

Configuración básica SSH para servidores

Antes de empezar	2
¿Qué es SSH?	2
¿Qué realizaremos como ejemplo práctico?	2
Configuración inicial y estructura	2
Asignación de IP	3
En Ubuntu Server:	3
En Rocky Server:	4
Instalación SSHD y activación automática	5
En ubuntu:	5
En Rocky:	6
Acceso SSH por defecto	6
Cuestiones de seguridad	7
Puerto	7
Cuestiones exclusivas Rocky Server	8
SE linux	8
• Firewall	9
• Usuario root	9
Fingerprint ssh	9

Antes de empezar

En este documento se resumirá mediante un ejemplo práctico las bases de configuración SSH para computadores, que normalmente se aplican a servidores linux.

No utilizaremos un dispositivo real para ello, si no un hipervisor (VirtualBox), pero todo lo explicado aquí es extrapolable a una situación real sustituyendo los pasos realizados en el hipervisor por la correspondiente configuración en los equipos físicos.

Esta información proviene de lo aprendido en las prácticas de la asignatura Ingeniería de Servidores, impartida en el grado en Ingeniería informática en la Universidad de Granada.

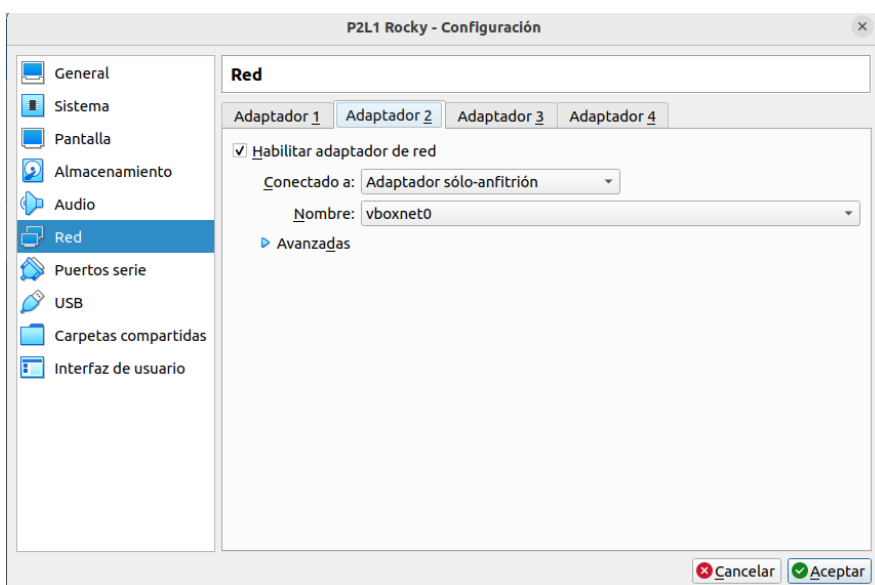
¿Qué es SSH?

SSH consiste de una forma muy básica, en la conexión remota entre servidores, principalmente nos permite controlar una terminal de un dispositivo desde otro dispositivo.

¿Qué realizaremos como ejemplo práctico?

Vamos a conectar dos servidores, uno en Rocky Server y otro en Ubuntu Server mediante conexión SSH, configurando los aspectos de seguridad básicos e imprescindibles para realizar la conexión de manera segura.

Configuración inicial y estructura



Partiremos con dos máquinas virtuales una en Rocky Server y otra en Ubuntu Server con sus respectivas instalaciones por defecto.

Respecto a la estructura de su conectividad, contarán con 2 interfaces de red. La NAT convencional en el adaptador 1, y una segunda conexión sólo anfitrión que añadiremos nosotros manualmente en el adaptador 2 mediante el hipervisor.

Nota: No olvidar confirmar que la casilla “cable conectado” está rellena en el apartado avanzadas.

Mediante el comando **ip a** podemos observar la actual estructura de redes

```
albertopm@localhost ~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:be:c2:88 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 86292sec preferred_lft 86292sec
    inet6 fe80::a00:27ff:febe:c288/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ad:24:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 492sec preferred_lft 492sec
    inet6 fe80::361a:7f5a:b00c:e97c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

albertopm@buntuerver:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:66:e4:07 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 86356sec preferred_lft 86356sec
    inet6 fe80::a00:27ff:fe66:e407/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:a7:e8:17 brd ff:ff:ff:ff:ff:ff
albertopm@buntuerver:~$
```

En esta imagen podemos ver las dos interfaces definidas en ambos servidores.

