Actividad final Módulo Sistemas Informáticos

Curso 2021/2022

INTRODUCCIÓN

Una de las formas más sencillas de realizar copias de seguridad consiste en comprimir los directorios que se quieran respaldar y enviar el fichero comprimido resultante a otra máquina. Incluso a varias máquinas ubicadas en lugares distintos, con el objeto de que un evento fatal no nos haga perder todas las copias de seguridad.

Lo ideal es que este proceso se lleve a cabo de forma automática, para lo cual usaremos cron, que es el demonio/servicio por excelencia para llevar a cabo tareas periódicas en sistemas GNU/Linux.

Para el envío de información usaremos scp, que nos permite copiar ficheros entre ordenadores. Con la importantísima característica de que esta información va cifrada. En la implantación de cualquier solución IT debemos tener siempre en cuenta los requisitos no funcionales en general, y la seguridad en particular.

Además nos tenderemos que apoyar en un script que nos permita encapsular la doble funcionalidad “empaquetado” y “envío” en un sólo programa. Así, podremos hacer ejecutar estas dos tareas bajo el nombre de una sóla con cron.

Por último, este proceso tiene que ser desatendido, o sea, debe llevarse a cabo sin interaccionar con el usuario. Esto significa que debemos prever que el programa puede pedir confirmaciones o puede pedir claves de usuario y tenemos que darle solución a dichas cuestiones.

EJERCICIO ÚNICO:

Abre un documento de texto y ve pegando los pantallazos que muestren las evidencias de la realización de cada ejercicio.

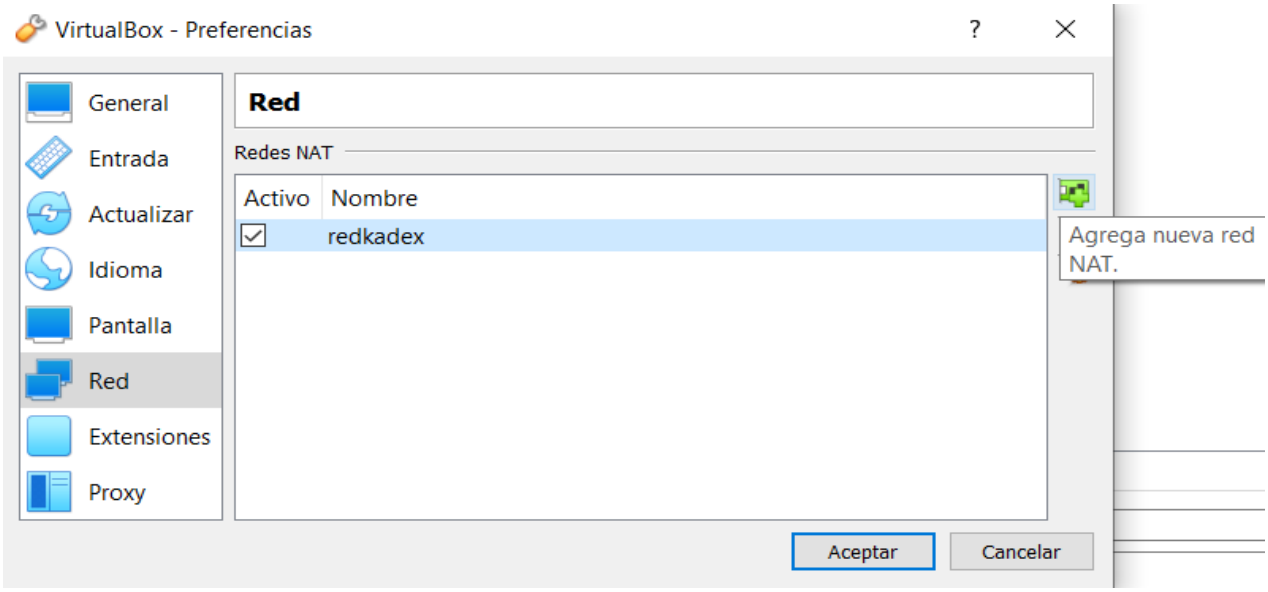
Habremos de usar dos máquinas virtuales, configurarlas en VirtualBox como Red NAT. Se denominarán “maq-A-act-final-tu-nombre” y “maq-B-act-final-tu-nombre”. Ambas máquinas contendrán una instalación Ubuntu 20.04 recién terminada y con las actualizaciones de software correspondientes llevadas a cabo.

- Para cambiar el nombre de la máquina, tenemos que modificar el fichero ‘hostname’ que se encuentra en la carpeta /etc de nuestro sistema Ubuntu:

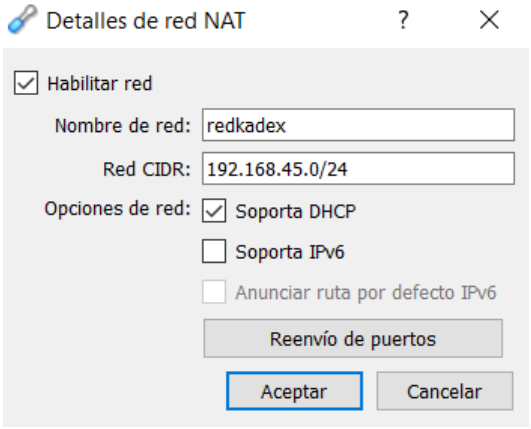
```
alberto@alberto-VirtualBox:/$ cd /etc
alberto@alberto-VirtualBox:/etc$ cat hostname
alberto-VirtualBox
alberto@alberto-VirtualBox:/etc$ sudo nano hostname
[sudo] password for alberto:
alberto@alberto-VirtualBox:/etc$ cat hostname
maq-B-act-final-AlbertoFernandezPalaciosAquino
alberto@alberto-VirtualBox:/etc$
```

(Después, tenemos que reiniciar la máquina virtual para que se apliquen los cambios).

- Para añadir las máquinas a la red, primero tenemos que crear una red mediante la interfaz gráfica de VirtualBox, en **archivos>preferencias>red**

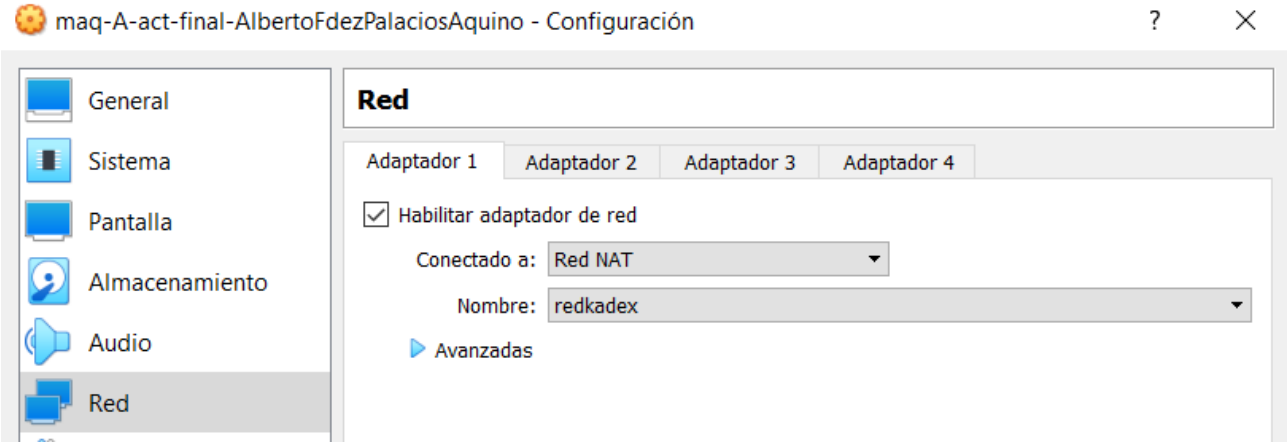


Ajustamos también las preferencias de nuestra red (yo he puesto la IP de red que se pide para la actividad):



(Esta captura no es evaluable, así que me permito ponerla sin ocupar todo el ancho).

