



ugr

Universidad  
de Granada

SERVIDORES WEB DE ALTAS PRESTACIONES  
GRADO EN INGENIERÍA INFORMÁTICA

## Prácticas resueltas

Autor

Carlos Sánchez Páez



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

CURSO 2019-2020

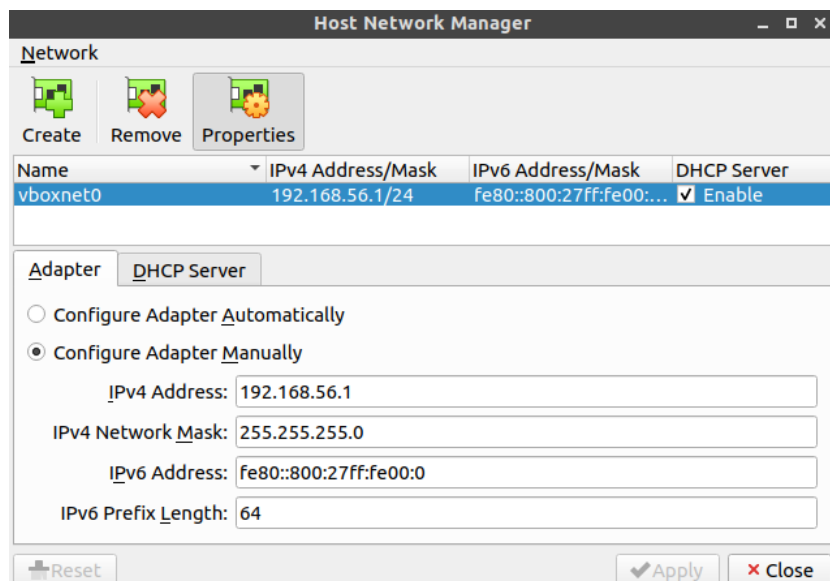
# Índice

# 1. Práctica 1

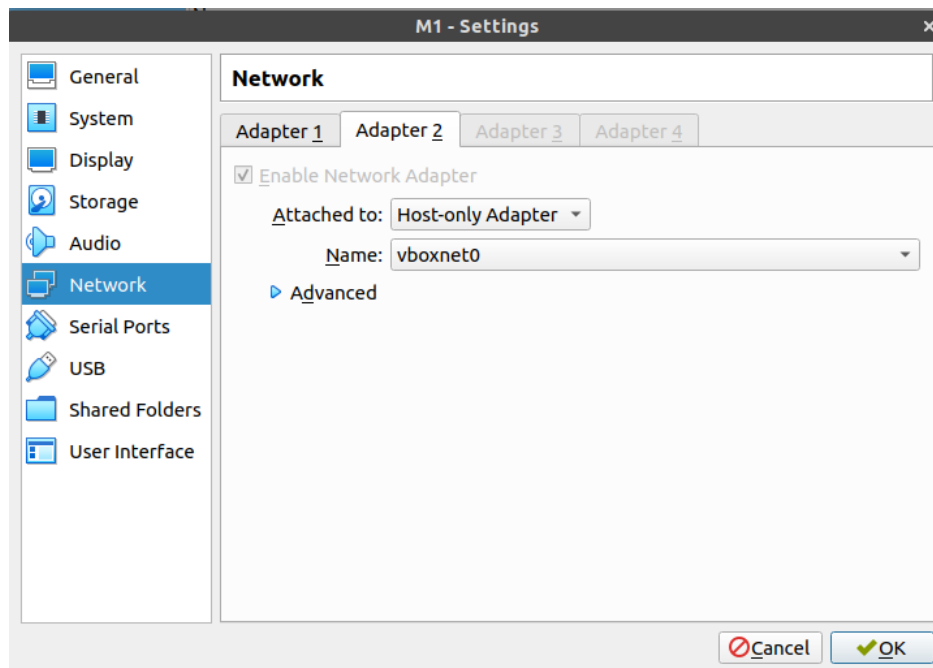
En esta práctica configuraremos dos máquinas virtuales (*M1* y *M2*). También crearemos una interfaz *host-only* para que las máquinas puedan conectarse entre ellas.

En esta guía configuraremos la interfaz sólo anfitrión manualmente en una máquina. En la otra dejaremos que el asistente de instalación lo haga por nosotros.

1. Comenzaremos creando las máquinas virtuales desde VirtualBox. Las proveeremos de al menos 512MB de RAM y 10GB de disco duro.
2. Descargamos la imagen ISO de Ubuntu Server 18.04 y la montamos en la unidad de disco de la primera máquina.
3. Arrancamos la máquina e iniciamos el asistente de instalación. Establecemos nuestro nombre de usuario de GitHub como *username* y *m1* como nombre del servidor. La clave será *Swap1234*
4. Cuando termine la instalación apagamos la máquina.
5. En VirtualBox abrimos el administrador de redes sólo anfitrión (dentro del menú Archivo).
6. Creamos un adaptador y activamos DHCP para que asigne una IP a nuestra máquina.



