SSH

SSH (**Secure Shell**) en un protocolo de red encriptado para realizar **comunicaciones de datos de forma segura**, conexiones a través de **línea de comandos** (*shell*), **ejecución de comandos** y otros servicios de red seguros entre dos equipos conectados en red. La comunicación se realiza a través de un canal seguro **sobre una red insegura**, utilizando un **servidor** (ejecutando un servidor SSH) y un **cliente** (ejecutando una aplicación cliente SSH).

El **uso más frecuente** es para acceder de forma remota al *shell* de sistemas Linux. También se puede utilizar para conectarse a equipos con Windows, aunque en ese caso es más frecuente el uso del *Escritorio remoto*. Los sistemas Linux más modernos, que dispongan de interfaz gráfica, también ofrecen servicios similares al escritorio remoto. El uso de SSH se hace más frecuente que el escritorio remoto cuando no se necesita un interfaz gráfico, ya que la conexión se hace de manera más **rápida**. Por ejemplo, es muy habitual para la configuración de servidores, en los que sólo es necesario editar archivos de texto, y ejecutar una serie de comandos.

SSH se diseñó como un **reemplazo del servicio *Telnet*** y otros protocolos de *shell* remoto, todos ellos menos seguros que SSH ya que, por ejemplo, transmitían las contraseñas de conexión en texto plano. SSH proporciona un protocolo de comunicación que proporciona confidencialidad e integridad en los datos sobre una red insegura, como puede ser Internet.

Cómo funciona SSH

Una conexión SSH se establece en varias fases:

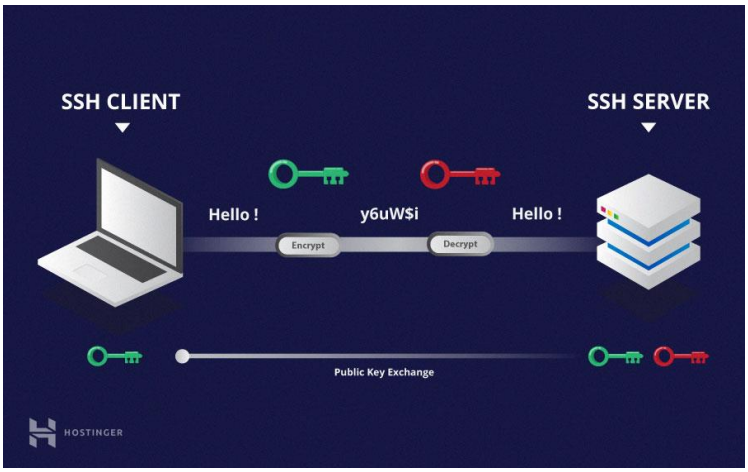
- En primera instancia, se determina la identidad entre el servidor y el cliente para establecer un canal seguro (capa segura de transporte).
- En segunda instancia, el cliente inicia sesión en el servidor.

Establecer un canal seguro

El establecimiento de una capa segura de transporte comienza con la fase de negociación entre el cliente y el servidor para ponerse de acuerdo en los métodos de cifrado que

quieren utilizar. El protocolo SSH está diseñado para trabajar con un gran número de algoritmos de cifrado, por esto, tanto el cliente como el servidor deben intercambiar primero los algoritmos que admiten.

Después, para establecer una conexión segura, el servidor envía al cliente su clave de host (clave pública del servidor). El cliente genera una clave de sesión (clave simétrica) de 256 bits que cifra con la clave pública del servidor y luego la envía al servidor junto con el algoritmo utilizado. El servidor descifra la clave de sesión con su clave privada y envía al cliente un mensaje de confirmación cifrado con la clave de sesión. Después de esto, las comunicaciones restantes se cifran gracias a un algoritmo de cifrado simétrico, mediante la clave de sesión compartida entre el cliente y el servidor.



La seguridad de la transacción se basa en la confianza del cliente y el servidor en que las claves host (claves públicas) de cada una de las partes son válidas. Así, cuando se conecta por primera vez con el servidor, el cliente muestra generalmente un mensaje en el que le pide que acepte la comunicación (y posiblemente le presenta un hash de la clave host del servidor):

```
Host key not found from the list of known hosts. Are you sure you
want to continue connecting (yes/no)?
```

Para obtener una sesión segura propiamente dicha, es mejor pedirle directamente al administrador del servidor que valide la clave pública presentada. Si el usuario cliente valida la conexión, el cliente guarda la clave host del servidor para evitar tener que repetir esta fase.

