos. Podemos administrar remotamente usando otras herramientas como el Telnet, pero no aportan el nivel de fiabilidad y confidencialidad que nos aporta el protocolo SSH al tener una comunicación encriptada. El servidor sshd instalado con OpenSSH estará siempre que lo tengamos activado a la escucha de posibles clientes que se quieran conectar a él.

Este protocolo es muy útil para los administradores de servidores para poder gestionar el servidor remotamente. Supongamos que el servidor que tiene alojada la página web del centro no responde a las peticiones que se le hacen y nosotros, administradores de ese servidor, estamos en un aula distinta a la que se encuentra dicho servidor. Para reiniciar el servicio web de nuestro servidor tendríamos que abandonar la clase e ir hasta el aula donde se encontrara el servidor para poder reiniciarlo. En cambio, usando SSH con un simple comando desde cualquier equipo del aula donde nos encontráramos podríamos reiniciar el servicio. Podríamos poner muchos más ejemplos donde SSH nos puede sacar de más de un apuro.

Más información:

<http://es.wikipedia.org/wiki/OpenSSH>

<http://www.openssh.org/es/>

INSTALACIÓN

Lo primero que haremos es comprobar si tenemos instalado los paquetes de servidor y cliente de SSH.

aptitude show [nombre-paquete]

aptitude show openssh-server |more

Donde [nombre-paquete] sería openssh-client y openssh-server

En la imagen vemos como el valor **State: installed** indica que el paquete esta instalado.

```
Package: openssh-server
State: installed
Automatically installed: no
Version: 1:5.8p1-7ubuntu1
Priority: opcional
Section: net
Maintainer: Colin Watson <cjwatson@ubuntu.com>
Uncompressed Size: 897 k
Depends: libc6 (>= 2.8), libcomerr2 (>= 1.01), libgssapi-krb5-2 (>= 1.8+dfsg),
        libkrb5-3 (>= 1.6.dfsg.2), libpam0g (>= 0.99.7.1), libselinux1 (>=
        1.32), libssl1.0.0 (>= 1.0.0), libwrap0 (>= 7.6-4~), zlib1g (>=
        1:1.1.4), debconf (>= 1.2.0) | debconf-2.0, openssh-client (=
        1:5.8p1-7ubuntu1), upstart-job, libpam-runtime (>= 0.76-14),
        libpam-modules (>= 0.72-9), adduser (>= 3.9), dpkg (>= 1.9.0), lsb-base
        (>= 3.2-13), procps
Recommends: xauth, ssh-import-id
Suggests: ssh-askpass, rssh, molly-guard, openssh-blacklist,
        openssh-blacklist-extra, ufw, monkeysphere
Conflicts: rsh-client (< 0.16.1-1), sftp, ssh (< 1:3.8.1p1-9), ssh-krb5 (<
        1:4.3p2-7), ssh-nonfree (< 2), ssh-socks, ssh2
Replaces: openssh-client (< 1:3.8.1p1-11), ssh, ssh-krb5
Provides: ssh-server
Description: servidor de terminal seguro (SSH) para acceder de forma segura desde
e máquinas remotas
Esta es la versión adaptable de OpenSSH, una implementación libre del protocolo
«Secure Shell» como especifica el grupo de trabajo secsh del IETF.

Ssh (Secure Shell) es una aplicación para acceder y ejecutar comandos en una
máquina remota. Provee conexiones encriptadas y seguras entre dos huéspedes no
--Más--
```

Si no hemos instalado el servicio durante la instalación del sistema operativo, podemos recurrir al comando que vimos antes **tasksel** ([sudo tasksel install openssh-server](#)), o hacerlo mediante **apt-get** usando el comando:

sudo apt-get install openssh-client

Para instalar el **cliente** y

sudo apt-get install openssh-server

para hacernos con el **servidor**.

CONFIGURACIÓN

Para configurar como se comportará por defecto nuestro servidor OpenSSH tenemos que editar el archivo **/etc/ssh/sshd_config**

Antes de ponernos a editar vamos a hacer una **copia de seguridad** con la configuración original del archivo por si en algún momento queremos volver a la configuración por

defecto, además cambiaremos los **permisos** sobre ese archivo para que solo sea de **lectura** y así evitar modificarlo por error. Para hacerlo nos vamos a un terminal y escribimos:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.ori
```

```
sudo chmod a-w /etc/ssh/sshd_config.ori
```

Vemos como hemos hecho una copia del archivo original:

```
enrique@ServidorCep:/etc/ssh$ ls
moduli          sshd_config      ssh_host_dsa_key.pub  ssh_host_rsa_key
ssh_config       sshd_config.ori  ssh_host_ecdsa_key    ssh_host_rsa_key.pub
ssh_config.ori  ssh_host_dsa_key ssh_host_ecdsa_key.pub ssh_import_id
enrique@ServidorCep:/etc/ssh$ _
```

Y vemos también como los permisos de la copia del archivo original son de sólo lectura:

```
enrique@ServidorCep:/etc/ssh$ ls -l
total 168
-rw-r--r-- 1 root root 125749 2011-07-29 18:02 moduli
-rw-r--r-- 1 root root   1669 2011-07-29 18:02 ssh_config
-rw-r--r-- 1 root root   1669 2012-02-05 20:06 ssh_config.ori
-rw-r--r-- 1 root root   2489 2012-02-01 00:35 sshd_config
-r--r--r-- 1 root root   2489 2012-02-05 20:10 sshd_config.ori
-rw----- 1 root root    668 2012-02-01 00:35 ssh_host_dsa_key
-rw-r--r-- 1 root root    606 2012-02-01 00:35 ssh_host_dsa_key.pub
-rw----- 1 root root    227 2012-02-01 00:35 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    178 2012-02-01 00:35 ssh_host_ecdsa_key.pub
-rw----- 1 root root   1679 2012-02-01 00:35 ssh_host_rsa_key
-rw-r--r-- 1 root root    398 2012-02-01 00:35 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    302 2011-01-10 19:48 ssh_import_id
enrique@ServidorCep:/etc/ssh$ _
```

Archivos de configuración de OpenSSH

OpenSSH dispone de dos conjuntos diferentes de ficheros de configuración: uno para el cliente (ssh, scp y sftp) y otro orientado completamente al servidor.

Archivos de configuración del lado del servidor

La ubicación de los ficheros referentes al servidor se encuentran en la ruta:

/etc/ssh/

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

moduli	Contiene grupos Diffie-Hellman usados para el intercambio de la clave Diffie-Hellman que es imprescindible para la construcción de una capa de transporte seguro. Cuando se intercambian las claves al inicio de una sesión SSH, se crea un valor secreto y compartido que no puede ser determinado por
---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	ninguna de las partes individualmente. Este valor se usa para proporcionar la autenticación del host.
ssh_config	El archivo de configuración del sistema cliente SSH por defecto. Este archivo se sobrescribe si hay alguno ya presente en el directorio principal del usuario.
sshd_config	El archivo de configuración para el demonio sshd.
ssh_host_dsa_key	La clave privada DSA usada por el demonio sshd
ssh_host_dsa_key.pub	La clave pública DSA usada por el demonio sshd
ssh_host_key	La clave privada RSA usada por el demonio sshd para la versión 1 del protocolo SSH.
ssh_host_key.pub	La clave pública RSA usada por el demonio sshd para la versión 1 del protocolo SSH.
ssh_host_rsa_key	La clave privada RSA usada por el demonio sshd para la versión 2 del protocolo SSH.
ssh_host_rsa_key.pub	La clave pública RSA usada por el demonio sshd para la versión 2 del protocolo SSH.

Archivos de configuración del lado del cliente

La ubicación de los ficheros referentes al cliente se encuentran almacenados en el directorio de trabajo de cada usuario:

/home/usuario/

/home/enrique/

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

authorized_keys	Este archivo contiene una lista de claves públicas autorizadas. Cuando un cliente se conecta al servidor, el servidor autentica al cliente chequeando su clave pública firmada almacenada dentro de este archivo.
id_dsa	Contiene la clave privada DSA del usuario.
id_dsa.pub	La clave pública DSA del usuario.
id_rsa	La clave RSA privada usada por ssh para la versión 2 del protocolo SSH.
id_rsa.pub	La clave pública RSA usada por ssh para la versión 2 del protocolo SSH.
identity.	La clave privada RSA usada por ssh para la versión 1 del protocolo SSH.
identity.pub	La clave pública RSA usada por ssh para la versión 1 del protocolo SSH.
known_hosts	Este archivo contiene las claves de host DSA de los servidores SSH a los cuales el usuario ha accedido. Este archivo es muy importante para asegurar que el cliente SSH está conectado al servidor SSH correcto.

Configuración de fichero sshd_config

A continuación vamos a ver como configurar el servidor SSH de manera que evitemos algunos de los agujeros de seguridad más frecuentes que se pueden plantear usando este servicio que se caracteriza por ser bastante más seguro que otros protocolos de comunicación remota.

El fichero sshd_config lo podremos localizar en en la siguiente ruta:

/etc/ssh/

El siguiente paso será abrir el fichero con la ayuda del editor de textos (**nano**, **vi**, **gedit**)

sudo nano /etc/ssh/sshd_config

A partir de este punto comenzaremos a proteger nuestro servidor SSH.

Cambiando el puerto por defecto

Cuando un atacante intenta por primera vez acceder a un sistema, uno de los pasos más frecuentes a realizar por el atacante es un escaneo de puertos (usando el paquete **nmap** por ejemplo, o incluso a través de un teléfono móvil con aplicaciones como **iNet** para terminales iPhone, o **Overlook Find** para terminales Android). Con este escaneo el atacante puede saber que puertos tiene abiertos el servidor y por lo tanto que tipo de servicios está ofreciendo. Por ejemplo, si al realizarle un escaneo a una máquina detecta el puerto 80 abierto, lo más probable es que esta máquina este actuando de servidor web. El puerto usado por SSH por defecto es el 22, vamos a ver como cambiarlo de manera que no quede asociado al servicio y ponerle las cosas un poco más difíciles a algún intruso.

Como hasta el puerto 1024 la mayoría están reservados por protocolos de uso común, se recomienda usar un puerto cualquiera por encima del 1024, así que elegiremos uno cualquiera por encima de dicho puerto como por ejemplo el 2323. Para hacerlo tendremos que cambiar el parámetro **Port** del fichero de configuración el cual deberá quedar así:

Estas líneas son líneas comentadas. Siempre que este la almohadilla delante

no tendrán efecto dentro de la configuración.

What ports, IPs and protocols we listen for

Port 2323

```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 2323
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
```

Desactivando el Protocolo 1

Hay dos versiones de ssh en cuanto a su protocolo de comunicación, estas son:

- Versión 1
- Versión 2.

Como la versión 1 del protocolo de OpenSSH han dejado de darle soporte los creadores, puede que alguno de los algoritmos de codificación usados en dicho protocolo presenten algunas brechas de seguridad que podrían llegar a ser un problema. Para evitar el uso de este protocolo, limitaremos en la configuración el uso de la versión 2 del protocolo editando el archivo de configuración en su parámetro Protocol.

```
# En las últimas versiones de OpenSSH ya viene desactivado el Protocol 1.
# ListenAddress ::
# ListenAddress 0.0.0.0
```

Protocol 2

Deshabilitando el acceso a root

Normalmente en los sistemas **Linux** cuando se instala el sistema se crea un superusuario llamado **root** con privilegios totales sobre la máquina, de parecida forma a como en los sistemas **Windows** se crea un usuario **Administrador** al instalar el sistema pero con la diferencia de que el usuario **root** tiene mayor control y permisos para modificar hasta las partes más internas del sistema. Para los atacantes una forma de entrar en un sistema es usar este usuario, del cual ya saben su existencia, para poder modificar cualquier cosa una vez obtenido acceso como **root**. Por lo tanto proteger el acceso de este usuario y aplicarle una contraseña fuerte es necesario si queremos evitar posibles ataques asociados a esta “vulnerabilidad”. Nosotros vamos a evitar que alguien se pueda loguear directamente con este usuario cambiando el parámetro “**PermitRootLogin**” dentro de la configuración a “**no**”. De esta forma si alguna vez queremos usar el usuario **root** primero será necesario entrar con otro usuario y una vez dentro pasar a modo superusuario con el comando “**su -**”, o usando “**sudo**” para ordenes concretas como estamos haciendo en el curso. El fichero de configuración deberá quedar de la siguiente manera:

```
# Authentication:
```

```
LoginGraceTime 120
```

```
PermitRootLogin no
```

```
#StrictModes yes
```

```
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no_
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
```

Definiendo un número máximo de intentos de conexión

Muchos de los ataques que se pueden hacer a un sistema son usando el método de fuerza bruta que consiste en ir probando combinaciones posibles de claves hasta dar con la acertada. Estas pruebas que a un humano le llevarían mucho tiempo, para un ordenador con un procesador potente frente a una clave débil pueden ser segundos o minutos. Por eso el uso de una clave suficientemente larga , combinando números, símbolos y letras mayúsculas/minúsculas hacen que la mayoría de los ataques por fuerza bruta se estrellen. Además si restringimos el número de intentos de loguearse evitamos que se estén probando claves continuamente, con lo que dificultamos aún más este tipo de ataques.

Para llevar a cabo estos cambios tendrá que editar el parámetro “**MaxAuthTries**” del fichero de configuración el cual deberá quedar de la siguiente manera:

```
PermitRootLogin no
```

```
MaxAuthTries 2
```


El número 2 indica la cantidad de veces que podemos equivocarnos al ingresar el usuario y/o contraseña, en este caso después de dos intentos, se perderá o cerrará la conexión. Es totalmente posible volver a intentarlo, pero con solo dos intentos por vez.

Activando el modo estricto

La opción `StrictModes` especifica si el servidor ssh debe comprobar los permisos de usuarios en su directorio home y archivos `rhosts` antes de aceptar el login. Ésta opción siempre deberá ser afirmativa porque algunas veces un usuario puede dejar su directorio con permisos de escritura para todos. Para llevar a cabo estos cambios tendrá que editar el parámetro **StrictModes** del fichero de configuración el cual deberá quedar de la siguiente manera:

`PermitRootLogin no`

`StrictModes yes`

`MaxAuthTries 2`