7. Vamos a los ajustes de red de la máquina y "conectamos" el adaptador que acabamos de crear.



8. Arrancamos la máquina. Ejecutamos el siguiente comando para ver las interfaces conectadas:

```
m1> sudo ifconfig -a
```

```
csp98@m2:~$ sudo ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe7a:4726 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:7a:47:26 txqueuelen 1000 (Ethernet)
    RX packets 33 bytes 10975 (10.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 40 bytes 4642 (4.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 08:00:27:4b:e5:bc txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    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 172 bytes 15276 (15.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 172 bytes 15276 (15.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

csp98@m2:~$ _
```

9. Vemos que tenemos una nueva interfaz (*enp0s8*) pero no tiene IP asignada. Para arreglar esto usaremos *netplan*.
10. Creamos un archivo de configuración para ella:

```
m1> sudo nano /etc/netplan/host-only.yaml
```

11. Introducimos este contenido en el archivo. Debemos usar espacios en vez de tabuladores.

```

network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      dhcp4: true

```

12. Guardamos los cambios y los aplicamos con el comando

```
m1> sudo netplan apply
```

13. Ejecutamos *sudo ifconfig -a* y comprobamos que ya tenemos IP asignada:

```

csp98@m2:~$ sudo ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe7a:4726  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:7a:47:26  txqueuelen 1000  (Ethernet)
    RX packets 48  bytes 13110 (13.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 63  bytes 6831 (6.8 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.56.103  netmask 255.255.255.0  broadcast 192.168.56.255
    inet6 fe80::a00:27ff:fe4b:e5bc  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:4b:e5:bc  txqueuelen 1000  (Ethernet)
    RX packets 2  bytes 1180 (1.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 13  bytes 1530 (1.5 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 172  bytes 15276 (15.2 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 172  bytes 15276 (15.2 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

csp98@m2:~$ _

```

14. Instalamos la pila LAMP:

```

m1> sudo apt install -y openssh-server openssh-client
↪ apache2 mysql-server mysql-client

