



07:46

[3428]

Aufnahme

M 1/5 F4.0 100

A 100 200 400 800 1600 3200 6400

# INSTALACIÓN Y CONFIGURACIÓN:

## SERVICIO SSH UBUNTU SERVER 20.04

# INSTALACIÓN DEL SERVIDOR DE SHELL SEGURO

El servidor de shell seguro o SSH (**Secure SHell**) es un servicio muy similar al servicio telnet ya que permite que un usuario acceda de forma remota a un sistema Linux pero con la particularidad de que, al contrario que telnet, las comunicaciones entre el cliente y servidor viajan encriptadas desde el primer momento de forma que si un usuario malintencionado intercepta los paquetes de datos entre el cliente y el servidor, será muy difícil que pueda extraer la información ya que se utilizan sofisticados algoritmos de encriptación.

La popularidad de ssh ha llegado a tal punto que el servicio telnet prácticamente no se utiliza. Se recomienda no utilizar nunca telnet y utilizar ssh en su lugar.

Para que un usuario se conecte a un sistema mediante ssh, deberá disponer de un cliente ssh. Desde la primera conexión, y mediante encriptación asimétrica, las comunicaciones se encriptan incluido el proceso de autenticación del usuario cuando proporciona su nombre y su contraseña. También se proporciona una clave de encriptación simétrica para encriptar las comunicaciones del resto de la sesión mediante encriptación simétrica por su menor necesidad de procesamiento.

# INTALACIÓN DEL SERVIDOR Y CLIENTE SSH

Para instalar el servidor y el cliente ssh debemos instalar mediante apt-get el paquete ssh que contiene tanto la aplicación servidora como la aplicación cliente:

```
alumno@servidor:~$ apt-get install ssh
```

Los archivos de configuración son:

- /etc/ssh/ssh\_config: Archivo de configuración del cliente ssh
- /etc/ssh/sshd\_config: Archivo de configuración del servidor ssh

Antes de modificar cualquier parámetro en los ficheros de configuración se recomienda hacer copia de seguridad de los mismos.

# ARRANQUE Y PARADA MANUAL DEL SERVIDOR SSH

El servidor ssh, al igual que todos los servicios en Debian y derivados, dispone de un script de arranque y parada en la carpeta /etc/init.d.

```
alumno@servidoresr:~$ /etc/init.d/ssh restart
```

```
alumno@servidoresr:~$ /etc/init.d/ssh stop
```

Aunque también podemos hacerlo directamente con comandos:

**service ssh (restart | stop | status)**

# CONEXIÓN AL HOST REMOTO

Ponemos el comando correspondiente al usuario con el que queremos conectar y al host remoto.

```
alumno@servidor:~$ ssh alumno@192.168.1.1
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
ECDSA key fingerprint is SHA256:tPfl2XbtX/6k7aJALfIAr5s6tCIrTGQhXDtScGm6J6M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? _
```

A continuación, si es la primera vez que nos conectamos nos pregunta por la confianza en la clave que nos envía. Si aceptamos, pedirá contraseña. En caso de introducir un *password* correcto accederemos al servicio.

```
profesor@192.168.1.1's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of sáb 05 nov 2022 09:14:37 UTC

System load:  0.03               Processes:            118
Usage of /:   60.1% of 8.76GB    Users logged in:     1
Memory usage: 24%               IPv4 address for enp0s3: 192.168.1.1
Swap usage:   0%                IPv4 address for enp0s8: 10.0.3.15

120 updates can be applied immediately.
38 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
```

# FICHERO DE CONFIGURACIÓN DEL SERVIDOR SSH

El archivo de configuración del servidor SSH se encuentra en `/etc/ssh` (como siempre) y se llama `sshd_config`.

Este fichero lo podemos modificar con cualquier editor de texto como **gedit** (modo gráfico), **vim**, **nano**, etc.