- A continuación, añadimos las dos máquinas virtuales a esta red NAT:



Ambas máquinas se configurarán con las direcciones IP 192.168.45.1 y 192.168.45.2. Deberán tener visibilidad entre ellas. Adjunta captura de pantalla de un ping de una máquina a la otra para evidenciar que tienen visibilidad.

NOTA ACLARATORIA: He configurado la máquina A en la IP 192.168.45.2, y la máquina B en la dirección 192.168.45.3 → Pese a que en el enunciado se especifican otras direcciones para ambas máquinas, lo he hecho así ante la necesidad de configurar un Gateway para tener conexión a Internet, en la dirección IP 192.168.45.1 (considerado una buena práctica). Lo he hablado con mi profesor; Fernando Usero, quien en pleno uso de sus facultades mentales me ha confirmado que le parece bien esta forma de proceder.

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ cd /etc/netplan/
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ ls
01-network-manager-all.yaml
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ sudo nano 01-network-manager-all.yaml
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ cp 01-network-manager-all.yaml 01-network-manager-all.yaml.backup
cp: cannot create regular file '01-network-manager-all.yaml.backup': Permission denied
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ sudo cp 01-network-manager-all.yaml 01-network-manager-all.yaml.backup
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ ls
01-network-manager-all.yaml  01-network-manager-all.yaml.backup
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/netplan$ sudo nano 01-network-manager-all.yaml
```

(Utilizamos netplan para la configuración de la red de nuestras máquinas virtuales Ubuntu) (He creado una copia de seguridad, por si acaso).

```
  GNU nano 4.8                 /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.45.2/24]
      nameservers:
        addresses: [8.8.8.8]
```

(Hacemos lo mismo para ambas máquinas, pero cambiando la dirección IP. Luego se verá cuando apliquemos el comando ifconfig).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/$ sudo netplan try
Do you want to keep these settings?


Press ENTER before the timeout to accept the new configuration


Changes will revert in 116 seconds
Configuration accepted.
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/$
```

(Utilizamos este comando para aplicar los cambios al .yaml // también podemos usar directamente *'sudo netplan apply'*, pero entonces no nos verificará que todo está bien escrito).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.45.2  netmask 255.255.255.0  broadcast 192.168.45.255
        inet6 fe80::a00:27ff:fe8f:8576  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8f:85:76  txqueuelen 1000  (Ethernet)
        RX packets 105  bytes 18076 (18.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 49  bytes 6985 (6.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 145  bytes 12263 (12.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 145  bytes 12263 (12.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(Máquina A)

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.45.3  netmask 255.255.255.0  broadcast 192.168.45.255
        inet6 fe80::a00:27ff:fe8f:8576  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8f:85:76  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 5766 (5.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 115  bytes 18978 (18.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 183  bytes 21458 (21.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 183  bytes 21458 (21.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(Máquina B)

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ ping 192.168.45.3
PING 192.168.45.3 (192.168.45.3) 56(84) bytes of data.
64 bytes from 192.168.45.3: icmp_seq=1 ttl=255 time=0.274 ms
64 bytes from 192.168.45.3: icmp_seq=2 ttl=255 time=0.253 ms
64 bytes from 192.168.45.3: icmp_seq=3 ttl=255 time=0.289 ms
64 bytes from 192.168.45.3: icmp_seq=4 ttl=255 time=0.221 ms
^C
--- 192.168.45.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3049ms
rtt min/avg/max/mdev = 0.221/0.259/0.289/0.025 ms
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

(Ping entre ambas máquinas, evidenciando su visibilidad).

En la máquina virtual denominada “maq-A-act-final-tu-nombre”:

1. Configurar un servidor de SSH para poder acceder como usuario que no sea root y que el acceso sea por el puerto 21. Adjuntar en este apartado la captura de pantalla relativa a la configuración del demonio SSH.

Primero, tenemos que instalar el servidor SSH, para ello, ejecutamos el comando ‘*sudo apt install openssh-server’*. Es posible que nos de un pequeño error, y esto puede deberse a que, al estar en una RED NAT, no tenemos configurada por defecto una puerta de enlace que nos permita acceder a Internet y, por ende, descargarnos el paquete para instalar SSH en nuestra máquina virtual. Esto lo solventamos de la siguiente manera:

```
E: Failed to fetch http://es.archive.ubuntu.com/ubuntu/pool/main/o/openssh/ssh_8.2p1-4ubuntu0.4_
all.deb  Temporary failure resolving 'es.archive.ubuntu.com'
```

(Uno de los errores que nos podría aparecer).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo route add default gateway 192.168.45.1
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.45.1    0.0.0.0         UG    0      0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.45.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

(con el comando ‘route’ podemos configurar un Gateway (puerta de enlace) que nos conecte a Internet. Generalmente, es una buena práctica que esté configurada en la primera dirección IP de nuestra red, como viene especificado en el comando). Después de esto, deberíamos tener acceso a Internet.

IMPORTANTE: si después de usar este comando, utilizamos por alguna razón el comando *‘sudo netplan apply/try’* → La configuración que acabamos de aplicar al gateway desaparecerá, al no estar escrito en el fichero .yaml , y tendríamos que volver a aplicar el comando para añadir la puerta de enlace por defecto. Una forma de solventar esto sería escribir la IP del Gateway directamente en el fichero .yaml, de la siguiente manera:

```
  GNU nano 4.8                          01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.45.2/24]
      nameservers:
        addresses: [8.8.8.8]
      gateway4: 192.168.45.1
```

(Una vez hemos hecho esto, solo queda hacer *‘sudo netplan apply/try’* y ya tendríamos guardada la configuración, y acceso a Internet).