```

15. Pasamos ahora a la configuración de *m2*. Para ello seguimos los pasos anteriores. Como el adaptador sólo anfitrión ya está creado, el asistente de instalación lo configurará automáticamente. Lo único que tenemos que hacer es "conectarlo" a la máquina como hicimos antes.
16. Cuando termine la instalación, ejecutamos el comando anterior para instalar la pila LAMP.
17. Probamos la conexión SSH conectando las máquinas entre sí. Para facilitar las conexiones podemos guardar las IPs en variables:

```
M1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

command 'lconfig' from deb ipmitool
command 'lconfig' from deb wireless-tools

Try: sudo apt install <deb name>

csp98@m1:~$ ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe90:bc8b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:90:bc:8b txqueuelen 1000 (Ethernet)
    RX packets 24752 bytes 36303433 (36.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2341 bytes 154241 (154.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.100 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::a00:27ff:fe28:9232 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:28:92:32 txqueuelen 1000 (Ethernet)
    RX packets 73 bytes 15087 (15.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 71 bytes 8695 (8.6 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 352 bytes 31248 (31.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 352 bytes 31248 (31.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

csp98@m1:~$ echo "M2_IP=192.168.56.103" >> .bashrc ; source .bashrc
csp98@m1:~$ echo $M2_IP
192.168.56.103
csp98@m1:~$ _

M2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m2:~$ ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe7a:4726 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:7a:47:26 txqueuelen 1000 (Ethernet)
    RX packets 52 bytes 13410 (13.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 7201 (7.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.103 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::a00:27ff:fe4b:e5bc prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:4b:e5:bc txqueuelen 1000 (Ethernet)
    RX packets 2 bytes 1180 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1600 (1.6 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 172 bytes 15276 (15.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 172 bytes 15276 (15.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

csp98@m2:~$ echo "M1_IP=192.168.56.100" >> .bashrc ; source .bashrc
csp98@m2:~$ echo $M1_IP
192.168.56.100
csp98@m2:~$ _
```

```
M1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m1:~$ ssh $M2_IP
csp98@192.168.56.103's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

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

System information as of Thu Mar 12 16:26:12 UTC 2020

System load:  0.0          Processes:      93
Usage of /:   33.1% of 9.78GB  Users logged in: 1
Memory usage: 16%          IP address for enp0s3: 10.0.2.15
Swap usage:  0%            IP address for enp0s8: 192.168.56.103

 * Latest Kubernetes 1.18 beta is now available for your laptop, NUC, cloud
   instance or Raspberry Pi, with automatic updates to the final GA release.

   sudo snap install microk8s --channel=1.18/beta --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.

   https://multipass.run/

15 packages can be updated.
8 updates are security updates.

Last login: Thu Mar 12 16:13:02 2020
csp98@m2:~$

M2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m2:~$ ssh $M1_IP
The authenticity of host '192.168.56.100 (192.168.56.100)' can't be established.
ECDSA key fingerprint is SHA256:s8dN2AG0vTWbIEb67Tm9C9EMUTvzrh97Ji+Sig8fGM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.100' (ECDSA) to the list of known hosts.
csp98@192.168.56.100's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

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

System information as of Thu Mar 12 16:26:23 UTC 2020

System load:  0.08         Processes:      94
Usage of /:   33.0% of 9.78GB  Users logged in: 1
Memory usage: 16%          IP address for enp0s3: 10.0.2.15
Swap usage:  0%            IP address for enp0s8: 192.168.56.100

 * Latest Kubernetes 1.18 beta is now available for your laptop, NUC, cloud
   instance or Raspberry Pi, with automatic updates to the final GA release.

   sudo snap install microk8s --channel=1.18/beta --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.

   https://multipass.run/

14 packages can be updated.
4 updates are security updates.

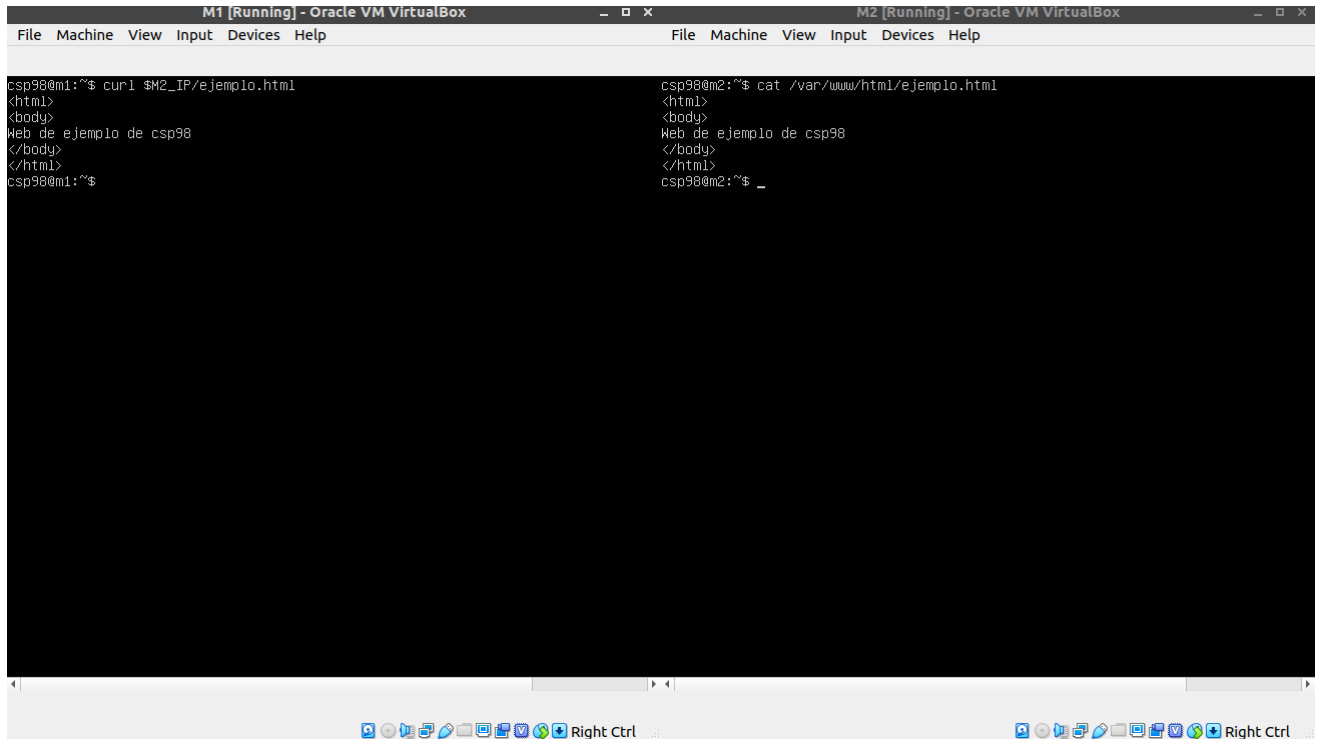
Last login: Thu Mar 12 15:52:08 2020
csp98@m1:~$ _
```