Recuerda hacer una copia de seguridad antes de empezar a modificarlo, por ejemplo, en tu home para no tener varias versiones del mismo archivo en un directorio clave.

```
#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem        sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#      X11Forwarding no
#      AllowTcpForwarding no
#      PermitTTY no
```



# FICHERO DE CONFIGURACIÓN DEL SERVIDOR SSH

Veamos algunos de los parámetros más importantes de este fichero:

- **Port:** indica el puerto de escucha del servicio SSH. Por defecto el puerto de escucha es el 22 aunque podemos cambiarlo por seguridad.
- **Protocol** (versiones antiguas): indica la versión de protocolo que se utiliza. Normalmente se utiliza la versión 2 que es la más segura.
- **X11Forwarding:** este parámetro permite ejecutar aplicaciones gráficas en el servidor. Lo veremos más adelante.
- **PermitRootLogin:** este parámetro indica si el administrador (root) puede conectarse al servidor SSH. Por motivos de seguridad no debemos permitir que el administrador pueda conectarse al servidor (*PermitRootLogin no*). Si tenemos esta opción habilitada en caso de ataque informático tienen la mitad del trabajo hecho, pues el usuario **root** existe en todas las máquinas GNU/Linux, tan sólo le queda averiguar la contraseña. Por eso es recomendable deshabilitar esta opción. No os preocupéis los que tenéis en mente usar SSH para hacer un uso administrativo, podéis hacerlo con vuestra cuenta y sudo.

# FICHERO DE CONFIGURACIÓN DEL SERVIDOR SSH

**AllowUsers:** con este parámetro indicamos aquellos usuarios que podrán conectarse al servidor. El resto de usuarios no podrán hacerlo. La lista de usuarios que tendrán permiso para conectarse al servidor deben ir en la misma línea separados por espacios. Si, por ejemplo, queremos que sólo puedan conectarse a nuestro servidor SSH el usuario con login **user1** y el usuario con login **user2**, pondremos:

**AllowUsers user1 user2.**

**DenyUsers:** igual que el anterior pero para impedir accesos.

**LoginGraceTime:** es el número de segundos que tiene un usuario remoto para hacer login en el servidor SSH. Poned ese valor a pocos segundos, no tardamos mucho en hacer login si sabemos la cuenta y la password. De esta forma evitamos ciertos scripts que se aprovechan de ese tiempo. El valor típico en términos de seguridad es 30, aunque podéis poner incluso menos si estáis más conformes.

**PrintLastLog** [yes/no]: Mostrar cuando entra el usuario la fecha de su último acceso ssh.



# FICHERO DE CONFIGURACIÓN DEL SERVIDOR SSH

**MaxAuthTries:** es el número de intentos que tiene el usuario remoto para hacer login. Quien intente conectar debe acordarse de su *login* y *password*, por lo que es tontería darle un número grande de intentos. En principio con dos son más que suficientes. Si al segundo intento no lo ha conseguido se cortará la conexión SSH. Siempre se puede volver a conectar y reintentarlo, pero así nos quitamos de encima ciertos scripts que intentan encontrar el *login* por fuerza bruta a base de ensayo y error.

**MaxStartups:** define el número máximo de usuarios que pueden estar conectados simultáneamente al servidor SSH. Si estamos hablando de un ordenador personal donde sólo vas a conectar tú, pues lo lógico sería que como mucho hubiera una.

**Banner /PATH/fichero.txt:** Mostrar un Banner previo a la autenticación.

**ClientAliveInterval:** espera un número de segundos transcurridos los cuales con inactividad del cliente hará que la sesión expire.

# CONEXIÓN AL SERVIDOR MEDIANTE SSH

Para conectar desde un PC cliente al servidor mediante **ssh**, debemos ejecutar el comando **ssh** seguido del nombre o dirección IP del servidor.

La conexión se realizará con el mismo nombre de usuario que estemos utilizando en el PC cliente. Ejemplo, supongamos que el usuario henki, desde el PC llamado aula5pc3, quiere conectarse al servidor cuya IP es 192.168.1.239:

**henki@aula5pc3: ~\$ ssh 192.168.1.239**

Cuando nos pida el password tendremos que introducir el password del usuario henki en la máquina destino, por lo que debe existir, lógicamente, también en dicha máquina.

