U06 Linux SSH

Descargar PDF

ÍNDICE

SERVIDOR SSH

INSTALACIÓN

Dependiendo de la distribución GNU/Linux que se utilice, el proceso de instalación del servidor puede variar. Para los sistemas Debian y derivados (Ubuntu, Linux Mint, etc.) se utiliza el sistema de gestión de paquetes `apt`, mientras que en Alpine Linux se emplea `apk`.

En Debian, el paquete que proporciona el servidor SSH se llama `openssh-server` y el que proporciona el cliente se llama `openssh-client`. El paquete `openssh-server` depende del paquete `openssh-client` y, por lo tanto, al instalar el servidor se instalará automáticamente el cliente. Existe un metapaquete llamado `ssh` que instala ambos paquetes. En este caso instalar ssh u openssh-server es equivalente, ya que ambos instalarán el servidor y el cliente.

En Alpine Linux, el paquete que proporciona el servidor SSH se llama `openssh-server` y el que proporciona el cliente se llama `openssh-client`. También tenemos el metapaquete que instala ambos y que se llama `openssh`.

Distribución Debian

```
apt update
apt install openssh-server
```

Distribución Alpine

```
apk update
apk add openssh-server
```

En Debian, tras la instalación se crea y activa el servicio `ssh` (o `sshd`) gestionado por `systemd`, y se generan las claves de host en `/etc/ssh/`. Sin embargo, en Alpine Linux el proceso es diferente: tras instalar el paquete `openssh-server` no se genera automáticamente el servicio ni las claves de host. Por lo tanto, en Alpine Linux es necesario generar manualmente las claves de host con el comando `ssh-keygen` y luego iniciar el servicio `sshd` utilizando `rc-service`. Para hacer que el servicio `sshd` se inicie automáticamente al arrancar el sistema en Alpine Linux, se debe habilitar con `rc-update`.

Instalación SSH

En Debian, el proceso de instalación del servidor SSH se realiza con el siguiente comando:

```
apt install openssh-server
```

En Alpine Linux, el proceso de instalación es el siguiente:

```
apk add openssh-server
ssh-keygen -A
rc-service sshd start
rc-update add sshd default
```

La última línea asegura que el servicio `sshd` se inicie automáticamente al arrancar el sistema.

PROGRAMAS RELACIONADOS

Cuando instalamos el servidor OpenSSH en un sistema Debian (o derivados) no solo se pone en marcha el servicio, sino que se añaden varias utilidades que cubren las tareas habituales relacionadas con SSH: ofrecer el

servicio, conectarse a otros equipos, transferir ficheros y gestionar las claves de autenticación.

- **sshd (servidor SSH).** Es el servicio o demonio que queda en segundo plano escuchando normalmente en el puerto 22 a la espera de conexiones entrantes. Se administra como cualquier otro servicio del sistema (por ejemplo, con `systemctl start ssh`, `systemctl stop ssh`, etc.) y se configura principalmente a través del fichero `/etc/ssh/sshd_config`.
- **ssh (cliente SSH).** Es el programa que se utiliza para conectarse de forma remota a otro equipo. Permite tanto abrir una sesión interactiva (tipo consola) como ejecutar un único comando en la máquina remota, recibiendo la salida en nuestra terminal local.
- **scp (copia de archivos sobre SSH).** Proporciona una forma sencilla de copiar archivos y directorios entre el equipo local y el remoto utilizando SSH como canal cifrado. La sintaxis es similar a la de `cp`, pero indicando el equipo remoto con la forma `usuario@host:ruta`.
- **sftp (SFTP: "FTP sobre SSH").** Ofrece un cliente interactivo de transferencia de archivos, similar a un cliente FTP tradicional, pero utilizando SSH como transporte seguro. Desde este entorno se pueden listar directorios, subir y descargar archivos, borrar, renombrar, etc., con comandos como `ls`, `cd`, `get` o `put`.
- **ssh-keygen (creación y gestión de claves).** Se emplea para crear y administrar pares de claves pública/privada que permiten la autenticación sin contraseña. También genera las claves de host que identifican al propio servidor (las que se almacenan en `/etc/ssh/ssh_host_*`). Permite elegir el tipo de clave (`rsa`, `ed25519`, etc.), su tamaño, la ruta de almacenamiento y la frase secreta que protege la clave privada.
- **ssh-agent (agente de claves en memoria).** Es un proceso auxiliar que guarda en memoria las claves privadas descifradas mientras dura la sesión de trabajo. De esta forma, el usuario introduce la frase secreta de la clave una sola vez y, a partir de ahí, las conexiones `ssh`, `scp` o `sftp` pueden utilizarla sin volver a pedirla.
- **ssh-add (gestión de claves del agente).** Es la herramienta que se utiliza para añadir o eliminar claves privadas gestionadas por `ssh-agent`. Por ejemplo, `ssh-add ~/.ssh/id_ed25519` carga en el agente la clave privada indicada tras introducir su frase secreta.
- **ssh-keyscan (recogida de claves públicas de servidores).** Permite obtener de forma rápida las claves públicas de host de uno o varios servidores remotos. Resulta útil para poblar o revisar el fichero `known_hosts` sin necesidad de establecer una conexión interactiva completa.