Por el contrario, dependiendo de su configuración, el servidor puede, a veces, verificar que el cliente es quien dice ser. Si el servidor tiene una lista de hosts autorizados para la conexión, cifrará el mensaje utilizando la clave pública del cliente (que se encuentra en la base de datos de claves del host) para verificar si el cliente es capaz de descifrarla con su clave privada (esto se llama *challenge)*.

Autentificación

Una vez que se ha establecido la conexión segura entre el cliente y le servidor, el cliente debe conectarse al servidor para obtener un derecho de acceso. Existen diversos métodos:

1. el método más conocido es la **contraseña** tradicional. El cliente envía un nombre de acceso y una contraseña al servidor a través de la conexión segura y el servidor verifica que el usuario en cuestión tiene acceso al equipo y que la contraseña suministrada es válida.
2. un método menos conocido, pero más flexible, es el uso de **claves públicas**. Si el cliente elige la clave de autentificación, el servidor creará un *challenge* y le dará acceso al cliente si éste es capaz de descifrar el *challenge* con su clave privada.

Instalación del servidor SSH en Ubuntu

Para instalar el servidor SSH en una máquina con Ubuntu basta con indicar la orden:

```
sudo apt-get install openssh-server
```

Puedes consultar su configuración:

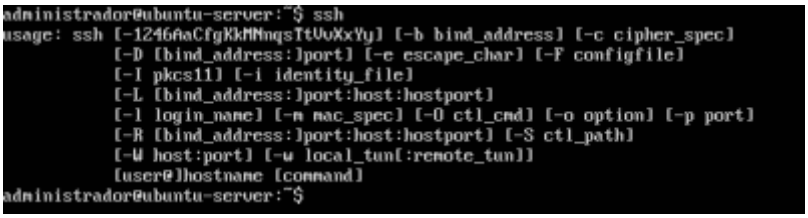
```
sudo nano /etc/ssh/sshd_config
```

Clientes SSH

Para realizar la conexión a un equipo remoto que disponga del software de servidor SSH, es necesario disponer de una aplicación que haga las funciones de cliente SSH.

OpenSSH

Ubuntu y **Mac OS X** ofrece de manera predeterminada el **cliente OpenSSH**, por lo que no es necesario instalar nada más para hacer las funciones de equipo cliente SSH. Para ejecutarlo, tan sólo debes indicar el comando *ssh.*
En **Windows**, para usar SSH desde el Símbolo del sistema, debes instalar ***OpenSSH for Windows*** obteniendo el instalador binario desde su página de descarga.

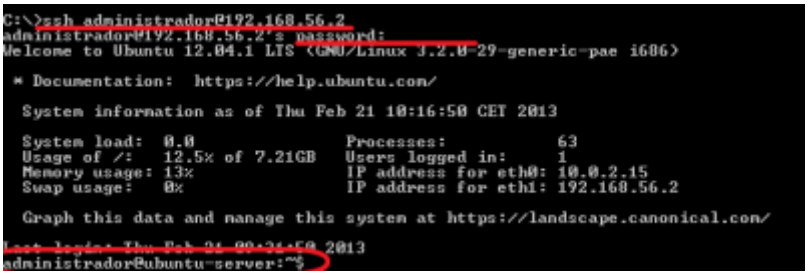


Como puedes ver, al intentar ejecutarlo sin ningún parámetro, se muestra la ayuda donde puedes observar las distintas opciones de uso. En la página de ayuda de OpenSSH puedes encontrar el significado de cada opción, o cargando el manual con: ***man ssh***.

Para realizar la **conexión con un servidor**, debes indicar la **dirección IP o nombre de dominio** del servidor, y si el nombre del **usuario remoto** que quieres utilizar es **diferente** al usuario local debes indicar previamente el nombre del usuario remoto seguido del símbolo @, siguiendo el siguiente formato.

ssh *usuario_remoto@nombre_equipo_remoto*
ssh *usuario_remoto@direccion_IP_equipo_remoto*

Una vez realizada la conexión, podrás **realizar cualquier operación remotamente** como si tuvieras abierta una sesión del terminal en el servidor, que puede encontrarse en cualquier punto alejado de tu equipo local, aunque siempre deben tener conexión de red entre ambos.