Ahora ya podemos instalar el servicio SSH en nuestra máquina virtual:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Voy a comprobar por qué puerto está escuchando el servicio ssh:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo netstat -atunp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      634/cupsd
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      588/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      3486/sshd: /usr/sbi
tcp        1      0 192.168.45.2:54516      34.104.35.123:80        CLOSE_WAIT  2259/chrome --type=
tcp        0      0 127.0.0.1:22            127.0.0.1:43910         ESTABLISHED 3323/sshd: alberto
tcp        0      0 127.0.0.1:43910         127.0.0.1:22            ESTABLISHED 3322/ssh
tcp        0      0 192.168.45.2:37376      54.148.148.62:443       ESTABLISHED 2605/firefox
tcp6       0      0 ::1:631                 :::*                    LISTEN      634/cupsd
tcp6       0      0 :::22                   :::*                    LISTEN      3486/sshd: /usr/sbi
```

(en la última línea de esta captura, se puede ver que está escuchando por el puerto 22, que es su puerto predeterminado. Pero para esta actividad, se nos ha especificado que lo configuremos en el puerto 21. Vamos allá):

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ cd /etc/ssh
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ ls
moduli       ssh_config.d  sshd_config.d       ssh_host_ecdsa_key.pub  ssh_host_ed25519_
ssh_config   sshd_config   ssh_host_ecdsa_key  ssh_host_ed25519_key    ssh_host_rsa_key
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ sudo nano sshd_config
```

(Tenemos que modificar el fichero *'sshd_config'*, que es donde se especifica la configuración del DEMONIO SSH (el otro fichero, *'ssh_config'* es para la configuración del CLIENTE, no es el que queremos tocar).

Una vez dentro, 'descomentamos' la línea que indica el numero de puerto y cambiamos el 22 por el 21, tiene que quedar así:

```
# The strategy used for options in the
# OpenSSH is to specify options with th
# possible, but leave them commented.
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 21
#AddressFamily any
```

En la siguiente captura podemos comprobar que se ha cambiado con éxito el puerto por el que escucha el servicio SSH:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ sudo netstat -atunp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      634/cupsd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      3983/sshd: /usr/sbi
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      588/systemd-resolve
tcp        0      0 192.168.45.2:42656      35.86.38.2:443          ESTABLISHED 2605/firefox
tcp        1      0 192.168.45.2:54516      34.104.35.123:80        CLOSE_WAIT  2259/chrome --type=
tcp        0      0 127.0.0.1:22            127.0.0.1:43910         ESTABLISHED 3323/sshd: alberto
tcp        0      0 127.0.0.1:43910         127.0.0.1:22            ESTABLISHED 3322/ssh
tcp6       0      0 ::1:631                 :::*                    LISTEN      634/cupsd
tcp6       0      0 :::21                   :::*                    LISTEN      3983/sshd: /usr/sbi
```

Captura de pantalla que atestigua que no se permite el acceso a el usuario root:

```
PermitRootLogin no
#StrictModes yes
MaxAuthTries 5000
```

Ahora compruebo que el servicio está activo:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-03 15:48:47 CEST; 16s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 4227 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 4228 (sshd)
      Tasks: 1 (limit: 7040)
     Memory: 1.0M
     CGroup: /system.slice/ssh.service
             └─4228 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```

```
may 03 15:48:47 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[4228]: Server listening on :: port 21.
may 03 15:48:47 maq-A-act-final-AlbertoFernandezPalaciosAquino systemd[1]: Started OpenBSD Secure Shell server.
```

Ahora compruebo, haciéndome ssh a mí mismo, que funciona el servidor:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ ssh localhost
ssh: connect to host localhost port 22: Connection refused
```

(IMPORTANTE: Este error SOLO APARECERÁ si hemos cambiado el puerto por defecto del servidor ssh). Para que funcione correctamente, tenemos que establecer también el puerto 21 en el fichero *'ssh_config'*. Lo hago a continuación para que se pueda ver con claridad:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ sudo nano ssh_config
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ sudo systemctl restart ssh
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ ssh localhost
alberto@localhost's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

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

0 updates can be applied immediately.
```

(Nótese que he reiniciado el demonio ssh tras modificar el archivo de configuración).

La modificación ha sido la misma que hicimos previamente para el fichero *'sshd_config'*:

```
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_ecdsa
#   IdentityFile ~/.ssh/id_ed25519
   Port 21
#   Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com
#   EscapeChar ~
```

(Previamente ponía *'#port 22'*, y estaba comentado, porque es el puerto por defecto para este servicio en Ubuntu).

Y compruebo de nuevo que funciona correctamente el servidor, haciéndome ssh a mí mismo (me va a pedir la contraseña de mi usuario de Ubuntu):

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ ssh localhost
alberto@localhost's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

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

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

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Tue May  3 15:54:16 2022 from 127.0.0.1
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

Pero de momento sólo nos interesa modificar el puerto 21 para el demonio ssh (es decir, el archivo *sshd_config*), así que voy a dejar el fichero '*ssh_config'* como estaba. Si hace falta cambiarlo o no, se hará más adelante.

Nota extra: recordemos que para salir del ssh, basta con escribir *'exit'* en la terminal:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ exit
logout
Connection to localhost closed.
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

2. Levantar el servicio

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo service ssh stop
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Tue 2022-05-03 16:07:46 CEST; 4s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 4291 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
    Process: 4292 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited, status=0/SUCCESS)
   Main PID: 4292 (code=exited, status=0/SUCCESS)
```

(Primero, lo he parado, para poder 'levantarlo', tal y como me pide este apartado. Lo estoy haciendo ahora con el comando *'service'* por utilizar diferentes opciones).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo service ssh start
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-03 16:08:30 CEST; 1s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 5011 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 5012 (sshd)
```

(Servicio iniciado).

3. Configurar el servicio para que arranque de forma predeterminada en cualquier arranque del sistema.

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

(A veces, con la instalación de ssh ya está configurado para que se inicie por defecto el servicio en cada arranque, pero ejecutando el comando de la captura que acabo de añadir nos aseguramos de que así sea).

4. Crear un directorio en el home del usuario que se llame "backups".

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/home$ ls
alberto
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/home$ cd alberto
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ mkdir backups
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ ls
backups  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

En la máquina virtual denominada "maq-B-act-final-tu-nombre"

5. Comprimir con un sólo comando la carpeta Descargas de tu home en un tar.bz2, que se llame "backup.tar.bz2". Se deja a criterio del estudiante el tamaño de la carpeta Descargas. Adjuntar captura de pantalla que evidencia la creación del fichero tar.bz2.

Primero me he descargado varias fotos de Internet, ya que la carpeta estaba vacía.

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ cd Downloads/
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$ ls
wp2204018.jpg  wp5442811.jpg  wp7122774.jpg  wp8834313.png  wp9558025.jpg
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$
```

Utilizaré bzip2 para hacer esto, podemos obtener información con el siguiente comando:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$ bzip2 --help
bzip2, a block-sorting file compressor.  Version 1.0.8, 13-Jul-2019.

   usage: bzip2 [flags and input files in any order]
```

DATO: los archivos .bz2 proporcionan una mayor compresión que sus homólogos gzip. Cuando queremos empaquetar y comprimir, en un solo comando, con gzip, utilicamos el modificador *-z* en el comando *tar*. Por el contrario, cuando queremos usar bzip2 (como es el caso), utilizamos el modificador *-j*. (más info en https://www.hostinger.es/tutoriales/como-usar-comando-tar-linux).

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ tar -cvjf backup.tar.bz2 Downloads/
Downloads/
Downloads/wp7122774.jpg
Downloads/wp2204018.jpg
Downloads/wp8834313.png
Downloads/wp9558025.jpg
Downloads/wp5442811.jpg
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ ls
backup.tar.bz2  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$
```

(No lo muestro en ninguna captura, pero como ta tengo el .tar.bz2, voy a borrar las fotos que hay en la carpeta Downloads).

