Servicio SSH

El servicio SSH (Secure Shell) nos permite poder conectar a una máquina de manera remota si existe conectividad directa entre los 2 equipos, de manera que podemos realizar operaciones sobre un sistema de manera remota, sin necesidad de desplazarnos físicamente hasta su ubicación.

Funcionamiento

El servicio SSH se compone de diferentes capas, que podemos definir de la siguiente forma:

- Capa de transporte
- Capa de autenticación
- Capa de conexión

Capa de transporte

La función principal de la capa de transporte es la de ofrecer un entorno fiable y seguro entre cliente y servidor para poder realizar el proceso de autenticación. En un primer lugar, el cliente contacta con el servidor, el que responde enviando una huella (fingerprint) que lo identifica de manera única. El cliente, si es la primera vez que conecta con él, deberá guardar esta información y vincularla a ese servidor (IP) para futuras conexiones. De esta manera se crea una primera comprobación de la veracidad del servidor (cada vez que el cliente se conecte va a recibir esa huella, por lo que ha de coincidir con la que tiene guardada).

Capa de autenticación

Una vez se ha establecido una conexión entre cliente y servidor se realiza el proceso de autenticación. En este proceso el cliente se identifica con unas credenciales que le permitan acceder al servidor. Estas credenciales pueden ser un usuario y contraseña válidos para iniciar sesión o puede ser una clave SSH que el servidor reconozca. El sistema de autenticación por medio de claves funciona de la siguiente manera:

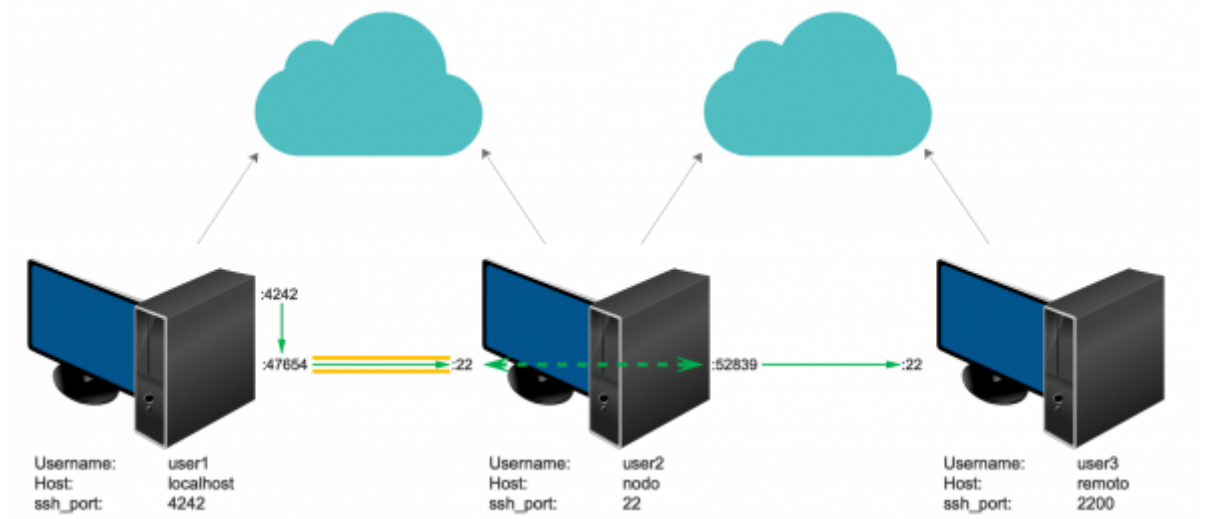
Capa de conexión

Una vez se ha establecido la conexión y el cliente se ha autenticado, cliente y servidor establecen un túnel a través del cual se envían, de manera cifrada, los paquetes entre ambos. SSH necesita que cliente y servidor tengan conectividad entre ellos para poder establecer una conexión. Sin embargo,

es posible realizar conexiones entre máquinas que no tienen visibilidad directa a través de diferentes tipos de túneles.

