

## **UD 05\_07 – Redes**

### **Práctica: administración remota con SSH.**

#### **Contenido**

1	INTRODUCCIÓN.....	2
2	SSH (PARTE I): ADMINISTRAR UBUNTU DESDE UN EQUIPO DIFERENTE.....	2
3	SSH (PARTE II): ACCEDER DESDE OTRO EQUIPO CON UBUNTU 16.04 LTS.....	10
4	SSH (PARTE III): ACCEDER DESDE OTRO EQUIPO CON WINDOWS 10.....	14

## 1 INTRODUCCIÓN

---

Este contenido está extraído de SomeBooks.es.

Para realizar esta práctica, asegúrate de que la máquina virtual que estás utilizando tiene en la configuración de red el “**adaptador puente**” y que las direcciones IP de tus equipos pertenecen a la misma red.

Observa que, en esta práctica, para iniciar los servicios se utiliza el comando `service` en lugar del `systemctl` que se utilizó en la práctica de Samba. Aunque `systemctl` puede proporcionar mayor control, muchas veces estos comandos se pueden utilizar indistintamente.

## 2 SSH (PARTE I): ADMINISTRAR UBUNTU DESDE UN EQUIPO DIFERENTE

---

Fuente: <http://somebooks.es/ssh-parte-i-administrar-ubuntu-desde-un-equipodiferente/>

Publicado por **P. Ruiz** en 3 agosto, 2016

El *protocolo SSH*<sup>1</sup> proporciona un modo sencillo y razonablemente seguro de conectarse a un ordenador, desde otro equipo de la red, con el propósito de administrarlo.

La principal ventaja que aporta es que toda la información que intercambian los ordenadores implicados viaja cifrada. Esta característica es muy importante cuando pretendemos realizar tareas administrativas de forma remota en el ámbito de nuestra red local, pero, sobre todo, cuando accedemos a través de Internet.

Quizás, el único inconveniente que puedas encontrar es que el acceso se produce en modo texto, aunque, si necesitas velocidad y simplicidad, más que un inconveniente, se trata de una ventaja.

Hoy, en [SomeBooks.es](http://SomeBooks.es), te enseñamos a instalar y configurar OpenSSH en un ordenador que ejecuta *Ubuntu Mate 16.04 LTS*, así, podrás utilizarlo con tu *Raspberry Pi* si has seguido nuestros artículos [Crear una imagen de instalación de Ubuntu Mate para Raspberry Pi desde Ubuntu](#) o [desde Windows](#). Sin embargo, los pasos son idénticos en cualquier otro *sabor* de *Ubuntu*. En

---

<sup>1</sup> SSH son las siglas de Secure SHell. OpenSSH es la versión abierta y libre del protocolo SSH

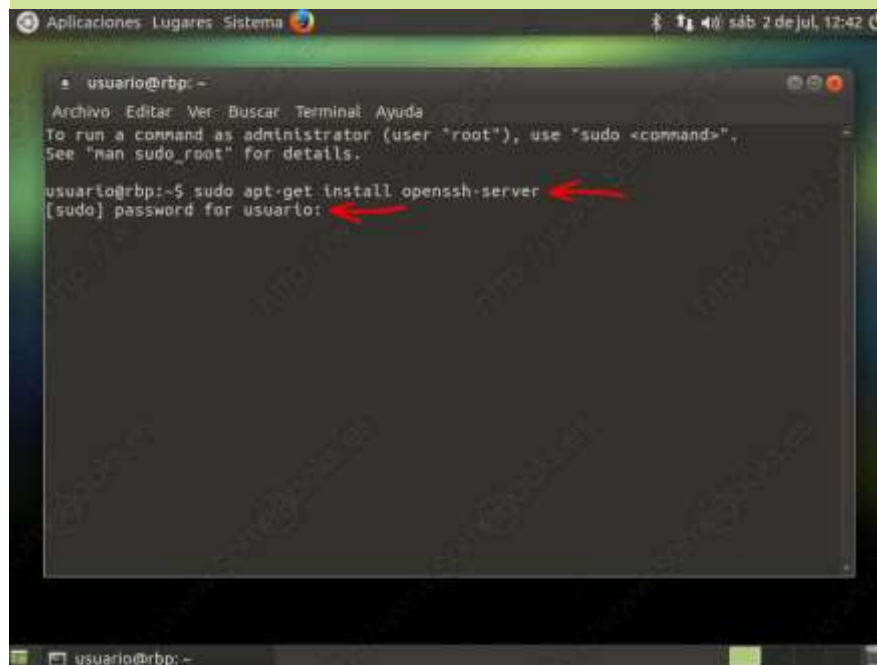
próximos artículos de esta serie te mostraremos cómo acceder a él desde *Windows* o desde *Ubuntu*, según te interese.

En realidad, la instalación es tan sencilla como ejecutar:

```
sudo apt update
```

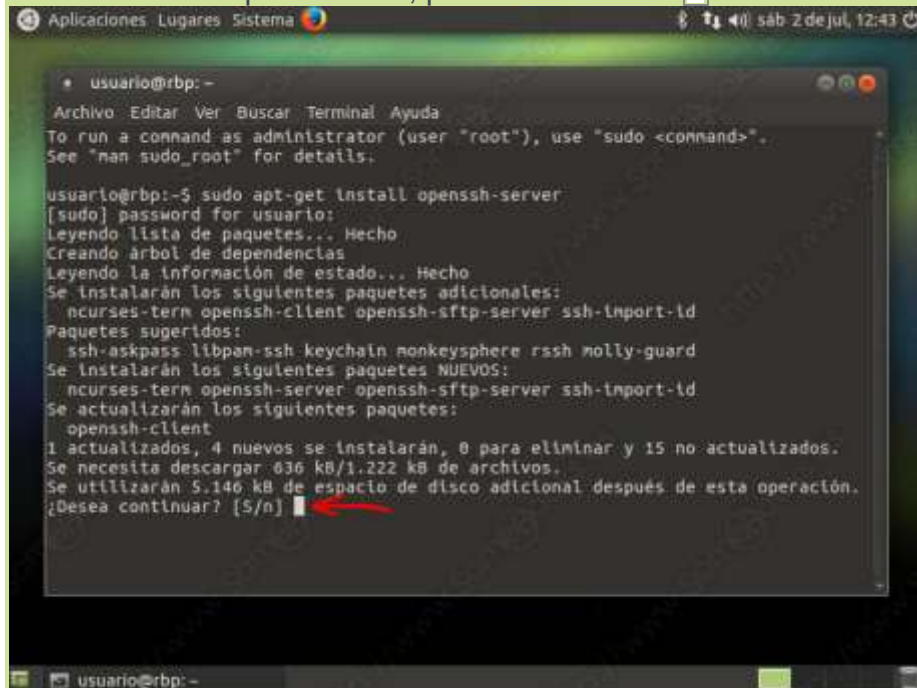
```
sudo apt install openssh-server
```

## 1 Instalamos el paquete **openssh-server**



Durante la instalación, el sistema nos informa de las dependencias que deberán resolverse para completar la tarea.