6. Enviar el tarball del apartado anterior a la máquina virtual denominada "maq-A-act-final-tu- nombre", concretamente al directorio backups que creamos anteriormente.

¡Llega el ansiado momento de utilizar SSH para acceder a la otra máquina virtual! Pero, si lo intentamos ahora, nos va a salir un error, mira:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$ ssh 192.168.45.2
ssh: connect to host 192.168.45.2 port 22: Connection refused
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$
```

(Esto se debe a que hemos configurado previamente SSH, cambiando el puerto al número 21. Pero ahora tenemos que hacer los cambios pertinentes en nuestra máquina B).

Yo he modificado tanto el fichero *ssh_config* como el *sshd_config*. Lo he hecho así porque estoy comprobando que funciona utilizando *ssh localhost*, y para que funcionen tienen que estar configurados tanto servidor como cliente en el mismo puerto. Con esto quiero decir que probablemente para conectar con la máquina A bastaría con configurar solo el cliente de ssh, y no el servidor. Pero no lo sé, no lo he comprobado.

Cuando ejecutemos el comando ssh, ponemos la dirección IP de la máquina A. Nos va a preguntar si estamos seguros, y le decimos que por supuesto:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ ssh 192.168.45.2
The authenticity of host '[192.168.45.2]:21 ([192.168.45.2]:21)' can't be established.
ECDSA key fingerprint is SHA256:jvuSZlcCpOo1R8FF0ZibNoV6YRF0CAItbzSouYDz1Is.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.45.2]:21' (ECDSA) to the list of known hosts.
alberto@192.168.45.2's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

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

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

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Tue May  3 15:59:37 2022 from 127.0.0.1
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$
```

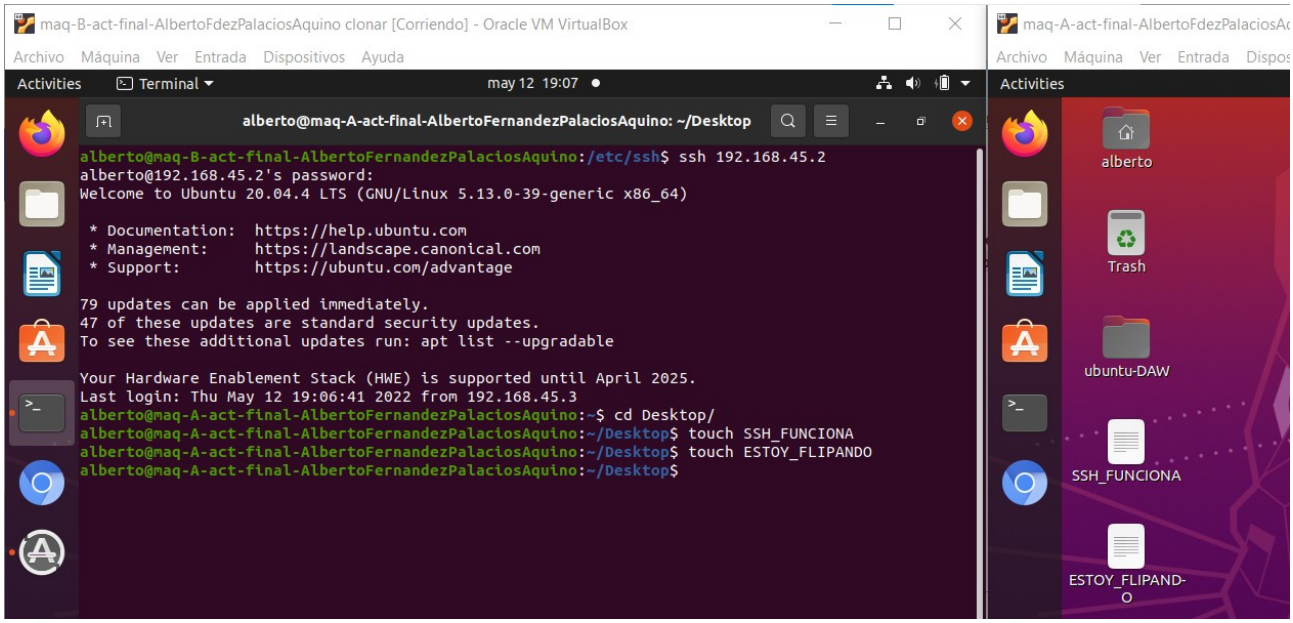
(Ya estamos conectados, es el momento de hacer algún comando de prueba. Por ejemplo, el del ejercicio de la actividad estaría bien…).

Pero antes, la comprobación más sencilla:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ cd Desktop/
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ touch SSH_FUNCIONA
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ touch ESTOY_FLIPANDO
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$
```

(Hecho en la máquina B, mediante SSH a la máquina A).

Si ahora me voy a la máquina A…



Ahora, lo que se nos pedía, **copiar el tarball al directorio backups que habíamos creado en la máquina A:**

Utilizaremos el comando SCP (Secure CoPy), que es una extensión del comando SSH y permite que podamos copiar archivos entre distintos servidores o máquinas de forma segura. También pide una autenticación previa:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ sudo scp alberto@192.168.45.3:
/home/alberto/backup.tar.bz2 alberto@192.168.45.2:/home/alberto/backups/
[sudo] password for alberto:
The authenticity of host '[192.168.45.3]:21 ([192.168.45.3]:21)' can't be established.
ECDSA key fingerprint is SHA256:jvuSZlcCpOo1R8FF0ZibNoV6YRF0CAItbzSouYDz1Is.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.45.3]:21' (ECDSA) to the list of known hosts.
alberto@192.168.45.3's password:
Permission denied, please try again.
Permission denied, please try again.
alberto@192.168.45.2: Permission denied (publickey,password).
lost connection
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$
```

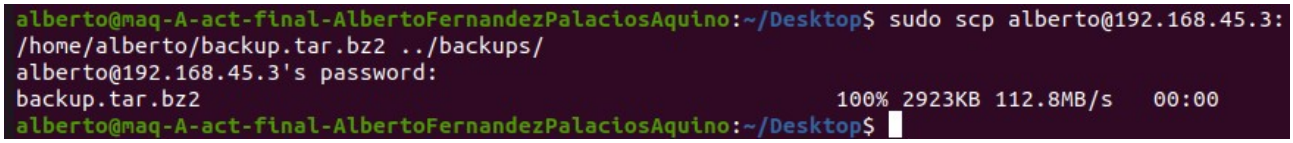
(He utilizado el comando con las rutas absolutas para que se vea la sintaxis que sigue). Podemos ver cómo utilizo el *nombre_de_usuario@ip_de_maquina:ruta_absoluta_o_relativa.* Tambien podría haber usado el nombre de la máquina virtual tras el @, en lugar de su IP.