```
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
MaxAuthTries 2

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
```

[Cancelado]

Limitando el tiempo para autenticarse con SSH

Vamos a limitar el tiempo que estará activa la pantalla de login para que un usuario se loguee en el sistema siendo suficiente 60 segundos, o incluso menos si sólo nosotros vamos a tener acceso. Para llevar a cabo estos cambios tendrá que editar el parámetro **LoginGraceTime** del fichero de configuración el cual deberá quedar de la siguiente manera:

```
# Authentication:
```

```
LoginGraceTime 60
```

```
PermitRootLogin no
```

```
StrictModes yes
```

```
MaxAuthTries 2
```

Especificando las direcciones IP de las interfaces que queremos que estén a la escucha:

Para una mayor seguridad si queremos que el servidor solo acepte conexiones SSH por alguna de sus interfaces de red podemos incluir la IP de la interfaz de red en el parámetro **ListenAddress**. Imaginemos que nuestro servidor dispone de dos direcciones IP siendo una pública y otra privada. Una forma de que sólo los ordenadores de la red privada (192.168.2.0 en nuestro caso) pudieran acceder por SSH, sería incluir la IP privada del servidor (192.168.2.100) en el parámetro ListenAddress. De esta forma el archivo de configuración debería quedar así:

Port 2323

```
#AddressFamily any
```

```
# ListenAddress 192.168.2.100
```

Si por el contrario queremos que el servidor esté a la escucha por cualquiera de sus interfaces, dejaremos esta línea comentada (con la almohadilla # delante). Nosotros para nuestras pruebas la dejaremos comentada puesto que en principio sólo vamos a acceder nosotros a través de la red de área local.

INICIANDO EL SERVICIO

Reiniciamos el servicio

Debemos ser cautelosos a la hora de modificar el archivo de configuración ya que

un error en él podría dejar el servicio apagado. Una vez que hemos modificado a nuestra conveniencia el archivo tenemos que reiniciar el servicio para que los cambios surtan efecto. Para reiniciar el servicio usamos el comando:

```
sudo service sshd restart
```

o

```
sudo /etc/init.d/ssh restart
```

Otros comandos que podemos usar son :

start	Inicia el servicio.
stop	Detiene el servicio.
restart	Reinicia el servicio.- La diferencia con reload está en que al ejecutar un restart este mata todos los procesos relacionado con el servicio y los vuelve a generar de nueva cuenta.
reload	Recarga el servicio.- La diferencia con restart radica en que al ejecutar un reload este solamente carga las actualizaciones hechas al fichero de configuración del servicio sin necesidad de matar los procesos relacionados con el mismo, por lo que podría entenderse que hace el cambio en caliente.
condrestart	Reinicio Condicional.- Solamente se inicia si el servicio se encuentra ejecutándose.
status	Da a conocer el estado en el que se encuentra el servicio.

Si queremos que el servicio se inicie siempre al arrancar el servidor ejecutamos el comando:

```
sudo chkconfig sshd on
```

con esto ya no haría falta iniciarlo manualmente pero si reiniciarlo si hacemos modificaciones en la configuración.

Para comprobar el estado del servicio ssh podemos verlo usando el comando