**2** Cuando nos pregunte si deseamos continuar instalando las dependencias, pulsamos la tecla **S**.

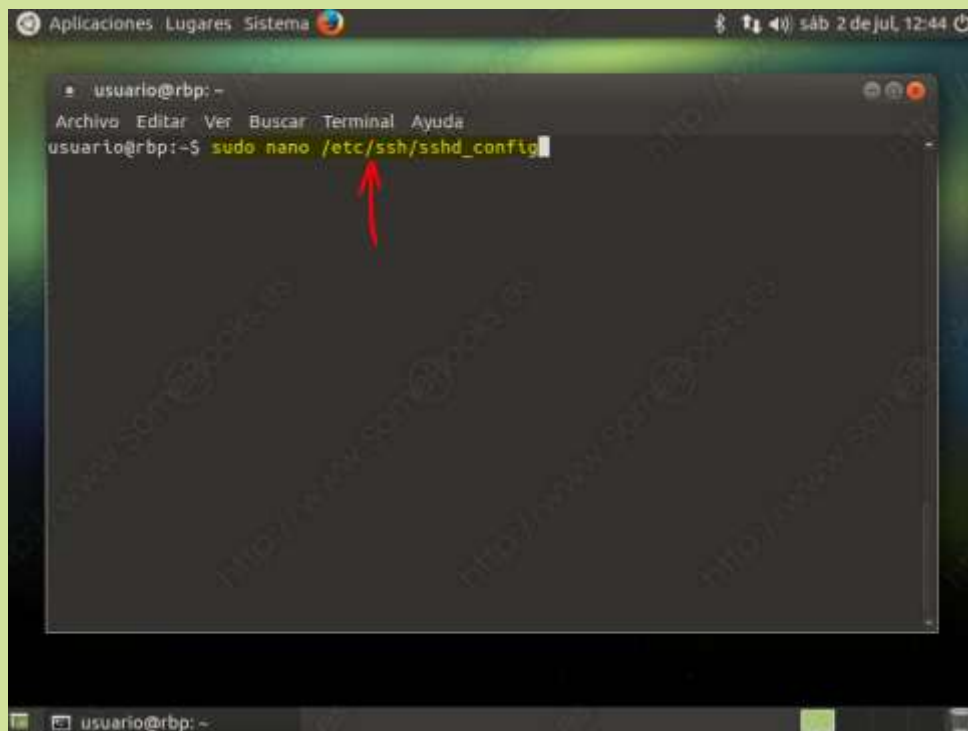


```
usuario@rbp: ~$ sudo apt-get install openssh-server
[sudo] password for usuario:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
ncurses-term openssh-client openssh-sftp-server ssh-import-id
Paquetes sugeridos:
ssh-askpass libpam-ssh keychain monkeysphere rssh molly-guard
Se instalarán los siguientes paquetes NUEVOS:
ncurses-term openssh-server openssh-sftp-server ssh-import-id
Se actualizarán los siguientes paquetes:
openssh-client
1 actualizados, 4 nuevos se instalarán, 0 para eliminar y 15 no actualizados.
Se necesita descargar 636 kB/1.222 kB de archivos.
Se utilizarán 5.146 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Con esto ya sería suficiente. El equipo estaría listo para recibir solicitudes de conexión. Sin embargo, aún es recomendable hacer algún ajuste para mejorar un poco más la seguridad. Para lograrlo, sólo tenemos que editar el archivo `/etc/ssh/sshd_config`, por ejemplo, con el siguiente comando:

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

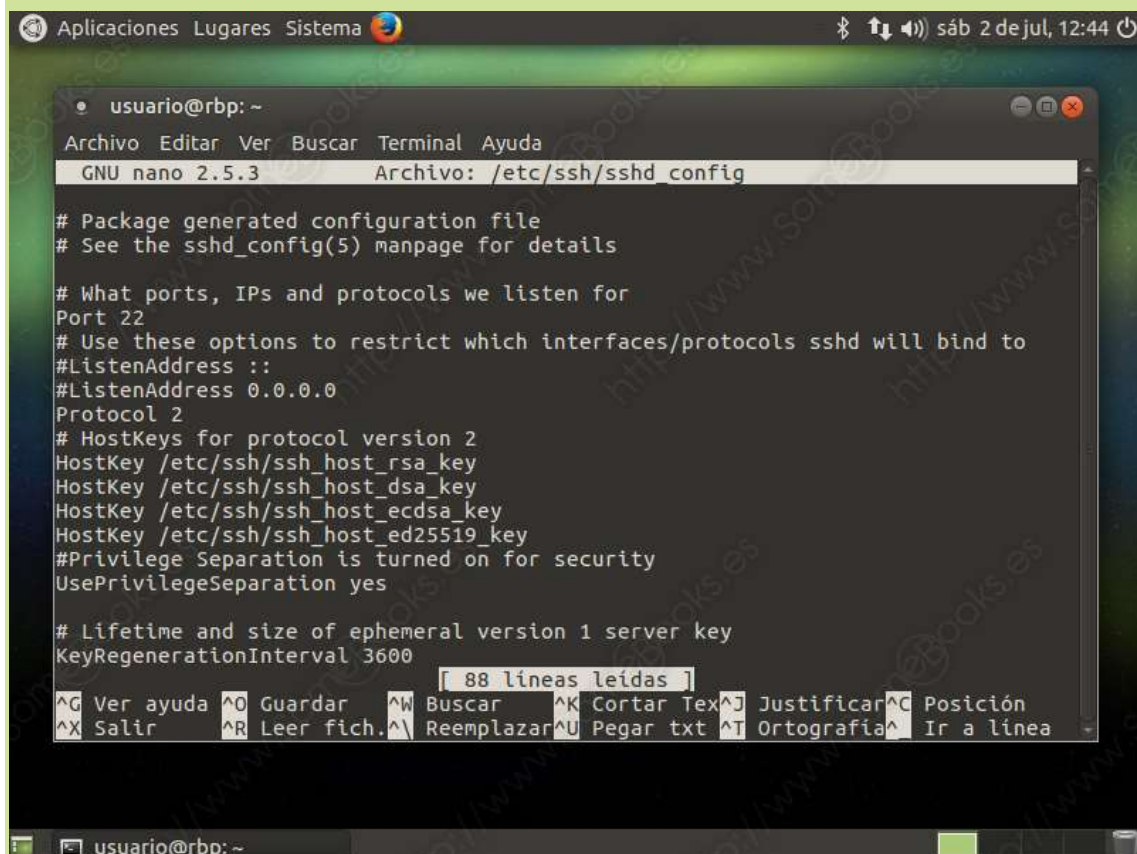
### 3 Editamos el archivo `/etc/ssh/sshd_config` con el editor *nano*.



A terminal window titled 'usuario@rbp: ~' with a menu bar: 'Archivo Editar Ver Buscar Terminal Ayuda'. The command 'sudo nano /etc/ssh/sshd\_config' is entered and highlighted in yellow. A red arrow points to the command. The window title bar shows 'Aplicaciones Lugares Sistema' and the date 'sáb 2 de jul, 12:44'.

El editor se abrirá y nos mostrará el contenido del archivo.

### 4 Contenido inicial de `sshd_config`.



A terminal window titled 'usuario@rbp: ~' with a menu bar: 'Archivo Editar Ver Buscar Terminal Ayuda'. The title bar shows 'GNU nano 2.5.3' and 'Archivo: /etc/ssh/sshd config'. The content of the file is displayed, including comments and configuration options. A status bar at the bottom shows '88 líneas leídas' and a list of keyboard shortcuts.

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

# What ports, IPs and protocols we listen for
Port 22
# 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
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

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

88 líneas leídas

Ver ayuda Guardar Buscar Cortar Text Justificar Posición  
Salir Leer fich. Reemplazar Pegar txt Ortografía Ir a línea

El archivo `sshd_config` contiene la configuración que utilizará *OpenSSH* para funcionar. En él, deberíamos cambiar al menos los siguientes datos:

- El parámetro `Port`, contiene el puerto que emplearemos para la conexión. De forma predeterminada se utiliza el 22, pero, como una medida básica para impedir accesos indebidos, lo cambiaremos por un valor diferente. Podemos elegir cualquier valor entre 1025 y 65536 (por ejemplo, el 5000).
- El parámetro `AllowUsers`, si existe, limita la conexión remota sólo a los usuarios relacionados a continuación. De forma opcional, podemos añadir la IP de los ordenadores desde los que pueden conectarse dichos usuarios (aunque necesitaríamos que dichos clientes tuviesen también una IP fija).

```
Port 5000
```

```
AllowUsers usuario@192.168.1.100
```

Si necesitásemos incluir más de un usuario, los añadiríamos a continuación del primero, separados por espacios en blanco, repitiendo el mismo formato *nombre\_de\_usuario@dirección\_ip*.

En nuestro ejemplo, sólo limitaremos por nombre de usuario.

## 5 Realizamos los cambios en `Port` y añadimos la entrada `Allowusers`.



```
usuario@rbp: -
GNU nano 2.5.3 Archivo: /etc/ssh/sshd_config Modificado

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

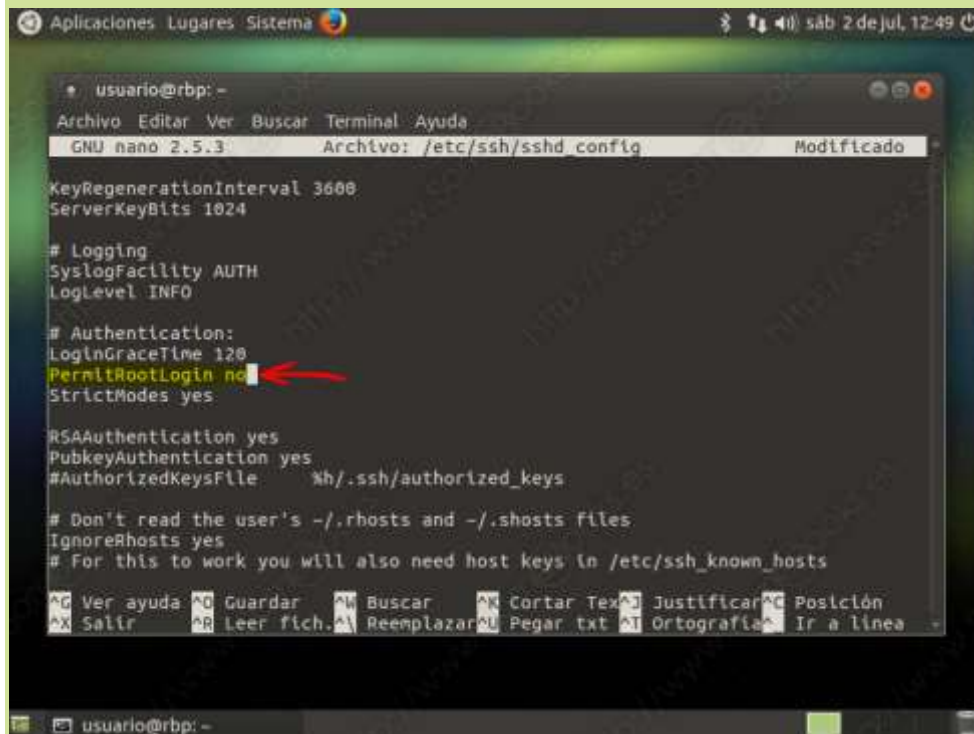
# What ports, IPs and protocols we listen for
Port 5000
AllowUsers usuario