Además de estos programas principales, la instalación de OpenSSH incorpora otras utilidades auxiliares (como `ssh-askpass`, `ssh-keysign`, etc.) que normalmente no se ejecutan de manera directa, sino que son invocadas internamente por el propio sistema en escenarios de autenticación más avanzados.

Para publicar claves en un servidor y poder usar autenticación basada en claves, lo más sencillo es utilizar `ssh-copy-id`. En caso de que no esté disponible, siempre se puede recurrir a `scp` y editar manualmente el fichero `authorized_keys` del usuario remoto.

ARCHIVOS DE CONFIGURACIÓN

La instalación del servidor crea y utiliza una serie de ficheros de configuración y claves que determinan cómo se comporta `sshd` y quién puede conectarse. En Debian 11, los más relevantes son:

- `/etc/ssh/sshd_config`
  Archivo de configuración principal del servidor SSH (`sshd`), donde se definen aspectos como el puerto de escucha, los métodos de autenticación permitidos o las directivas de acceso de los usuarios.
- `/etc/ssh/ssh_host_*_key`
  Conjunto de claves privadas de host utilizadas para identificar de forma única al servidor frente a los clientes.
- `/home/<usuario>/.ssh/authorized_keys`
  Fichero que contiene las claves públicas autorizadas para la conexión sin contraseña de un usuario concreto.

CLIENTE SSH

INSTALACIÓN

El cliente OpenSSH suele venir instalado por defecto en Debian 11. Si no estuviera disponible, se puede instalar el paquete `openssh-client`:

```
sudo apt update
sudo apt install openssh-client
```

PROGRAMAS DISPONIBLES

Una vez instalado el cliente OpenSSH, el sistema dispone de un conjunto de herramientas orientadas al uso desde la máquina cliente. Muchas de ellas ya se han mencionado al describir el servidor, pero desde el punto de vista del usuario que se conecta conviene destacar las siguientes:

- `ssh`: programa principal para establecer conexiones remotas interactivas o ejecutar comandos en otros equipos.
- `scp`: utilidad para copiar archivos y directorios de forma segura entre el cliente y el servidor.
- `sftp`: cliente interactivo de transferencia de archivos sobre SSH, pensado para subir, descargar y gestionar ficheros remotos.
- `ssh-copy-id`: herramienta que copia la clave pública del usuario al servidor, facilitando la configuración de la autenticación sin contraseña.
- Herramientas de apoyo a la gestión de claves: `ssh-add`, `ssh-keyscan`, `ssh-keygen`, `ssh-agent`.