18. Creamos en *M2* un documento HTML (*/var/www/html/ejemplo.html*) con el siguiente contenido:

```
<HTML>
<BODY>
Web de ejemplo de <nombre usuario> para SWAP
</BODY>
```

</HTML>

19. Hacemos *curl* desde la otra máquina y comprobamos que recibimos la respuesta esperada:



```
M1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m1:~$ curl $M2_IP/ejemplo.html
<html>
<body>
Web de ejemplo de csp98
</body>
</html>
csp98@m1:~$

M2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m2:~$ cat /var/www/html/ejemplo.html
<html>
<body>
Web de ejemplo de csp98
</body>
</html>
csp98@m2:~$ _
```

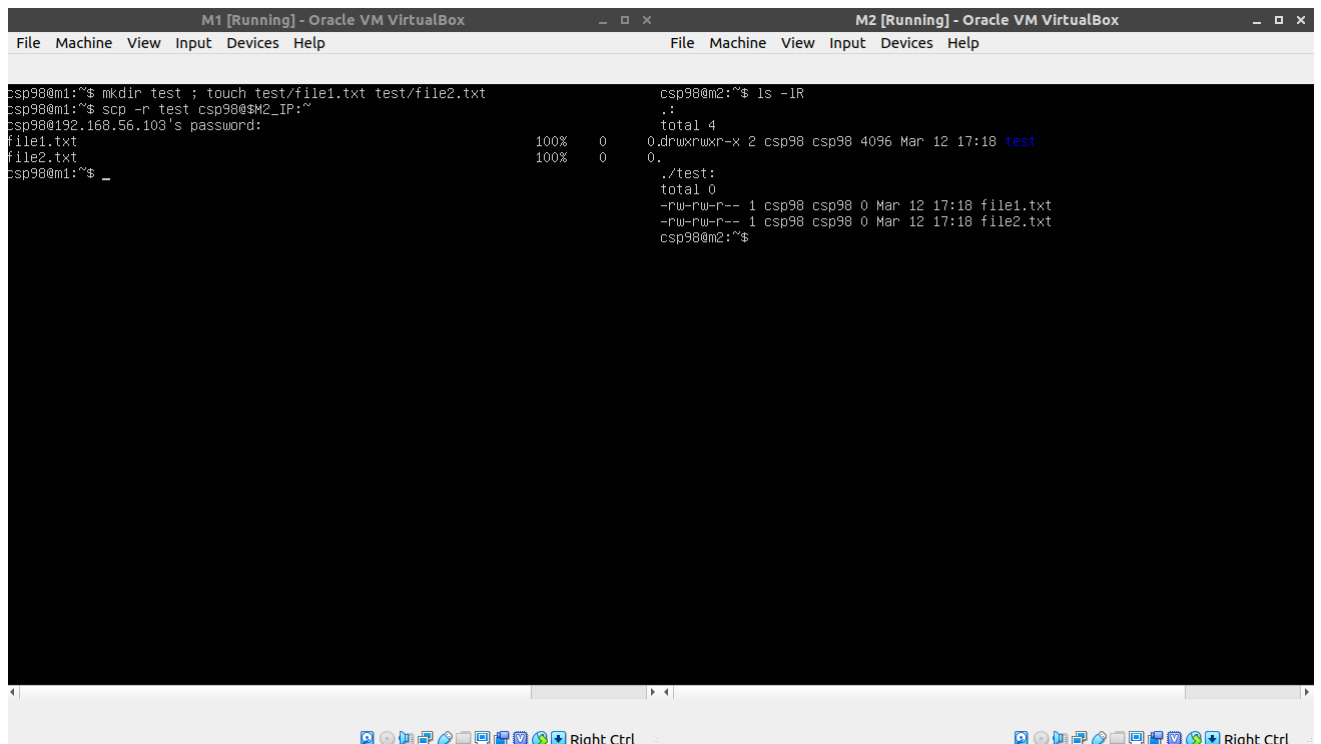
## 2. Práctica 2

En esta práctica clonaremos información entre las máquinas.

1. Comenzamos copiando un directorio de *m1* a la carpeta de usuario de *m2*:

```
m1 > mkdir test ; touch test/file1.txt test/file2.txt
m1 > scp -r test $M2_IP:~
```

Ejecutamos *ls -lR* en *M2* para ver el resultado:



2. Para no tener que introducir la contraseña en cada conexión SSH configuraremos la autenticación mediante clave público-privada RSA:

```
m2 > ssh-keygen -b 4096 -t rsa
```

Pulsamos Enter tres veces.