# 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
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^M Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

En el parámetro `PermitRootLogin`, cambiaremos su valor a `no`, para impedir accesos remotos con el superusuario.

## 6 Asignamos el valor `no` a `PermitRootLogin`.



```
usuario@rbp:~$ nano /etc/ssh/sshd_config
GNU nano 2.5.3 Archivo: /etc/ssh/sshd_config Modificado

KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes

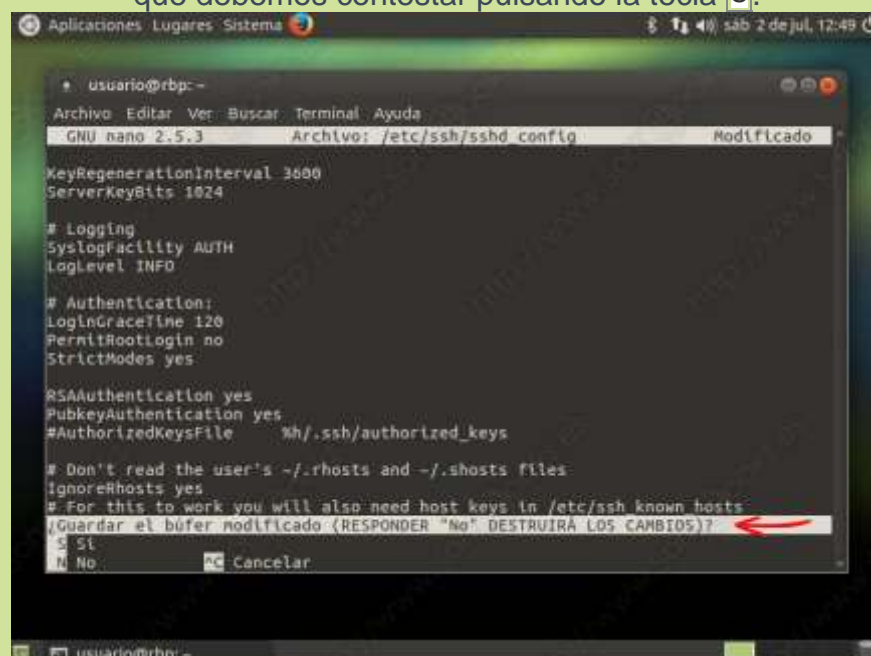
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys

# 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

Ver ayuda Guardar Buscar Cortar Text Justificar Posición
Salir Leer fich. Reemplazar Pegar txt Ortografía Ir a línea
```

Cuando hayamos terminado las modificaciones, pulsamos la combinación de teclas `Ctrl` + `X` para salir

## 7 El editor nos preguntará si queremos guardar los cambios, a lo que debemos contestar pulsando la tecla `S`.



```
usuario@rbp:~$ nano /etc/ssh/sshd_config
GNU nano 2.5.3 Archivo: /etc/ssh/sshd_config Modificado

KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes

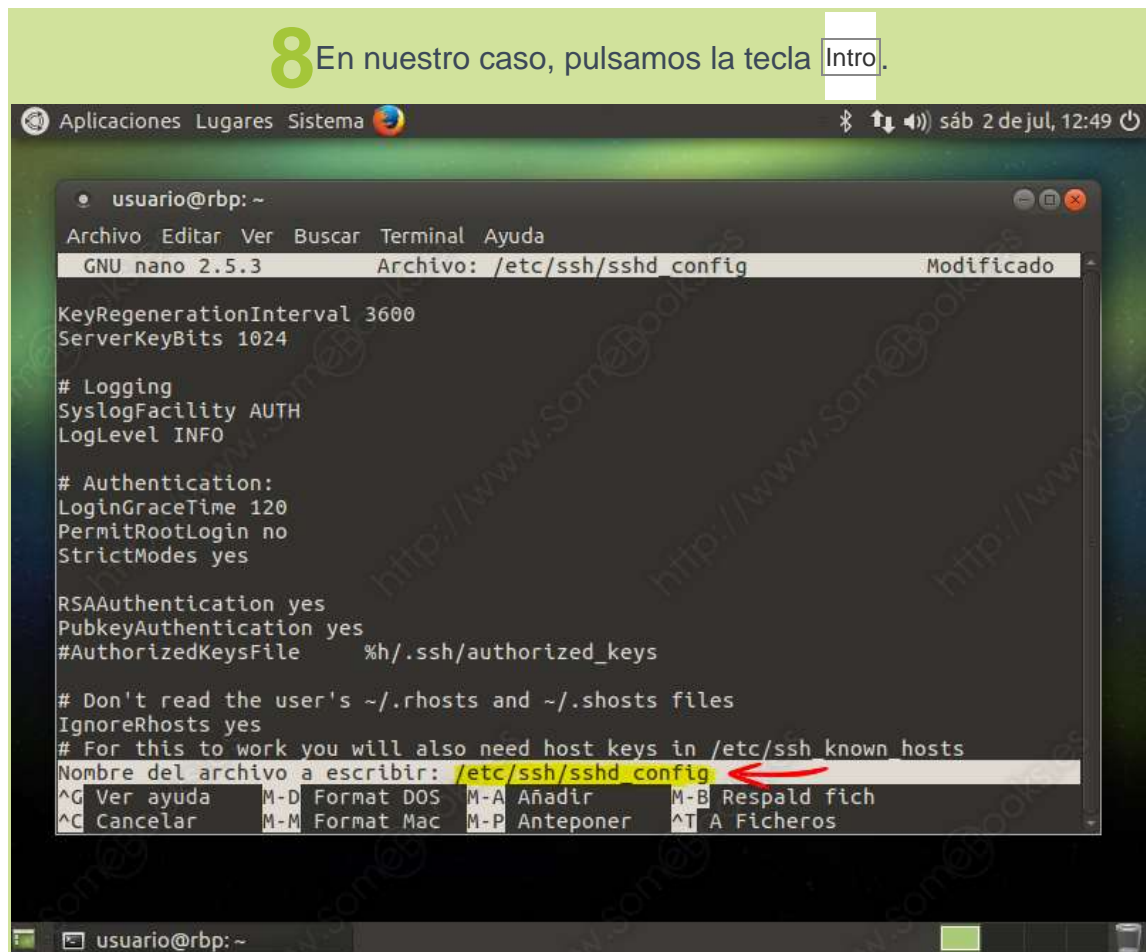
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys

# 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

Guardar el búfer modificado (RESPONDER "No" DESTRUIRÁ LOS CAMBIOS)?
S Si
N No Cancelar
```



Después de esto, el editor nos pide el nombre que queremos darle al archivo, por si no queremos sobrescribir el original.