ARCHIVOS DE CONFIGURACIÓN DEL CLIENTE

El comportamiento del cliente SSH también se controla mediante distintos ficheros de configuración y claves que se encuentran repartidos entre la configuración global del sistema y el directorio personal de cada usuario. Los más habituales son:

- `/etc/ssh/ssh_config`
  Fichero de configuración global del cliente SSH para todo el sistema, aplicable a todos los usuarios salvo que se sobreescriba con ajustes personales.
- `/home/<usuario>/.ssh/known_hosts`
  Lista de servidores que el usuario ya ha autenticado y que se consideran seguros (almacena las claves públicas de host).
- `/home/<usuario>/.ssh/config`
  Archivo de configuración específico del cliente SSH para un usuario concreto, en el que se pueden definir alias de hosts, puertos alternativos, usuarios por defecto, etc.
- `/home/<usuario>/.ssh/id_[tipo_clave]`
  Clave privada del usuario. La palabra *tipo_clave* puede ser `rsa`, `ecdsa`, `ed25519`, etc.
- `/home/<usuario>/.ssh/id_[tipo_clave].pub`
  Clave pública del usuario asociada a la clave privada anterior. Es la que se copia al servidor para la autenticación basada en claves.

CONEXIÓN Y DESCONEXIÓN

El uso más habitual del cliente SSH consiste en abrir una sesión remota en otro equipo. Para ello, se ejecuta el comando `ssh` indicando el usuario y la máquina a la que se desea conectar. Si no se indica explícitamente el

usuario remoto, se utilizará por defecto el mismo nombre de usuario con el que estamos conectados en el sistema local.

Una vez establecida la conexión, trabajaremos en la consola del equipo remoto como si estuviéramos sentados delante de él. Cuando terminemos, la sesión se cierra utilizando comandos como `exit` o `logout`, o bien cerrando directamente la terminal.

Conexión SSH

En el siguiente ejemplo nos conectaremos, desde un cliente Alpine Linux, al servidor ssh (Debian) con el usuario `sshuser1`. La ip del servidor es la 192.168.1.5.
En las distribuciones Alpine Linux, el cliente SSH no viene instalado por defecto, por lo que el primer paso es instalarlo.

```
Alpine:~# apk add openssh-client
(1/4) Installing openssh-keygen (10.2_p1-r0)
(2/4) Installing libedit (20251016.3.1-r0)
(3/4) Installing openssh-client-common (10.2_p1-r0)
(4/4) Installing openssh-client-default (10.2_p1-r0)
Executing busybox-1.37.0-r30.trigger
OK: 176.4 MiB in 99 packages
Alpine:~#
```

A continuación, se establece la conexión con el servidor SSH con el usuario `sshuser1`. Se supone que el servidor está configurado para aceptar conexiones SSH y que existe el usuario `sshuser1`.

```
Alpine:~# ssh sshuser1@192.168.1.5
The authenticity of host '192.168.1.5 (192.168.1.5)' can't be established.
ED25519 key fingerprint is SHA256:GV0L9lTQT+DNDPqXQTZgITPBEIu+u0ix7wh9RC42j×jQ
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.5' (ED25519) to the list of known hosts.
...
...
sshuser1@192.168.1.5's password:
Linux SSHSERVER 5.10.0-38-amd64 #1 SMP Debian 5.10.249-1 (2026-02-10) x86_64
...
...
sshuser1@SSHSERVER:~$
```

En este intercambio de mensajes puede verse que es la primera vez que el cliente se conecta al servidor `192.168.1.5`. El servidor pide al cliente que acepte su clave pública y esta se añade al fichero ~/.ssh/known_hosts del cliente.
Fijémosnos que una vez conectados ha cambiado el prompt de la terminal.

El cliente SSH guarda en el fichero ~/.ssh/known_hosts las claves públicas de los servidores a los que se conecta. Para un mismo servidor puede haber varias líneas: una por cada tipo de clave de host que el servidor tenga configurado (por ejemplo, ED25519, ECDSA o RSA). En la primera conexión, como el cliente todavía no conoce ninguna clave, muestra un aviso indicando que la autenticidad del host no puede establecerse y enseña la huella (fingerprint) de una de esas claves. Si aceptamos (yes), esa clave se añade a known_hosts. En las versiones actuales de OpenSSH, una vez aceptada la primera clave, el servidor puede enviar por el canal cifrado el resto de claves de host que tiene, y el cliente también las guarda automáticamente; por eso, aun habiéndonos conectado solo una vez, es normal ver varias claves para la misma IP o nombre en known_hosts.

