

## **CASOS PRÁCTICOS CON DEBIAN 9**

**Iván Miranda**

## Caso práctico 1

*Convierte la IP dinámica que tienes en modo estático. Para ello averigua la IP dinámica asignada ¡Asegúrate que tienes tu máquina virtual en modo **bridge**!*

*Para ello... deberás conocer de qué manera conoces la IP que se te ha asignado de manera dinámica a través del servidor DHCP de la clase. Anótala y configura el fichero `/etc/network/interfaces` de tal manera que el direccionamiento sea estático.*

*Anota los comandos que has usado para:*

- *Instalar ifconfig que no viene instalado en Debian 9.*
- *Apagar el interfaz de red cableada*
- *Activar el interfaz de red cableada.*

*Tradicionalmente en todas las distribuciones basadas en Debian, las interfaces de red cableadas se llamaban `ETH0`, `ETH1`... Pero esto ha cambiado. ¿De qué manera se llaman los interfaces de red en Debian 9 ahora?*

Maneras de saber la ip asignada por el servidor DHCP:

- `hostname -I`
- `ip addr show`

Instalar ifconfig en Debian9: parece ser que ya no viene instalado por defecto en Debian9 porque se considera obsoleto, como ocurre también con `route`. Efectivamente, si queremos instalarlo, lo tendremos que hacer manualmente. Según las fuentes de internet, hay que instalar el paquete `net-tools`. Para ello:

```
sudo apt install net-tools
```

Tras ello, ejecutamos el `ifconfig` y vemos que la ip es: 192.168.1.41

Otro de los cambios fundamentales de Debian 9 con respecto a sus versiones anteriores es precisamente el nombre de las interfaces de red. Ya no se llaman como antaño `eth0`, `eth1` o `ethX`. Es el sistema operativo el que renombra las interfaces. Si le hacemos un `cat` a `/etc/network/interfaces`, vemos que el nombre de la interfaz ahora es `enp0s3`.

En la página principal de Debian podemos encontrar bastante información sobre cómo configurar todos los aspectos relacionados con redes. La mayoría de parámetros relacionados con la configuración de red se pueden modificar en `/etc/network/interfaces`. Puede, entre otras cosas, fijar una ip estática u obtenerla por DHCP, establecer la información del enrutamiento, configurar el enmascaramiento IP, poner rutas por defecto, etc.

## Análisis del archivo *interfaces* y sus parámetros

En el archivo de la máquina Debian 9 presenta este aspecto:

```
auto lo

iface lo inet loopback

allow-hotplug enp0s3

iface enp0s3 inet dhcp
```

Cambia eth0 por enp0s3. Si hacemos un `man interfaces` en la línea de comandos, veremos nada más comenzar la información cómo configurar una ip estática para el servidor. Sería así:

```
auto lo

iface lo inet loopback

allow-hotplug enp0s3

iface enp0s3 inet static

    address 192.168.1.50/24

    gateway 192.168.1.1
```

**Las líneas auto** sirven para especificar las interfaces físicas que se activarán cuando se ejecute el comando *ifup -a*, que se ejecuta automáticamente al arrancar el sistema. Así, quedan especificadas las tarjetas que se activan desde el inicio de la máquina. Se pueden poner todas las líneas de este tipo que se deseen y se pueden acumular las interfaces.

**Líneas allow-auto:** equivalentes a las líneas auto.

**Líneas allow-hotplug:** similares a las anteriores. Las interfaces incluidas en estas líneas se activan cuando se producen los eventos *hotplug* en las interfaces de red, como por ejemplo la detección de la tarjeta de red por parte del kernel o la conexión del cable de red. Cuando se cumple esta condición o estos eventos, el sistema ejecuta el comando *ifup*.

**Líneas iface:** sirven para definir nombres lógicos de interfaces de red junto con su configuración particular.

Estructura de las líneas iface:

iface + config\_name + address\_family + method\_name

- Config\_name: nombre lógico de la interfaz.
- Address\_family: especifica la configuración IPv4. Normalmente es *inet*, pero también puede ser *inet6* o *ipx*.
- Method\_name: método de configuración de la interfaz. Depende del valor asignado en address\_family. Concretamente, para *inet*, tenemos los valores:
  - Loopback: usado solo para la interfaz lo.
  - Static: usado para establecer ip fija.
  - Dhcp: usado para establecer ip dinámica.

A continuación, se tienen que introducir las parejas opción/valor, que dependerán de la familia y del método elegido.

Las dos primeras líneas del fichero `/etc/network/interfaces` deben ser:

```
auto lo
```

```
iface lo inet loopback
```

¿Qué hacen estas líneas? Activan la interfaz lo (bucle local), que usaremos en tareas de diagnóstico de conectividad y validez de los protocolos de comunicación.

Si vamos al método *static*, presenta las siguientes opciones:

- Address: dirección ip que pretendes fijar. Obligatorio.
- Netmask: máscara de la dirección IP. Obligatorio.
- Gateway: puerta de enlace. Solo puede haber una regla por defecto de enrutamiento.
- Network: dirección de la red a la que pertenece la dirección IP.
- Broadcast: dirección de broadcast de la red.
- Hwaddress: establece la dirección MAC de la tarjeta. Para las tarjetas ethernet, habrá que incluir la palabra *ether* como tipo de la interfaz.

- Pre-up: ejecuta el comando antes de que la tarjeta se configure con el comando *ifup*. Si el comando falla, *ifup* no se ejecuta, se manda mensaje de error y la tarjeta no queda configurada.
- Up: ejecuta comando durante la configuración de la tarjeta con el comando *ifup*
- Post-up comando: ejecuta el comando después de que la tarjeta se configure con el comando *ifup*. Si el comando falla, *ifup*, imprime mensaje de error y la tarjeta no se marca como configurada.
- Pre-down comando: ejecuta el comando antes de que la tarjeta se desconfigure con el comando *ifdown*. Si el comando falla, *ifdown* aborta, imprime mensaje de error y la tarjeta se marca como desconfigurada.
- Down: ejecuta el comando durante la desconfiguración de la tarjeta con el comando *ifdown*.
- Post-down: ejecuta el comando después de que la tarjeta se desconfigure con el comando *ifdown*. Si el comando falla, *ifdown* aborta, imprime mensaje de error y la tarjeta no se marca como desconfigurada.

¿Qué ocurre con las opciones *pre-up*, *up*, *post-up*, *pre-down*, *down* y *post-down*?

- Son opcionales
- Valen para cualquier familia y método.
- Pueden repetirse tantas veces como se quiera.
- Se ejecutan de forma ordenada, en su momento, y según el orden de aparición en el fichero.