Como vemos, se nos pide una autenticación y le damos a aceptar, y luego nos pedirá la contraseña de la máquina desde la que queremos enviar los ficheros. (También, arriba del todo, me ha pedido la contraseña de la máquina de destino, pero eso es porque he usado el comando sudo). Lo ponemos todo y… ¡Ha fallado! Vaya, ahora toca arreglarlo...

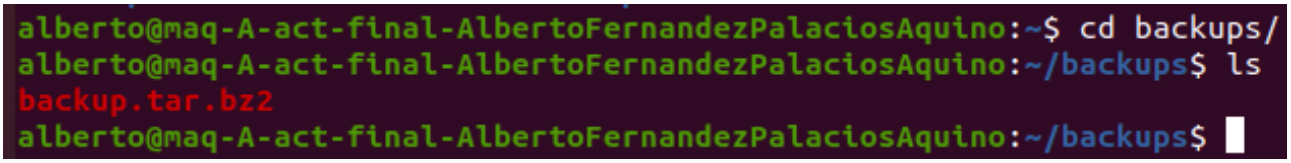
Mientras lo voy arreglando, puedes ir consultando este artículo para enterarte de todo lo relacionado con el comando SCP:
https://www.javierrguez.com/copiar-archivos-de-un-servidor-a-otro-en-linux-con-el-comando-scp/#:~:text=SCP%20con%20puerto-,Comando%20SCP%20para%20copiar%20ficheros%20entre%20servidores,forma%20segura%2C%20pidiendo%20autenticaci%C3%B3n%20previa.

¡Solucionado!

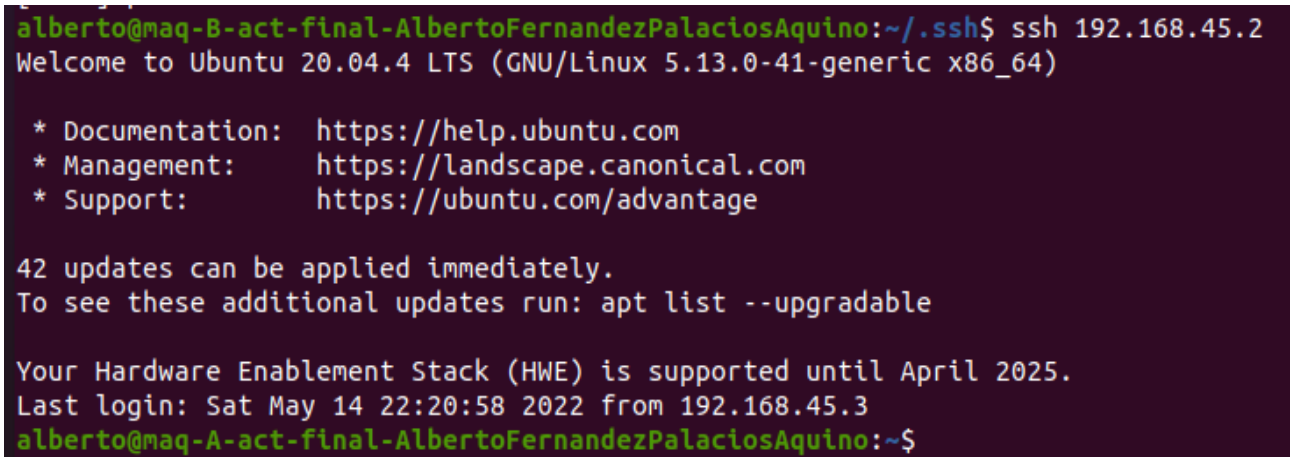
He tocado el fichero sshd_config, pero nada ha servido. No ha funcionado hasta que he usado una ruta relativa en lugar de absoluta para la máquina A (la de destino):



(Me salgo de la carpeta en la que estaba y pongo la ruta relativa al directorio backups). La demostración:

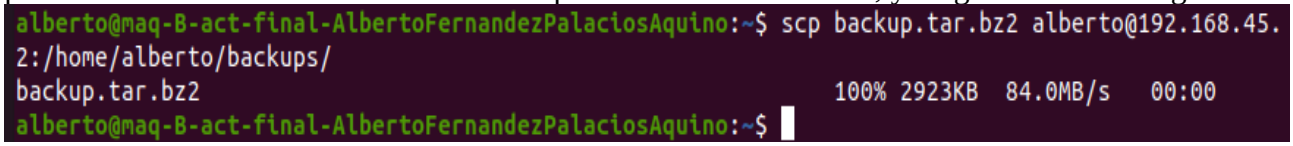


7. Repetir el paso anterior para que la copia se haga sin pedir la clave de usuario. Busca en Internet cómo se hace. Este paso puede requerir ejecutar acciones en la máquina " maq-A-act-final-tu nombre".
Adjunta las siguientes capturas de pantallas:
◦ **Una que evidencian que la ejecución se lleva a cabo sin solicitar clave de usuario**

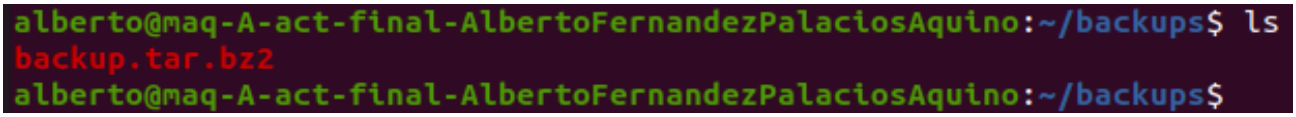


(Esta captura únicamente demuestra que ya puedo conectar las máquinas mediante SSH sin que me pida ninguna clave).

Para mostrar el comando que copia el fichero backup.tar.bz2 de la máquina B a la máquina A, primero me he salido de la conexión SSH que acababa de establecer, y luego he hecho lo siguiente:



(directamente usando el comando scp).



(Aquí estoy comprobando que efectivamente se ha copiado correctamente).

◦ **Las que sean necesarias para mostrar los comandos necesarios para que no se solicite clave**

Si no queremos iniciar sesión con contraseña, tendremos que autenticarnos de alguna manera. Esa manera son las claves SSH. Básicamente vamos a configurar un conjunto de claves (keys) que "van juntas", de manera que una es pública y la otra privada, y se necesita configurar ambas para poder acceder a la máquina mediante SSH sin que nos pidan una contraseña.

Esta analogía se entenderá mejor: funcionan, más que como dos llaves (keys), como una puerta (public key) y una llave que abre esa puerta (private key). ¡Vamos a configurarlo!