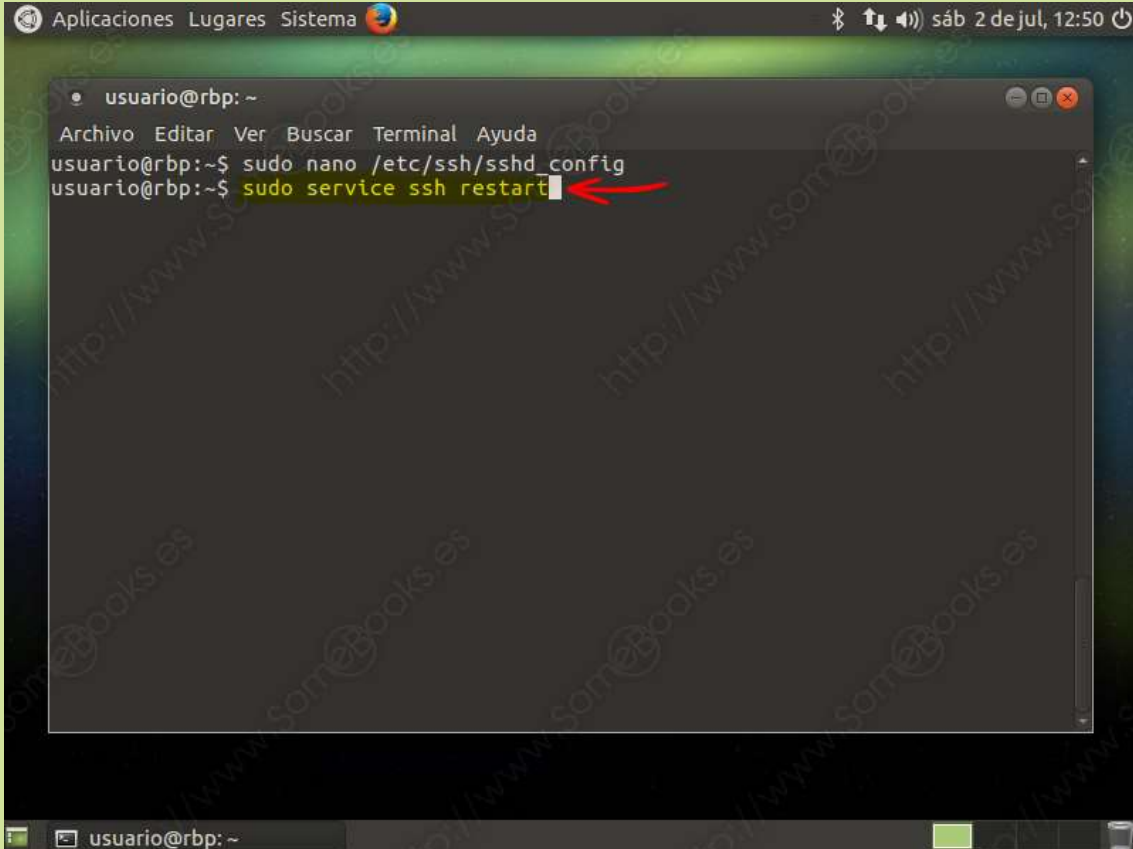


Al salir del editor, sólo nos quedará reiniciar el servicio para que empiece a realizar su trabajo. Para lograrlo, usamos el siguiente comando:

```
sudo service ssh restart
```



## 9 Reiniciamos el servicio.



```
usuario@rbp: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
usuario@rbp:~$ sudo nano /etc/ssh/sshd_config  
usuario@rbp:~$ sudo service ssh restart
```

Y con esto, el equipo estará listo para comenzar a recibir conexiones desde otros equipos de la red.

### 3 SSH (PARTE II): ACCEDER DESDE OTRO EQUIPO CON UBUNTU 16.04 LTS

---

Fuente: <http://somebooks.es/ssh-parte-ii-acceder-desde-otro-equipo-con-ubuntu-16-04-lts/>

Publicado por **P. Ruiz** en 5 agosto, 2016

Hace unos días, [en la primera parte](#) de este artículo, te explicábamos cómo instalar *OpenSSH* en un ordenador con *Ubuntu* para poder administrarlo de forma segura a través de una terminal de texto remota.

Si en el archivo `sshd_config` limitaste la *dirección IP* desde la que podrían iniciarse conexiones, el contenido de este artículo sólo te funcionará desde dichos ordenadores.

Hicimos la instalación sobre un ordenador que ejecutaba *Ubuntu Mate 16.04 LTS*, con la intención de que pudieras aplicarlo sobre tu *Raspberry Pi*. Así, podrías comenzar a verle las ventajas al contenido de nuestros artículos previos [Crear una imagen de instalación de Ubuntu Mate para Raspberry Pi desde Ubuntu](#) o [desde Windows](#). Sin embargo, te recuerdo que los pasos son idénticos para cualquier otro *sabor* de *Ubuntu*.

Pues bien, si ya has seguido los pasos del primer artículo, hoy te enseñamos a acceder de forma remota desde cualquier otro ordenador de la red. En este caso un ordenador que ejecute *Ubuntu 16.04 LTS*, aunque, también en este caso, sería prácticamente igual desde cualquier otro *Ubuntu*.

Es importante decir que, la mayoría de las distribuciones *Linux* ya incorporan de forma predeterminada el *cliente SSH*, por lo que no es frecuente que necesitemos instalarlo.

Por lo tanto, para utilizarlo, sólo tenemos que escribir algo como esto:

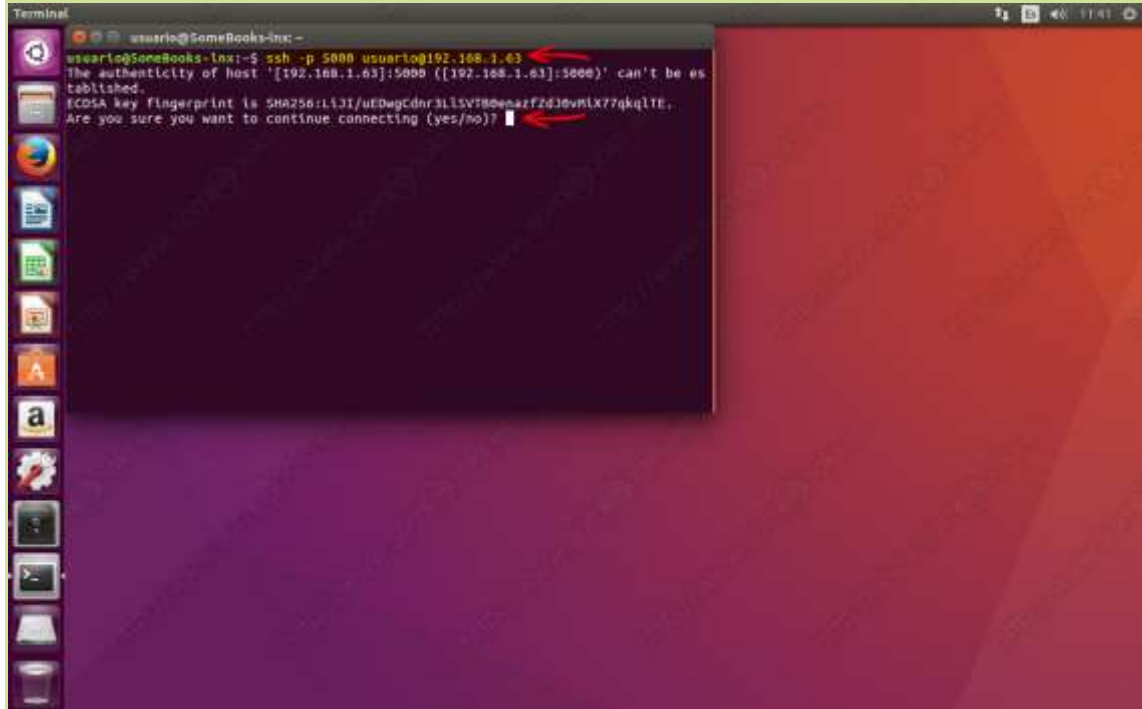
```
ssh -p 5000 usuario@192.168.1.10
```

Como se puede suponer, el argumento `-p` se utiliza para indicar el número del puerto que utiliza el servidor para escuchar solicitudes de conexión *SSH* (recuerda que lo configuramos así en el archivo `sshd_config`).

Después, indicaremos la cuenta de usuario que queremos utilizar para autenticarnos y la *dirección IP* del equipo al que accederemos (ambos separados por un carácter `@`).

Como es lógico, la cuenta que indiquemos debe estar definida con anterioridad en el ordenador al que queremos acceder y, en su caso, incluida en la variable `Allowusers`, en el archivo `sshd_config` de dicho ordenador.