Además de las líneas *up*, *down*, y demás; existe la posibilidad de ejecutar comandos cuando una tarjeta de red se está activando o desactivando. Se realiza mediante la introducción de scripts. Serían del tipo:

- `etc/network/if-down.d`
- `etc/network/if-post-down.d`
- `etc/network/if-pre-down.d`
- `etc/network/if-up.d`

En el método *dhcp*, las opciones serían.

- `hwaddress`
- `pre-up`, `post-up`
- `pre-down`, `down`, `post-down`

Por tanto, según lo visto, para tirar o levantar las interfaces de red, tendremos que usar los siguientes comandos:

- Para tirar la interfaz de red cableada: ifdown.
- Para levantar la interfaz de red cableada: ifup.

**Ejemplo práctico:**

```
auto lo
```

```
iface lo inet loopback
```

```
allow-hotplug enp0s3
```

```
iface enp0s3 inet static
```

```
    address 192.168.1.50/24
```

```
    netmask 255.255.255.0
```

```
    gateway 192.168.1.1
```

Fuentes: Wikipedia, google, <http://fpg.66ghz.com/DebianRed/etcnetworkinterfaces.html?i=1>,  
man interfaces.

## Caso práctico 2

*Para instalar aplicaciones necesitamos tener el sistema **apt** correctamente configurado.*

*Asegúrate que tienes los repositorios correctamente escritos en el fichero correspondiente.*

La configuración del sistema apt se encuentra en la siguiente ubicación:

`/etc/apt/sources.list`

En este archivo se configuran los parámetros necesarios para el correcto funcionamiento de este gestor de paquetes.

Uno de los puntos más fuertes de Debian es la versatilidad que ofrece el gestor de instalación nativo apt. Todo elemento dentro de Debian se construye o incluye en paquetes que son instalados en el sistema.

APT significa *Advanced Package Tool*. Es, como se ha señalado, un gestor de instalación de paquetes. Tiene las siguientes utilidades:

- Instalar aplicaciones
- Borrar aplicaciones
- Actualizar aplicaciones
- Corrección de errores en la configuración de los paquetes.

APT permite la resolución de problemas de dependencia y permite obtener los paquetes solicitados de los repositorios señalados. Delega tanto la instalación como la eliminación de los paquetes en *dpkg*. Normalmente, apt se ejecuta a través del shell, aunque podemos encontrar interfaces gráficas para su empleo.

Como ya hemos señalado antes, el archivo *sources.list* es usado por el sistema apt como parte de su ejecución. Este archivo contiene una lista de fuentes de las cuales se obtienen los paquetes. Es de vital importancia, por tanto, tenerlo correctamente configurado para que las peticiones realizadas al sistema se lleven a cabo de forma apropiada.

### **Análisis de parámetros en el sistema APT**

#### **Tipos de archivo**

La primera entrada de cada línea representa el tipo de archivo:

- Deb: significa que la url proporcionada contiene paquetes precompilados. Estos paquetes son instalados por defecto cuando se usan gestores de instalación como apt-get o aptitude.
- Deb-src: indica la fuente que contiene los cambios requeridos para la instalación del paquete. Usa archivos de control .dsc y diff.gz.

## URL del repositorio

La siguiente entrada es una url que apunta al sitio del que se descargarán los paquetes. Se puede consultar la lista principal de los repositorios Debian en <https://www.debian.org/mirror/list>.

## Distribución

La distribución puede ser nombrada mediante el nombre de versión o alias, o bien por el tipo de clase.

El archivo tendría que tener, normalmente, este aspecto:

```
deb http://deb.debian.org/debian stretch main
```

```
deb-src http://deb.debian.org/debian stretch main
```

```
deb http://deb.debian.org/debian stretch-updates main
```

```
deb-src http://deb.debian.org/debian stretch-updates main
```

```
deb http://security.debian.org/debian-security/ stretch/updates main
```

```
deb-src http://security.debian.org/debian-security/ stretch/updates main
```

Ahora bien, si quiero tener también los componentes non-free y los componentes main, tendré que añadir las siguientes líneas:

```
deb http://deb.debian.org/debian stretch main contrib non-free
```

```
deb-src http://deb.debian.org/debian stretch main contrib non-free
```

```
deb http://deb.debian.org/debian stretch-updates main contrib non-free
```

```
deb-src http://deb.debian.org/debian stretch-updates main contrib non-free
```

```
deb http://security.debian.org/debian-security/ stretch/updates main contrib non-free
```

```
deb-src http://security.debian.org/debian-security/ stretch/updates main contrib non-free
```

Actualizo, por tanto, sources.list y, a continuación tendré que ejecutar el siguiente comando para que los cambios surjan efecto:

```
sudo apt-get update
```

Esto asegurará que el índice del sistema apt esté correctamente sincronizado.



## Añadir repositorios customizados

Llevar a cabo esta operación no es siempre aconsejable. En lugar de ello, lo que se puede hacer aparte es crear un archivo en el directorio `/etc/apt/sources.list.d`. Volviendo a la inclusión de repositorios personalizados, tendríamos que abrir nuestro archivo `sources.list` y añadir la línea de comando correspondiente. Por ejemplo, si quisiéramos añadir docker:

```
deb https://apt.dockerproject.org/repo debian-stretch main
```

## Importar keys apt o apt-keys

Cuando se trabaja con los repositorios `sources.list`, en algún punto entran en juego las gpg keys. Estas son gestionadas mediante el comando `apt-key`. La sintaxis es:

```
# apt-key adv --keyserver <server-address>--recv-keys <key-id>
```

Siguiendo con el ejemplo anterior, para descargar las claves de docker, ejecutaríamos:

```
# apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys  
58118E89F3A912897C070ADB76221572C52609D
```

Finalmente, actualizamos apt e instalamos el paquete correspondiente:

```
# apt-get update && apt-get install docker-engine
```

En general, trabajar con `sources.list` es sencillo. El único aspecto más o menos complejo es el de colocar la distribución correcta. Si el repositorio se instala en una distribución estable con paquetes inestables, esto puede llevar a la rotura del sistema o a numerosos problemas de dependencias.

### 2.1 ¿Cuáles son estos?

*Asegúrate que entiendes para qué sirven los comandos `apt update`, `apt upgrade` y `apt remove`.*

- **apt-update:** apt trabaja sobre una base de datos que contiene todos los paquetes de instalación disponibles. Si esta base de datos no está actualizada, el sistema no sabrá, lógicamente, si se han incluido nuevos paquetes. Por tanto, la primera cosa que tenemos que hacer en cuanto a apt concierne, es ejecutar el comando `apt update` para actualizar dicha base de datos y tener los repositorios al día. Requiere permisos de superusuario.

Al ejecutarlo, verás en consola la información del paquete proporcionada por los distintos servidores. Hay tres tipos de líneas que pueden aparecer y que nos interesan:

- Hit: indica que no hay cambios en la versión del paquete en comparación a la versión previa.
- Ign: indica que se ignora el paquete. Las razones son varias. O bien la nueva versión es tan reciente que ni se molesta en actualizarla o bien se ha dado un error en la transferencia del archivo, pero este es tan trivial que ni lo menciona.