Esto sería si no hubiéramos cambiado el puerto, ya que intentaría conectar por el puerto 22 que es el puerto por defecto del cliente. Podéis cambiarlo si queréis para que conecte por defecto por el puerto que le digáis en lugar del 22.

# CONEXIÓN AL SERVIDOR MEDIANTE SSH

La otra opción, que es lo más normal, es simplemente indicarle en la línea de conexión qué puerto ha de usar y el usuario con el que nos identificaremos en la máquina destino.

```
$ ssh -p puerto tu_usuario@ip_servidor_remoto
```

La primera vez que se conecte alguien desde dicho PC cliente, se instalará el certificado de autenticación del servidor, lo cual es normal si se trata de la primera vez.

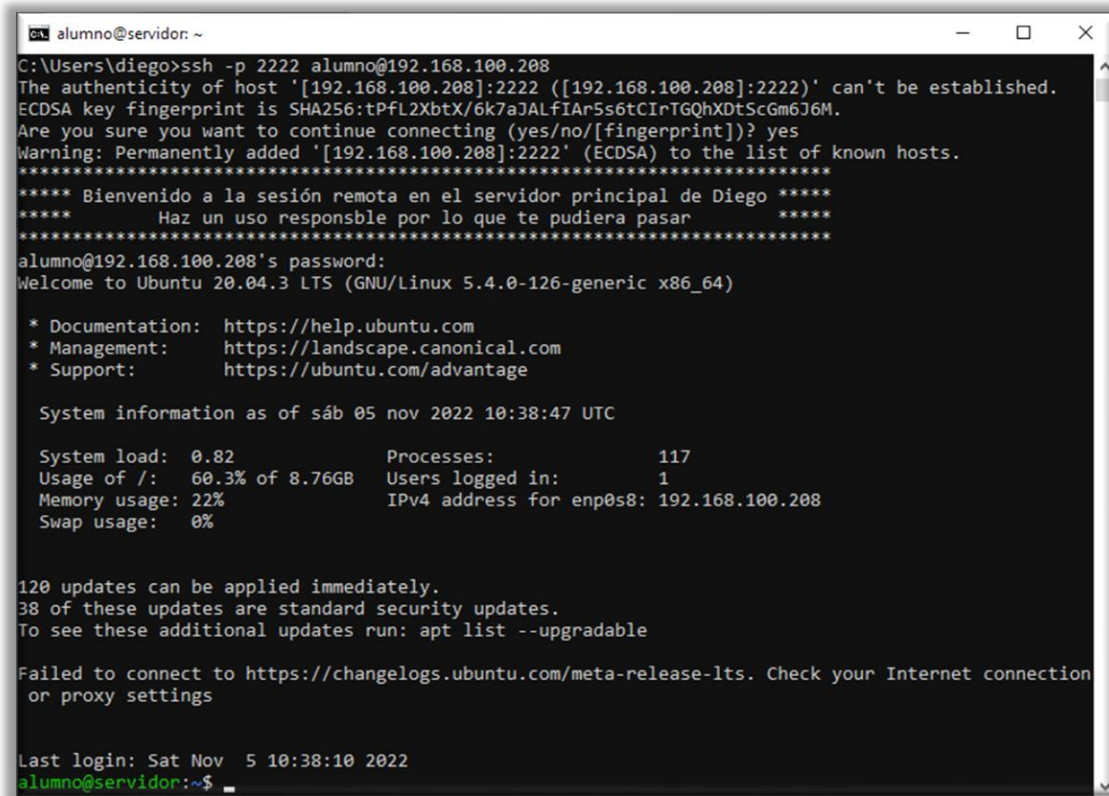
A la pregunta *“Are you sure you want to continue connecting (yes/no)?”* debemos responder *yes* ya que de lo contrario la comunicación se cortará.

Si ya nos hemos conectado anteriormente otras veces y vuelve a realizar ésta pregunta, significa que alguien se está haciendo pasar por el servidor (nuestro servidor ha sido hackeado) o que se ha reconfigurado el servidor (cambio de nombre, IP, etc...).