## 1 Escribimos el comando `ssh` anterior.

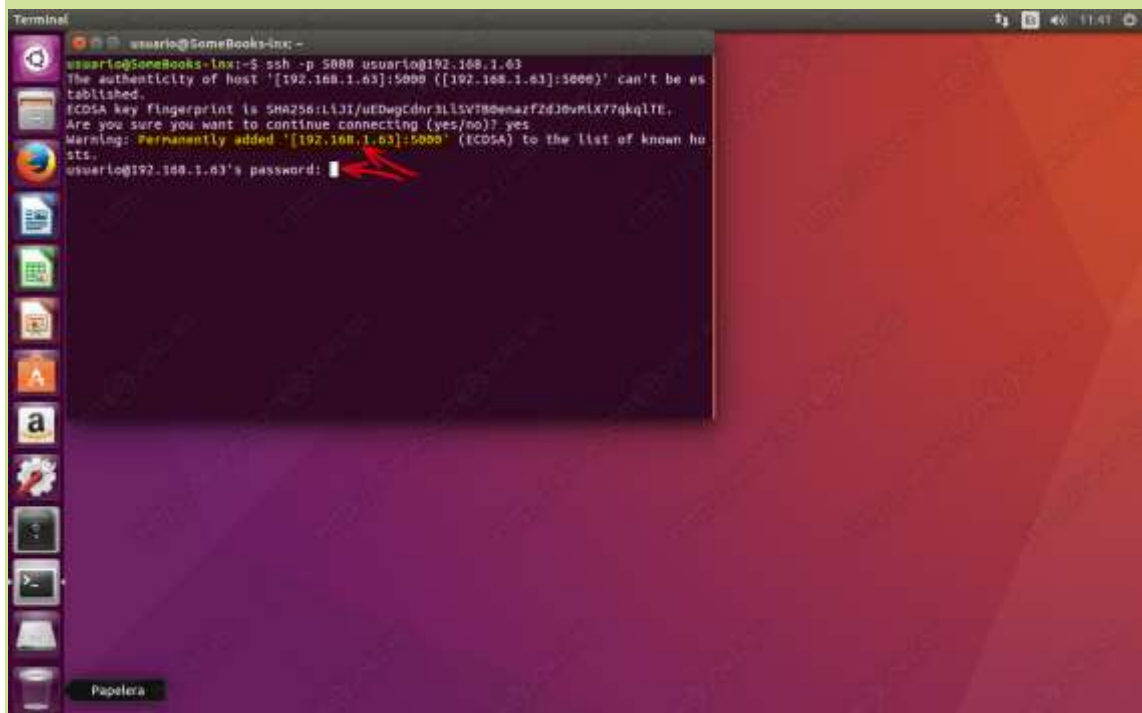


De forma predeterminada, *SSH* utiliza un método de cifrado basado en clave pública. Como es la primera vez que nos conectamos, el *cliente SSH* nos informa de que no conoce la clave pública que le está ofreciendo el servidor y la muestra para que podamos validarla.

Dado que estamos en un entorno de red controlado y sí estamos seguros de que el servidor al que nos estamos conectando es confiable, contestamos Sí (yes) a la pregunta de si queremos continuar con la conexión.

Al aceptar la conexión, el *cliente SSH* guarda esta clave pública en el archivo `known_hosts`, que se encuentra dentro de la carpeta `.ssh` en el perfil del usuario con el que estamos iniciando la conexión (en este caso, `/home/usuario/.ssh/known_hosts`). De esta forma, la próxima vez que iniciemos sesión en el mismo servidor, ya no volverá a preguntarnos.

## 2 A continuación, ya podemos identificarnos en el servidor.



Obviamente, para la autenticación usaremos la contraseña que corresponda con la cuenta a la que nos estamos conectando en el equipo remoto.

## 3 Si todo es correcto, el sistema nos da la bienvenida y nos ofrece el *prompt* listo para que comencemos a escribir comandos.



Observa que el *prompt* ha cambiado: En las primeras imágenes se mostraba el *prompt* del equipo desde el que accedíamos porque es ahí donde ejecutamos el *cliente SSH* (en nuestro ejemplo, `usuario@SomeBooks-lnx`), mientras que al final de esta última imagen se muestra el *prompt* del ordenador al que estamos accediendo (en nuestro caso, `usuario@rbp:~`). Esto nos muestra claramente que la conexión está activa (aunque en ambos casos estemos usando una cuenta llamada usuario, pero esto es una simple coincidencia).

A partir de aquí, podremos escribir ordenes en el *ordenador remoto*, desde el *ordenador local*, como si estuviésemos físicamente delante del *ordenador remoto*. Por ejemplo, en la siguiente imagen hemos utilizado el comando `ls` para ver el contenido de la carpeta del usuario.



Usando esta sencilla técnica, puedes tener un ordenador en un lugar poco accesible y administrarlo de forma remota y cómoda desde un ordenador diferente. Por ejemplo, puedes usarlo para administrar la *Raspberry Pi* que tienes conectada al televisor de tu salón, desde tu ordenador de sobremesa o tu portátil.

## 4 SSH (PARTE III): ACCEDER DESDE OTRO EQUIPO CON WINDOWS 10

---

Fuente: <http://somebooks.es/ssh-parte-iii-acceder-desde-otro-equipo-con-windows-10/>

Publicado por **P. Ruiz** en 8 agosto, 2016

Hace unos días, [en la primera parte](#) de este artículo, te explicábamos cómo instalar *OpenSSH* en un ordenador con *Ubuntu* para poder administrarlo de forma segura a través de una terminal de texto remota. También te hemos explicado cómo [Acceder desde otro equipo con Ubuntu 16.04 LTS](#) al equipo que habías configurado durante aquél primer artículo.

Pues bien, siguiendo con la serie, hoy aprenderemos a acceder en modo terminal (o línea de comandos, si lo prefieres) al equipo remoto, pero desde un ordenador equipado con el sistema operativo *Windows 10*.

En realidad, lo único que necesitamos es un programa como *PuTTY*.

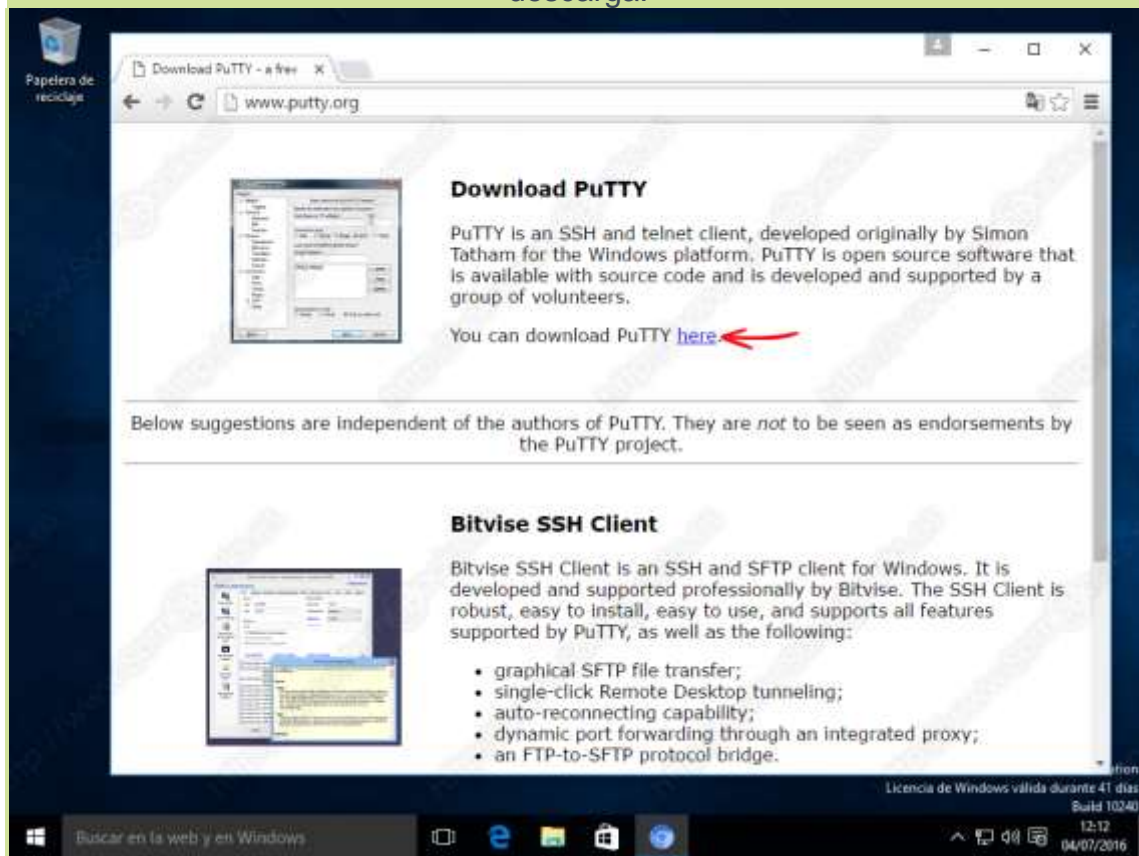
También podemos encontrar *PuTTY* en los repositorios de *Ubuntu* y en *OSX* a través de *MacPorts*

Se trata de un cliente telnet y SSH programado originalmente por *Simon Tatham* para sistemas *Windows*, aunque ahora lo mantienen un grupo de voluntarios. Dado que es software libre, podemos obtenerlo gratuitamente tanto su versión ejecutable como su código fuente.