```
status ssh
```

```
enrique@ServidorCep:~/.ssh$ status ssh
ssh start/running, process 434
enrique@ServidorCep:~/.ssh$
```

Si el servicio esta iniciado aparecerá **start/running** si lo tenemos parado en cambio veremos el estado como **stop/waiting**.

UTILIZANDO SSH

Una vez que hemos visto como instalar y configurar el servicio es la hora de empezar a probarlo. Para ello necesitamos tener el equipo servidor y el cliente en red y recordar que en el sistema cliente debe tener también un cliente ssh para acceder al servidor usando ese protocolo.

ESTABLECIENDO LA CONEXIÓN

Como siempre existen varias formas de establecer una comunicación vía SSH. En el curso veremos algunas de las más sencillas y usadas.

Conexión por PuTTY

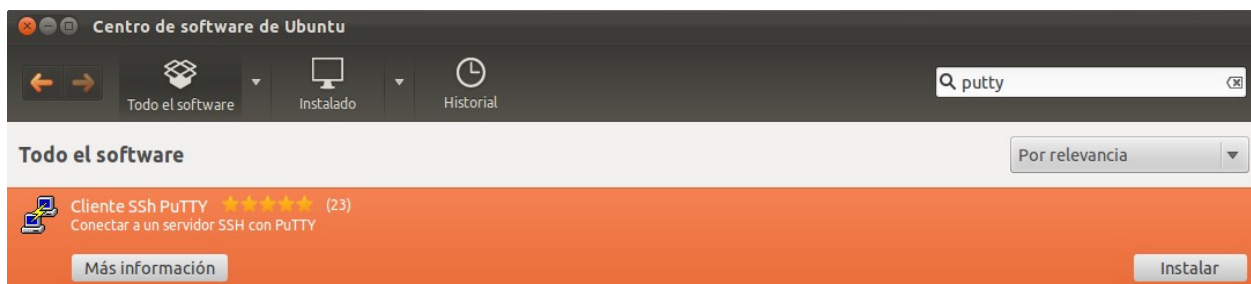
En **Linux y Windows** podemos establecer una comunicación mediante un cliente SSH con interfaz gráfica como **Putty** que es uno de los más estandarizados para este tipo de conexiones. Se puede descargar gratuitamente desde su página oficial:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

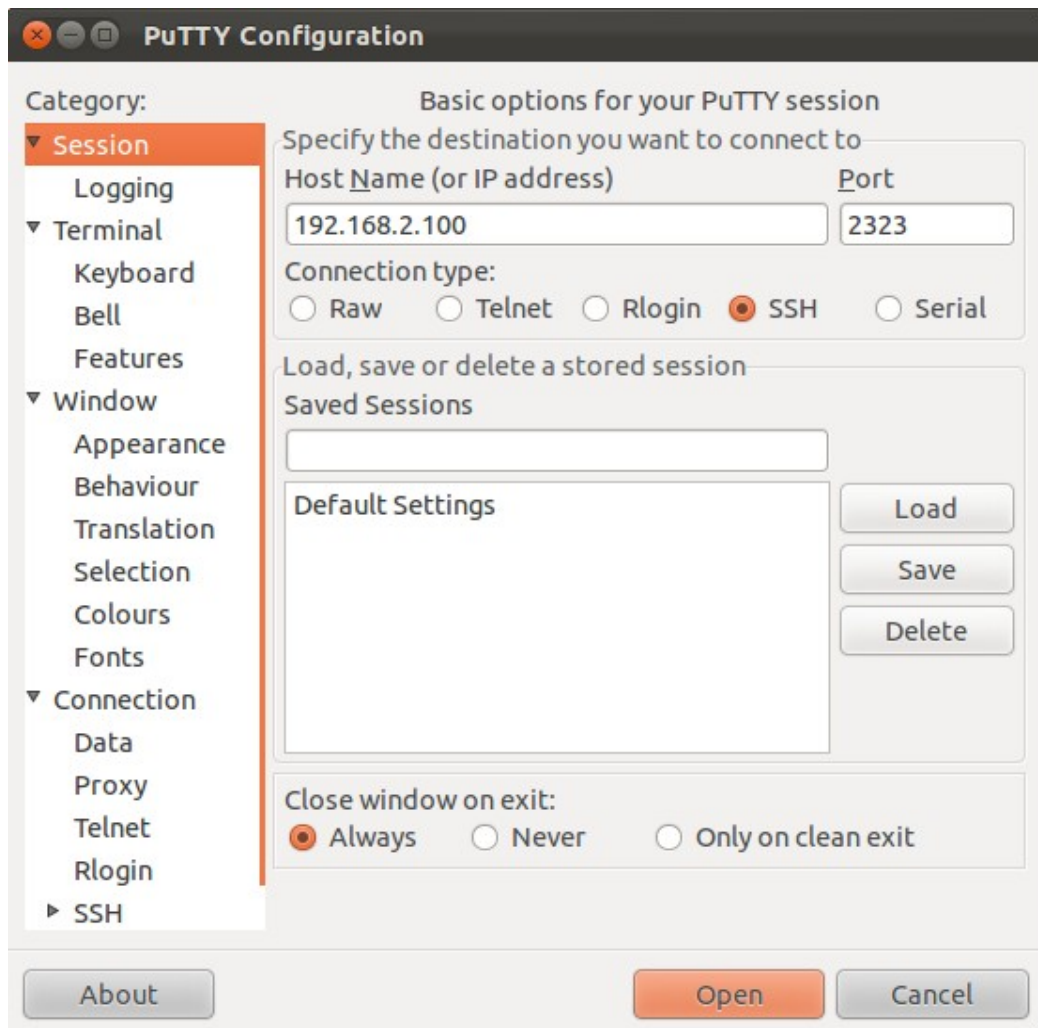
Las funciones principales están realizadas por los mismos ficheros PuTTY:

- PuTTY**- los clientes Telnet y SSH
- PSCP**- un cliente SCP, para copia de ficheros segura por línea de comandos
- PSFTP**- un cliente SFTP, para sesiones de transferencia de ficheros generales como en FTP
- PuTTYtel**- un cliente de solo Telnet
- Plink**- una interfaz de línea de comandos al PuTTY back ends
- Pageant**- un agente de autenticación SSH para PuTTY, PSCP y Plink
- PuTTYgen**- una utilidad de generación de claves RSA y DSA. Lo veremos un poco después.
- pterm**- un emulador de terminal X.

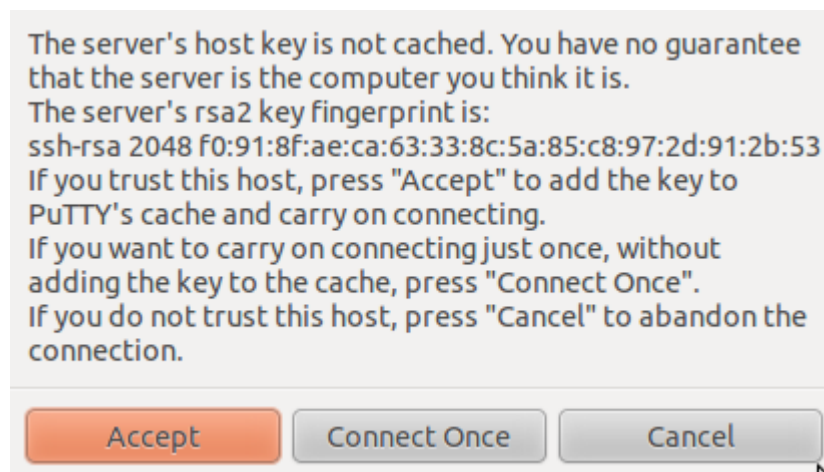
Vemos como podemos buscar Putty directamente en el centro de software de Ubuntu y seleccionarlo para instalarlo.



Abrimos el programa y configuramos la **IP del servidor** (192.168.2.100) y el **puerto** (2323). Pinchamos en **Open**.



Nos saldrá un aviso para avisarnos que el equipo al que queremos conectarnos no está en la base de datos del programa y que podría no ser seguro por no ser el equipo que nosotros pensamos. Así que nos da la opción de **Aceptar** añadiendo la llave a la caché de putty para futuras conexiones, **conectarnos solo por esta vez** sin guardar la llave, o **cancelar**.

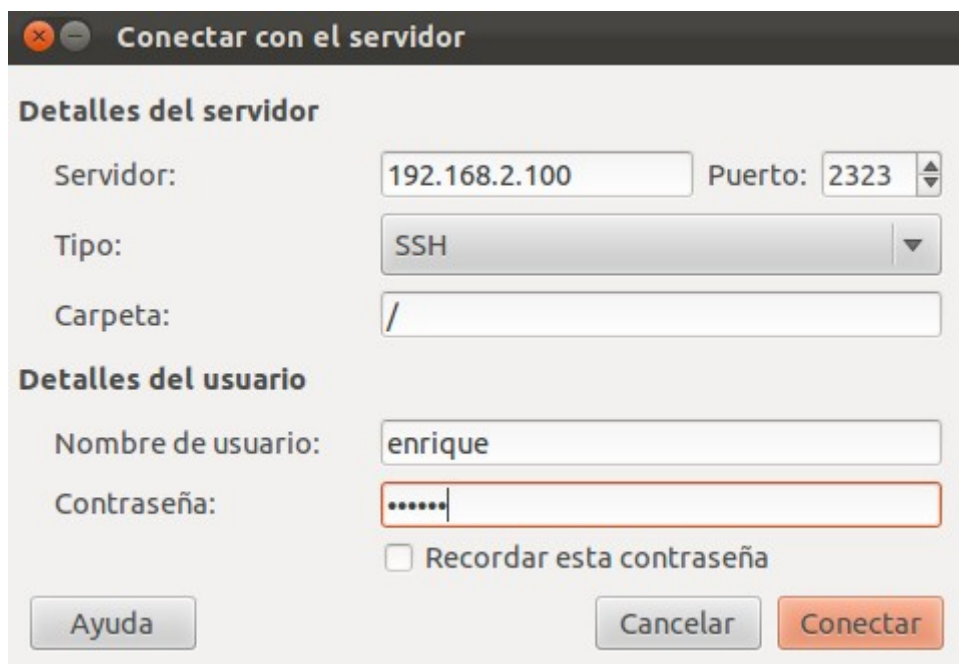


Más información:

<http://es.wikipedia.org/wiki/PuTTY>

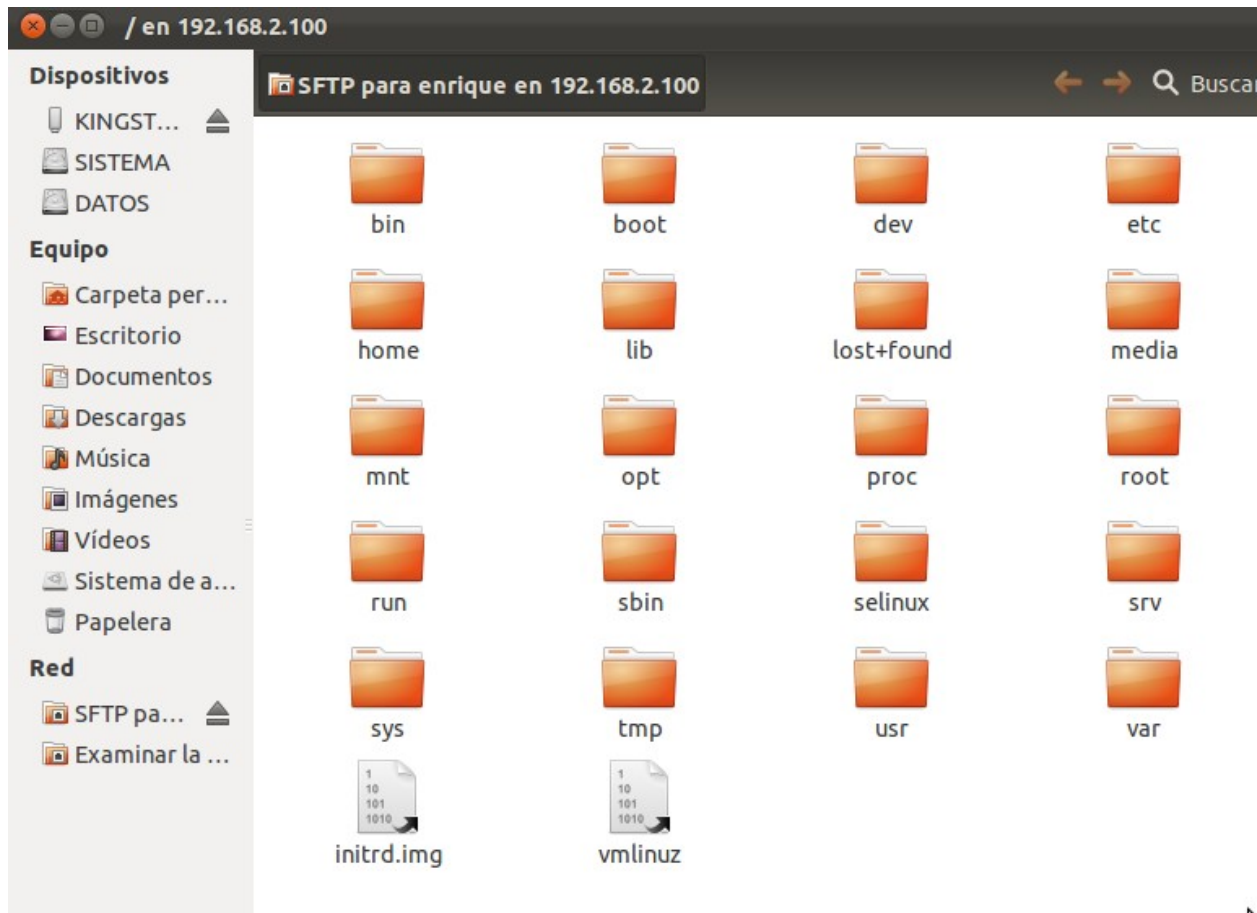
Conexión por Nautilus

En Linux además, podemos usar directamente **nautilus** para acceder por ssh al servidor abriendo cualquier carpeta y seleccionando “**Archivo**” del menú superior izquierdo “**Conectar con el servidor**”. Seleccionamos el protocolo SSH introducimos la IP, el puerto y la carpeta que visualizaremos inicialmente.



The image shows a dialog box titled "Conectar con el servidor" (Connect to the server). It is divided into two sections: "Detalles del servidor" (Server details) and "Detalles del usuario" (User details). In the "Detalles del servidor" section, the "Servidor:" field contains "192.168.2.100", the "Puerto:" field contains "2323", and the "Tipo:" dropdown menu is set to "SSH". The "Carpeta:" field contains a forward slash "/". In the "Detalles del usuario" section, the "Nombre de usuario:" field contains "enrique", and the "Contraseña:" field is filled with dots. There is a checkbox labeled "Recordar esta contraseña" (Remember this password) which is currently unchecked. At the bottom of the dialog, there are three buttons: "Ayuda" (Help), "Cancelar" (Cancel), and "Conectar" (Connect).

Cuando nos conectamos usando nautilus automáticamente se ejecutará el protocolo SFTP, que veremos ahora, para ver el contenido de la carpeta que le pongamos en la ventana de configuración de la conexión anterior.



Conexión por terminal

También podemos establecer una conexión usando un **terminal** con el comando:

```
ssh usuario remoto@ip_servidor
```

en nuestro caso

```
ssh enrique@192.1682.100
```

Como nosotros hemos cambiado el puerto por defecto debemos indicarlo en el comando de la siguiente forma:

```
ssh -p [puerto a la escucha] usuario remoto@ip_servidor
```

en nuestro caso estando en la red local

```
ssh -p 2323 enrique@192.168.2.100
```

La primera vez que nos conectemos nos dará un aviso diciendo que el host no se encuentra entre los host conocidos. Escribimos “**yes**” para confirmar que se conecte y tras introducir la clave se establecerá la conexión.

```
enrique@enrique-Inspiron-1520:~$ ssh -p 2323 enrique@192.168.2.100
enrique@192.168.2.100's password:
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Mon Feb  6 18:09:48 CET 2012

System load:   0.0                Processes:      78
Usage of /home: 2.5% of 2.75GB    Users logged in: 1
Memory usage:  13%               IP address for eth0: 192.168.2.100
Swap usage:    0%

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Mon Feb  6 18:00:11 2012 from 192.168.2.106
enrique@ServidorCep:~$
```

A partir de ese momento el ordenador cliente, que puede ser el host anfitrión de la máquina virtual en nuestro caso, o cualquier equipo de la red local, habrán establecido una conexión y podrán administrar remotamente el servidor iniciando servicios, copiando o moviendo archivos, cambiando permisos, etc.

Para ver como realmente esta conexión ha sido establecida vamos a usar un comando que nos permite ver que conexiones están activas y más datos al respecto. Vamos a usar el comando:

```
netstat -a |more
```

Podremos observar como existe una conexión entre nuestro servidor (192.168.2.100) y el equipo de la red cliente desde donde se ha realizado la conexión ssh. (192.168.2.106). En la imagen vemos esa conexión a través del puerto 2323 en el servidor y uno aleatorio del cliente (52117 en este ejemplo).


```

Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local Dirección remota Estado
tcp 0 0 localhost:mysql *:* ESCUCHAR
tcp 0 0 *:netbios-ssn *:* ESCUCHAR
tcp 0 0 *:pop3 *:* ESCUCHAR
tcp 0 0 *:imap2 *:* ESCUCHAR
tcp 0 0 *:www *:* ESCUCHAR
tcp 0 0 *:2323 *:* ESCUCHAR
tcp 0 0 *:smtp *:* ESCUCHAR
tcp 0 0 *:microsoft-ds *:* ESCUCHAR
tcp 0 0 *:imaps *:* ESCUCHAR
tcp 0 0 *:pop3s *:* ESCUCHAR
tcp 0 0 192.168.2.100:2323 192.168.2.106:52117 ESTABLECIDO
tcp6 0 0 [::]:pop3 [::]:* ESCUCHAR
tcp6 0 0 [::]:imap2 [::]:* ESCUCHAR
tcp6 0 0 [::]:2323 [::]:* ESCUCHAR
tcp6 0 0 [::]:imaps [::]:* ESCUCHAR
tcp6 0 0 [::]:pop3s [::]:* ESCUCHAR
udp 0 0 192.168.2.25:netbios-ns *:*
udp 0 0 192.168.2.10:netbios-ns *:*
udp 0 0 *:netbios-ns *:*
udp 0 0 192.168.2.2:netbios-dgm *:*
udp 0 0 192.168.2.1:netbios-dgm *:*
udp 0 0 *:netbios-dgm *:*
Activar zócalos de dominio UNIX (servidores y establecidos)
Proto RefCnt Flags Type State I-Node Ruta
unix 2 [ ACC ] FLUJO ESCUCHANDO 7424 /var/run/dovecot/log
n/ssl-params
unix 2 [ ACC ] FLUJO ESCUCHANDO 7426 /var/run/dovecot/log
--Más--[Utilizar q o Q para salir]

```

Para salir de una conexión establecida basta con usar el comando **exit**.