# CONEXIÓN AL SERVIDOR SSH DESDE WINDOWS

Desde PC con Windows es posible conectarse por ssh a servidores Linux mediante el programa Putty. Se trata de un cliente **ssh** para Windows que permite acceder en modo texto al sistema Linux desde sistemas Windows.

También podemos acceder desde consola de comandos, ya sea símbolo del sistema o PowerShell.



```

C:\Users\diego>ssh -p 2222 alumno@192.168.100.208
The authenticity of host '[192.168.100.208]:2222 ([192.168.100.208]:2222)' can't be established.
ECDSA key fingerprint is SHA256:tPfl2XbtX/6k7aJALfIAR5s6tCIrTGQhXdtScGm6J6M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.100.208]:2222' (ECDSA) to the list of known hosts.
***** Bienvenido a la sesión remota en el servidor principal de Diego *****
***** Haz un uso responsable por lo que te pudiera pasar *****
alumno@192.168.100.208's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

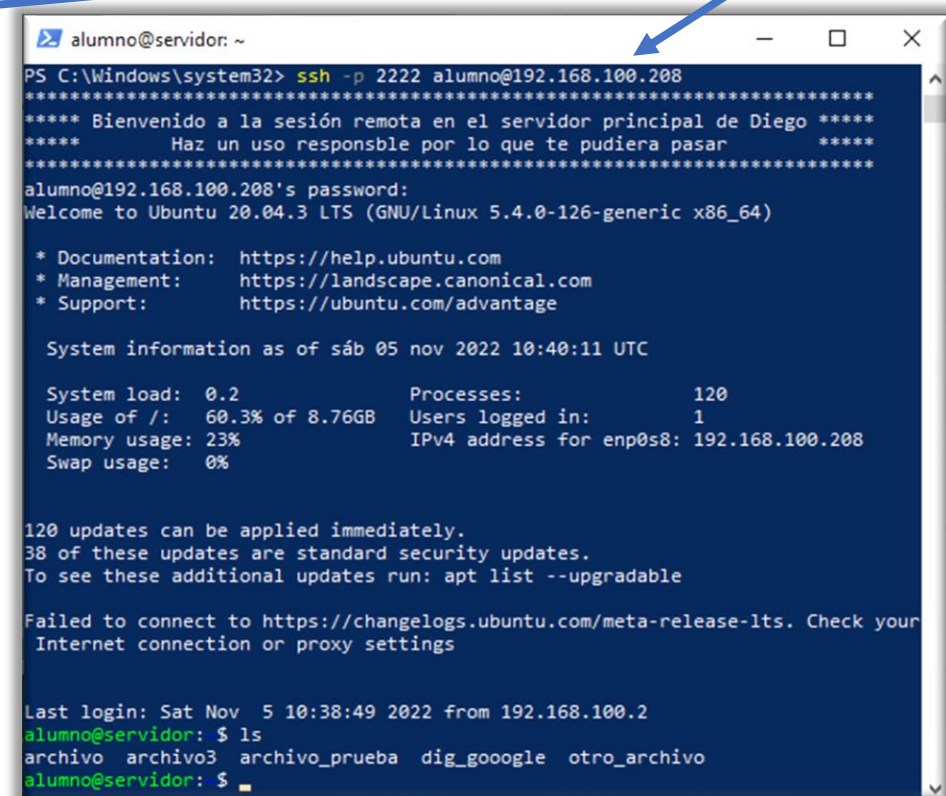
System information as of sáb 05 nov 2022 10:38:47 UTC

System load: 0.82          Processes:           117
Usage of /:  60.3% of 8.76GB Users logged in:       1
Memory usage: 22%         IPv4 address for enp0s8: 192.168.100.208
Swap usage:  0%

120 updates can be applied immediately.
38 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings

Last login: Sat Nov  5 10:38:10 2022
alumno@servidor:~$
  