- **Get:** indica que hay una nueva versión disponible. Descargará la información de la versión, no el paquete como tal, ya que lo que está haciendo es actualizar la información de la base de datos y no descargando el paquete.
- **apt-upgrade:** una vez se ha actualizado la base de datos con apt update, puedes optimizar los paquetes instalados. La mejor manera de hacerlo es apt upgrade. Existe otra opción más compleja, que es apt full-upgrade, y que añade ciertas funcionalidades a la operación convencional update, como por ejemplo borrar paquetes que ya están instalados.

¿Qué diferencias me encuentro entre apt update y apt upgrade?

Apt update solo actualiza la información de la base de datos. Apt upgrade sí que toma los paquetes instalados de la máquina local y ejecuta la actualización de los mismos. Resumiendo, la primera informa; la segunda ejecuta.

- **apt-remove:** es el comando para eliminar paquetes instalados en el sistema. Tiene funciones de autocompletado, tal y como ocurre con apt install. Un comando similar es apt purge, que sirve para lo mismo, pero con ciertas diferencias.

¿Qué diferencias encuentro entre apt remove y apt purge?

La primera elimina los binarios del paquete y deja archivos de configuración residuales. La segunda lo elimina absolutamente todo.

## *2.2 ¿Qué significan los parámetros main contrib y non-free que hay al final de las líneas de repositorios?*

### **Componentes**

Normalmente encontramos tres componentes que pueden ser empleados en Debian:

- **Main:** contiene los paquetes que forman parte de la distribución Debian. Estos paquetes están compilados en DFSG.
- **Contrib:** son también compilados en DFSG, pero contienen paquetes que no se encuentran en el repositorio principal.
- **Non-free:** contiene software no compilado en DFSG.

## *2.3. Como tenemos el sistema sin escritorio, vamos a instalarle uno muy sencillo y ligero. Se llama XFCE4. Asegúrate que tienes los repositorios actualizados antes de instalar.*

Tenemos que asegurarnos de que los repositorios están actualizados: hacemos un apt-get update con permisos de superusuario.

A continuación, ejecutamos la instalación: `sudo apt-get install + nombre del paquete`. En este caso, ejecutamos:

```
apt install xfce4
```

Sin embargo, no todo es tan fácil. Aquí me surgió un problema con la resolución de nombres por parte del servidor dns durante la previa ejecución del comando `sudo apt update`, en el cual los accesos previstos en el archivo `sources.list` no se llevaban a cabo porque por alguna razón no respondía correctamente el dns.

Leyendo en foros, veo que se hacen distintas pruebas con ping a los servidores dns para ver si estos responden. Algunos proponen cambiarlos si esto ocurre, por ejemplo al típico 8.8.8.8 de google. Tras hacer las pruebas correspondientes, veo que ningún ping responde bien.

Decido volver a configurar la ip como dinámica para ver si había conflicto entre esto y la red nat configurada en la máquina virtual y, efectivamente, ahora los pings y el `apt update` se hacen como deben. Ahora la instalación de `xfce4` va perfecta.

Pero yo quiero tener ambas cosas: mi ip fija y mi sistema apt funcionando como debe. Le asigno entonces un ip dentro del rango establecido en la configuración. Por ejemplo, 10.0.2.50. Tras ello, le hago un reboot a la máquina y pruebo a hacerle ping a distintos servidores. Sin embargo, las pruebas salen mal. Al configurar la ip como estática, los servidores dns no responden y la red se muestra inaccesible en las pruebas con ping.

Consigo arreglarlo tras muchas pruebas modificando el archivo `/etc/network/interfaces`, poniendo la ip como estática y añadiendo estos atributos a la conexión:

- Address
- Netmask
- Gateway

Justo después, ejecuto `service networking restart` y, después, `ifup enp0s3` (nuevo nombre de la tarjeta de red simulada). Al hacerlo estrictamente en ese orden, las pruebas con ping transmiten al fin.

*2.4 Una vez que esté instalado el entorno gráfico, vamos a instalar el navegador Chrome. Para ello tendrás que seguir las instrucciones que encontrarás aquí:*

*Pero lo vamos a hacer un poco más complicado...*

*En vez de usar **apt** vamos a usar un instalador de paquetes manual. Esto tiene algunos problemas que ahora te encontrarás.*

*Para ello:*

- *Descarga el .deb que hay en este enlace.*
- *Instala el paquete manualmente usando dpkg -i.*
- *¿Ha habido algún problema? ¿Qué es lo que ha pasado?*


*¿Qué beneficio tiene entonces instalar usando **apt**?*

Normalmente, el problema fundamental que nos podemos encontrar al usar dpkg es que este no resuelve dependencias.

Cuando usamos el comando apt para instalar un paquete, internamente está usando dpkg. Cuando instalas un paquete con apt, primero creará una lista de todas las dependencias y las descargará del repositorio. Una vez que este proceso se ha realizado, se llamará al comando dpkg para instalar esos archivos, satisfaciendo todas las dependencias.

Para entender mejor todos estos conceptos, viene bien saber exactamente qué es una dependencia, un concepto muy recurrente en este entorno. Una dependencia es un archivo que el paquete requiere para ser instalado. En resumen, es una porción de software sobre la cual se apoya la instalación del paquete.

Vamos a hacer el caso práctico. Ejecuto el comando dpkg -i + ruta del archivo .deb. Para ello primero tengo que descargar el archivo .deb, cosa que no podré hacer porque no tengo navegador en Debian 9. Aquí ha sido útil la herramienta de Putty pscp para realizar la transferencia del archivo desde la máquina xp a esta.



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrador>cd ..
C:\Documents and Settings>cd administrador/escritorio
C:\Documents and Settings\Administrador\Escritorio>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 44A3-794C

