



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

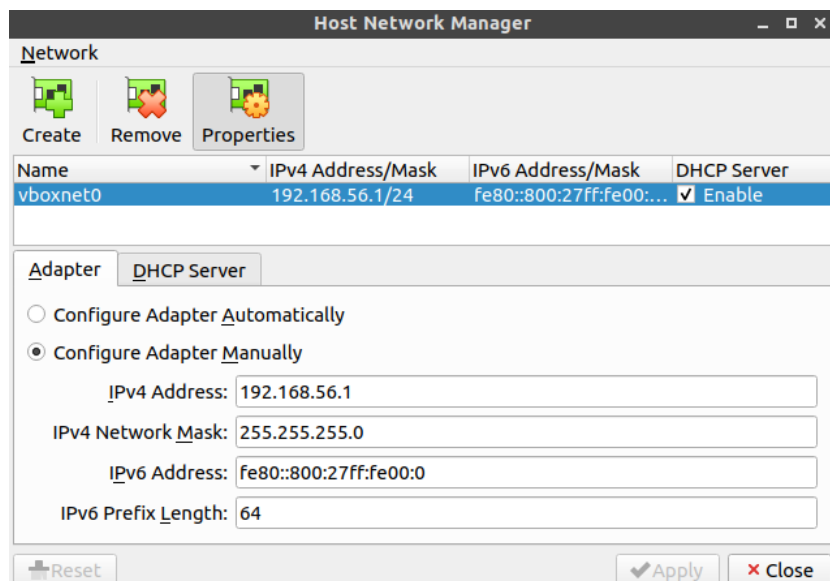
1. Práctica 1	2
2. Práctica 2	6
3. Práctica 3	9

1. Práctica 1

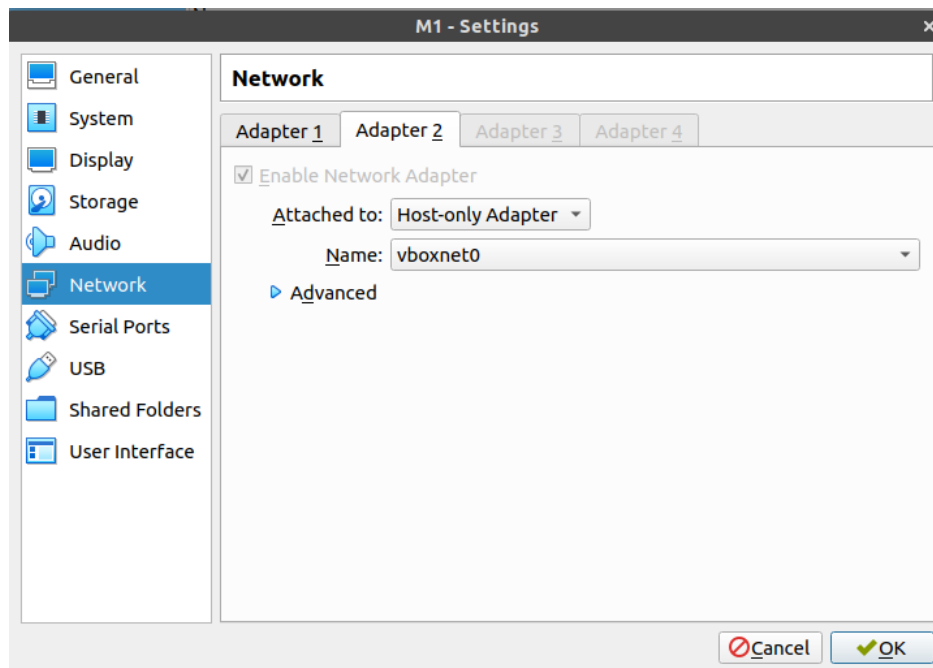
En esta práctica configuraremos dos máquinas virtuales (*M1* y *M2*). También crearemos una interfaz *host-only* para que las máquinas puedan conectarse entre ellas.