```



```

PS C:\Windows\system32> ssh -p 2222 alumno@192.168.100.208
***** Bienvenido a la sesión remota en el servidor principal de Diego *****
***** Haz un uso responsable por lo que te pudiera pasar *****
alumno@192.168.100.208's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of sáb 05 nov 2022 10:40:11 UTC

System load: 0.2          Processes:           120
Usage of /:  60.3% of 8.76GB Users logged in:       1
Memory usage: 23%         IPv4 address for enp0s8: 192.168.100.208
Swap usage:  0%

120 updates can be applied immediately.
38 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

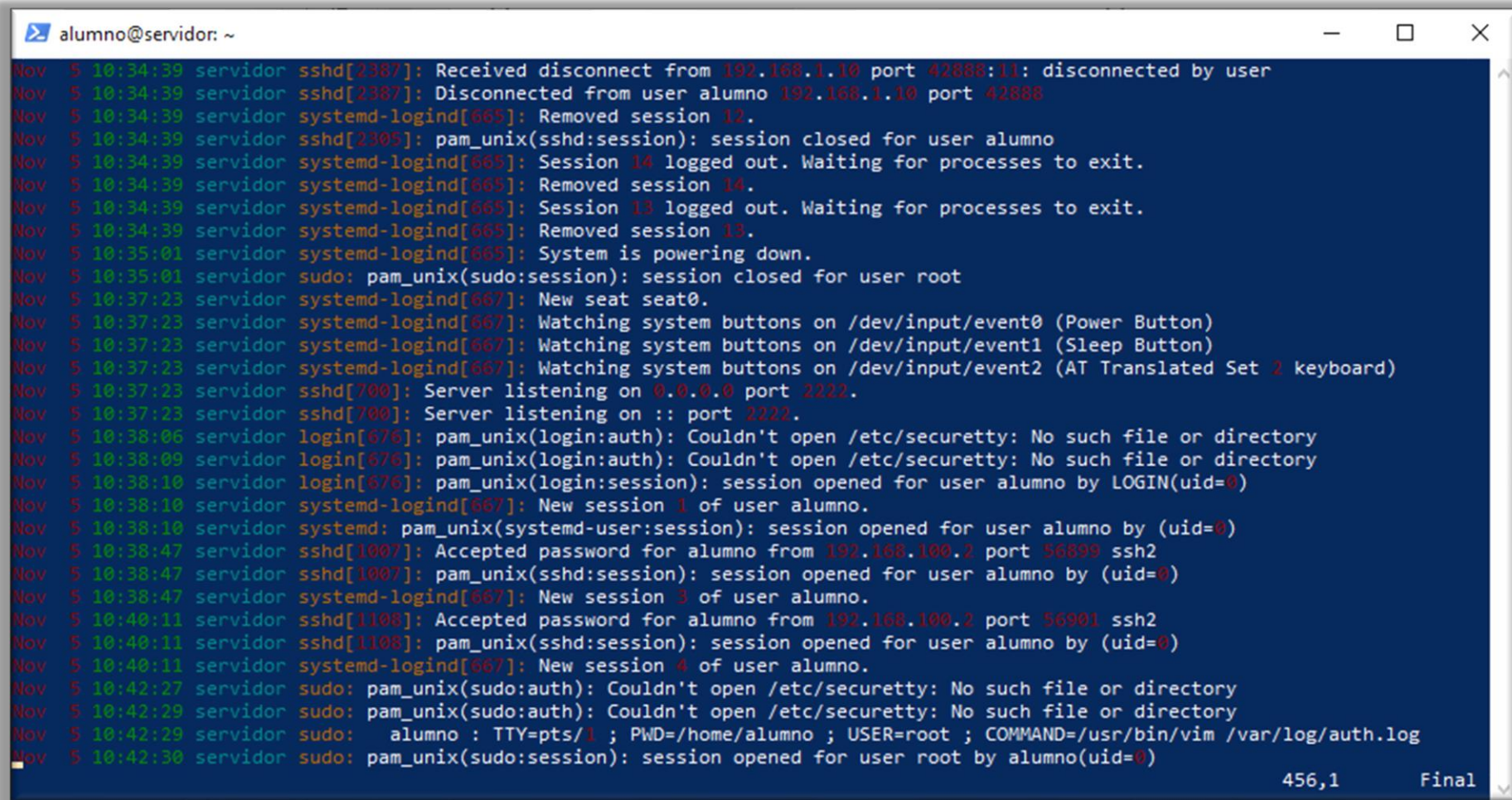
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Sat Nov  5 10:38:49 2022 from 192.168.100.2
alumno@servidor:~$ ls
archivo archivo3 archivo_prueba dig_gooogle otro_archivo
alumno@servidor:~$
  