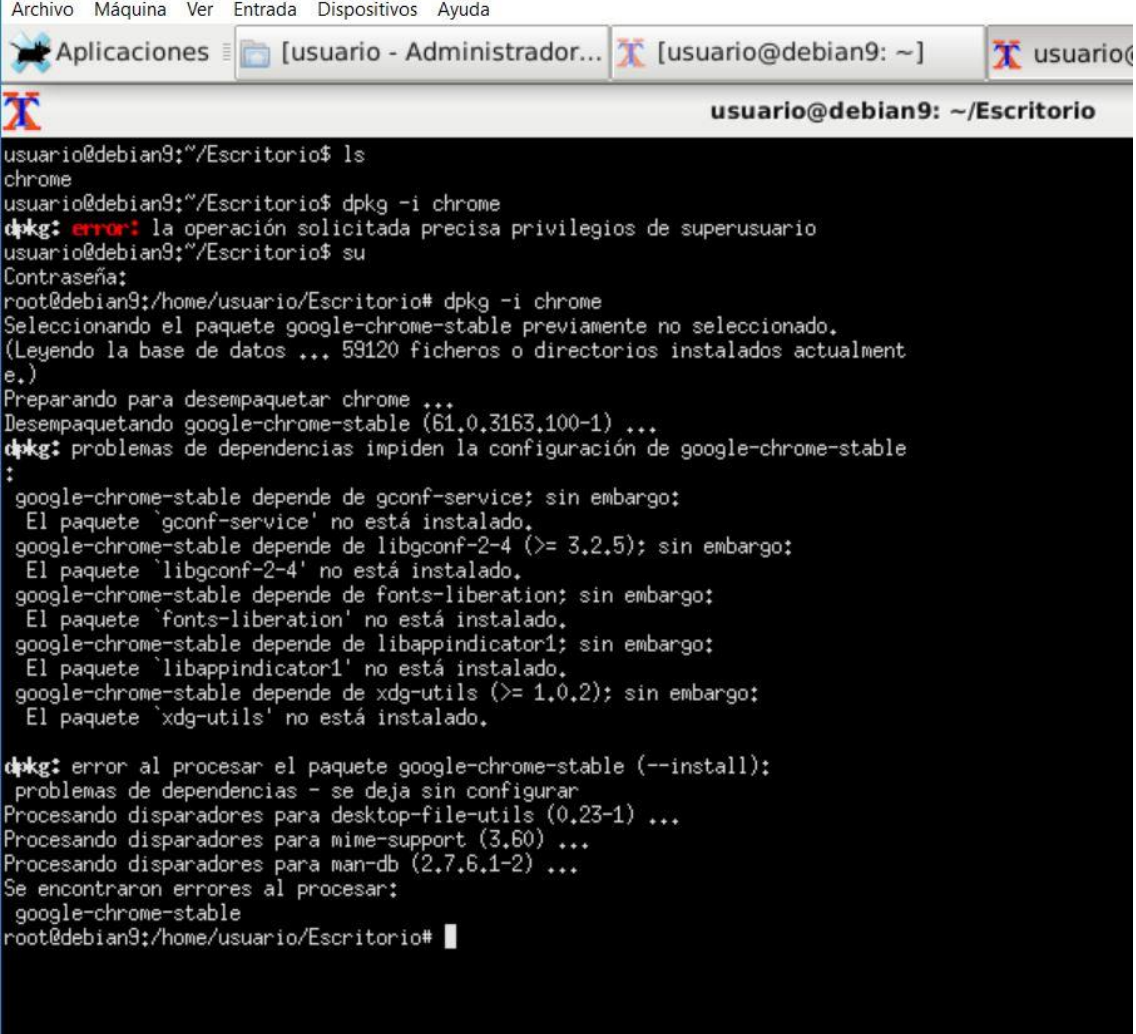
Directorio de C:\Documents and Settings\Administrador\Escritorio
07/10/2018  16:34    <DIR>          .
07/10/2018  16:34    <DIR>          ..
07/10/2018  16:34             65.267.668 google-chrome-stable_current_amd64.deb
                1 archivos             65.267.668 bytes
                2 dirs             1.361.670.144 bytes libres

C:\Documents and Settings\Administrador\Escritorio>pscp google-chrome-stable_cur
rent_amd64.deb usuario@10.0.2.50:/home/usuario
usuario@10.0.2.50's password:
google-chrome-stable_curr ! 63737 kB ! 7967.2 kB/s ! ETA: 00:00:00 ! 100%
C:\Documents and Settings\Administrador\Escritorio>

```

Si ejecuto el comando, efectivamente saltan directamente los problemas de dependencias, algo que apt automatizaba de modo previo al proceso de instalación. Para solucionarlo, instalando con el apt install debería funcionar. Sin embargo, no es así, ya que no encuentra el paquete.

Investigando, parece que hay una opción de apt install -f que corrige las dependencias ausentes en los paquetes desconfigurados. Así, si ejecuto estos dos comandos consecutivamente sí que instala Chrome perfectamente:



```
usuario@debian9:~/Escritorio$ ls
chrome
usuario@debian9:~/Escritorio$ dpkg -i chrome
dpkg: error: la operación solicitada precisa privilegios de superusuario
usuario@debian9:~/Escritorio$ su
Contraseña:
root@debian9:/home/usuario/Escritorio# dpkg -i chrome
Seleccionando el paquete google-chrome-stable previamente no seleccionado.
(Leyendo la base de datos ... 53120 ficheros o directorios instalados actualment
e.)
Preparando para desempaquetar chrome ...
Desempaquetando google-chrome-stable (61.0.3163.100-1) ...
dpkg: problemas de dependencias impiden la configuración de google-chrome-stable
:
google-chrome-stable depende de gconf-service; sin embargo:
El paquete 'gconf-service' no está instalado.
google-chrome-stable depende de libgconf-2-4 (>= 3.2.5); sin embargo:
El paquete 'libgconf-2-4' no está instalado.
google-chrome-stable depende de fonts-liberation; sin embargo:
El paquete 'fonts-liberation' no está instalado.
google-chrome-stable depende de libappindicator1; sin embargo:
El paquete 'libappindicator1' no está instalado.
google-chrome-stable depende de xdg-utils (>= 1.0.2); sin embargo:
El paquete 'xdg-utils' no está instalado.