Si queremos ver las claves de host de un servidor antes de conectarnos (o sin abrir una sesión interactiva), podemos usar el comando ssh-keyscan. Este comando pregunta al servidor qué claves de host tiene y las muestra todas, una por línea, sin modificar por sí mismo nuestro known_hosts. Es habitual que ssh-keyscan servidor muestre más claves de las que vemos en ~/.ssh/known_hosts: ssh-keyscan enseña todas las claves que el servidor anuncia, mientras que known_hosts solo contiene las que nuestro cliente ha ido aprendiendo y aceptando a lo largo de sus conexiones.

Clave pública del servidor

En este ejemplo comprobaremos que la clave pública del servidor que tengo en `~/.ssh/known_hosts` es la misma que se muestra al ejecutar `ssh-keyscan` para el servidor `192.168.1.5`.

```
Alpine:~# cat ~/.ssh/known_hosts
192.168.1.5 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPSp0Ned52etjvIkefzCnk6p4AhWlu0uZzY4fmBCb0RC
192.168.1.5 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmIzdhA6YNTYAAAAlbmIzdhA6YNTYAAABBB0EPAK85iPeMQSfsz22dGAW4532Bfcr1fjHdWgnYh3oRPKoPs5COi+fsFeDStNnWk7D+MxuQ1x9uSB6
Alpine:~#
```

A continuación prguntamos al servidor por su clave pública con el comando `ssh-keyscan`:

```
Alpine:~# ssh-keyscan 192.168.1.5
# 192.168.1.5:22 SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u5
# 192.168.1.5:22 SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u5
192.168.1.5 ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDKDQcqvXhV+4JQ5UPGTi24y8jVGnM9cI1acBCC8s/L0bIrp7pvMAQm5EwXpjp3WXh0QOs/UadKrSKBcy46+p7to/ix7YZ4vowzMW8FC
# 192.168.1.5:22 SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u5
192.168.1.5 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmIzdhAwNTYAAAAlbmIzdhAwNTYAAABBBDEPAK85iPeMQSfsz22dGAW4532Bfcr1fjHdWgnYh3oRPKoPs5COi+fsFeDStNnWk7D+MxuQ1x9uSB6Za
# 192.168.1.5:22 SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u5
192.168.1.5 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPSp0Ned52etjvIkefzCnk6p4AhWlu0uZzY4fmBCb0RC
# 192.168.1.5:22 SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u5
Alpine:~#
```

Podemos ver que las dos claves públicas que se muestran al ejecutar `ssh-keyscan` coinciden con las que tenemos en el fichero `~/.ssh/known_hosts`, lo que confirma que el servidor es el mismo al que nos hemos conectado anteriormente y que no ha habido ningún cambio en sus claves de host.

Para cerrar una sesión SSH, se puede usar el comando `exit` o `logout`, o simplemente cerrar la terminal.

Conexión SSH

Tras cerrar la sesión, nos volveremos a encontrar en la terminal del cliente, con el prompt habitual.

```
sshuser1@SSHSERVER:~$ exit
logout
Connection to 192.168.1.5 closed.
Alpine:~# ssh sshuser1@192.168.1.5
...
...
sshuser1@192.168.1.5's password:
Linux LSSHSERVER 5.10.0-38-amd64 #1 SMP Debian 5.10.249-1 (2026-02-10) x86_64
...
```

```
...
sshuser1@LSSHSERVER:~$
```

Ahora, al volver a conectarnos al servidor, el cliente SSH ya reconoce la clave pública del servidor (porque la guardó en `~/.ssh/known_hosts` en la conexión anterior) y no muestra el aviso de autenticidad del host. Sin embargo, como no hemos configurado aún la autenticación sin contraseña, el servidor nos pide la contraseña del usuario `sshuser1` para completar la conexión.