```



# ARCHIVO CON LOGS DE CONEXIÓN

Tras conectar, podemos ver que usuarios han accedido al mismo al comprobar los logs de conexión. Estos se encuentran en el archivo `/var/log/auth.log`



```
alumno@servidor: ~
Nov 5 10:34:39 servidor sshd[2387]: Received disconnect from 192.168.1.10 port 42888:11: disconnected by user
Nov 5 10:34:39 servidor sshd[2387]: Disconnected from user alumno 192.168.1.10 port 42888
Nov 5 10:34:39 servidor systemd-logind[665]: Removed session 12.
Nov 5 10:34:39 servidor sshd[2305]: pam_unix(sshd:session): session closed for user alumno
Nov 5 10:34:39 servidor systemd-logind[665]: Session 14 logged out. Waiting for processes to exit.
Nov 5 10:34:39 servidor systemd-logind[665]: Removed session 14.
Nov 5 10:34:39 servidor systemd-logind[665]: Session 13 logged out. Waiting for processes to exit.
Nov 5 10:34:39 servidor systemd-logind[665]: Removed session 13.
Nov 5 10:35:01 servidor systemd-logind[665]: System is powering down.
Nov 5 10:35:01 servidor sudo: pam_unix(sudo:session): session closed for user root
Nov 5 10:37:23 servidor systemd-logind[667]: New seat seat0.
Nov 5 10:37:23 servidor systemd-logind[667]: Watching system buttons on /dev/input/event0 (Power Button)
Nov 5 10:37:23 servidor systemd-logind[667]: Watching system buttons on /dev/input/event1 (Sleep Button)
Nov 5 10:37:23 servidor systemd-logind[667]: Watching system buttons on /dev/input/event2 (AT Translated Set 2 keyboard)
Nov 5 10:37:23 servidor sshd[700]: Server listening on 0.0.0.0 port 2222.
Nov 5 10:37:23 servidor sshd[700]: Server listening on :: port 2222.
Nov 5 10:38:06 servidor login[676]: pam_unix(login:auth): Couldn't open /etc/securetty: No such file or directory
Nov 5 10:38:09 servidor login[676]: pam_unix(login:auth): Couldn't open /etc/securetty: No such file or directory
Nov 5 10:38:10 servidor login[676]: pam_unix(login:session): session opened for user alumno by LOGIN(uid=0)
Nov 5 10:38:10 servidor systemd-logind[667]: New session 1 of user alumno.
Nov 5 10:38:10 servidor systemd: pam_unix(systemd-user:session): session opened for user alumno by (uid=0)
Nov 5 10:38:47 servidor sshd[1007]: Accepted password for alumno from 192.168.100.2 port 56899 ssh2
Nov 5 10:38:47 servidor sshd[1007]: pam_unix(sshd:session): session opened for user alumno by (uid=0)
Nov 5 10:38:47 servidor systemd-logind[667]: New session 3 of user alumno.
Nov 5 10:40:11 servidor sshd[1108]: Accepted password for alumno from 192.168.100.2 port 56901 ssh2
Nov 5 10:40:11 servidor sshd[1108]: pam_unix(sshd:session): session opened for user alumno by (uid=0)
Nov 5 10:40:11 servidor systemd-logind[667]: New session 4 of user alumno.
Nov 5 10:42:27 servidor sudo: pam_unix(sudo:auth): Couldn't open /etc/securetty: No such file or directory
Nov 5 10:42:29 servidor sudo: pam_unix(sudo:auth): Couldn't open /etc/securetty: No such file or directory
Nov 5 10:42:29 servidor sudo: alumno : TTY=pts/1 ; PWD=/home/alumno ; USER=root ; COMMAND=/usr/bin/vim /var/log/auth.log
Nov 5 10:42:30 servidor sudo: pam_unix(sudo:session): session opened for user root by alumno(uid=0)
```

456,1 Final

# OTROS SERVICIOS DE SSH

Ya vimos que es posible realizar copia segura de archivos y utilizar un servicio de FTP seguro. También ejecutar de forma remota aplicaciones gráficas anteponiendo el parámetro `-X` (previa configuración en el servidor).