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:

The screenshot shows two Oracle VM VirtualBox windows. The left window, titled 'M1 [Running] - Oracle VM VirtualBox', shows a terminal session where the user runs 'sudo apt install <deb name>' and then 'ifconfig -a'. The output shows network interfaces 'enp0s3' and 'enp0s8' with their respective IP addresses and configurations. The right window, titled 'M2 [Running] - Oracle VM VirtualBox', shows a similar terminal session where the user runs 'ifconfig -a'. The output shows network interfaces 'enp0s3', 'enp0s8', and 'lo' with their respective IP addresses and configurations. Both windows show the standard VirtualBox menu (File, Machine, View, Input, Devices, Help) and a taskbar at the bottom with various application icons and a 'Right Ctrl' button.

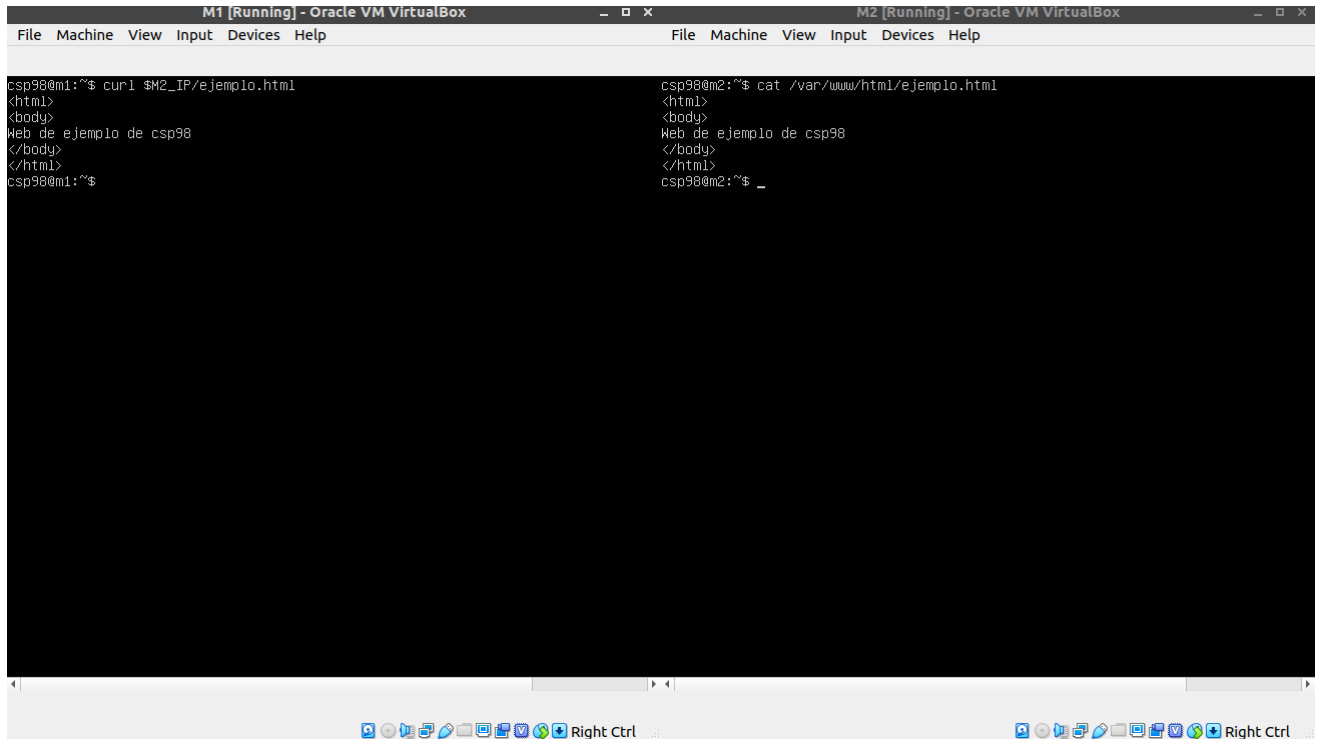
The screenshot shows two Oracle VM VirtualBox windows. The left window, titled 'M1 [Running] - Oracle VM VirtualBox', shows a terminal session where the user runs 'ssh \$M2_IP' and 'ssh \$M1_IP'. The output shows the Ubuntu login prompt and system information. The right window, titled 'M2 [Running] - Oracle VM VirtualBox', shows a similar terminal session where the user runs 'ssh \$M1_IP' and 'ssh \$M2_IP'. The output shows the Ubuntu login prompt and system information. Both windows show the standard VirtualBox menu (File, Machine, View, Input, Devices, Help) and a taskbar at the bottom with various application icons and a 'Right Ctrl' button.