GENERACIÓN DE CLAVES PÚBLICAS Y PRIVADAS

Si queremos evitar el uso de contraseñas en cada conexión y aprovechar la autenticación basada en claves, en la máquina cliente se siguen tres pasos principales:

- generar un par de claves
- copiar la clave pública al servidor
- conectarse utilizando la clave privada

Generar un par de claves (pública y privada) con el comando `ssh-keygen`
El comando `ssh-keygen` se utiliza para crear un par de claves criptográficas (pública y privada) que permiten la autenticación sin contraseña en SSH. Su sintaxis básica es la siguiente:

```
ssh-keygen [opciones]
```

donde las opciones permiten configurar el tipo de clave, su tamaño, la frase secreta que la protege, el nombre del archivo donde se guardará la clave privada, etc. En las versiones actuales de OpenSSH, si se ejecuta sin opciones se generará por defecto una clave de tipo `ed25519` y se guardará en el directorio `~/.ssh/` con el nombre `id_ed25519` para la clave privada y `id_ed25519.pub` para la clave pública. Antiguamente se generaba por defecto una clave RSA de 2048 bits, pero en las versiones más recientes se ha cambiado a `ed25519` por ser un algoritmo más moderno y seguro.

La siguiente tabla resume las opciones más comunes que se pueden usar con `ssh-keygen` para personalizar la generación de claves:

Parámetro	Descripción
`-t`	Define el tipo de clave a generar (por ejemplo, `rsa`, `ecdsa`, `ed25519`).
`-b`	Indica el tamaño de la clave en bits (por ejemplo, 2048 para RSA).
`-f`	Especifica el nombre del archivo donde se guardará la clave privada (la clave pública se guardará con el mismo nombre pero con la extensión `.pub`).
`-N`	Permite establecer una frase secreta para proteger la clave privada. Si se indica `-N ""`, la clave privada no tendrá frase secreta y podrá usarse sin necesidad de introducir una contraseña. Sin embargo, esta práctica no es recomendable en entornos de producción o con información sensible, ya que si alguien obtiene acceso a la clave privada podría usarla sin restricciones. En general, es aconsejable proteger la clave privada con una frase secreta y utilizar un agente de claves (`ssh-agent`) para gestionar su uso de forma segura.
`-q`	Activa el modo silencioso, evitando la salida de mensajes informativos durante la generación de claves.

Parámetro	Descripción
`-Z`	Especifica el formato de la clave privada (por ejemplo, `PEM` o `RFC4716`).

Generación par de claves pública/privada

En el siguiente ejemplo se ejecuta `ssh-keygen` en Alpine Linux sin opciones adicionales. El programa genera por defecto un par de claves de tipo `ed25519`, pregunta por la ruta donde guardarlas y solicita una frase secreta para proteger la clave privada:

```
Alpine:~# ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase for '/root/.ssh/id_ed25519' (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:PsBVJBCjD38gsMrRAM+XVJQaWuzMpnfIkj0zudlbc root@Alpine
The key's randomart image is:
+--[ED25519 256]--+
| .o      o.      |
| o o     o o.    |
| o  .   . o o .  |
| .  o +  . ooo . |
|  .  o S + . .   |
|       + o  .    |
|      + oo= o.   |
|       +.ooo.E   |
|         ..oo    |
+----[SHA256]-----+
Alpine:~#
```

Como se puede ver no hace falta indicar ninguna opción, ya que los valores por defecto son los que se desean en este caso.

Copiar la clave pública al servidor

Para que el servidor SSH pueda reconocer nuestra clave pública y permitir la autenticación sin contraseña, es necesario copiar esa clave al servidor y añadirla al fichero `authorized_keys` del usuario remoto. Para ello partimos de la clave pública generada en el paso anterior (por ejemplo, `~/.ssh/id_ed25519.pub`) y la transferimos al servidor. Existen dos formas habituales de hacerlo:

- mediante `ssh-copy-id`, que automatiza todo el proceso;
- o de forma manual, copiando la clave con `scp` y actualizando después el fichero `authorized_keys` en el servidor.