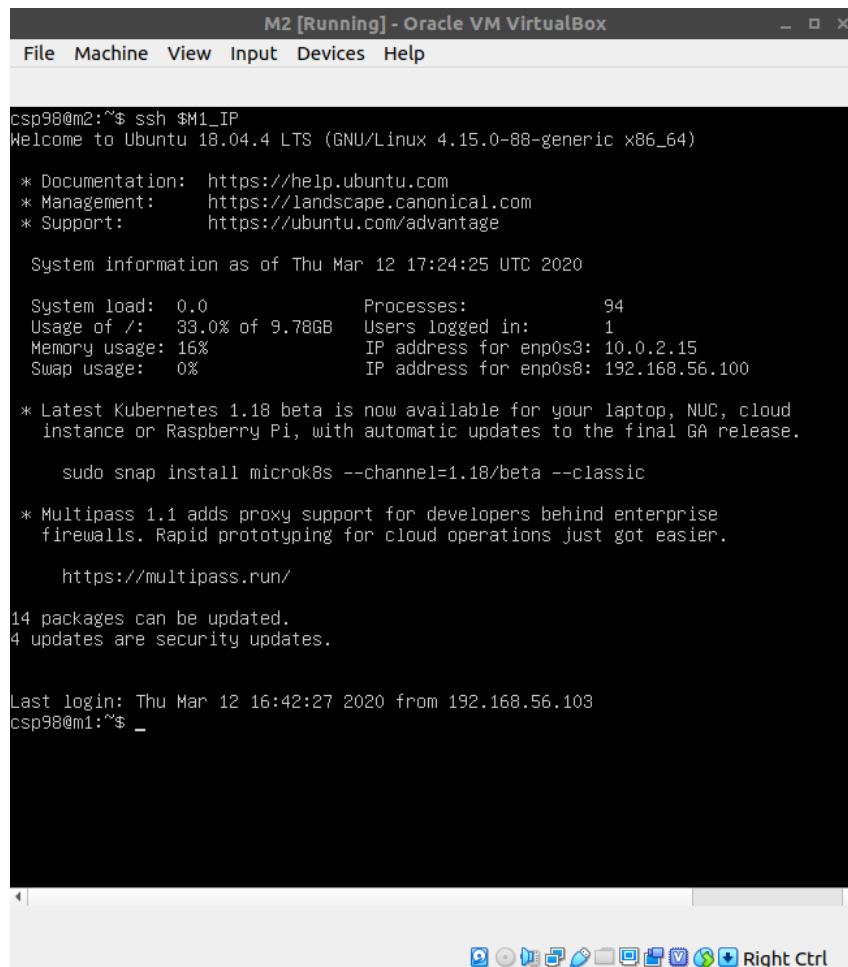
3. Copiamos la clave a M1:

```
m2 > ssh-copy-id $M1_IP
```

Introducimos la clave *Swap1234*

4. Probamos a realizar la conexión. Veremos que no nos pide la clave.





```
M2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

csp98@m2:~$ ssh $M1_IP
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

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

System information as of Thu Mar 12 17:24:25 UTC 2020

System load:  0.0               Processes:    94
Usage of /:   33.0% of 9.78GB   Users logged in: 1
Memory usage: 16%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.56.100

 * Latest Kubernetes 1.18 beta is now available for your laptop, NUC, cloud
   instance or Raspberry Pi, with automatic updates to the final GA release.

   sudo snap install microk8s --channel=1.18/beta --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.

   https://multipass.run/

14 packages can be updated.
4 updates are security updates.

Last login: Thu Mar 12 16:42:27 2020 from 192.168.56.103
csp98@m1:~$
```

5. Por último programamos una tarea de clonación que se ejecutará cada dos horas. De esta forma el contenido de M1 se replicará en M2 cada dos horas. Usaremos *crontab* para ello:
6. Lo primero que debemos hacer es configurar la clave público privada por ssh en M1 para que no se pida en cada copia (igual que en el paso anterior).
7. Después damos permisos a nuestro usuario para escribir en el directorio:

```
m2 > sudo chown $USER:$USER -R /var/www
```

8. Instalamos la herramienta *rsync* en M2:

```
m2 > sudo apt install -y rsync
```

9. Configuramos el clonado:

```
m2 > sudo nano /etc/crontab
```

Añadimos la siguiente línea:

```
00 */12 * * * csp98 rsync -avz -e ssh 192.168.56.100:/var/  
↪ www /var/
```

10. Si queremos probar el funcionamiento del comando podemos establecer una periodicidad menor, crear un archivo en M1 y ver que se replica en M2.