- Loopback: Esta interfaz siempre está.
- enp0s3: es la interfaz NAT del adaptador de red 1.
- enp0s8: Es la interfaz sólo anfitrión que añadimos en el adaptador de red 2.

De esta forma tenemos dos conexiones, una convencional, hacia internet, y otra como red interna, entre los dos computadores.

Asignación de IP

Como podemos ver en el comando **ip a**, la interfaz nos ha dado, en el caso de Rocky una dirección IP de forma automática que nosotros no queremos y en el caso de Ubuntu no ha especificado IP en las interfaces que hemos añadido, de modo que les asignaremos a cada uno una IP estática.

Para hacer las modificaciones puedes utilizar el editor de texto que quieras. En nuestro caso usaremos nano en ubuntu y vi en Rocky.

En Ubuntu Server:

Esto se hace desde el archivo de configuración que podemos encontrar en `/etc/netplan/00-installer-config.yaml` (después del netplan puedes hacer tabulador, es el único archivo en ese fichero).

De modo que para asignar la IP estática editaremos el fichero. Destacar que como es un .yaml, se rige por tabulaciones, por lo que estas son importantes.

Añadiremos la nueva interfaz, haremos que la ip sea estática (`dhcp4: false`) y le asignaremos la IP deseada (`addresses: [IP/máscara]`). El fichero quedaría tal que así:

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: false
      addresses: [192.168.56.105/24]
  version: 2
```

Importante introducir después el comando `sudo netplan apply` para que se realicen los cambios.

En Rocky Server:

En el caso de rocky no tiene un solo fichero para todas las interfaces, si no uno por interfaz. Podemos acceder a este mediante la siguiente ruta:
`/etc/sysconfig/network-scripts/ifcfg-enp0s8`

Aquí introduciremos la configuración que deseamos, entre ello lo principal que es la IP y la máscara.

```
TYPE=Ethernet
BOOTPROTO=none
NAME=enp0s8
DEVICE=enp0s8
ONBOOT=yes
IPADDR=192.168.56.110
NETMASK=255.255.255.0
```

Ahora podemos guardar el fichero y reiniciar.

```
rocky [Corriendo] - Oracle VM VirtualBox
Rocky Linux 9.0 (Blue Onyx)
Kernel 5.14.0-70.13.1.el9_0.x86_64 on an x86_64

localhost login: albertopm
Password:
Last login: Tue Nov  8 13:21:51 on tty1
[albertopm@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp8s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:be:c2:88 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic
        valid_lft 86391sec preferred_lft 86391sec
    inet6 fe80::a00:27ff:febe:c288/64 scope link noprefix
        valid_lft forever preferred_lft forever
3: enp8s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:ad:24:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.110/24 brd 192.168.56.255 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fead:2462/64 scope link
        valid_lft forever preferred_lft forever
[albertopm@localhost ~]$

albertopm@ubuntuserver:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:2a:08:33 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic
        valid_lft 85591sec preferred_lft 85591sec
    inet6 fe80::a00:27ff:fe2a:8333/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    link/ether 08:00:27:c0:31:2a brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.105/24 brd 192.168.56.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec0:312a/64 scope link
        valid_lft forever preferred_lft forever
albertopm@ubuntuserver:~$
```

De este modo tendríamos nuestras dos ips asignadas, y nuestras máquinas estarían conectadas mediante una red interna.

Podemos comprobar el correcto funcionamiento de la red lanzando pings mutuos a cada una de las IPs:

```
[albertopm@localhost ~]$ ping 192.168.56.105
PING 192.168.56.105 (192.168.56.105) 56(84) bytes of data.
 64 bytes from 192.168.56.105: icmp_seq=1 ttl=64 time=2.27 ms
 64 bytes from 192.168.56.105: icmp_seq=2 ttl=64 time=1.87 ms
 64 bytes from 192.168.56.105: icmp_seq=3 ttl=64 time=1.85 ms
 64 bytes from 192.168.56.105: icmp_seq=4 ttl=64 time=1.84 ms
 64 bytes from 192.168.56.105: icmp_seq=5 ttl=64 time=0.945 ms
 64 bytes from 192.168.56.105: icmp_seq=6 ttl=64 time=0.396 ms
 64 bytes from 192.168.56.105: icmp_seq=7 ttl=64 time=1.83 ms
 64 bytes from 192.168.56.105: icmp_seq=8 ttl=64 time=0.885 ms
--- 192.168.56.105 ping statistics ---
 8 packets transmitted, 8 received, 0% packet loss, time 7825ms
rtt min/avg/max/mdev = 0.396/1.884/2.267/0.492 ms