Copia de claves al servidor (método automático)

En este ejemplo usamos `ssh-copy-id` desde Alpine Linux para copiar la clave pública `id_ed25519.pub` del usuario root al servidor `192.168.1.5`, de forma que el usuario remoto `sshuser1` pueda autenticarse mediante clave en lugar de contraseña.

```
Alpine:~# ssh-copy-id sshuser1@192.168.1.5
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
expr: warning: '^ERROR: ' using ^^ as the first character
of a basic regular expression is not portable; it is ignored
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
*** WARNING: connection is not using a post-quantum key exchange algorithm.
*** This session may be vulnerable to "store now, decrypt later" attacks.
*** The server may need to be upgraded. See https://openssh.com/pq.html
sshuser1@192.168.1.5's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'sshuser1@192.168.1.5'"
and check to make sure that only the key(s) you wanted were added.

Alpine:~#
```

El propio `ssh-copy-id` indica cuántas claves se han añadido al servidor (`Number of key(s) added: 1`) y recuerda que, a partir de ese momento, deberíamos poder entrar con `ssh sshuser1@192.168.1.5` sin que se nos pida la contraseña, ya que la autenticación se hará con la clave pública instalada.

Si por algún motivo `ssh-copy-id` no estuviera disponible, podemos recurrir a `scp` y realizar el proceso de manera manual.

Copia de claves al servidor (método manual)

En este caso no usamos `ssh-copy-id`. Copiamos manualmente la clave pública desde el cliente Alpine (usuario root) hasta el servidor `192.168.1.5` y la añadimos al fichero `authorized_keys` del usuario `sshuser1`.

Solución

Los siguientes pasos detallan el proceso:

- Copiar la clave pública generada en el cliente (por ejemplo `id_ed25519.pub`) al `home` del usuario remoto `sshuser1`:

  ```
  Alpine:~# scp ~/.ssh/id_ed25519.pub sshuser1@192.168.1.5:/home/sshuser1/id_ed25519.pub
  ```

- Conectarse al servidor `192.168.1.5` con el usuario `sshuser1` usando contraseña:

  ```
  Alpine:~# ssh sshuser1@192.168.1.5
  ```

- Crear el directorio `.ssh` (si no existe) y añadir la clave pública al fichero `authorized_keys`:

  ```
  sshuser1@192.168.1.5:~$ mkdir -p ~/.ssh
  sshuser1@192.168.1.5:~$ cat id_ed25519.pub >> ~/.ssh/authorized_keys
  ```

- Eliminar el fichero temporal con la clave pública y cerrar la sesión en el servidor:

  ```
  sshuser1@192.168.1.5:~$ rm id_ed25519.pub
  sshuser1@192.168.1.5:~$ exit
  ```

Realizar la conexión utilizando la clave privada

Una vez que el servidor conoce nuestra clave pública (porque la hemos instalado con `ssh-copy-id` o de forma manual), podemos conectarnos usando la clave privada en lugar de una contraseña. Si utilizamos la ruta y el

nombre por defecto (`~/.ssh/id_ed25519`, `~/.ssh/id_rsa`, etc.), normalmente basta con ejecutar `ssh usuario@servidor` y el propio cliente SSH buscará la clave adecuada.

En caso de que queramos utilizar un fichero de clave distinto al habitual, podemos indicarlo de forma explícita con la opción `-i` del comando `ssh`.

Conexión con clave privada

Conectarse desde el cliente Alpine al servidor `192.168.1.5` con el usuario `sshuser1`, utilizando la clave privada `id_ed25519` que el cliente encuentra en `~/.ssh`.

```
Alpine:~# ssh sshuser1@192.168.1.5
*** WARNING: connection is not using a post-quantum key exchange algorithm.
*** This session may be vulnerable to "store now, decrypt later" attacks.
*** The server may need to be upgraded. See https://openssh.com/pq.html
Linux L01S00-SSH-Linux 5.10.0-38-amd64 #1 SMP Debian 5.10.249-1 (2026-02-10) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb 18 00:10:30 2026 from 192.168.1.198
sshuser1@L01S00-SSH-Linux:~$
```

En este ejemplo la clave privada **no tiene frase de paso (passphrase)**, por eso el cliente puede usarla directamente y no nos pide ninguna contraseña local.