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alberto/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alberto/.ssh/id_rsa
Your public key has been saved in /home/alberto/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:lWSQAamYvfLrcO2O3pXWuOnanJynMGlQ++snzy0w4oI alberto@maq-B-act-final-AlbertoFernandezPalac
iosAquino
The key's randomart image is:
+---[RSA 3072]----+
|      .oo+o      |
|      . .o .     |
|   + ..   o      |
|  o o. . .       |
|    ... S        |
|  . .o.oo+       |
|  .+..*.*o.      |
|  Eoo=.O.B+.     |
|   o=+=+&*o..    |
+----[SHA256]-----+
```

Usamos el comando *ssh-keygen*. Lo hemos utilizado en la máquina B, es decir, la máquina desde la que quiero acceder al host.

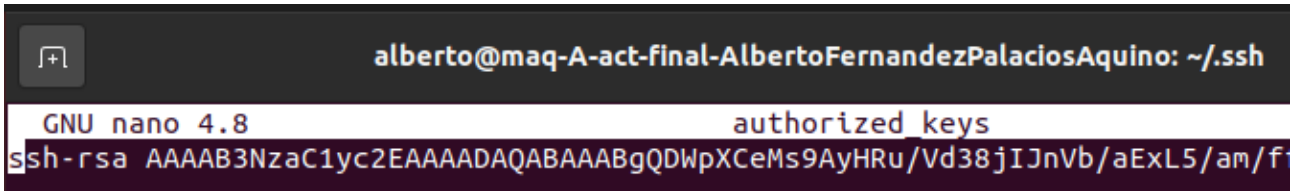
Me va a pedir que inserte una ruta donde almacenar la clave, le dejo la predeterminada. También un *passphrase*, que por motivos didácticos voy a dejar empty (aunque no es lo más recomendable) y por último me pide que añada de nuevo la *passphrase* (y así lo hago). Ahora me genera la *public/private key pair.*

Lo que tengo que hacer es copiar el fichero */home/$USER/.ssh/id_rsa.pub* de la máquina B en la máquina A. Este fichero se ha generado automáticamente en el comando anterior, y es la Public Key. (NOTA: el directorio .ssh está oculto en el home del usuario).

Esta clave tienes que mandarla a la máquina A, y guardarla en un fichero que se llama *authorized_keys* (crearlo si no existe). En la carpeta /*home/$USER/.ssh/ (Véase el siguiente enlace: https://docs.oracle.com/cd/E19683-01/806-4078/6jd6cjru7/index.html).*

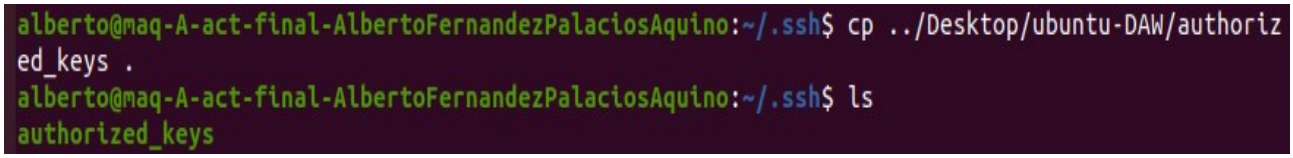
```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/etc/ssh$ cd /home/alberto/.ssh/
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/.ssh$ ls
known_hosts
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/.ssh$ touch authorized_keys
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/.ssh$ nano authorized_keys
```

(El fichero no está a priori, así que lo creo yo).



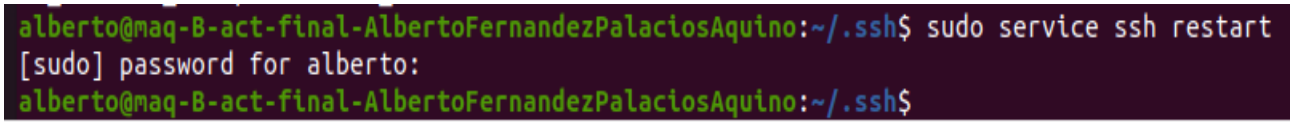
(La clave que se me había creado en la otra máquina).

Dato curioso: Yo la he mandado a la otra máquina virtual mediante una carpeta compartida entre la máquina virtual B y mi anfitrión Windows10, dicha carpeta también la he creado en la máquina A con el mismo nombre. Por lo que ha sido muy fácil:

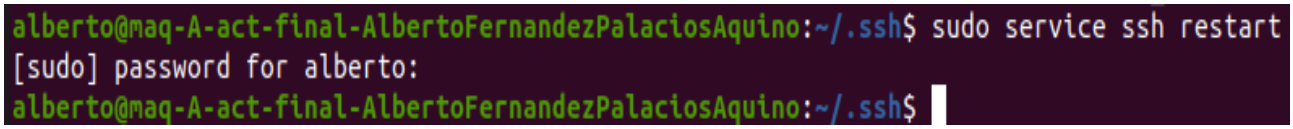


(Pero cada uno puede hacer este paso como quiera, incluso el propio demonio SSH tiene un comando que te permite hacerlo → *ssh-copy-id*). Para mas información, véase el siguiente enlace: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-ubuntu-20-04-es .

Ahora voy a reiniciar el servicio SSH en ambas máquinas:



(Máquina B).

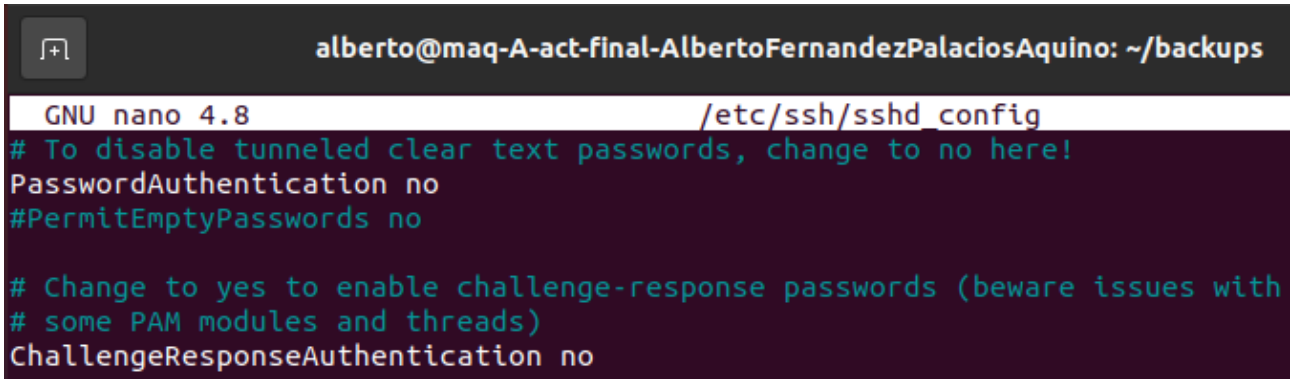


(Máquina A).

◦ **Las que sean necesarias para mostrar el contenido de los ficheros de configuración y de datos creados**

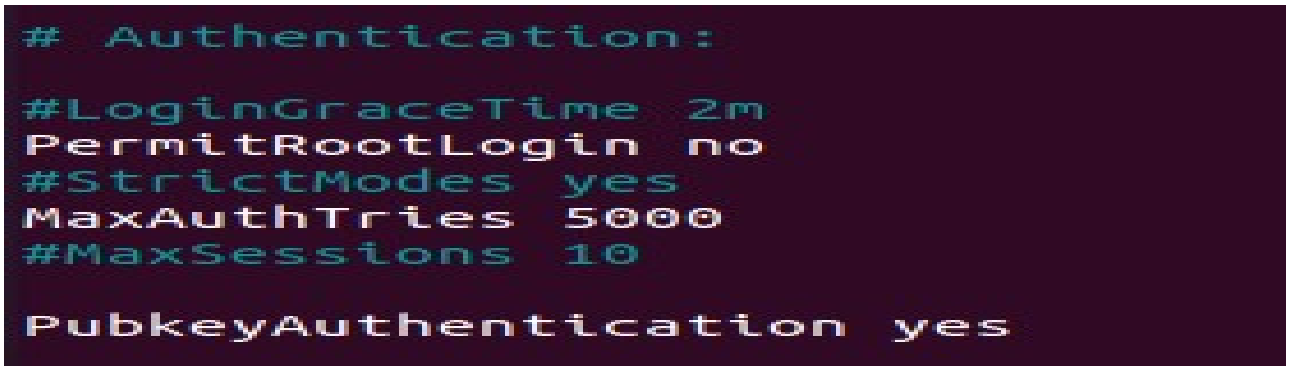
ACLARACIÓN: Si bien creo que no es necesario, yo he modificado ambos ficheros */etc/ssh/ssh_config* y */etc/ssh/sshd_config* de manera idéntica tanto para la máquina A como para la máquina B, dejándolos como muestro a continuación:

- **Fichero de configuración del server ssh (*sshd_config*)** (posiblemente este solo haya que modificarlo para la máquina A, la que va a hacer de servidor):

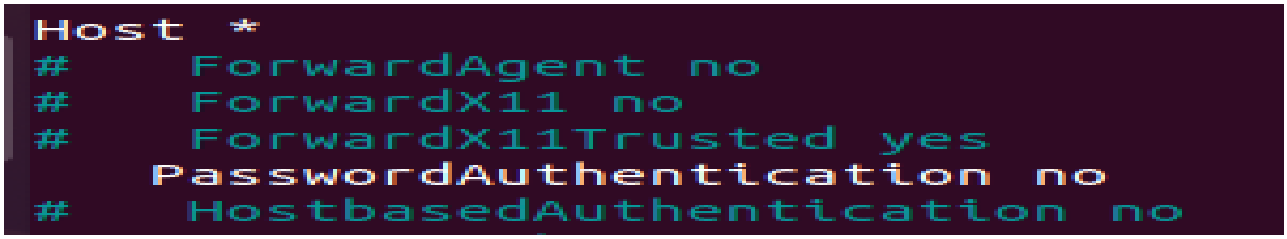


(He dejado el espacio de arriba deliberadamente para evidenciar que se trata del fichero *sshd_config* de la máquina A, en el que estoy desactivando la autenticación por contraseña).

En la siguiente imagen, lo que hago es activar la autenticación a través de Public Key. También se puede ver que he tocado la variable 'MaxAuthTries 5000'. Esto lo he puesto porque me iba fallando muchas veces la conexión conforme he ido aprendiendo a configurarlo, y no quería que se me bloquease ssh o dejara de poder usar la clave generada:



- **Fichero de configuración del cliente ssh (*ssh_config*)** (posiblemente este solo haya que modificarlo para la máquina B, la que va a hacer de cliente):



(Recordemos que estos ficheros son los mismos en los que modificamos el puerto para usar el 21 con ssh, y que dichos cambios siguen estando aplicados).

8. Crea un script que lleve a cabo la creación del tarball y el envío del fichero. El tarball ha de incluir la fecha y hora de creación del mismo en el nombre del archivo. Adjunta captura de pantalla que contenga el contenido del script.

(EXTRA → Breve introducción a los script shell de linux):

Se puede ampliar información en otros enlaces, como este: https://www.cyberciti.biz/faq/how-to-execute-a-shell-script-in-linux/ .

He creado un script sencillo de prueba:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ touch cool_script.sh
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ sudo nano cool_script.sh
```

```
  GNU nano 4.8                                cool_script.sh
#!/bin/bash
#Script para la super-hiper-mega-practica-que-te... digo, Actividad Final.
echo -e "Hola, dime tu nombre: "
read nombre
echo "Encantado, $nombre"
```