Túnel directo

SSH permite la posibilidad de enlazar un puerto de nuestro equipo *local* con el de un equipo *remoto* con el que no tenemos visibilidad. Para esto es necesario que exista un *nodo* visible para los dos (*local* y *remoto*). Éste recibe desde el equipo *local* la solicitud de enlazar un puerto de su máquina con otro del equipo *remoto*. En el siguiente diagrama se representa el proceso a seguir:



Para establecer el túnel debemos utilizar el comando ssh de la siguiente manera en nuestro equipo:

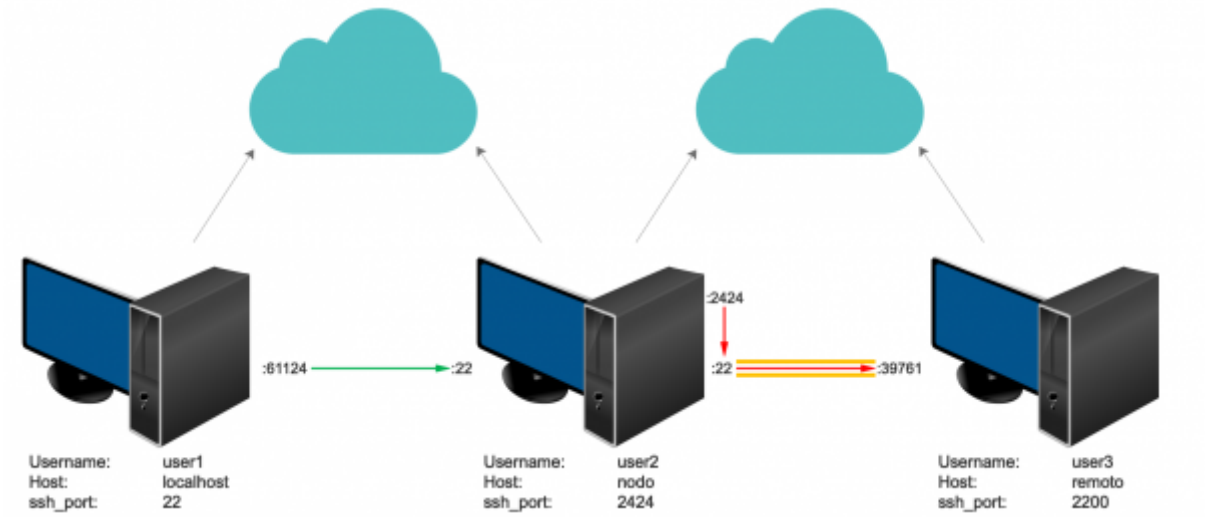
```
user1@localhost:~$ ssh -fN -L 4242:remoto:2200 user2@nodo
```

Una vez hecho esto, para conectar con el equipo *remoto* basta con hacer:

```
user1@localhost:~$ ssh -p 4242 user3@localhost
```

Túnel reverso

Cuando no tenemos la posibilidad de conocer la identidad del equipo *remoto* al que nos queremos conectar, tenemos la posibilidad de establecer un túnel reverso. En esta situación es el equipo *nodo* quien recibe, desde el equipo *remoto* la solicitud de establecimiento del túnel. Esto genera un vínculo entre 2 puertos entre los equipos *remoto* y *nodo*, de manera que desde nuestro equipo local nos podemos conectar al equipo *nodo*, y una vez allí establecer un ssh contra dicho puerto. En el siguiente diagrama se representa el proceso a seguir:



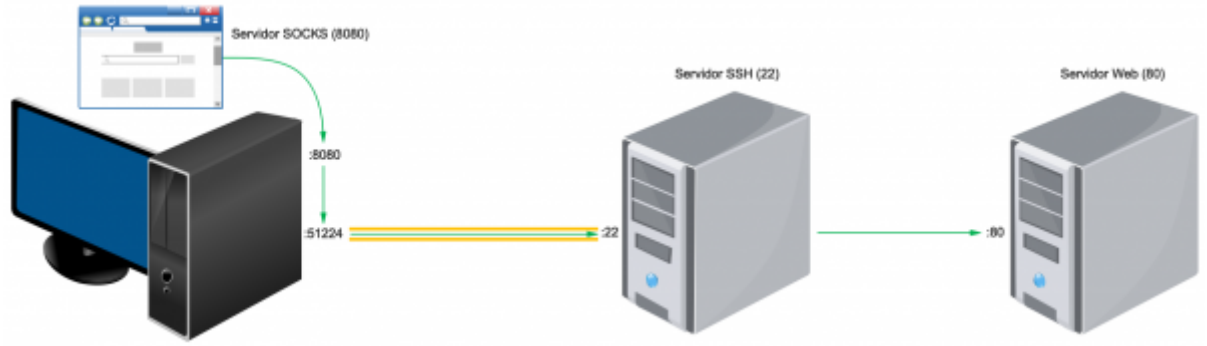
Para establecer el túnel debemos utilizar el comando ssh de la siguiente manera en el equipo *remoto*:

```
user3@remoto:~$ ssh -fN -R 2244:remoto:2200 user2@nodo
```

Ahora, para conectarnos al equipo *remoto* desde nuestro equipo haremos: </cli> user1@localhost:~$ ssh user2@nodo user2@nodo:~$ ssh -p 2244 user3@127.0.0.1 </cli>

Dynamic Port Forwarding

Otra forma de poder encapsular el tráfico a través de un túnel SSH es la opción del reenvío de tráfico por un puerto hacía un túnel SSH. Esta técnica permite vincular un puerto de la máquina local con una conexión SSH contra un servidor remoto



Para ello debemos utilizar el comando ssh de la siguiente manera:

```
user1@localhost:~$ ssh -D 8080 user2@server
```

Instalación y configuración

- openssh-server
- openssh-client

Comprobación del servicio

Una vez hemos configurado el servicio de manera correcta podemos iniciarlo y comprobar su funcionamiento. El servicio SSH está integrado dentro de systemd, por lo que podemos gestionarlo a través del comando systemctl

```
# systemctl start sshd.service
● ssh.service - OpenBSD Secure Shell server
 Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:
enabled)
 Active: active (running) since Sun 2020-04-10 18:49:19 CEST; 3h 20min ago
Process: 6133 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 6135 (sshd)
  Tasks: 1 (limit: 4915)
  CGroup: /system.slice/ssh.service
          └─6135 /usr/sbin/sshd -D
Apr 10 22:09:48 free sshd[12797]: Failed password for invalid user vnc from
192.241.235.11 port 54732 ssh2
Apr 10 22:09:48 free sshd[12797]: Received disconnect from 192.241.235.11
port 54732:11: Bye Bye [preauth]
Apr 10 22:09:49 free sshd[12807]: Invalid user liang from 91.231.113.113
port 48662
Apr 10 22:09:49 free sshd[12807]: input_userauth_request: invalid user liang
[preauth]
Apr 10 22:09:49 free sshd[12807]: pam_unix(sshd:auth): check pass; user
unknown
Apr 10 22:09:49 free sshd[12807]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser=
Apr 10 22:09:50 free sshd[12795]: Failed password for invalid user yumi from
118.25.36.79 port 57040 ssh2
Apr 10 22:09:50 free sshd[12795]: Received disconnect from 118.25.36.79 port
57040:11: Bye Bye [preauth]
Apr 10 22:09:51 free sshd[12807]: Failed password for invalid user liang
from 91.231.113.113 port 48662 ssh2
Apr 10 22:09:51 free sshd[12807]: Received disconnect from 91.231.113.113
port 48662:11: Bye Bye [preauth]
```

Verificación del servicio

Una vez comprobamos que el servicio está activo podemos consultar si el puerto configurado está en modo escucha a través del comando ss

```
# ss -tlnp | grep ssh
LISTEN  0      128        *:22        *:*
users:(("sshd",pid=28892,fd=5))
LISTEN  0      128       :::22       :::*
```

```
users:(("sshd",pid=28892,fd=6))
```

También podemos hacer uso de los logs del sistema para verificar si el servidor está asignando direcciones de manera correcta.

```
# grep sshd /var/log/auth.log
Apr 10 22:13:05 free sshd[12928]: Server listening on 0.0.0.0 port 2222
Apr 10 22:13:05 free sshd[12928]: Server listening on :: port 2222
```