USO DEL AGENTE DE CLAVES

Hasta ahora hemos utilizado una clave privada **sin frase de paso**. En este caso el agente de claves aporta poca ventaja: el cliente `ssh` puede leer la clave directamente del disco y conectarse sin pedir nada más.

El agente de claves `ssh-agent` se vuelve realmente útil cuando **protegemos la clave privada con una passphrase**. Entonces la situación es:

- Sin agente: cada vez que hacemos `ssh`, `scp`, `git pull` sobre ese servidor, se nos pide la passphrase de la clave.
- Con agente: introducimos la passphrase **una sola vez**, el agente guarda la clave descifrada en memoria y las siguientes conexiones ya no la piden.

Un flujo didáctico completo sería el siguiente.

1. Añadir una passphrase a una clave ya existente
Suponemos que ya tenemos creada la clave `~/.ssh/id_ed25519` sin passphrase. Podemos añadirle una frase de paso a posteriori con `ssh-keygen -p`:

Añadir passphrase a una clave existente

```
Alpine:# ssh-keygen -p -f ~/.ssh/id_ed25519
Key has comment 'root@Alpine'
```

```
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
Alpine:#
```

En este ejemplo se ha añadido una passphrase a la clave `id_ed25519` que antes no tenía. A partir de ahora, cada vez que el cliente SSH intente usar esa clave para autenticarse, nos pedirá la passphrase para descifrarla.

2. Ver el efecto: ahora cada conexión pide la passphrase

Si intentamos conectarnos de nuevo **sin agente**, el cliente nos pide la passphrase local antes de abrir la sesión SSH:

Conexión pidiendo la passphrase

```
Alpine:~# ssh sshuser1@192.168.1.5
*** WARNING: connection is not using a post-quantum key exchange algorithm.
*** This session may be vulnerable to "store now, decrypt later" attacks.
*** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key '/root/.ssh/id_ed25519':
Linux L01S00-SSH-Linux 5.10.0-38-amd64 #1 SMP Debian 5.10.249-1 (2026-02-10) x86_64
...
...
sshuser1@L01S00-SSH-Linux:~$
```

Ahora, antes de establecer la conexión SSH, el cliente nos pide la passphrase de la clave privada `id_ed25519` para poder usarla en la autenticación.

3. Utilizar el agente para no repetir la passphrase en cada conexión

Para evitar tener que escribir la passphrase en cada comando, arrancamos un agente y le añadimos la clave una sola vez. A partir de ese momento, el agente se encargará de gestionar la clave en memoria y las conexiones SSH podrán usarla sin volver a pedir la passphrase. El agente de claves se puede iniciar con `ssh-agent` y luego se añaden las claves con `ssh-add`.

Uso básico de ssh-agent

En este ejemplo se muestra cómo iniciar el agente de claves `ssh-agent`, añadir la clave privada `id_ed25519` al agente y luego conectarse al servidor sin necesidad de introducir la passphrase en cada conexión.

1. Iniciar el agente en la sesión actual del cliente Alpine:

```
Alpine:~# eval "$(ssh-agent -s)"
Agent pid 2711
```

El comando `eval "$(ssh-agent -s)"` arranca el agente de claves y configura las variables de entorno necesarias para que el cliente SSH pueda comunicarse con él. La salida muestra el PID del proceso del agente.

2. Añadir la clave privada `id_ed25519` al agente (ahora sí se pedirá la passphrase **una sola vez**):

```
Alpine:~# ssh-add ~/.ssh/id_ed25519
Enter passphrase for /root/.ssh/id_ed25519:
Identity added: /root/.ssh/id_ed25519 (root@Alpine)
```

3. Conectarse al servidor `192.168.1.5` como `sshuser1` sin necesidad de indicar la clave privada ni volver a escribir la passphrase:

```
Alpine:~# ssh sshuser1@192.168.1.5
*** WARNING: connection is not using a post-quantum key exchange algorithm.
*** This session may be vulnerable to "store now, decrypt later" attacks.
*** The server may need to be upgraded. See https://openssh.com/pq.html
Linux L01S00-SSH-Linux 5.10.0-38-amd64 #1 SMP Debian 5.10.249-1 (2026-02-10) x86_64
...
...
sshuser1@L01S00-SSH-Linux:~$
```

Ahora el cliente SSH se conecta al servidor utilizando la clave privada gestionada por el agente, sin pedir la passphrase en esta ni en las siguientes conexiones mientras el agente siga activo.