Para poder ejecutarlo, tengo que cambiarle los permisos. Tengo que hacer que pueda ser un fichero ejecutable, si no obtendré el siguiente error:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ ./cool_script.sh
bash: ./cool_script.sh: Permission denied
```

Yo he utilizado el siguiente comando:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ chmod +x cool_script.sh
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ ./cool_script.sh
Hola, dime tu nombre:
Fusero
Encantado, Fusero
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$
```

Bien, pues ahora se trata de que el script haga todo lo que se nos pide, ¡Allá vamos!

Voy a crear el script y a añadirle directamente el permiso que me interesa:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ touch backup_script.sh
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ nano backup_script.sh
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ chmod +x backup_script.sh
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$
```

El contenido del script es el siguiente:

Open backup_script.sh ~/Desktop Save

```
1 #!/bin/bash
2 #Script para la super-hiper-mega-practica-que-te... digo, Actividad Final.
3
4 echo "Iniciando el script..."
5 nombreFichero=$(date +"%Y-%m-%d-%H%M%S")_backup
6 cd $HOME/
7 tar -cvjf $nombreFichero.tar.bz2 Downloads/
8 scp $nombreFichero.tar.bz2 alberto@192.168.45.2:/home/alberto/backups
9
10 #Descomenta la siguiente linea si quieres que se borre el fichero de backup cuando se haya
   copiado en el host remoto:
11 #rm $nombreFichero.tar.bz2
12
13 #Descomenta la siguiente linea si quieres borrar todos los ficheros de Downloads tras terminar
   el proceso de copiar los datos en el host remoto:
14 #rm $HOME/Downloads/*
```

En la anterior captura se puede ver que he añadido algunas opciones en el script, que por defecto he puesto desactivadas (es a gusto del consumidor si se activan o no).

NOTA: se me antoja un poco incómodo que el *.tar.bz2* genere una darpeta Downloads nueva cada vez que se descomprime, pero era el criterio del apartado correspondiente… Probablemente, si estás siguiendo esta guía para configurarlo en tu propio equipo, no quieras que te genere la carpeta constantemente (habría que cambiar el comando tar o tocar el *backup_script.sh)*.

Verificación de que todo funciona:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$ ./backup_script.sh
Iniciando el script...
Downloads/
Downloads/wp7122774.jpg
Downloads/wp2204018.jpg
Downloads/wp8834313.png
Downloads/wp9558025.jpg
Downloads/wp5442811.jpg
2022-05-15-005644_backup.tar.bz2                    100% 2923KB 128.3MB/s   00:00
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Desktop$
```