```

enrique@ServidorCep:~$ exit
logout

```

Como hemos visto, una vez conectados podríamos reiniciar un servicio del servidor por ejemplo en caso de que se produjera algún fallo. Imaginad que tenemos alojada una página web en el servidor y no responde a las peticiones que se le hacen a través de un navegador web. Una vez establecida la conexión, usando el comando:

sudo /etc/init.d/apache2 restart

Reiniciaríamos el servidor **Apache** viendo si queda solucionado el problema.

OPERACIONES CON ARCHIVOS

Usando OpenSSH es posible subir archivos al servidor o descargarlos usando dos protocolos que son **SCP** y **SFTP**. Vamos a ver como usar cada uno de ellos y alguna de sus características.

Usando SCP (Shell Secure Copy)

Usando este sistema vamos a poder copiar archivos del servidor a nuestra máquina local y subir archivos de nuestra máquina local al servidor. A diferencia de usar el siguiente método que vamos a ver (SFTP), usando SCP debemos conocer la ruta de los archivos que queramos copiar, o subir. En cambio es un método que aporta mayor seguridad al mandar toda la información encriptada mediante SSH. La sintaxis para usar este comando es la siguiente:

Para copiar un recurso desde el servidor al equipo local

```
scp usuario@ip_servidor:ruta_del_recurso_remoto nombre_archivo_destino
```

Para copiar un archivo desde el equipo local al servidor:

```
scp ruta_del_recurso_local usuario@ip_servidor:ruta_destino_remota
```

Al igual que al establecer la conexión como nosotros hemos cambiado el puerto por defecto tenemos que introducir un parámetro más **(-P)** al usar el comando.

```
scp -P 2323 enrique@192.168.2.100:ruta_recurso_remoto
```

```
scp -P 2323 ruta_recurso_local enrique@192.168.2.100:ruta_destino
```

Imaginemos que queremos copiar la configuración del archivo original de OpenSSH (sshd_config.ori) desde el servidor a nuestro equipo local, el comando para hacerlo quedaría así:

```
scp -P 2323 enrique@192.168.2.100:/etc/ssh/sshd_config.ori sshd_config.ori
```

```
enrique@enrique-Inspiron-1520:~$ scp -P 2323 enrique@192.168.2.100:/etc/ssh/sshd
_config.ori sshd_config.ori
enrique@192.168.2.100's password:
sshd_config.ori                                100% 2489      2.4KB/s   00:00
enrique@enrique-Inspiron-1520:~$ ls
Descargas  examples.desktop  Plantillas  Vídeos
Documentos Imágenes          Público     VirtualBox VMs
Escritorio Música            sshd_config.ori
enrique@enrique-Inspiron-1520:~$
```

Si además el recurso que queremos subir o copiar es una carpeta y todo su contenido tenemos que usar el parámetro **"-r"** quedando así:

scp -P 2323 -r ruta_recurso_local enrique@192.168.2.100:ruta_destino

Usando SFTP (Security File Transfer Protocol)

Usando este protocolo perdemos en seguridad en la transferencia de los datos pero ganamos en opciones a la hora de manipularlos. Con SFTP podemos navegar por la estructura de archivos del servidor. La sintaxis del comando es:

sftp usuario@ip_servidor

Para especificar un puerto diferente al 22 cuando vamos a conectar usamos el parámetro “-o” .

sftp -o Port=2323 enrique@192.168.2.100

```
enrique@enrique-Inspiron-1520:~$ sftp -o Port=2323 enrique@192.168.2.100
enrique@192.168.2.100's password:
Connected to 192.168.2.100.
sftp> pwd
Remote working directory: /home/enrique
sftp> █
```

Para copiar un archivo del servidor al ordenador local la orden “get”:

get ruta_recurso_remoto ruta_destino_local

Para copiar un archivo desde el equipo local al servidor remoto:

put ruta_recurso_local ruta_destino_remoto

Hay otras ordenes que podemos usar con SFTP

cd[rutaRemota]	Cambia de directorio dentro del servidor remoto
lcd[rutaLocal]	Cambia de directorio en el equipo local
chgrp[grp] [rutaRemota]	Cambia el grupo de trabajo de un fichero remoto. El[grp]tiene que ser un Group ID
chmod[opciones] [rutaRemota]	Cambia los permisos de Lectura, Escritura o de Ejecución a un fichero remoto
chown[own] [rutaRemota]	Cambia el grupo de trabajo de un fichero remoto.

	El[own]tiene que ser un User ID
get[rutaRemota] [rutaLocal]	Copia un recurso remoto en un equipo local
mkdir[rutaLocal]	Crea una carpeta en el equipo local
lpwd	Imprime la ruta local en la cual estamos trabajando
mkdir[rutaRemota]	Crea una carpeta en el equipo remoto
put[rutaLocal] [rutaRemota]	Sube un fichero o archivo desde una ruta local hasta una ruta remota
pwd	Imprime la ruta remota en la cual estamos trabajando
exit	Salimos de SFTP
rename[rutaLocal] [rutaRemota]	Renombra un un fichero remoto
rmdir[rutaRemota]	Borra una carpeta remota
rm[rutaRemota]	Borra un fichero remoto

SSH sin passwords. Usando claves RSA.

Con OpenSSH podemos usar un tipo de conexión que no nos solicitara clave al conectarnos. Para hacer esto, necesitaremos usar una de los dos tipos de claves que nos ofrece OpenSSH, el tipo **RSA**, o el **DSA**. Nosotros en el curso veremos solo el primero por gastar menos tiempo de computo al usarlo, llevando además cifrada la información.

Aún así debemos ser conscientes que al usar este tipo de conexiones cualquiera que se hiciera con la clave privada podría acceder al sistema, por lo que hay que ser cauteloso.

Para usar este método lo primero es crear una clave RSA en el **cliente** con el comando:

```
ssh-keygen -t rsa
```

Nos preguntará por el nombre del archivo donde se guardará la clave y por un password para usarla. En ambos casos presionamos “enter” para dejar el directorio por defecto **/home/usuario/.ssh/id_rsa** y dejar sin password cuando se use la clave.

Las claves privadas se guardan en el directorio

```
/home/usuario/.ssh/id_rsa
```

y nunca se deben hacer públicas.

La llave pública que se acaba de crear también, se almacena en el directorio

/home/usuario/.ssh/id_rsa.pub

Esta será la llave que usaremos para ponerla en los clientes que queramos que tengan conexión al servidor OpenSSH sin necesidad de contraseña.