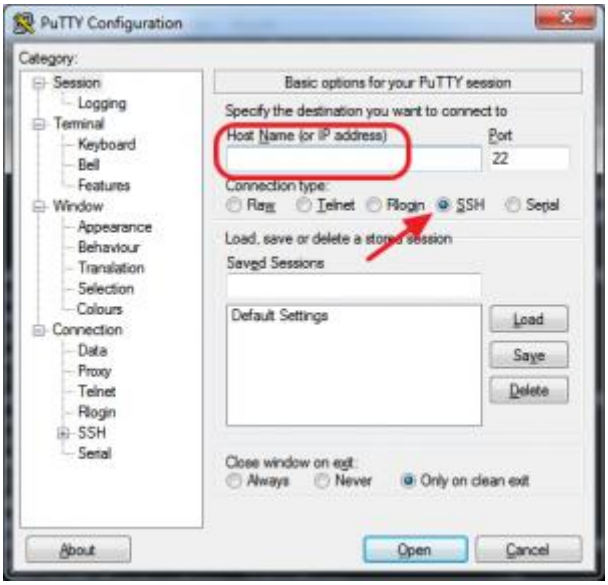


Observa que la **primera vez** que realizas la conexión con un equipo remoto, se te **informa** de que te has conectado a un equipo al que previamente no te habías conectado nunca. En ese caso debes **confirmar** que consideras que el equipo al que estás conectando es realmente el equipo deseado (debes escribir ***yes***). De esta manera, las próximas veces que te conectes a ese mismo servidor, no aparecerá este mensaje. Si un **intruso** quisiera hacer pasar un equipo diferente como si fuera el servidor al que te estás conectando, te volvería a aparecer el mensaje de que ese equipo no es conocido, y así detectarías al intruso, ya que se identifica a cada equipo con una clave a modo de huella (*fingerprint*).

Putty

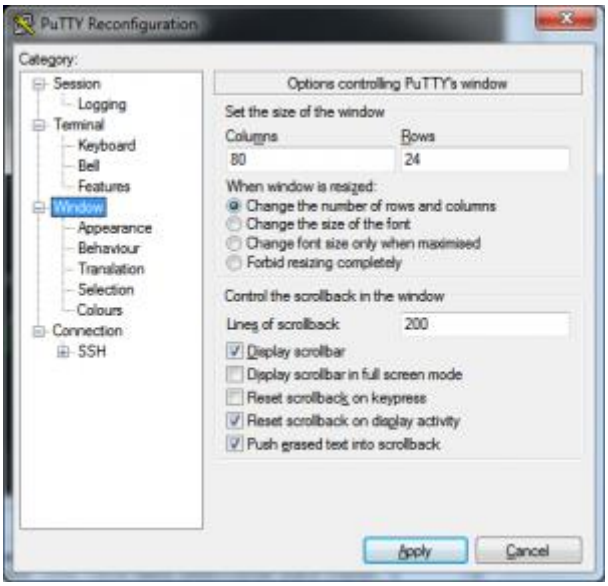
También puedes hacer la conexión a un servidor SSH de una manera más gráfica usando la aplicación **Putty**, disponible de forma **gratuita para Windows y Linux**.

La conexión al servidor se realiza indicando el nombre del servidor o su dirección IP.



Al igual que en el caso anterior, la primera vez que se hace la conexión, hay que aceptar la huella del servidor.

Facilita algunas cosas como usar el **portapapeles** (sólo hace falta seleccionar para copiar, y para pegar sólo hay que usar el clic derecho), cambiar el tamaño de la **ventana** o aspectos como los colores.



SCP y SFTP

El protocolo SCP (*Secure copy*) permite la **transferencia de archivos** entre un equipo local y otro remoto, o incluso entre dos equipos remotos. Está **basado en el protocolo SSH**.
Se puede hacer uso de **SCP desde la línea de comandos** a través de la aplicación con ese mismo nombre que incorpora OpenSSH o con la aplicación **PSCP** descargable desde la misma web que *Putty*.