dpkg: error al procesar el paquete google-chrome-stable (--install):
problemas de dependencias - se deja sin configurar
Procesando disparadores para desktop-file-utils (0.23-1) ...
Procesando disparadores para mime-support (3.60) ...
Procesando disparadores para man-db (2.7.6.1-2) ...
Se encontraron errores al procesar:
 google-chrome-stable
root@debian9:/home/usuario/Escritorio#
```

- dpkg -i google-chrome-stable\_current\_amd64.deb
- apt-get install -f

Duda: ¿por qué no ha funcionado simplemente con el apt-get install + nombre del archivo .deb si supuestamente añade las dependencias necesarias de forma automática?

## **Servidor SSH**

*SSH es un sistema que permite la administración remota de cualquier servidor basado en linux y que esté correctamente configurado y asegurado.*

*El servidor que vamos a elegir en este caso es openssh y su instalación es muy sencilla. Basta con escribir `apt install openssh-server`. Los ficheros de configuración se encuentran en `/etc/ssh`. El correspondiente al servidor es el archivo `sshd_config`.*

### **Caso práctico 3**

*Contesta a las siguientes preguntas relacionadas con el servidor SSH:*

- 1. ¿Qué paquetes tenemos que instalar para poner en funcionamiento el servidor ssh?*

El paquete que tenemos que instalar para establecer el servidor ssh es el mencionado anteriormente, el openssh mediante `apt install openssh-server`.

- 2. ¿De qué manera podemos hacer para iniciar-reiniciar-parar el servicio?*

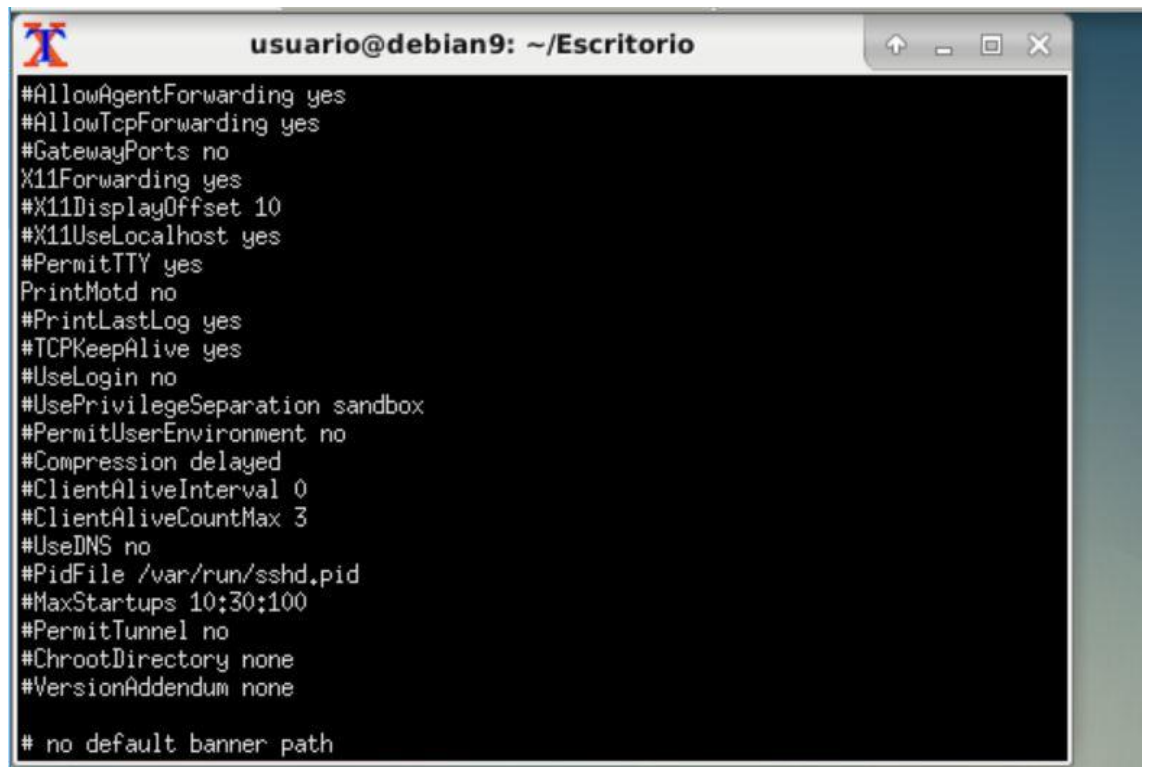
Hay varias maneras. Entre ellas podemos encontrar estas:

- `sudo service ssh start`
- `sudo systemctl start ssh`
- `sudo service ssh stop`
- `sudo systemctl stop ssh`
- `sudo systemctl restart ssh`

- 3. ¿Dónde se encuentra el archivo de configuración del servidor? ¿Qué apartados tiene?*

El archivo de configuración se encuentra en la carpeta `/etc/ssh` y se llama `sshd_config`. Si le hacemos un `cat`, encontramos diferentes categorías.

Aparecen líneas, cada una con su aplicación correspondiente, y con el símbolo `#` indicando comentario. Encontramos información relativa a puertos, direcciones, hostkeys, cifrado, logging, autenticación. En resumen, bastantes parámetros relacionados con la configuración del servidor ssh. Llegando al final del archivo encontramos una porción donde se pueden activar/desactivar bastantes opciones.