## 3. Práctica 3

En esta práctica configuraremos varios balanceadores de carga y los probaremos mediante tests de estrés.

### 3.1. Instalación de balanceadores

1. Comenzamos creando una nueva máquina (M3) e instalando Ubuntu Server. Le añadiremos también el adaptador *host-only*.
2. Almacenamos el alias de ambas IPs para ahorrar tiempo en futuros comandos:

```
m3 > echo "M1_IP=192.168.56.100" >> .bashrc ; source .bashrc
m3 > echo "M2_IP=192.168.56.103" >> .bashrc ; source .bashrc
```

3. Instalamos nginx y lo lanzamos:

```
m3 > sudo apt update ; sudo apt install nginx
m3 > sudo systemctl start nginx
```

4. Como solo queremos que funcione como balanceador de carga, desactivamos la funcionalidad de servidor web. Para ello editamos el archivo de configuración y comentamos la siguiente línea:

```
m3 > sudo nano /etc/nginx/nginx.conf
```

```
# include /etc/nginx/sites-enabled/*
```

5. Configuramos el *upstream*, máquinas entre las que se dividirá el tráfico:

```
m3 > sudo nano /etc/nginx/conf.d/default.conf
```

Insertamos el siguiente contenido:

```
upstream servidoresSWAP{
    server 192.168.56.100;
    server 192.168.56.103;
}

server{
    listen 80;
    server_name balanceador;
    access_log /var/log/nginx/balanceador.access.log;
    error_log /var/log/nginx/balanceador.error.log;
    root /var/www/;
    location /
    {
        proxy_pass http://servidoresSWAP;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```

        proxy_set_header X-Forwarded-For
            ↪ $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}

```

6. Reiniciamos el servicio para que los cambios surtan efecto:

```
m3 > sudo service nginx restart
```

7. Arrancamos M1 y M2. Para diferenciar a cual de ellas se envía la petición a través del balanceador de carga, modificaremos el archivo `/var/www/html/ejemplo.html` de cada una:

```

GNU nano 2.9.3 /var/www/html/ejemplo.html
<html>
<body>
<p>M2</p>
Web de ejemplo de csp98
</body>
</html>
csp98@m2:~$ _

GNU nano 2.9.3 /var/www/html/ejemplo.html
<html>
<body>
<p>M1</p>
Web de ejemplo de csp98
</body>
</html>
csp98@m1:~$ _

```

8. Consultamos la IP de M3 y entramos a `192.168.56.104/ejemplo.html` desde el navegador de nuestro host:

```
m3 > ip addr | grep 192
```

9. Recargamos varias veces la página y podremos ver como la petición la atiende un servidor u otro.
10. Probaremos ahora a repartir la carga de forma desigual. M1 atenderá el doble de peticiones que M2. Para ello editamos el siguiente archivo de configuración y añadimos la directiva *weight*:
11. Configuramos el *upstream*, máquinas entre las que se dividirá el tráfico:

```
m3 > sudo nano /etc/nginx/conf.d/default.conf
```

```
upstream servidoresSWAP{
    server 192.168.56.100 weight=2;
    server 192.168.56.103 weight=1;
}
```

12. Reiniciamos el servicio y probamos a acceder con el navegador a la IP anterior y refrescar la página.
13. También podemos especificar políticas para mantener la sesión:

- **Keep-alive:** las peticiones se enviarán al mismo servidor durante  $n$  segundos.

```
upstream servidoresSWAP{
    server 192.168.56.100;
    server 192.168.56.103;
    keepalive <n>;
}
```

- **IP-hash.** Una vez que un servidor atiende una IP, la atenderá siempre. No se recomienda su uso (no se balancea la carga de forma equitativa).

```
upstream servidoresSWAP{
    ip_hash;
    server 192.168.56.100;
    server 192.168.56.103;
}
```

14. Ahora pasamos a *haproxy*, otro balanceador. Comenzamos instalándolo:

```
m3 > sudo apt install -y haproxy
```

15. Configuramos el balanceador:

```
m3 > sudo nano /etc/haproxy/haproxy.cfg
```

Introducimos las siguientes líneas al final del archivo:

```
frontend http-in
    bind *:80
    default_backend servidoresSWAP
backend servidoresSWAP
    server m1 192.168.56.100:80 maxconn 32
    server m2 192.168.56.103:80 maxconn 32
```

16. Desactivamos *nginx* y reiniciamos *haproxy*:

```
m3 > sudo service nginx stop
m3 > sudo service haproxy restart
```

17. Comprobamos su funcionamiento:

```
~ > curl 192.168.56.104/ejemplo.html
<html>
<body>
<p>M1</p>
Web de ejemplo de csp98
</body>
</html>
~ > curl 192.168.56.104/ejemplo.html
<html>
<body>
<p>M2</p>
Web de ejemplo de csp98
</body>
</html>
~ >
```