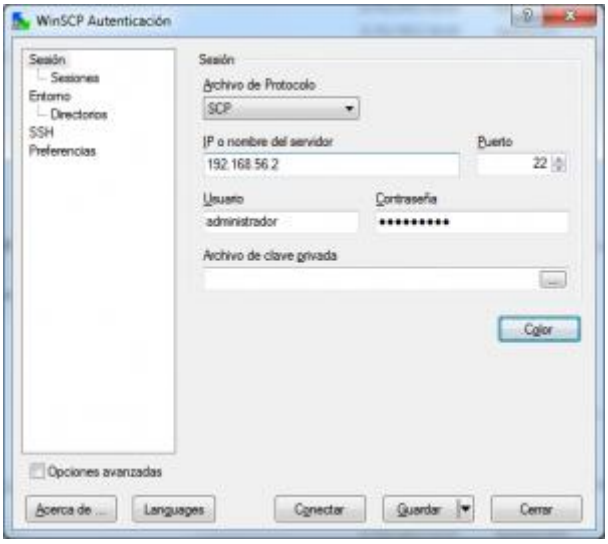
Su **utilización básica** sigue el siguiente formato:

Para **descargar** un archivo remoto:
scp *usuario@equipo:/ruta/archivo /ruta/archivo_local*

Para **subir** un archivo local a uno remoto:
scp */ruta/archivo_local usuario@equipo:/ruta/archivo*

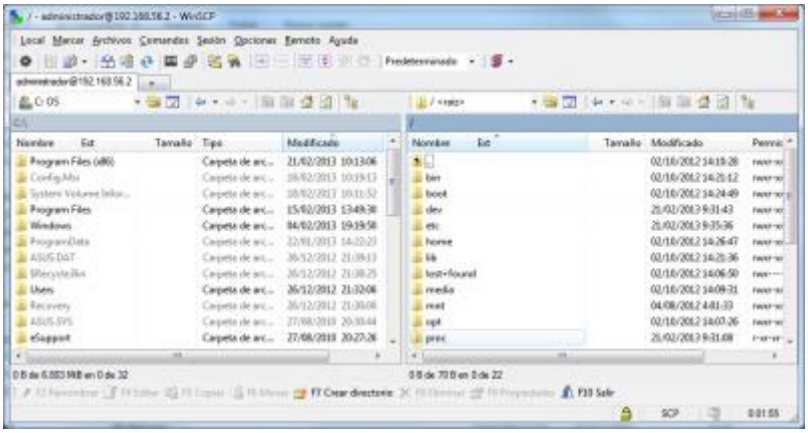
Para **copiar** entre equipos remotos:
scp *usuario@equipo1:/ruta/archivo usuario@equipo2:/ruta/archive*

De una manera más gráfica, se puede utilizar SCP con la aplicación **WinSCP** para Windows. Al iniciar la aplicación te solicitará el protocolo a utilizar (SCP, SFTP o FTP), la dirección del servidor, usuario y contraseña de acceso al equipo remoto.



Al ser un protocolo basado en SSH, la primera vez que se conecte con un servidor desconocido, solicitará la confirmación para almacenar la huella digital del equipo remoto.

Una vez abierta la aplicación, observarás que tiene la misma estructura y funcionamiento que una aplicación cliente FTP gráfica, donde puedes descargar y subir archivos arrastrando sus iconos de lado a lado.