Para descargar la versión *Windows*, sólo tenemos que acceder al área de descargas de su página web (<http://www.putty.org/>).



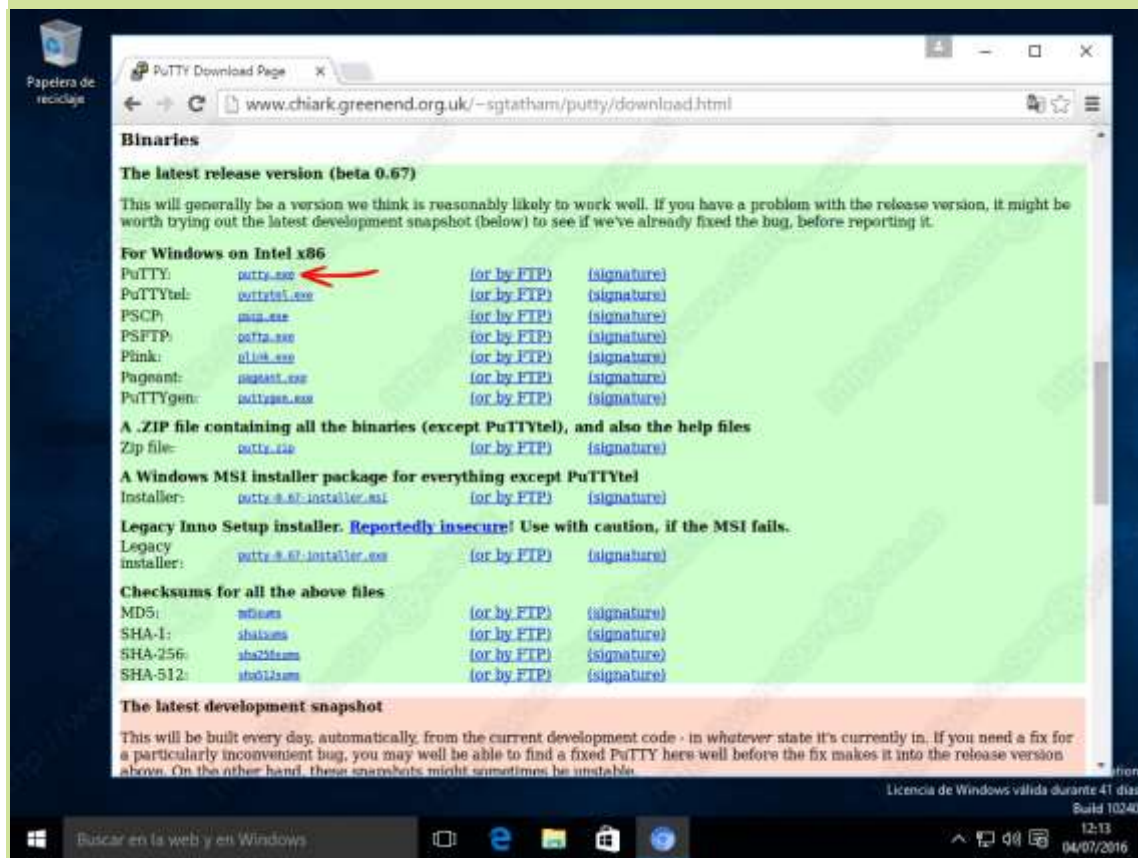
1 Una vez en su página web, comenzamos por hacer clic sobre el enlace de descarga.



Éste, nos lleva a una página donde encontramos multitud de posibilidades, divididas en dos áreas fundamentales: Una con fondo verde, que corresponde a la versión estable, y otra con fondo rosa, que contiene la versión de desarrollo y que se actualiza diariamente.

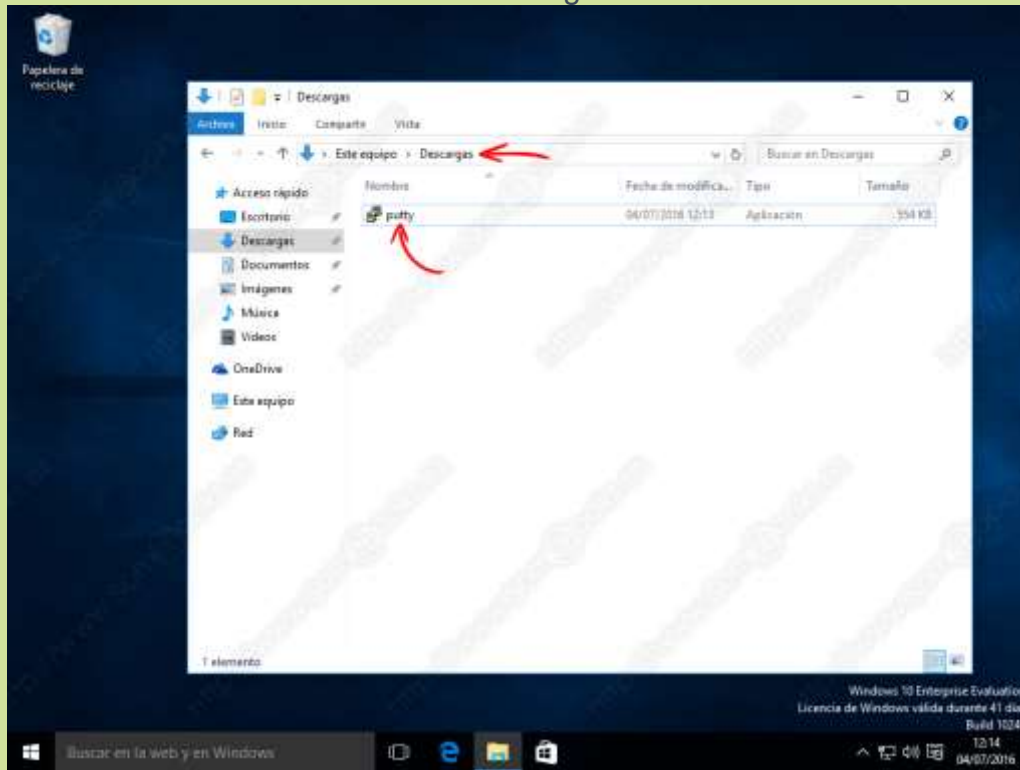
Además de la versión estándar (*putty.exe*), disponemos de clientes para el protocolo *Secure Copy* (*pscp.exe*), para utilizar *FTP* sobre *SSH* (*psftp.exe*) o para utilizar *telnet* sin cifrado (*puttytel.exe*), entre otros.

## 2 Nosotros nos decantamos por la versión estándar.



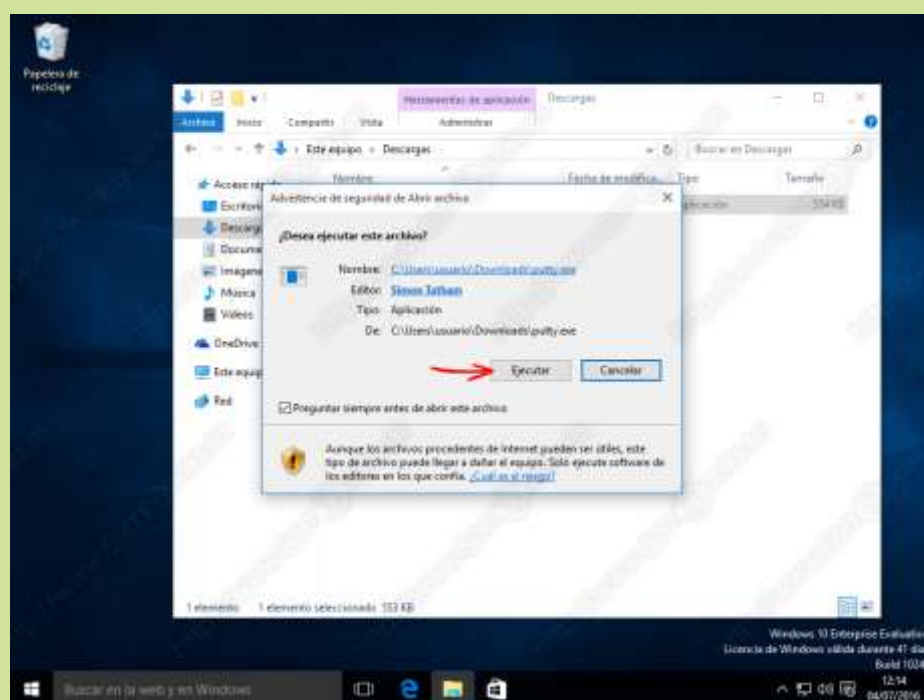
Una característica interesante de *PuTTY* es que, una vez descargado, no hay que instalarlo. Así, podemos guardar el ejecutable en una *memoria USB*, por ejemplo, y llevarlo con nosotros a cualquier ordenador donde podamos necesitarlo.