18. Por último usaremos un balanceador más, *pen*. Comenzamos instalándolo:

```
m3 > sudo apt install -y pen
```

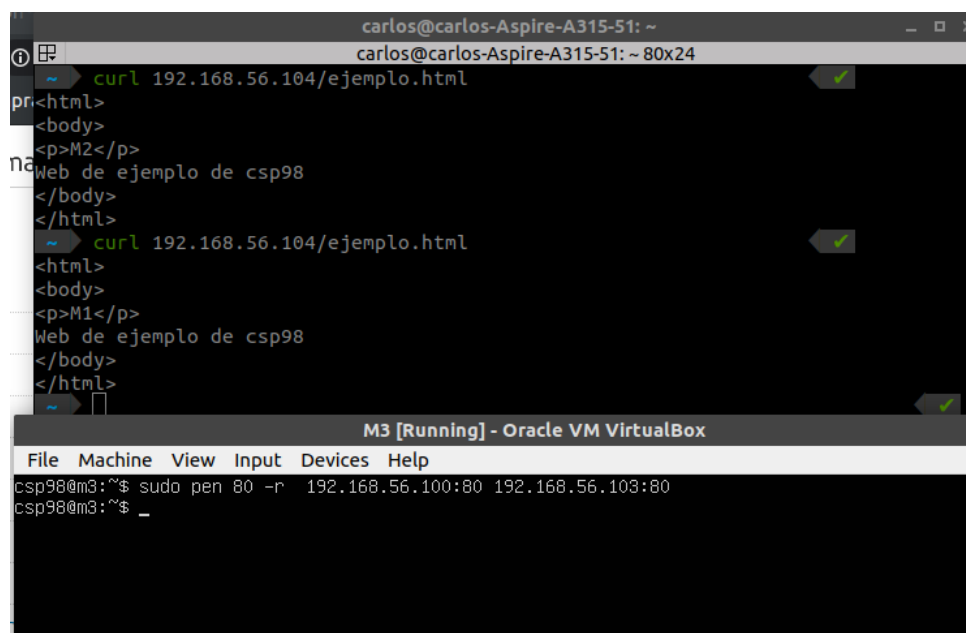
19. Paramos *haproxy*:

```
m3 > sudo service haproxy stop
```

20. Establecemos el balanceador con el algoritmo round-robin:

```
m3 > sudo pen 80 -r $M1_IP:80 $M2_IP:80
```

21. Comprobamos que funciona:



```
carlos@carlos-Aspire-A315-51: ~
carlos@carlos-Aspire-A315-51: ~ 80x24
~ > curl 192.168.56.104/ejemplo.html ✓
<html>
<body>
<p>M2</p>
Web de ejemplo de csp98
</body>
</html>
~ > curl 192.168.56.104/ejemplo.html ✓
<html>
<body>
<p>M1</p>
Web de ejemplo de csp98
</body>
</html>
~ >
```

```
M3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
csp98@m3:~$ sudo pen 80 -r 192.168.56.100:80 192.168.56.103:80
csp98@m3:~$ _
```

### 3.2. Test de rendimiento en balanceadores mediante *ab*

En esta sección someteremos los distintos balanceadores a una carga alta mediante *Apache Benchmark* y mediremos su rendimiento en función del tiempo de respuesta obtenido.

1. Comenzamos instalando la utilidad en nuestro host:

```
host > sudo apt install -y apache2-utils
```

2. Ejecutamos el siguiente test contra los distintos balanceadores:

```
host > ab -n 10000 -c 10 http://192.168.56.104/ejemplo.html
```

Los tiempos medios de respuesta obtenidos tras realizar tres ejecuciones y calcular la media aritmética fueron los siguientes:

| Balanceador    | Tiempo de respuesta por petición (ms) |
|----------------|---------------------------------------|
| <i>haproxy</i> | 5.054                                 |
| <i>nginx</i>   | 6.458                                 |
| <i>pen</i>     | 5.353                                 |

En este caso concluimos con que *haproxy* ofrece mejor rendimiento que *nginx* y *pen*.

## 4. Práctica 4

En esta práctica configuraremos la seguridad de nuestra granja mediante un certificado y un cortafuegos.