SFTP

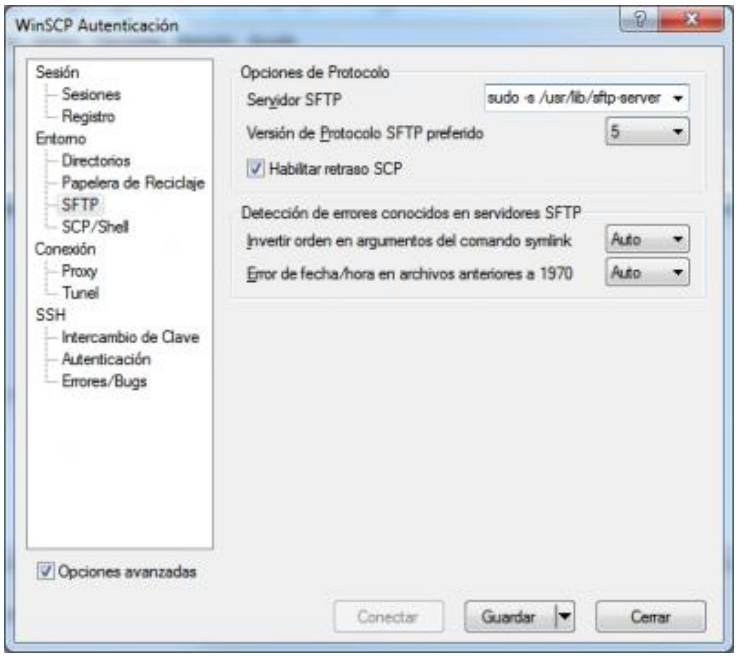
De manera muy similar se puede utilizar **SFTP (SSH File Transfer Protocol)** que también permite realizar transferencias de archivos usando el protocolo SSH, aunque posibilita acciones más avanzadas que SCP como la pausa en la descarga, aunque SCP posee un algoritmo más eficaz que permite hacer las descargas más rápidas.

Las aplicaciones clientes se pueden encontrar de la misma manera que SCP, es decir, con el comando **SFTP** de OpenSSH, **PSFTP** de Putty o de manera gráfica con **WinSCP** para Windows, o ***Cyberduck*** para Windows y Mac.

sudo en WinSCP con SFTP

Con WinSCP empleando el protocolo SFTP es posible realizar acciones que requieren permisos de usuario root (sudo) como editar determinados archivos del sistema. Para ello, antes de realizar la conexión, especifica en la **configuración** (activando las **opciones avanzadas**) dentro de la sección ***Entorno > SFTP***, que vas a usar como servidor SFTP el siguiente:

```
sudo -s /usr/lib/sftp-server
```



Además, en la máquina con Ubuntu debes indicar que tu usuario puede realizar acciones sudo sin necesitar una contraseña. Para ello, añade en el archivo ***/etc/sudoers*** la línea:

```
tu_usuario ALL=NOPASSWD: ALL
```

FTPS

El uso de SSL/TSL en FTP tiene sentido especialmente a la hora de comunicar las credenciales de autenticación así que si solo vamos a usar FTP anónimo no deberíamos preocuparnos.

Lo primero que debemos hacer es generar un certificado. vsFTPd utiliza el formato pem así que lo generamos con la siguiente orden:

```
Openssl req -x509 –nodes –days 365 –newkey rsa:2048 –keyout /etc/ssl/private/vsftpd.pem –out /etc/ssl/private/vsftpd.pem
```

Habrá que editar el archivo de configuración:

```
mv /etc/vsftpd.conf /etc/vsftpd.conf.bak
sudo nano /etc/vsftpd.conf
```

y escribir lo siguiente

```
listen=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
ssl_enable=YES
listen_port=990
#The following line enables implicit mode
implicit_ssl=YES
allow_anon_ssl=NO
#This will force secure data connections, not required, but recommended
force_local_data_ssl=YES
#This will force secure logins, not strictly required, but REALLY recommended
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
#La siguiente línea es necesaria para que Filezilla acepte la conexión
ssl_ciphers=AES128-SHA
rsa_cert_file=/etc/ssl/private/vsftpd.pem
```

Posteriormente tendremos que reiniciar el servicio: `service vsftpd restart`

Ahora necesitamos un **cliente en modo texto** que soporte este tipo de conexiones, para ello instalamos lftp, el más extendido en modo texto para Linux.