```
enrique@enrique-Inspiron-1520:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/enrique/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/enrique/.ssh/id_rsa.
Your public key has been saved in /home/enrique/.ssh/id_rsa.pub.
The key fingerprint is:
```

Para que un cliente pueda conectarse mediante la clave RSA que acabamos de crear debemos añadir su información en el servidor a las **claves autorizadas (authorized_keys)**, para esto debemos copiar la clave pública (**id_rsa.pub**) del cliente a la carpeta del servidor siguiente:

/home/usuario/.ssh/

/home/enrique/.ssh/

Para hacerlo podemos usar scp desde el cliente

scp -P 2323 /home/enrique/.ssh/id_rsa.pub
enrique@192.168.2.100:/home/enrique/.ssh/id_rsa.pub

Con este comando estamos copiando el archivo local id_rsa.pub al servidor a la carpeta /home/usuario/.ssh

```
enrique@enrique-Inspiron-1520:~$ scp -P 2323 /home/enrique/.ssh/id_rsa.pub enrique@192.168.2.100:/home/enrique/.ssh/id_rsa.pub
enrique@192.168.2.100's password:
id_rsa.pub                                100% 411      0.4KB/s   00:00
enrique@enrique-Inspiron-1520:~$
```

Una vez que tenemos el archivo que guarda la clave pública en la carpeta del servidor faltaría añadir los datos que contiene el archivo al archivo **authorized_keys**. Para hacer esto estando en la carpeta del servidor **/home/usuario/.ssh** usamos el comando

```
cat id_rsa.pub >> authorized_keys
```

```
enrique@ServidorCep: ~/.ssh$ cat id_rsa.pub >> authorized_keys
enrique@ServidorCep: ~/.ssh$ ls
authorized_keys id_rsa id_rsa.pub
enrique@ServidorCep: ~/.ssh$
```

Finalmente nos vamos a una terminal del equipo cliente y probamos que al conectarnos por ssh al servidor este ya no nos pide la clave:

```
enrique@enrique-Inspiron-1520:~$ ssh -p 2323 enrique@192.168.2.100
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Mon Feb  6 20:33:49 CET 2012

System load:   0.0               Processes:      80
Usage of /home: 2.5% of 2.75GB   Users logged in: 1
Memory usage:   14%             IP address for eth0: 192.168.2.100
Swap usage:    0%