### 4.1. Creación del certificado autofirmado para acceder por HTTPS

1. Comenzamos activando el módulo SSL de Apache en M1:

```
m1 > sudo a2enmod ssl; sudo service apache2 restart
```

2. Creamos el certificado:

```
m1 > sudo mkdir /etc/apache2/ssl
m1 > sudo openssl req -x509 -nodes -days 365 -newkey rsa
➔ :2048 -keyout /etc/apache2/ssl/apache.key -out /etc/
➔ apache2/ssl/apache.crt
```

3. Configuramos los siguientes datos:

- **Country name:** ES.
- **Province name:** Granada.
- **Locality name:** Granada.

- **Organization name:** SWAP.
  - **Organizational unit name:** P4.
  - **Common name:** nuestro usuario UGR.
  - **Email address:** nuestro email UGR.
4. Ahora utilizaremos ese certificado en M1. Para ello, debemos editar la configuración de nuestro servidor web y especificar la ruta de los certificados:

```
m1 > sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Sustituimos las siguientes líneas bajo *SSLEngine on*:

```
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

5. Activamos el sitio default-ssl y reiniciamos apache:

```
m1 > sudo a2ensite default-ssl
m1 > sudo service apache2 restart
```

6. Accedemos desde el host a *https://M1IP* > *yobtenemoslainformacióndelcertificado :*

*FIGURECERT.PNG* Ahora configuraremos la granja para que funciones bajo SSL. Comenzamos

```
7. m1 > sudo scp /etc/apache2/ssl/apache.crt <
    ↪ nombre_usuario@<IP M2>:/home/<nombre_usuario
    ↪ >/apache.crt
m1 > sudo scp /etc/apache2/ssl/apache.key <
    ↪ nombre_usuario@<IP M2>:/home/<nombre_usuario
    ↪ >/apache.key

m1 > sudo scp /etc/apache2/ssl/apache.crt <
    ↪ nombre_usuario@<IP M3>:/home/<nombre_usuario
    ↪ >/apache.crt
m1 > sudo scp /etc/apache2/ssl/apache.key <
    ↪ nombre_usuario@<IP M3>:/home/<nombre_usuario
    ↪ >/apache.key
```

Movemos los archivos de certificado a */etc/apache2/ssl* y realizamos los pasos anteriores en M2.

Creamos un nuevo server en nginx (M3):

```
m1 > sudo nano /etc/nginx/conf.d/default.conf
```

Copiamos las líneas que introdujimos en la práctica anterior, cambiamos *listen 80* por *listen 443 ssl* y añadimos lo siguiente:



```
ssl on;  
ssl_certificate /home/<nombre_usuario>/apache.crt;  
ssl_certificate_key /home/<nombre_usuario>/apache.key;
```

Reiniciamos el servicio:

```
m1 > sudo systemctl restart nginx
```

Comprobamos desde el navegador que ya podemos hacer peticiones HTTPS.

Ahora configuraremos un cortafuegos para que a M1 y M2 solo se pueda acceder desde el balanceador. Para ello elaboraremos un script:

```
#!/bin/bash  
# Run this script with superuser permissions!  
# Eliminamos las reglas existentes  
iptables -F;  
iptables -X;  
iptables -Z;  
iptables -t nat -F;  
# Denegamos todo el trficio entrante (poltica inicial)  
iptables -P INPUT DROP;  
iptables -P OUTPUT ACCEPT;  
iptables -P FORWARD DROP;  
# Permitir acceso desde las loopback addresses (localhost)  
iptables -A INPUT -i lo -j ACCEPT  
iptables -A OUTPUT -o lo -j ACCEPT  
# Puertos 22 (SSH), 80 (HTTP) y 443 (HTTPS) desde el balanceador  
iptables -A INPUT -i 192.168.56.104 -p tcp -m multiport --dports  
    ↪ 22,80,443 -j ACCEPT  
iptables -A INPUT -i 192.168.56.104 -p tcp -m multiport --sports  
    ↪ 22,80,443 -j ACCEPT
```

Hacemos que el script se ejecute al inicio del sistema:

```
m1, m2 > sudo crontab -e
```

Añadimos lo siguiente al final:

```
@reboot sudo /home/<nombre_usuario>/iptables_servers.sh
```

Finalmente creamos un script para M3 (sólo aceptará peticiones HTTP y HTTPS):

```
#!/bin/bash  
# Run this script with superuser permissions!  
# Eliminamos las reglas existentes  
iptables -F;  
iptables -X;  
iptables -Z;  
iptables -t nat -F;
```

```
# Denegamos todo el trfico entrante (poltica inicial)
iptables -P INPUT DROP;
iptables -P OUTPUT ACCEPT;
iptables -P FORWARD DROP;
# Permitir acceso desde las loopback addresses (localhost)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# Puertos 80 (HTTP) y 443 (HTTPS) desde cualquier IP
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
iptables -A INPUT -p tcp -m multiport --sports 80,443 -j ACCEPT
```