Mientras el agente siga activo, cualquier conexión que use esa clave se podrá realizar sin volver a introducir la frase de paso.

Además, el agente ofrece algunas utilidades adicionales:

- Para ver las claves públicas correspondientes a las claves cargadas en el agente se puede utilizar el comando `ssh-add -L`. Este comando muestra la lista de claves públicas que el agente tiene disponibles, una por línea, con su tipo, su huella y su comentario (normalmente el nombre del usuario y el host).

Ver claves públicas cargadas en el agente

En este ejemplo se muestra la salida del comando `ssh-add -L` después de haber añadido la clave privada `id_ed25519` al agente. La salida muestra la clave pública correspondiente a esa clave privada, con su tipo (ssh-ed25519), su huella (fingerprint) y su comentario (root@Alpine).

```
Alpine:~# ssh-add -L
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGKhSdAPH0EIIuydqa966YFaBCuRFoTo/mRWSD000Jgyt root@Alpine
Alpine:~#
```

Si queremos ver solo las claves privadas cargadas en el agente, se emplea el comando `ssh-add -l`. Este comando muestra una lista de las claves privadas que el agente tiene en memoria, indicando su tipo, su huella y su comentario. La diferencia con `ssh-add -L` es que este último muestra las claves públicas, mientras que `ssh-add -l` muestra las claves privadas (aunque no revela su contenido, solo su huella).

Ver claves privadas cargadas en el agente

En este ejemplo se muestra la salida del comando `ssh-add -l` después de haber añadido la clave privada `id_ed25519` al agente. La salida indica que hay una clave privada de tipo ED25519 cargada, con su huella (SHA256:PsB/UBCjD3BgsMrARM+XV/3QalWuzMpnfIkjozudlbc) y su comentario (root@Alpine).
La salida de `ssh-add -l` no muestra el contenido de la clave privada, solo su huella y su comentario, para

evitar revelar información sensible. Sin embargo, nos permite confirmar que la clave privada está cargada en el agente y lista para ser utilizada en las conexiones SSH.

```
Alpine:~# ssh-add -l
256 SHA256:PsB/UBCjD3BgsMrARM+XV/3QalWuzMpnfIkjozudlbc root@Alpine (ED25519)
Alpine:~#
```

- Para detener el agente y borrar de memoria todas las claves cargadas se emplea el comando `ssh-agent -k`. Este comando envía una señal al proceso del agente para que se cierre y borre las claves de memoria. Además, muestra un mensaje indicando que el agente ha sido detenido y las variables de entorno relacionadas han sido desconfiguradas.

Detener el agente de claves

En este ejemplo se muestra cómo detener el agente de claves `ssh-agent` con el comando `ssh-agent -k`. La salida indica que las variables de entorno `SSH_AUTH_SOCK` y `SSH_AGENT_PID` han sido desconfiguradas (unset) y que el proceso del agente con PID 2711 ha sido terminado (killed).
Después de ejecutar este comando, el agente ya no estará activo y las claves que tenía cargadas dejarán de estar disponibles para las conexiones SSH.

```
Alpine:~# ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 2711 killed;
Alpine:~#
```

Al intentar conectarnos de nuevo al servidor después de haber detenido el agente, el cliente SSH ya no podrá usar la clave privada que teníamos cargada en el agente, por lo que volverá a pedir la passphrase para poder usar esa clave.

```
Alpine:~# ssh sshuser1@192.168.1.5
*** WARNING: connection is not using a post-quantum key exchange algorithm.
*** This session may be vulnerable to "store now, decrypt later" attacks.
*** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key '/root/.ssh/id_ed25519':
```