**3** Para ejecutarlo, basta con hacer doble clic sobre el archivo que acabamos de descargar.



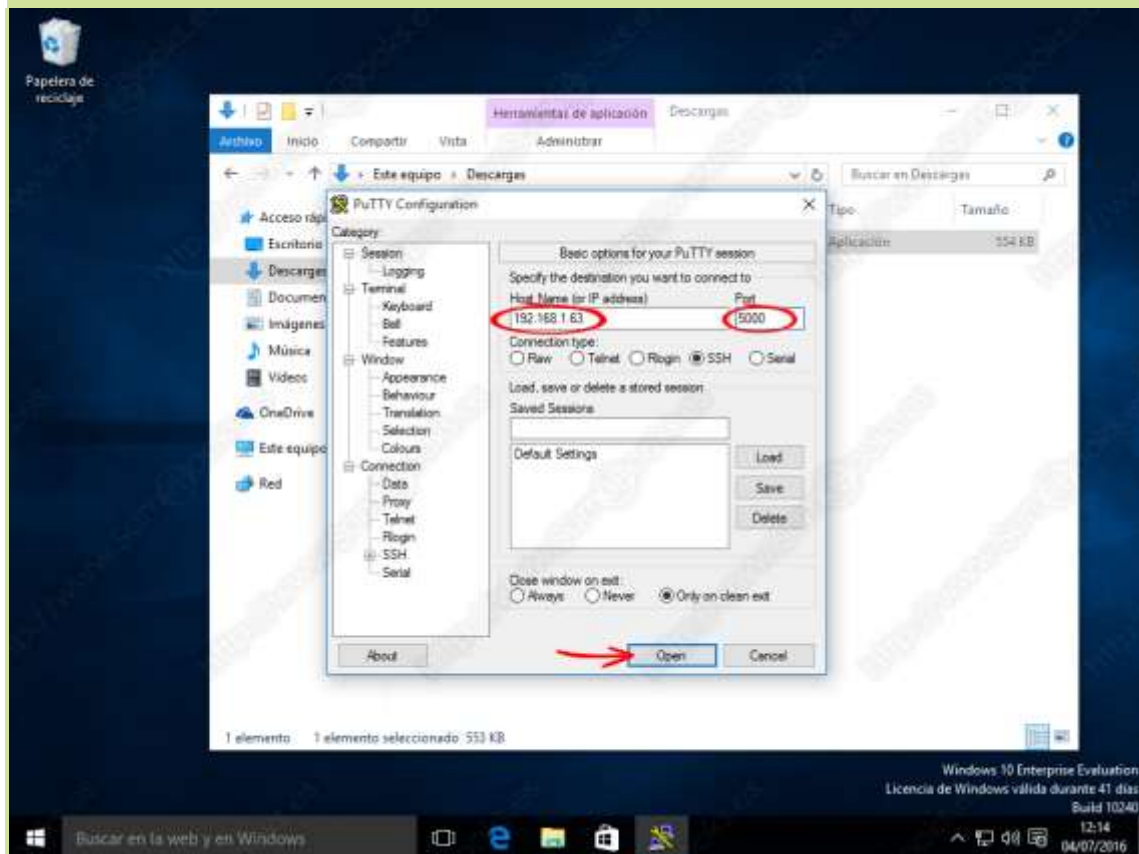
Como de costumbre, *Windows 10* nos pide que autoricemos la ejecución del programa, ya que proviene de un origen desconocido para él.

**4** Hacemos clic sobre el botón *Ejecutar*.



Esto hará que se abra la ventana que utiliza *PuTTY* para configurar la conexión. En ella escribiremos, como mínimo, la *dirección IP* del servidor y el puerto que está usando para atender intentos de conexión (recuerda que lo configuramos con el valor 5000)

**5** Una vez introducidos los valores, haremos clic sobre el botón *Open*.



Veremos que, inmediatamente, se abre una ventana de texto donde se va a establecer la comunicación con la terminal del servidor.

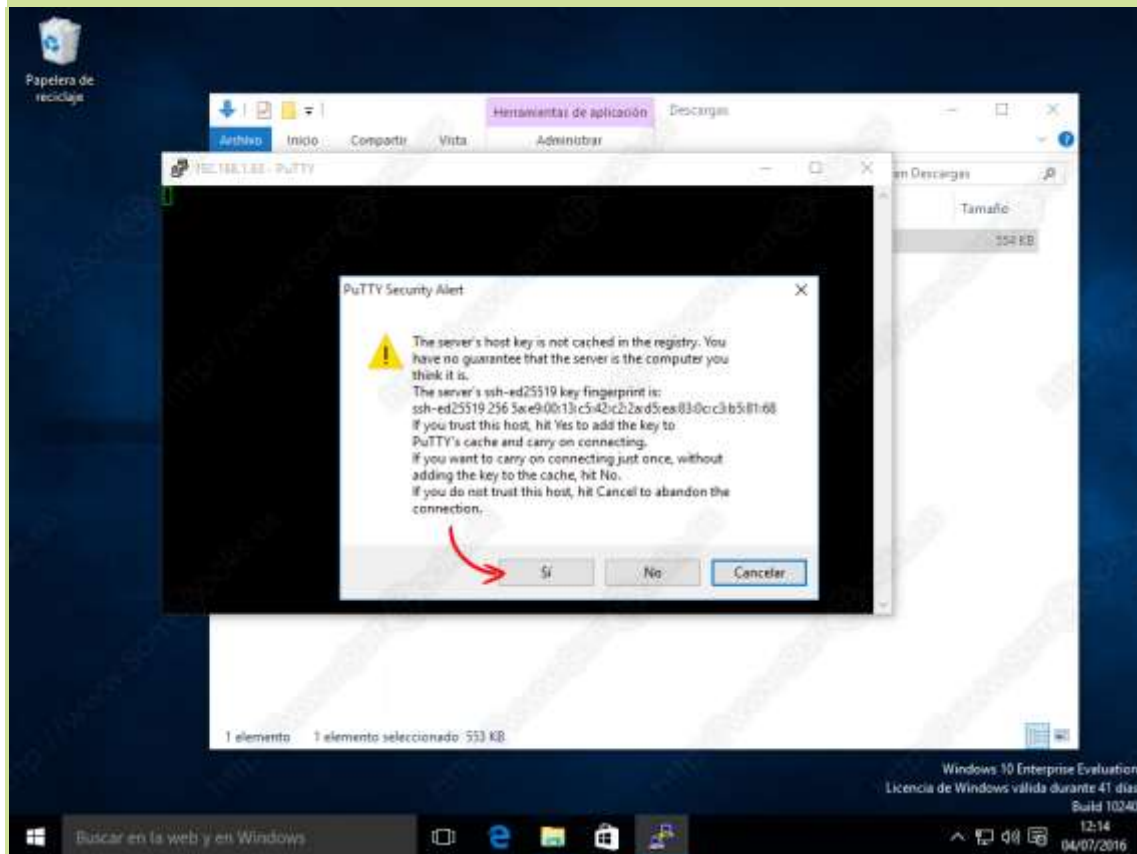
Sin embargo, como ya ocurrió en *Ubuntu*, dado que es la primera vez que intentamos acceder desde *Windows 10*, *PuTTY* nos muestra una ventana donde avisa de que no conoce la clave pública que le está enviando el servidor y la muestra para que podamos validarla.

Igual que antes, seguimos en un entorno de red controlado y estamos seguros de que el servidor al que nos estamos conectando es confiable. No obstante, *PuTTY* nos ofrece una opción más que la terminal de *Ubuntu 16.04 LTS*:

- Si hacemos clic sobre el botón *Sí*, añadiremos la clave al archivo de claves de *PuTTY* para que la próxima vez reconozca al servidor y no vuelva a mostrar este mensaje. Esta es la opción que equivale a la que usamos en *Ubuntu 16.04 LTS*.

- Si hacemos clic sobre el botón *No*, estableceremos igualmente la conexión con el servidor, pero *PuTTY* no guardará la clave pública del servidor.
- Si hacemos clic sobre el botón *Cancelar*, abortaremos el intento de conexión.