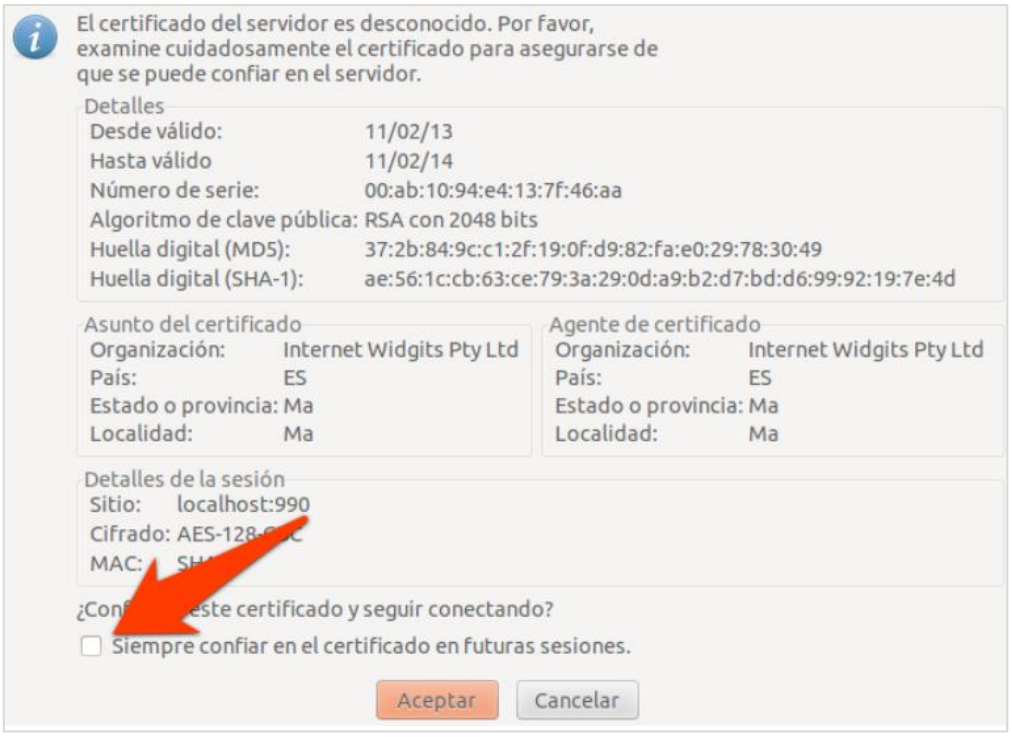
```
sudo apt-get update
sudo apt-get install lftp
```

ahora ya podemos probar nuestra conexión, con el consiguiente error de certificado si no es emitido por una CA:

```
lftp ftps://localhost
lftp localhost:~> user prueba
Clave:
lftp prueba@localhost:~> ls
ls: Error fatal: Certificate verification: Not trusted
lftp prueba@localhost:~> set ssl:verify-certificate no
```

En el **cliente gráfico** Filezilla establecemos los parámetros de conexión: URL del servidor, user, pass y puerto (generalmente el 21).

En el caso de establecer una conexión segura, si el certificado no es de confianza, nos aparecerá una ventana de aviso en la que veremos los datos del certificado y podremos marcar el certificado para que nos vuelva a preguntar si confiamos en él.



SFTP vs FTPS

SFTP y FTPS no son lo mismo, ni siquiera se parecen, aunque ambos sirvan para lo mismo (transferencia segura de ficheros).

- **SFTP:** Sus siglas significan SSH File Transfer Protocol, es completamente diferente del protocolo FTP (File Transfer Protocol). SFTP fue construido desde cero y añade la característica de FTP a SSH. Sólo usa un canal de comunicación, envía y recibe los mensajes en binario (y no en formato texto como hace FTP).

- **FTPS:** Es una extensión de FTP mediante SSL para el cifrado de los datos, utiliza dos canales, envía y recibe los mensajes en formato texto. FTPS normalmente es más conocido ya que usa los mismos comandos que FTP.

Ambos protocolos utilizan un **algoritmo asimétrico (RSA, DSA)**, un **algoritmo simétrico (AES por ejemplo)**, y un algoritmo de intercambio de claves. Para la autenticación, FTPS utiliza certificados **X.509**, mientras que SFTP utiliza *usuario* y *password* cifrados con la cifrado simétrico o bien las claves SSH (*challenge*) con certificados digitales.

¿Cuál es más seguro?

Técnicamente **SFTP es más avanzado que FTPS**, sin embargo algunos dispositivos pueden no ser compatibles con SFTP (como los móviles, consolas etc.) y sin embargo con FTPS sí lo son, como hemos dicho antes, FTPS es una extensión de FTP.