```
usuario@debian9: ~/Escritorio
#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#UsePrivilegeSeparation sandbox
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
```

4. *El servicio SSH por defecto escucha en el puerto 22. Modifícalo para que arranque en el puerto 10022 (u otro) y averigua cuál sería el comando utilizado para poder acceder al servidor. (ayuda: man ssh).*

Para modificar el puerto de acceso, podemos hacerlo precisamente en el archivo `sshd_config`. En la línea `Port`, cambiamos a 10022 y descomentamos borrando almohadilla.

La sintaxis para conectarnos desde otro equipo a este servidor ssh es `USERNAME@IP`. En mi caso, usé la máquina Debian 8 con servicio ssh también para conectarme a la Debian 9 tecleando [usuario@10.0.2.50](#).

Luego pude consultar si los cambios en el puerto habían surtido su efecto. Efectivamente, si corremos el comando `service ssh status` veremos el puerto en el que se encuentra recibiendo el servicio.

5. *¿Cómo se establece una conexión sencilla al servidor?*

Para establecer una conexión sencilla tenemos que seguir la sintaxis:  
`ssh USERNAME@SSHSERVERIP`

6. *Averigua cómo impedimos el acceso a un determinado usuario.*

#### **Permitir acceso**

Nos tenemos que ir al archivo de configuración `/etc/ssh/sshd_config` y añadir la siguiente línea:

`AllowUsers + nombre usuario`

Con esto le daremos vía libre al usuario para que acceda al servicio ssh. Se podrán introducir tantos nombres como se desee separados por espacios. También se puede permitir directamente el acceso a grupos. Por ejemplo:

AllowUsers root

#### **Denegar acceso**

Se añade la siguiente línea:

DenyUsers + nombre usuario

Sigue la misma mecánica que en el caso anterior.

En los dos supuestos, tras la operación será necesario reiniciar el servicio mediante:

sudo systemctl restart sshd

Se apunta, además, en muchas de las páginas consultadas, que permitir el acceso ssh al grupo root está considerado como mala práctica desde el punto de vista de la seguridad. Para ello, encontrarás una línea que reza PermitRootLogin. Lo único que tendrás que hacer es agregarle el valor no.

#### **7. ¿Qué programa usaremos para establecer conexiones por SSH desde máquinas Windows? ¿Cómo se realiza una conexión?**

Según las fuentes consultadas, en Windows no existe una herramienta o comando nativo que permita la conexión con un servidor ssh. Esta conexión se tendrá que hacer a través de algún software diseñado para este propósito. El más típico es el que hemos visto en clase: Putty.

El establecimiento de la conexión es bastante sencillo: en la interfaz inicial de Putty, en la pestaña de sesión te pedirá la ip del servidor al que quieres conectarte y el puerto. Habrá que seleccionar la opción ssh, evidentemente. Una vez rellenados los campos, abrimos la conexión y nos pedirá las credenciales de acceso al servidor.

#### **Acceso al servidor sin claves.**

*Para acceder al servidor tenemos que autenticarnos indicando un nombre de usuario con su contraseña que esté registrado en el equipo remoto servidor. Por ejemplo, para conectarnos al servidor 192.168.1.6 usando el usuario ssh\_user tenemos que teclear la siguiente orden: ssh ssh\_user@192.168.1.6*



*Al teclear esto, se nos pedirá la contraseña correspondiente a ese usuario. Si todo es correcto el servidor nos dará paso al servidor y podremos actuar según el nivel administrativo que tenga ese usuario en concreto.*

*Si esto lo tenemos que hacer repetidas veces, puede ser un poco engorroso tener que teclear cada vez que hacemos conexión la contraseña.*

*Podemos simplificar este proceso se puede hacer un sistema mediante el cual la conexión sea directa sin tener que escribir contraseñas. Esto es bueno por ejemplo, si queremos hacer copias de seguridad automatizadas a través de ssh usando el comando scp.*

*Seguiremos los siguientes pasos:*

*Para este ejercicio necesitaremos dos máquinas. Una hará de cliente y otra de servidor.*

*Además en la máquina cliente habrá un usuario creado p.e usuario\_cliente y en el servidor otro llamado p.e usuario\_servidor.*

Las máquinas empleadas para llevar a cabo la práctica serán la Debian 8 y la Debian 9, siendo la primera cliente y la segunda el servidor ssh.

Se crean los siguientes usuarios:

#### **Usuario cliente**

Nombre: ivanms-client

Password: ivanms-client

#### **Usuario servidor**

Nombre: ivanms-server

Password: ivanms-server

Los comandos usados serán:

- `sudo useradd + nombre usuario`
- `sudo passwd + password usuario`

Para ver el listado de usuarios en el sistema:

- `compgen -u`

Comprobar usuario actual:

- `whoami`

Cambiar de usuario:

- `su + nombre usuario`

Borrar usuario:

- `userdel + nombre usuario`

Cerrar sesión:

- `exit`

Se observa una peculiaridad en la creación de usuarios. Estos piden sus contraseñas respectivas en el login, excepto cuando el cambio se hace partiendo de root, donde no la pide.

*Desde el equipo cliente, con el usuario\_cliente ya creado y estando iniciada sid\_u sesión ejecutamos el comando: ssh-keygen -t rsa.*

En nuestro caso concreto, la máquina cliente tendrá el usuario ivanms-client iniciado.

*Este comando crea un par de claves pública-privada id\_rsa (clave privada que debe conservar el usuario en el directorio .ssh de su home e id\_rsa.pub que debemos pasar al directorio home del usuario al cual nos queremos conectar remotamente).*

Aquí te puedes encontrar con un problema bastante molesto y este es, que, a la hora de ejecutar el comando ssh-keygen, el sistema no permitía almacenar las claves por falta de permisos. Además, otro factor que había que tener en cuenta era la existencia del directorio .ssh/id\_rsa/ y del archivo en el que alojar las claves. Si este no existía, también arrojaba problemas.

La solución que encontré fue la siguiente: aplicar el comando chown -R ivanms-client /home/usuario desde root para así dar permisos de escritura al usuario ivams-client. A continuación, creé el directorio correspondiente, así como el archivo con extensión .txt para guardar las claves. Al ejecutar, lo guardó correctamente.

#### **Paso de la clave pública al servidor.**

A la hora de ver de nuevo dónde se encontraban almacenadas las claves privada y pública, surge un nuevo problema relacionado con el directorio .ssh. Este no se encuentra al ejecutar el comando ls. Esto es, según la información consultada en internet, porque los archivos que comienzan por "." son considerados ocultos en Linux. Para verlos, basta con ejecutar ls -a.

*Se trata ahora de pasar la clave al servidor e incluirla en el directorio .ssh que está dentro del home del usuario al que nos queremos conectar en el servidor (usuario\_remoto).*

*Para realizar este paso desde el cliente usamos el comando ssh-copy-id -i /home/usuario\_cliente/.ssh/id\_rsa.pub <nombre\_de\_usuario\_en\_el\_servidor@ip\_servidor>.*

Ejecutamos, por tanto, este comando:

```
ssh-copy-id -i keypair.txt.pub ivanms-server@10.0.2.50
```

El resultado no es el deseado, diciendo que no puede llevar a la cabo debido a que el directorio no existe. Pruebo de este modo:

```
ssh-copy-id -i keypair.txt.pub ivanms-server@10.0.2.50:/home/usuario/.ssh
```

Aparece un nuevo error. En este caso, dice que ha habido un fallo a la hora de abrir el id del archivo y que este no existe. Puede que sea por la configuración del puerto, que recordemos habíamos cambiado de 22 a 10022.

Intento esto:

```
ssh-copy-id -i keypair.txt.pub ivanms-server@10.0.2.50:/home/usuario/.ssh -p 10022
```

Sin éxito. Cambio los permisos de escritura, lectura y ejecución en la máquina 10.0.2.50, por si tuviera algo que ver. Sin éxito.

Tras buscar en internet sin encontrar la solución, decido ver qué puedo hacer por el mensaje que aparece. Habla de un directorio que falta: 'home/ivanms-client'. Decido cambiar los permisos y crearlo. Funciona. Da problemas con el directorio de destino, que también hay que crearlo: 'home/ivanms-server'. Esto último en la máquina servidor. El comando resultante sería: `ssh-copy-id -p 10022 -i /home/usuario/.ssh/id_rsa/keypair.txt.pub ivanms-server@10.0.2.50`

A continuación, pide la contraseña del servidor. Se introduce y la consola imprime un mensaje aconsejándote que te asegures de que la máquina funciona bien sin requerimiento de contraseñas en futuras conexiones.

Sin embargo, a la hora de conectar, no solo no lo hace de forma automáticamente, sino que al reiniciar la máquina Debian 9, ninguna contraseña funciona correctamente. Por lo que, ahora, toca restablecer de alguna manera todas las contraseñas y ver qué ha ocurrido.

Empiezo por reestablecer root iniciando el sistema en modo recovery. Pero la reestablezco correctamente y al introducirla, vuelve a fallar.

Vuelvo a acceder a la interfaz grub de inicio y pulso `-e`. Se cambia la línea que comienza por Linux y se introducen los valores `rw init=/bin/bash` para, justo después, reiniciar y reestablecer la contraseña. Se hace y ahora funciona en el login, pero no durante la ejecución de la terminal. Es decir, al principio sí la acepta, pero si cambias de usuario más tarde te la da como errónea. Aun así, compruebo la conexión ssh y tampoco funciona porque sigue pidiendo contraseña. Ejecuto el comando de nuevo, a ver si es que hubiera puesto algo mal. Sale lo mismo.

Intento entrar directamente.

Ninguna de las correcciones surte efecto. Tiro las dos máquinas y empiezo de nuevo.

### **Desde cero**

Creo los dos usuarios: el de cliente y el de servidor.

Vuelvo a ejecutar el comando para la creación de la clave pública privada:

```
ssh-keygen -t rsa
```

Esto te pide que introduzcas el archivo en el cual guardar las claves y que introduzcas también las passphrases. El error que da es por falta de permisos. Será, probablemente, porque el usuario ivanms-client no tiene permisos de escritura. Vamos a solucionar esto primero.

Para esto puedo crear un archivo .txt en /home/usuario desde root y modificarle los permisos haciéndole un `chmod 777`. Una vez hecho esto, ejecuto de nuevo el comando. Obtengo otro resultado: ahora dice que sí ha guardado mi identificación en el archivo keypair.txt (el que yo he creado). Sin embargo, dice que no ha podido guardar la clave pública en

/home/usuario/keypair.txt.pub. Se me ocurre que si creo yo el archivo .pub pudiera funcionar. Ejecuto la misma operación que antes desde root y, efectivamente, consigo guardar las claves. Ahora tenemos en el directorio /home/usuario las dos claves: la .txt la .pub. La .txt es la que debemos conservar y la .pub la que debemos mover al servidor de destino, según el enunciado de la práctica. Habrá, por tanto, que colocar esas dos claves en el directorio .ssh. Pero el usuario no tiene permisos para acceder a dicho directorio, por lo que habrá que modificar este aspecto para evitar futuros problemas. Lo podemos intentar, pero da error por falta de permisos. Volviendo atrás, va a ser mejor idea arreglar primero el tema de los permisos y luego realizar la creación de las claves.

Si el usuario en cuestión va a tener que realizar operaciones de escritura en la carpeta /home/usuario, lo más lógico es cambiar el propietario de ese directorio. Para ello, usamos chown. Escribiendo man chown vemos sus características: nos interesa la opción -R para la modificación recursiva. Así que escribo esto:

```
chown -R ivanms-client /home/usuario
```

Vamos a hacer la generación de las claves para ver cómo responde:

```
ssh-keygen -t rsa
```

Nos pide un archivo en el que volcar la información. Lo creo en el directorio .ssh. Las dos se crean correctamente.

Volviendo a lo anterior: se han creado dos claves, una pública y otra privada. La pública debe permanecer en el directorio /home/usuario/.ssh del usuario cliente y la pública es la que debe copiarse al servidor de destino. Se ejecuta entonces esto:

```
ssh-copy-id -p 10022 -i /home/usuario/.ssh/keypair.txt.pub ivanms-server@10.0.2.50
```

La respuesta es un error. Dice que no existe el directorio /home/ivanms-client/.ssh. Lo creo, a ver qué pasa. No me lo permite. Los permisos llegaban solo hasta /home/usuario. Cambio también los de /home. Creo el directorio y me deja. Ahora da otro problema: que no se puede crear el directorio .ssh y que no existe el directorio /home/ivanms-server en la máquina de destino. Modifico entonces los permisos para que pueda escribir y creo el directorio home/ivanms-server en la máquina servidor.

Tras todos los cambios, sigue sin funcionar: el servidor ssh sigue pidiendo la clave. Reinicio.

Sigue con fallos.

Ante esto, decido repetir el proceso ssh-keygen con la plantilla por defecto propuesta por la consola (.ssh/id\_rsa). El resultado ahora es distinto. Copio las claves al servidor, pero sigue fallando de algún modo, ya que no pide la contraseña pero sí la passphrase.

Repito el proceso omitiendo la passphrase. Funciona. Por fin.

Se han modificado, aparte, por recomendaciones vistas en internet, los siguientes permisos:

- chmod 644 para la clave pública
- chmod 600 para la clave privada
- chmod 700 para el directorio .ssh

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ivanms-client/.ssh/id_rsa): /home/usuario/keypair.txt
/home/usuario/keypair.txt already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usuario/keypair.txt.
Your public key has been saved in /home/usuario/keypair.txt.pub.
The key fingerprint is:
34:35:d9:0d:2f:7e:5b:02:d2:ae:f1:fa:36:67:4c:1f ivanms-client@debian8
The key's randomart image is:
+---[RSA 2048]---+
|                |
|      .o.o       |
|     ..o...      |
|    o . + .      |
|   . . + o       |
|  S . o o .      |
|   + ..E         |
|   ..o...        |
|   .o + .        |
|   .o.+          |
+-----+
$
```

```
ished.
ECDSA key fingerprint is a1:ab:67:05:9e:09:34:6f:40:72:93:05:2d:30:05:c1.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
ivanms-server@10.0.2.50's password:
Could not chdir to home directory /home/ivanms-server: No such file or directory
mkdir: no se puede crear el directorio «.ssh»: Permiso denegado
$ ssh-copy-id -p 10022 -i /home/usuario/.ssh/keypair.txt.pub ivanms-server@10.0.
2.50
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
ivanms-server@10.0.2.50's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh -p '10022' 'ivanms-server@10.0.2.
50'"
and check to make sure that only the key(s) you wanted were added.
$
```

*Este comando preparará todo lo necesario para que la conexión funcione. (Básicamente se trata de escribir la clave pública en el fichero `authorized_keys`)*

*Si todo ha ido bien, podemos probar la conexión desde el cliente al servidor escribiendo simplemente `ssh usuario_servidor@ip_servidor`. No se nos pedirá claves.*

*Para más información y detalles técnicos puedes visitar este enlace:*

*<http://nereida.deioc.ull.es/~pp2/perlexamples/node28.html>*

## Ejercicio práctico SSH / Script

*Programa un script que permita hacer una copia de seguridad automatizada del directorio /home/usuario/Documentos, comprimiéndolo con tar.gz. El script se ejecutará los domingos a las 00.00 horas. Una vez que se comprima el directorio se mandará por scp sin claves a un servidor de copias de seguridad.*

En primer lugar, habrá que mirar cómo comprimir los archivos del directorio. En mi caso, y siguiendo con lo anterior, tomaré como directorio a comprimir el siguiente:

/home/ivanms-client/

Para comprimir una serie de archivos y dotarles de la extensión .tar.gz, recopilamos primero los siguientes comandos:

### Archivos .tar.gz:

**Comprimir:** tar -czvf empaquetado.tar.gz /carpeta/a/empaquetar/

**Descomprimir:** tar -xzf archivo.tar.gz

### Archivos .tar:

**Empaquetar:** tar -cvf paquete.tar /dir/a/comprimir/

**Desempaquetar:** tar -xvf paquete.tar

### Archivos .gz:

**Comprimir:** gzip -9 index.php

**Descomprimir:** gzip -d index.php.gz

### Archivos .zip:

**Comprimir:** zip archivo.zip carpeta

**Descomprimir:** unzip archivo.zip

En mi caso, yo quiero comprimir todo el contenido de /home/ivanms-client. Lo guardaré en /home/ivanms-client/recovery. Para ello, ejecuto:

```
tar -czvf /home/ivanms-client/recovery/rpack.tar.gz /home/ivanms-client
```