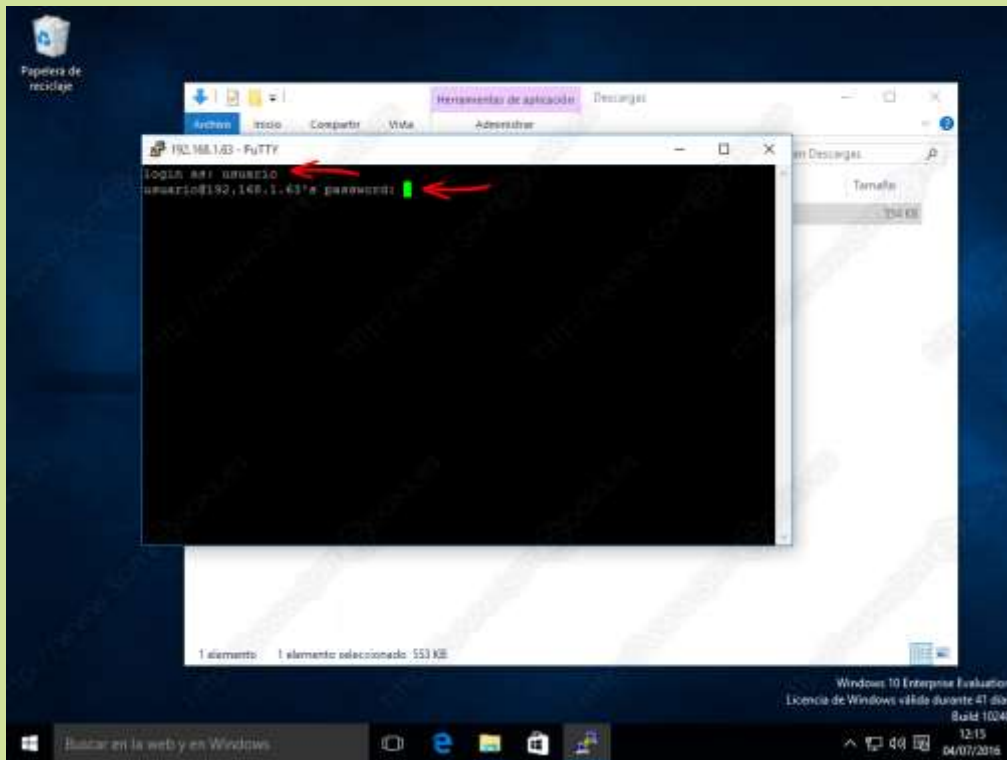
**6** En nuestro caso, hacemos clic sobre el botón *Sí*.



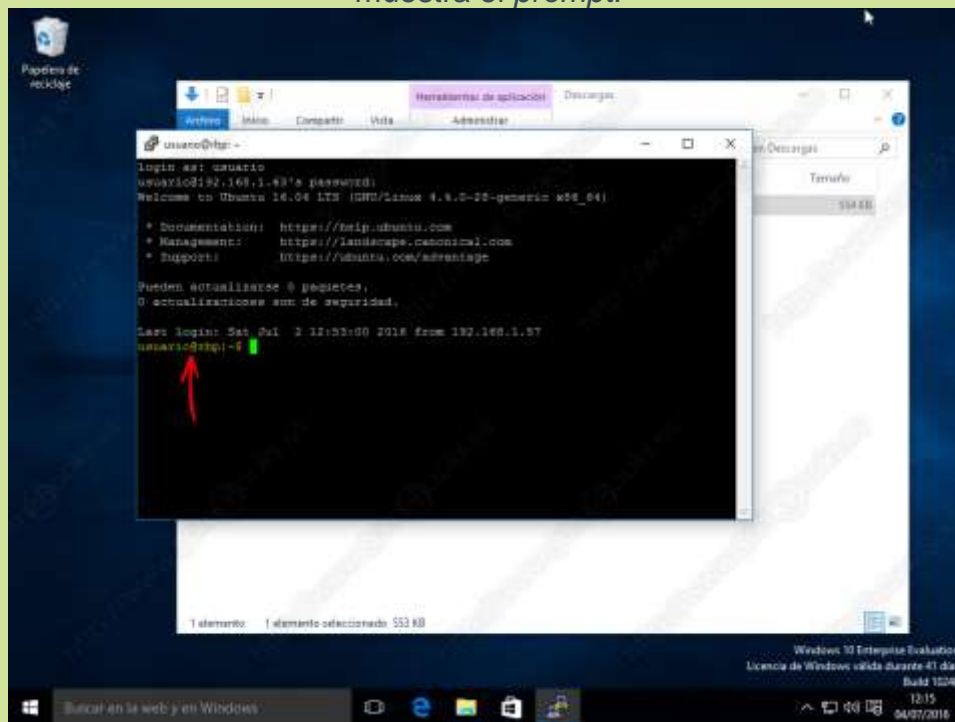
Al cerrar la ventana de aviso, la ventana de texto que había detrás nos solicita el nombre de usuario (*login as:*) y la contraseña (*password*).



## 7 Escribimos los datos apropiados y pulsamos la tecla *Intro*.



## 8 Veremos cómo aparece el mensaje de bienvenida del *servidor*, que nos muestra el *prompt*.



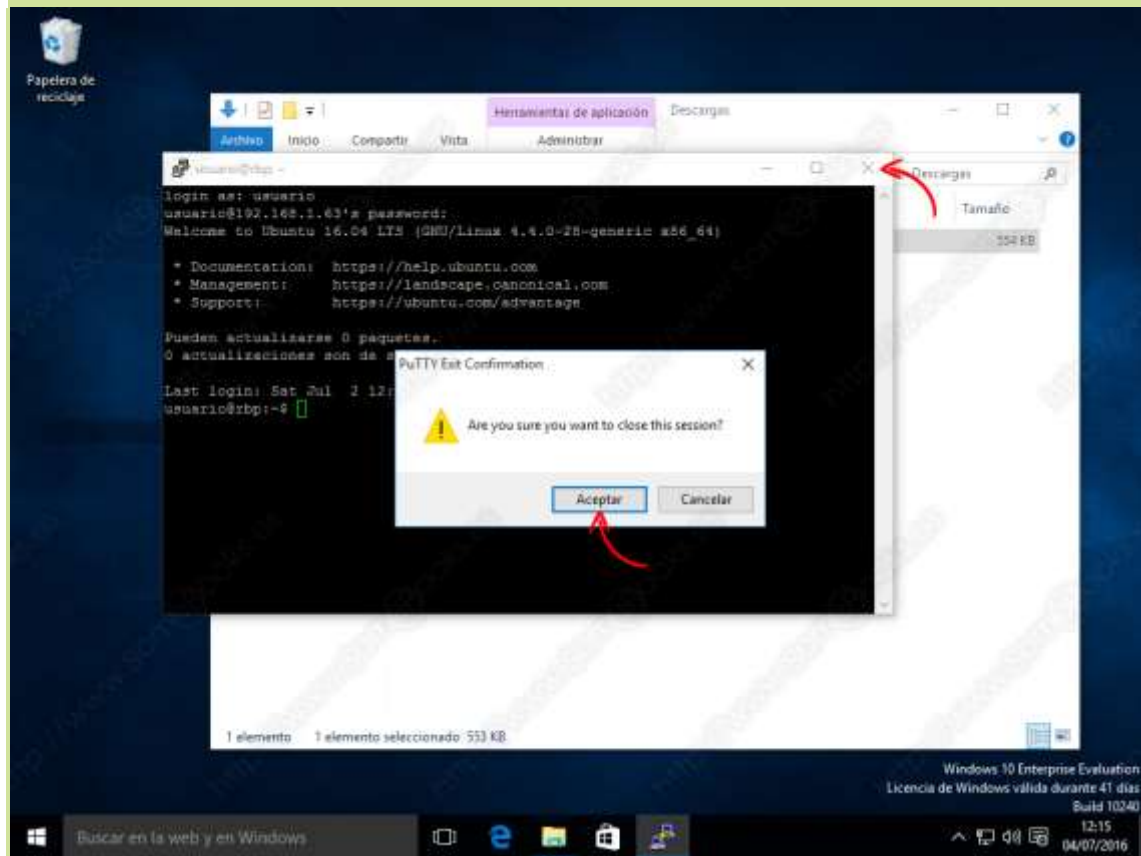
Observa que el *prompt* se corresponde con el del equipo remoto. Ya podemos comenzar su administración.



Cuando acabemos, basta con escribir la orden `exit`. Esto hará que se cierre la sesión en el *servidor* y la ventana de texto de *PuTTY*.

Si en lugar de escribir la orden `exit` cerramos directamente la ventana de texto, PuTTY nos preguntará si queremos cerrar la sesión.

## 9 Haremos clic sobre *Aceptar*.



Si hemos intentado cerrar la ventana por error, basta con hacer clic sobre el botón *Cancelar* para volver a la ventana de texto.