(Script ejecutado en la máquina B, está configurado para que no importe desde qué directorio se ejecute).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ cd backups
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ ls
2022-05-15-005644_backup.tar.bz2
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ tar -xf 2022-05-15-005644_backup.tar.bz2
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ ls
2022-05-15-005644_backup.tar.bz2  Downloads
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ cd Downloads/
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups/Downloads$ ls
wp2204018.jpg  wp5442811.jpg  wp7122774.jpg  wp8834313.png  wp9558025.jpg
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups/Downloads$
```

(Aparece correctamente en la máquina A).

Enlaces de interés de esta parte:
- https://atareao.es/tutorial/scripts-en-bash/variables-en-bash/
- https://blogs.upm.es/estudiaciencia/variables-en-bash/
- https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es

9. Configura cron para que ejecute el script cada hora. Adjunta captura de pantalla que evidencie la correcta configuración del cron. Adjunta captura de los logs, filtrando la parte correspondiente a la ejecución del script, con al menos dos ejecuciones del mismo. Asimismo, es necesaria una segunda captura de pantalla de un ls (con detalles) que muestre que los ficheros se están volcando al directorio de backups. Deben aparecer tantos ficheros como ejecuciones del script se hayan realizado.

Cron es un demonio de Linux que nos sirve para programar tareas. Este servicio está basado en el tiempo y nos permite ejecutar dichas tareas de forma automática. (Más información en: https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-ubuntu-1804-es). (Y también → https://www.linuxtotal.com.mx/?cont=info_admon_006).

Para continuar, debemos instalar cron, con *sudo apt update* y luego *sudo apt install cron*.

Es posible que ya estuviera instalado en la máquina (como me ha pasado a mí). En cualquier caso, lo siguiente es asegurarnos de que lo configuramos para ejecutarse en segundo plano:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ sudo systemctl enable cron
Synchronizing state of cron.service with SysV service script with /lib/systemd/systemd-sysv-inst
all.
Executing: /lib/systemd/systemd-sysv-install enable cron
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$
```

(So far so good).

Ahora, hay varias formas de utilizar cron. Probablemente, a estas alturas del tutorial estemos cansados y deseando hacer nuestra vida mas fácil, por lo que podríamos plantearnos utilizar la solución menos compleja:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ cd /etc/cron.
cron.d/       cron.daily/   cron.hourly/  cron.monthly/ cron.weekly/
```

Cuando instalamos Cron, en el directorio /etc tendremos las carpetas que se ven en la captura que he añadido inmediatamente arriba. En teoría bastaría con añadir nuestro script a la carpeta correspondiente para que este se ejecute diariamente, semanalmente, cada hora, etc.

Yo lo he probado, pero desafortunadamente no me ha salido a la primera y, en cualquier caso, lo voy a configurar de otra manera que nos va a permitir ser más específicos con nuestros requerimientos.

Debemos utilizar el comando *crontab -e*, que nos va a generar un fichero en el que podemos configurar, mediante la sintaxis de Cron, qué script queremos ejecutar y en qué momento, con qué periodicidad, etc:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~$ crontab -e
no crontab for alberto - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.tiny
  3. /bin/ed

Choose 1-3 [1]: 1
crontab: installing new crontab
```

(La primera vez que lo ejecutamos para nuestro usuario, nos saldrá esto)

Yo he elegido el editor de texto nano, por ser el que utilizo con más frecuencia. Mi fichero lo he configurado de la siguiente manera (había unas instrucciones comentadas que he eliminado):

```
alberto@maq-B-act-final-AlbertoFernandezPalacios/
  GNU nano 4.8                  /tmp/crontab.RXj7HS/crontab
0 * * * * /home/alberto/Desktop/backup_script.sh
```

(De este modo, lo que hará será ejecutar el *backup_script.sh* cada hora). Dejar el script en el Escritorio no es lo más recomendable, porque podría moverlo sin querer y la configuración que he especificado para Cron ya no funcionaría, pero lo he hecho así por motivos didácticos.

Guardamos el fichero y, para asegurarnos de que vaya a funcionar, reiniciamos el demonio Cron:

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$ sudo systemctl restart cron
[sudo] password for alberto:
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:~/Downloads$
```

(Ahora solo falta esperar...)

Et voilà! Aquí tenemos la demostración de que se ha ejecutado correctamente la primera vez:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~$ cd backups/
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ ls
2022-05-15-160001_backup.tar.bz2
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$
```

(Se puede observar que el fichero indica la fecha y hora en que se ha creado, ahora debería ejecutarse cada hora).

La siguiente captura muestra 4 ficheros correspondientes a 4 ejecuciones del script, he hecho un ls con detalles en esta ocasión:

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$ ls -l
total 11696
-rw-rw-r-- 1 alberto alberto 2993328 may 15 16:00 2022-05-15-160001_backup.tar.bz2
-rw-rw-r-- 1 alberto alberto 2993525 may 15 17:00 2022-05-15-170001_backup.tar.bz2
-rw-rw-r-- 1 alberto alberto 2993493 may 15 18:00 2022-05-15-180001_backup.tar.bz2
-rw-rw-r-- 1 alberto alberto 2993512 may 15 19:00 2022-05-15-190001_backup.tar.bz2
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:~/backups$
```

Los logs que queremos consultar se encuentran en el directorio */var/logs* (al menos en la distro de Ubuntu, tenemos que ver cuál de todos los ficheros es el que vamos a utilizar, el siguiente enlace tiene una tabla que puede resultar de utilidad: https://geekland.eu/logs-en-linux/).

```
alberto@maq-A-act-final-AlbertoFernandezPalaciosAquino:/var/log$ cat auth.log | grep 1[6,7,8,9]:00
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[3610]: Accepted publickey for alberto from 192.168.45.3 port 33370 ssh2: RSA SHA2
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[3610]: pam_unix(sshd:session): session opened for user alberto by (uid=0)
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino systemd-logind[684]: New session 11 of user alberto.
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[3663]: Received disconnect from 192.168.45.3 port 33370:11: disconnected by user
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[3663]: Disconnected from user alberto 192.168.45.3 port 33370
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino sshd[3610]: pam_unix(sshd:session): session closed for user alberto
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino systemd-logind[684]: Session 11 logged out. Waiting for processes to exit.
May 15 16:00:01 maq-A-act-final-AlbertoFernandezPalaciosAquino systemd-logind[684]: Removed session 11.
```

(Captura extra) → Esta es la mayor información que podemos obtener desde la máquina A, y lo hacemos mirando el fichero auth.log, donde podemos ver, para cada hora, la conexión establecida mediante SSH con la máquina B, y como dicha conexión se cierra. No podemos obtener más información porque los comando realmente se están ejecutando desde la máquina B (el comando *scp*).

Para conseguir la captura que muestre la ejecución del script de Cron, tenemos que ir a los logs de la máquina B: (Las últimas líneas indican cuándo el script se ha ejecutado de seguido).

```
alberto@maq-B-act-final-AlbertoFernandezPalaciosAquino:/var/log$ cat syslog.1 | grep "backup_script.sh"
May 15 02:18:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[9967]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 02:30:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[10012]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 13:30:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[2878]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 15:00:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[3308]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 16:00:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[3684]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 17:00:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[3837]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 18:00:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[3994]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
May 15 19:00:01 maq-B-act-final-AlbertoFernandezPalaciosAquino CRON[4106]: (alberto) CMD (/home/alberto/Desktop/backup_script.sh)
```