Desde PCs con Windows es posible conectarse por **ssh** a servidores Linux de forma gráfica mediante Cygwin. Se trata de un conjunto de programas libres que simulan un Unix para Windows con servidor gráfico X y cliente **ssh** para Windows entre otras cosas, que permite acceder en modo gráfico al sistema Linux desde sistemas Windows.

Otros servidores X gratuitos para Windows son Xming y Mocha.



# INSTALACIÓN CON CERTIFICADO

Para evitar tener que introducir continuamente la contraseña cuando deseamos conectar con un servidor remoto por **ssh**, o para programar tareas automatizadas, existe la posibilidad de autenticarse por certificado, para ello debemos:

1. Crear un certificado de usuario en el PC cliente.
2. Copiar el certificado en el PC servidor.

Para que el servidor **ssh** acepte la autenticación por medio de certificado, deberá tener activada la opción ***PubkeyAuthentication yes***, es decir, deberá tener dicho parámetro en el archivo de configuración de **ssh**.

```
PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
```

# CREAR CERTIFICADO EN EL PC CLIENTE

Para crear un certificado que permita autenticar al usuario, debemos ejecutar el comando **ssh-keygen** (man ssh-keygen). Dicho comando creará dentro de nuestra carpeta **home**, en una carpeta llamada **.ssh**, dos archivos:

- uno llamado **id\_rsa** que será la clave privada de nuestro certificado.
- y otro llamado **id\_rsa.pub** que será la clave pública de nuestro certificado. Éste último archivo será el que hay que copiar en el servidor remoto.

```
alumno@servidoresr:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alumno/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alumno/.ssh/id_rsa
Your public key has been saved in /home/alumno/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ceuDPWe2QqFxzs6UNp/k361HVEo64zSqMoyAsBu9JNs alumno@servidoresr
```

# COPIAR EL CERTIFICADO EN EL PC SERVIDOR

Para poder identificarse en el servidor desde el cliente sin tener que introducir la contraseña, debemos copiar el archivo ***id\_rsa.pub*** que hemos creado en el cliente, en la carpeta home del usuario con el que queremos conectar en el servidor dentro de una carpeta llamada ***.ssh*** en un archivo llamado ***authorized\_keys***.

Para copiar dicho archivo del cliente al servidor, podemos hacerlo con **scp**.

```
alumno@servidor:~/.ssh$ sudo scp -P 2222 id_rsa.pub alumno@192.168.1.1:/home/alumno/.ssh/authorized_keys
The authenticity of host '[192.168.1.1]:2222 ([192.168.1.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:tPfl2XbtX/6k7aJALfIAr5s6tCIrTGQhXDtScGm6J6M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.1.1]:2222' (ECDSA) to the list of known hosts.
*****
***** Bienvenido a la sesión remota en el servidor principal de Diego *****
***** Haz un uso responsable por lo que te pudiera pasar *****
*****
alumno@192.168.1.1's password:
id_rsa.pub 100% 743 426.6KB/s 00:00
```