albertopm@ubuntuserver:~$ ping 192.168.56.110
PING 192.168.56.110 (192.168.56.110) 56(84) bytes of data.
 64 bytes from 192.168.56.110: icmp_seq=1 ttl=64 time=1.01 ms
 64 bytes from 192.168.56.110: icmp_seq=2 ttl=64 time=0.955 ms
 64 bytes from 192.168.56.110: icmp_seq=3 ttl=64 time=1.04 ms
 64 bytes from 192.168.56.110: icmp_seq=4 ttl=64 time=0.962 ms
--- 192.168.56.110 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.955/0.990/1.037/0.033 ms
albertopm@ubuntuserver:~$
```

Instalación SSHD y activación automática

En ubuntu:

el SSH cliente viene instalado por defecto en ubuntu server, por lo que procederemos a instalar el SSH servidor:

```
sudo apt update
```

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

Ahora si tabulamos tras escribir ssh, podemos ver cómo aparece el demonio sshd, esto quiere decir que se ha instalado correctamente. Además, el servicio se ha iniciado de forma

automática. Para que el servicio se inicie siempre de forma automática al arrancar el servidor:

```
sudo systemctl sshd start -permanent
```

En Rocky:

En Rocky Server ya está instalado sshd y su servicio está activo por defecto (algo muy a tener en cuenta al hacer un servidor en Rocky server).

Podemos comprobar el servicio en ambos servidores mediante el comando

```
sudo systemctl status sshd
```

```
albertopm@ubuntuserver:~$ sudo systemctl status sshd
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-11-08 12:02:29 UTC; 47min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 684 (sshd)
    Tasks: 1 (limit: 4563)
   Memory: 4.4M
      CPU: 29ms
   CGroup: /system.slice/ssh.service
           └─684 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

nov 08 12:02:29 ubuntuserver systemd[1]: Starting OpenBSD Secure Shell server...
nov 08 12:02:29 ubuntuserver sshd[684]: Server listening on 0.0.0.0 port 22.
nov 08 12:02:29 ubuntuserver sshd[684]: Server listening on :: port 22.
nov 08 12:02:29 ubuntuserver systemd[1]: Started OpenBSD Secure Shell server.
```

Acceso SSH por defecto

Podemos comprobar el correcto funcionamiento del servicio utilizándolo entre los servidores. Para ello:

```
ssh <usuario>@<ip>
```

siendo usuario el usuario de la otra máquina al que quiero acceder, e ip su correspondiente IP.

Ejemplo de conexión:

```
[albertopm@localhost ~]$ ssh albertopm@192.168.56.105
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ED25519 key fingerprint is SHA256:zYqkj1c6xFK4aLsRAoowgBpHABFkqmbphUZgrUrZTJI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.105' (ED25519) to the list of known hosts.
albertopm@192.168.56.105's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

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

System information as of mar 08 nov 2022 12:53:56 UTC

System load:  0.0               Processes:            104
Usage of /:   36.3% of 8.02GB   Users logged in:     1
Memory usage: 7%               IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%               IPv4 address for enp0s8: 192.168.56.105

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

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

Last login: Tue Nov  8 12:02:39 2022
albertopm@ubuntuserver:~$
```

En este caso nos hemos conectado a la terminal del servidor ubuntu server, de forma remota, desde el terminal de Rocky server.

Cuestiones de seguridad

Esta configuración básica presenta muchos inconvenientes de seguridad y hay ciertas modificaciones mínimas que son casi obligatorias para dotar al sistema de una mínima capa de seguridad, teniendo en cuenta lo peligroso que puede ser una conexión ssh no autorizada.

Puerto

SSH utiliza por defecto el puerto 22, cambiar el puerto que es necesario utilizar para la conexión es una buena medida de seguridad, pues así no es evidente el puerto que estás usando.

Esto se hace desde el archivo de configuración `/etc/ssh/sshd_config`. Para asignar un puerto descomentamos la línea `#Port 22`, y modificamos el 22 por el puerto que vayamos a usar. Es recomendable utilizar un puerto alto ya que estos suelen estar libres.

Si probamos la conexión de Rocky server a Ubuntu Server, esta funciona correctamente, sin embargo de Ubuntu server a Rocky Server no. Podría ser que el sistema operativo que estés utilizando tenga sus propias cuestiones exclusivas como Rocky, si no funciona correctamente el ssh, deberías revisarlas.

Cuestiones exclusivas Rocky Server

- SE linux

Rocky Server tiene una capa extra de seguridad llamada SE Linux que nos está impidiendo realizar la conexión ssh. Para operar con Se Linux necesitamos el binario `semanage`, así que buscamos que paquete contiene:

```
dnf provides semanage
```

Y encontramos que el paquete `policycoreutils-python-utils` lo contiene, de modo que lo instalamos

```
sudo dnf install policycoreutils-python-utils
```

Con `sudo semanage port -l` obtenemos una lista de los puertos asignados, de modo que buscamos los asignados a ssh con un `grep`:

```
sudo semanage port -l | grep "ssh"
```

Y vemos como efectivamente solo está asignado el puerto 22. Para añadir el puerto que nosotros queremos asignar:

```
sudo semanage port -a -t ssh_port_t -p tcp 22022
```

-a: añadir

-t <nombre puerto>: el puerto el concreto

-p <modelo>: puerto

- Firewall

A diferencia de Ubuntu Server, que tiene su firewall desactivado por defecto, Rocky Linux lo tiene activado.

Para listar las reglas firewall:

```
sudo firewall-cmd --list-all
```

Y como podemos observar, nuestro puerto 22022 no es uno de los permitidos, así que lo añadimos:

```
sudo firewall-cmd --add-port=22022/tcp --permanent
```

Si queremos un extra de seguridad respecto al servidor en Ubuntu Server, podríamos activar el firewall con el UFW y añadirle el puerto 22022.

```
sudo ufw enable
```

```
sudo ufw allow 22022
```

Usuario root

La posibilidad del acceso al usuario root es también un claro punto de vulnerabilidad, pues una persona no autorizada al acceso, con algunos conocimientos podría saber de la existencia de este usuario en nuestro dispositivo. Además es el superusuario lo cual puede ser peligroso, por lo que desactivaremos la posibilidad de acceder al usuario root.

Para ello en el archivo `/etc/ssh/sshd_config` descomentamos la opción `PermitRootLogin`, y le pondremos `no`.

Nota: Recordar que es importante restaurar el servicio en los servidores ssh para que se realicen los cambios (`sudo systemctl restart sshd`).

Fingerprint ssh

Las contraseñas, en general, siguen siendo un punto vulnerable, pues estas pueden ser descubiertas de muchas formas. Para ello utilizamos las fingerprints.

Se trata de un método de acceso por autorización, es decir, sin contraseña. Lo que sucede es que el equipo 1 al intentar conectarse al equipo dos, cifra un paquete con la llave privada y lo envía. El equipo 2 puede descifrar este paquete ya que posee la llave pública del Equipo 1, y por tanto se autoriza el acceso del equipo 1 al equipo 2 sin necesidad de introducir la contraseña.

ssh utiliza un cifrado simétrico, mediante la fingerprint estableceremos un cifrado asimétrico, teniendo una llave privada (que tiene siempre la llave cliente) y una llave pública, que copiaremos en el servidor que queremos que pueda conectarse.

Para generar estos dos archivos:

ssh_keygen

- Dejamos el archivo por defecto.
- No le ponemos contraseña (aunque podría ser una capa extra de seguridad, si se nos extraviase la llave podría ser muy peligroso ya que podría accederse al servidor).

Podemos observar como se ha creado tanto la llave pública como la llave privada con ls -la .ssh/

Para enviar la llave tenemos un script predefinido que lo hace por nosotros:

```
ssh-copy-id -i .ssh/id_rsa.pub -p <puerto> <usuario>@<ip>
```

```
ssh-copy-id -i .ssh/id_rsa.pub -p 22022 albertopm@192.168.56.105
```

Tras esto podemos deshabilitar el acceso con contraseña para que así solo pueda acceder por ssh mediante la fingerprint.

En el fichero etc/ssh/sshd_config ponemos password authentication en no.

Finalmente podemos hacer conexiones sin introducir la contraseña de un equipo a otro.

Ejemplo de conexión automática:

```
ssh albertopm@192.168.56.105 -p 22022
```