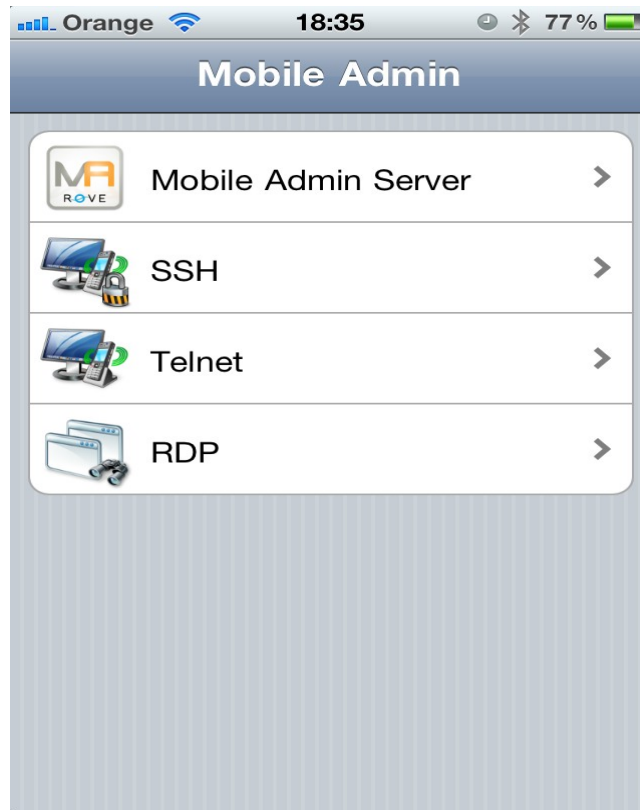
Graph this data and manage this system at https://landscape.canonical.com/
Last login: Mon Feb  6 20:12:37 2012 from 192.168.2.106
enrique@ServidorCep:~$
```

CONTROLANDO NUESTRO SERVIDOR REMOTAMENTE CON UN DISPOSITIVO MOVIL.

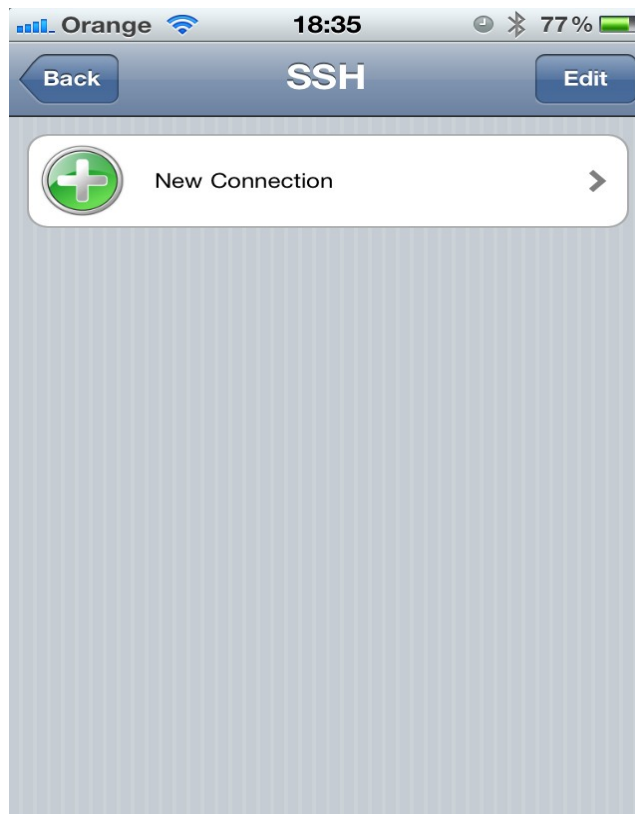
Ahora que hemos visto como administrar remotamente un servidor podemos dar un paso mas aprendiendo a hacerlo desde un teléfono móvil para no tener que tener algún equipo encendido para realizar tareas en el servidor.

Existen muchas aplicaciones de los dos sistemas operativos mas usados en terminales móviles en la actualidad, tanto de **Android** como de **iOS**, para realizar conexiones SSH. En el curso solo vamos a ver un ejemplo que se puede aplicar a cualquiera de ellas. Concretamente la aplicación que hemos usado es para el iOS de los **Iphone/Ipad** y se llama **Rove Mobile Admin** y no la hemos elegido por ser la mas potente sino por ser una de las gratuitas para este sistema.

Una vez instalada la aplicación la iniciamos y vemos como nos da la posibilidad de realizar conexiones a través de varios protocolos como son **RDP**, **Telnet** y **SSH** que es el que nos interesa.



Seleccionamos **SSH** y elegimos **crear nueva conexión**.



Introducimos la dirección ip del **Host servidor 192.168.2.100** para el nuestro y el **puerto usado 2323** para nuestra configuración.

Solo falta introducirle el **nombre de usuario y el login** para establecer un conexión y obtener una shell del servidor donde poder por ejemplo reiniciar un servicio.

Orange 18:36 77%

SSH

Host 192.168.2.100

Port 2323

User Name enrique

Password ●●●●●●

1 2 3 4 5 6 7 8 9 0

- / : ; () € & @ "

#+= . , ? ! ' < x

ABC espacio Aceptar

Orange 18:36 77%

SSH

Rows 24

Scrollback Lines 72

Enter Sends CR >

Connect

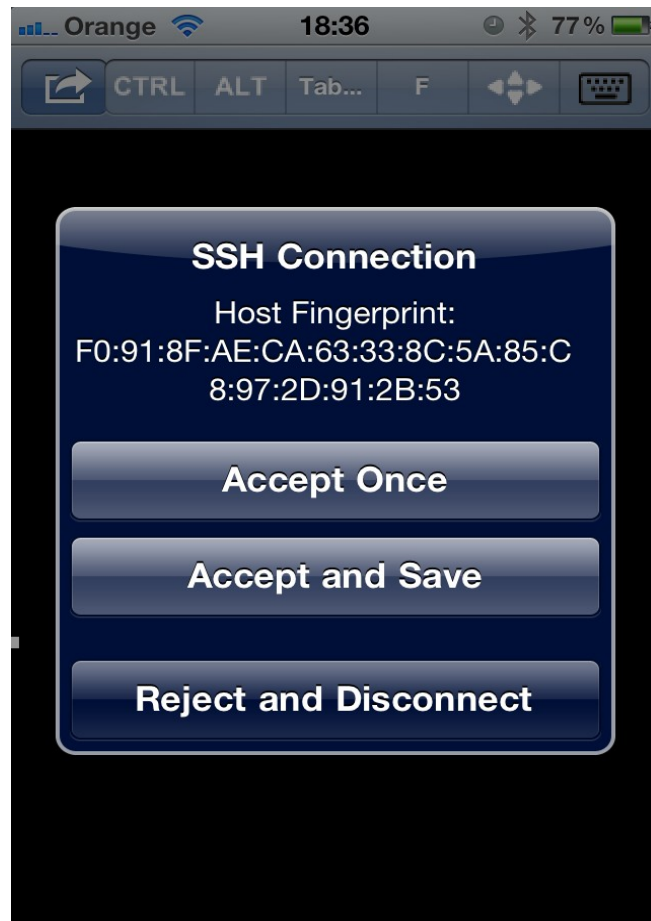
1 2 3 4 5 6 7 8 9 0

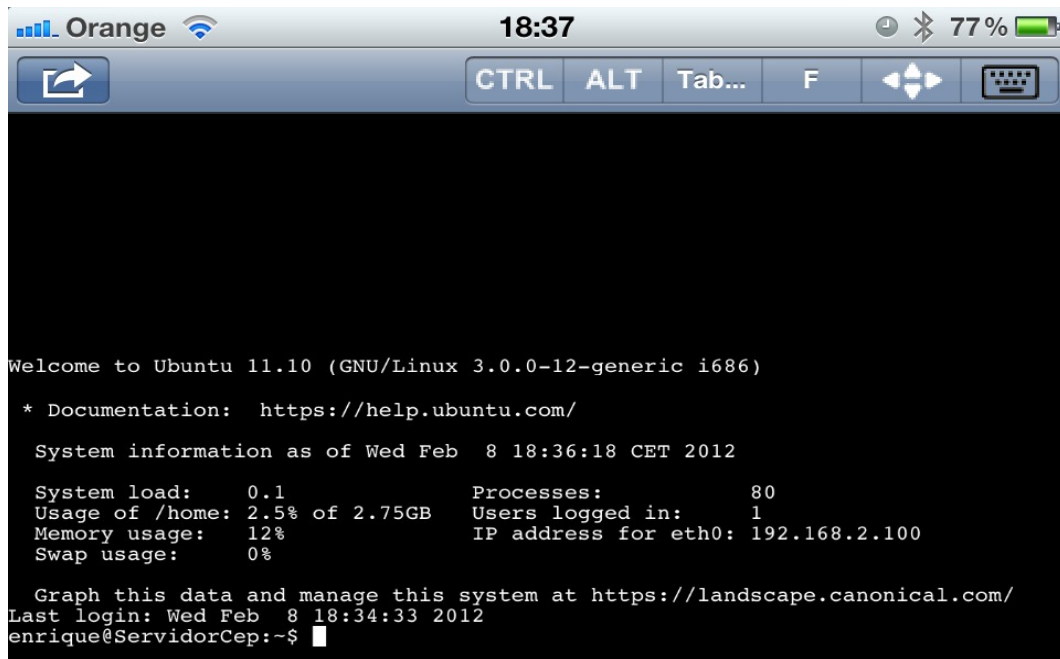
- / : ; () € & @ "

#+= . , ? ! ' < x

ABC espacio Aceptar

Nos avisa si queremos aceptar esa conexión sólo por esta vez, o guardarla para futuras conexiones. Aceptamos y guardamos (**Accept and Save**)





The screenshot shows an iPhone screen with a terminal application open. The status bar at the top displays 'Orange' as the carrier, signal strength bars, a Wi-Fi icon, the time '18:37', a Bluetooth icon, and a battery level of '77%'. The terminal window has a title bar with 'CTRL', 'ALT', 'Tab...', 'F', and navigation icons. The terminal content shows a 'Welcome to Ubuntu 11.10' message, documentation link, system information as of Wed Feb 8 18:36:18 CET 2012, and system statistics including load, memory usage, and IP address. The prompt is 'enrique@ServidorCep:~\$'.

```
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Wed Feb  8 18:36:18 CET 2012

System load:        0.1               Processes:            80
Usage of /home:    2.5% of 2.75GB     Users logged in:     1
Memory usage:     12%                IP address for eth0: 192.168.2.100
Swap usage:        0%

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Wed Feb  8 18:34:33 2012
enrique@ServidorCep:~$
```

Ya podemos ver la terminal de comandos de nuestro servidor en la pantalla del iPhone

Y ejecutar algún comando como reiniciar el servidor web



This screenshot shows the same iPhone terminal window at a later time, '18:42', with the battery at '76%'. The system information is repeated. The prompt is now 'enrique@ServidorCep:~\$' and the command 'sudo /etc/init.d/apache2 restart' has been entered. A virtual QWERTY keyboard is visible at the bottom of the screen.

```
System information as of Wed Feb  8 18:36:18 CET 2012

System load:        0.1               Processes:            80
Usage of /home:    2.5% of 2.75GB     Users logged in:     1
Memory usage:     12%                IP address for eth0: 192.168.2.100
Swap usage:        0%

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Wed Feb  8 18:34:33 2012
enrique@ServidorCep:~$ sudo /etc/init.d/apache2 restart
```



Esto es especialmente útil cuando tenemos que administrar un servidor y no estamos en un sitio con algún equipo conectado a la red. Solo con nuestro teléfono móvil y una conexión **3G** podemos hacerlo.

Con estos conceptos ya podemos administrar remotamente un servidor de forma remota usando interfaz gráfico, o mediante comandos en Windows y Linux.

Este artículo esta licenciado bajo Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License.

Servidores Linux Enrique Brotons