Esto crea el paquete con la copia de seguridad. Ahora tenemos que ejecutar el comando pertinente para mandar la información al servidor de copias de seguridad. Usamos el comando scp:

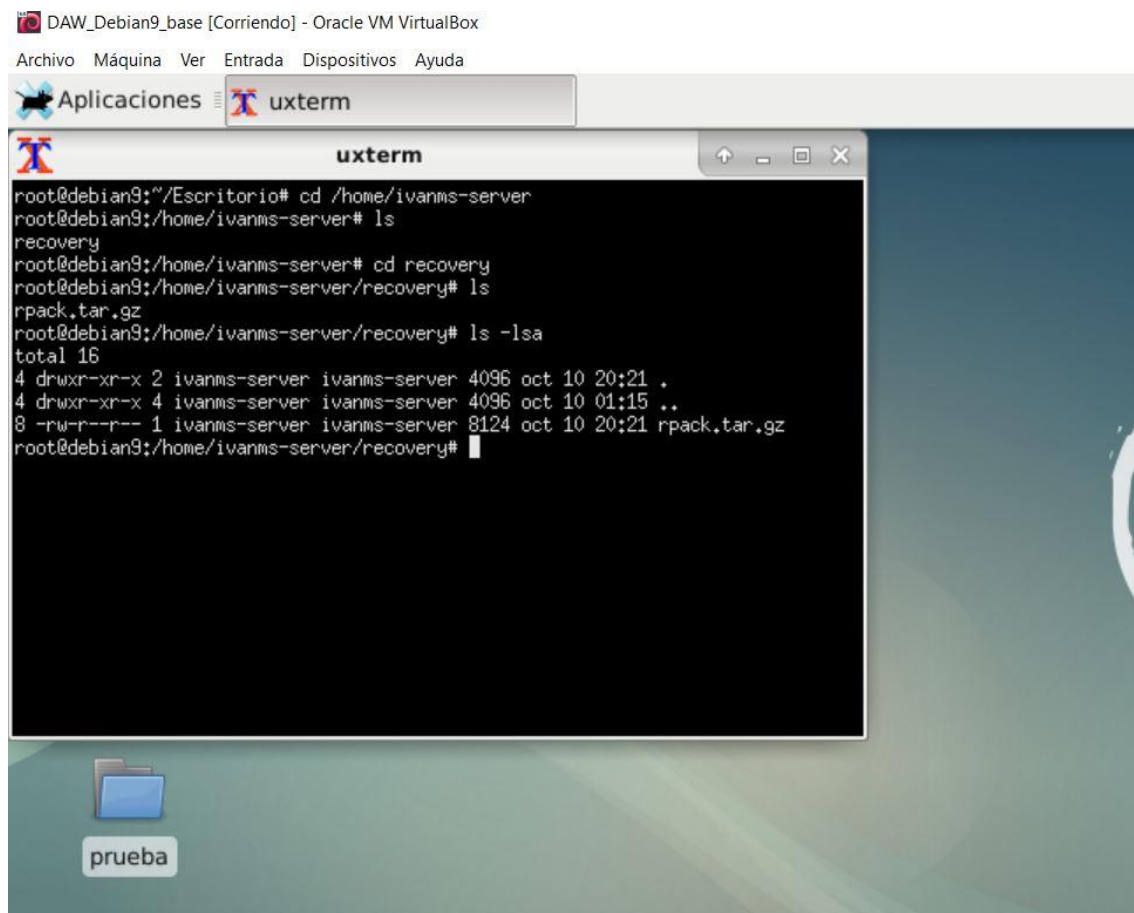
```
scp -P 10022 /home/ivanms-client/recovery/rpack.tar.gz ivanms-server@10.0.2.50:/home/ivanms-server/recovery
```

Esto lo tenemos que meter en un script que colocaremos en una carpeta llamada script colgando de /ivanms-client. El archivo se llamará copyexe.sh. Una vez lo hemos creado, solo tenemos que modificarlo para que se ejecute cada domingo a las 00:00. Le hacemos un `chmod 777 copyexe.sh` para poder ejecutarlo luego.

Para generar tareas periódicas, existen varias opciones en Linux. La más apropiada parece `crontab`, que permite realizar trabajos repetitivos a lo largo del tiempo. A la hora de probarlo, surge un primer problema: la salida `stdout` no aparece por ninguna parte. Tras mucha búsqueda, encuentro por internet que los daemons (el `cron` en este caso que trabaja en segundo plano para el correcto funcionamiento del comando `crontab`) no disponen de la salida convencional `stdout` y que, por tanto, no pueden imprimir mensajes. La única opción que se podría plantear sería la de redireccionar la información dentro del propio script a un `log.txt` externo.

Ahora vamos con el funcionamiento de `crontab`. Leyendo en la información que proporciona `man crontab`, la opción `crontab -e` permite la edición de las tareas relativas al usuario actual. Nosotros queremos que sea el usuario que tenga la clave privada establecida con el servidor de destino, por lo que la tarea deberá ser creada en dicha cuenta. La línea será más o menos así:

```
00 00 * * 7 /home/ivanms-client/script/copyexe.sh >> /home/ivanms-client/script/log.txt
```





La última parte es la redirección. El principio indica minutos/horas/días del mes/mes/días de la semana. Tras asignarle fecha y hora a la ejecución de la tarea, se comprueba y esta ejecuta la función correctamente. Para hacer las modificaciones pertinentes en el directorio `/var/spool/cron/crontabs/ivanms-client`, tienes dos modos: desde root o desde el propio usuario. En el primer caso, se aplica la opción `-u` del comando:

```
crontab -eu ivanms-client
```

Pero surge una última cuestión: ¿qué pasa si no estoy logueado como el usuario que nos interesa para realizar la transferencia programada de archivos? Pues nada. Gracias a dios, de esta parte se encarga Linux, puesto que el sistema de crontab separa las tareas asignadas a cada uno de los usuarios, de manera que estos archivos son independientes entre ellos. Se hace la prueba y el archivo llega correctamente a la fecha y hora estipulada en el servidor de destino, a pesar de estar logueado como root.

Dudas:

- ¿Cuáles son las máquinas que debo tener en modo bridge? ¿XP y Debian 9?
- ¿Creo yo el archivo interfaces? ¿Cómo apago una interfaz de red cableada?
- Significado de las líneas `auto lo / iface lo inet loopback / allow-hotplug enp0s3`.
- ¿Qué tienen en común los directorios acabados en `.d` en Linux? Se ven bastante.