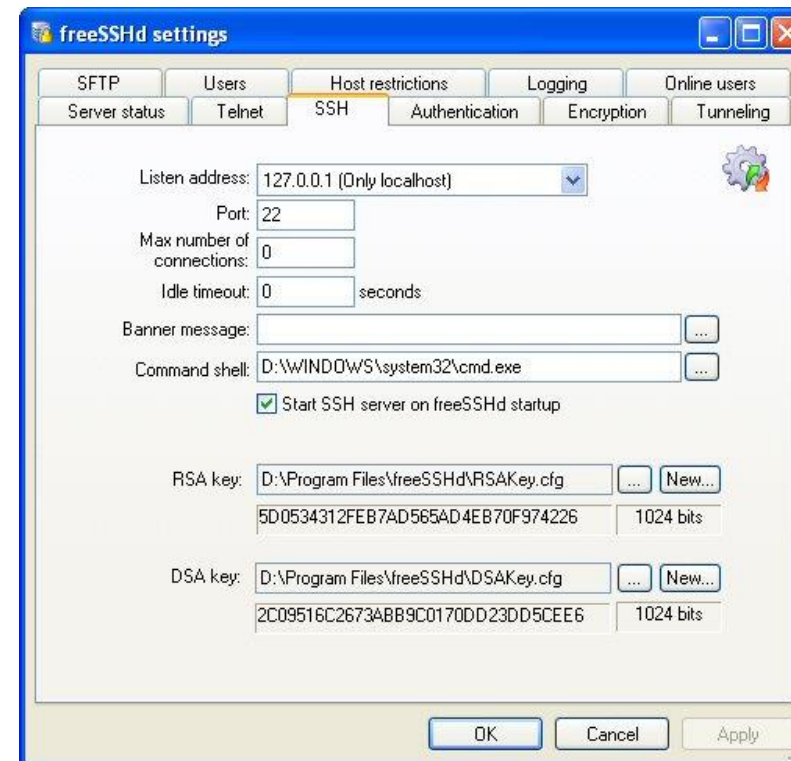
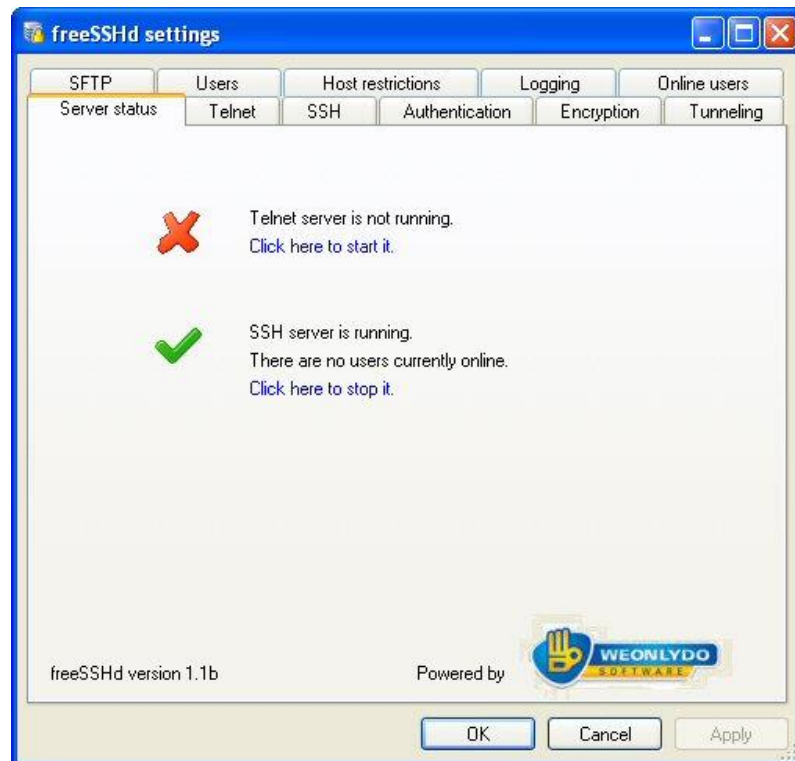
Ya podemos conectarnos al servidor sin necesidad de introducir la contraseña.

# CONECTAR DESDE LINUX A UN SERVIDOR SSH WINDOWS

Para ello debemos instalar un servidor **SSH** en Windows. **freeSSHd Server** es una buena opción gratuita.

Para conectarnos al servidor **SSH** usaremos un cliente **SSH** como **Putty** o un cliente Linux (terminal).

Para copiar ficheros desde o hacia el servidor **SSH** podemos utilizar un programa como **Winscp**.



# SERVICIO SECURE SHELL (SSH)

FIN