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:~$ _
```

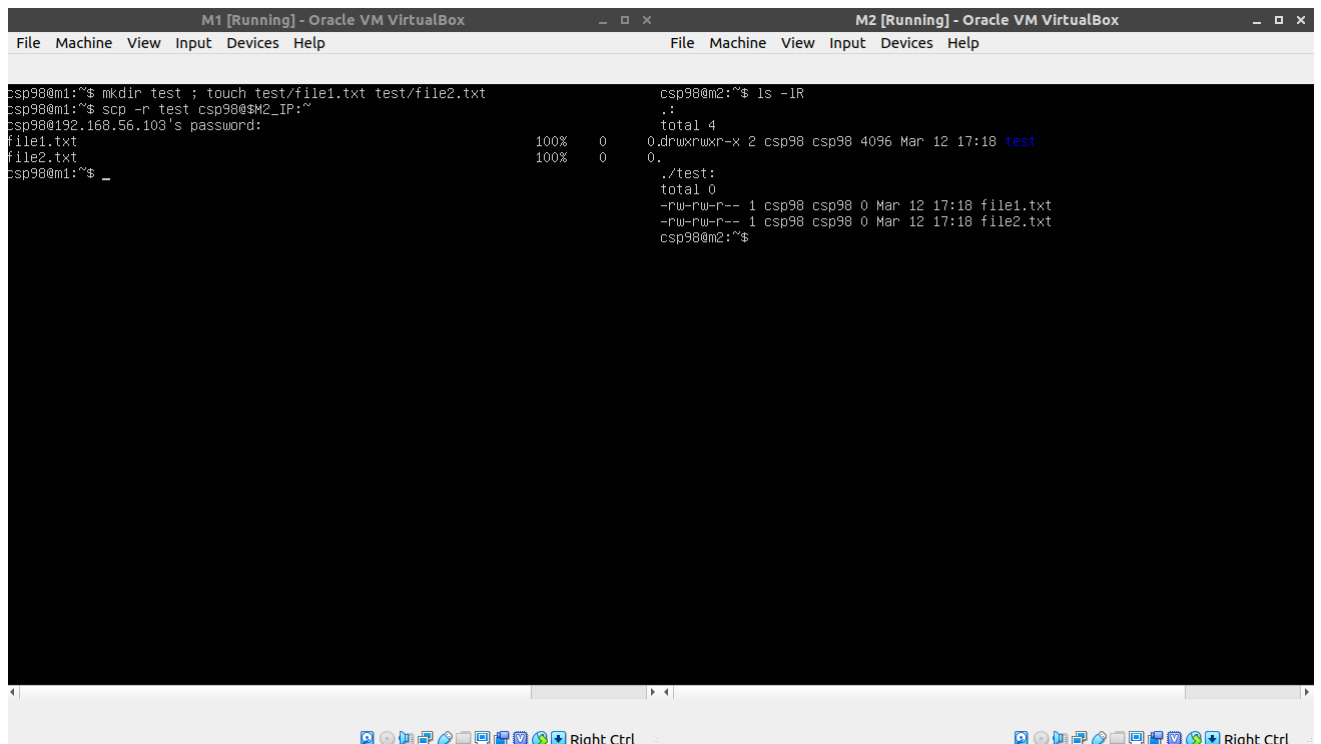
2. Práctica 2

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
```

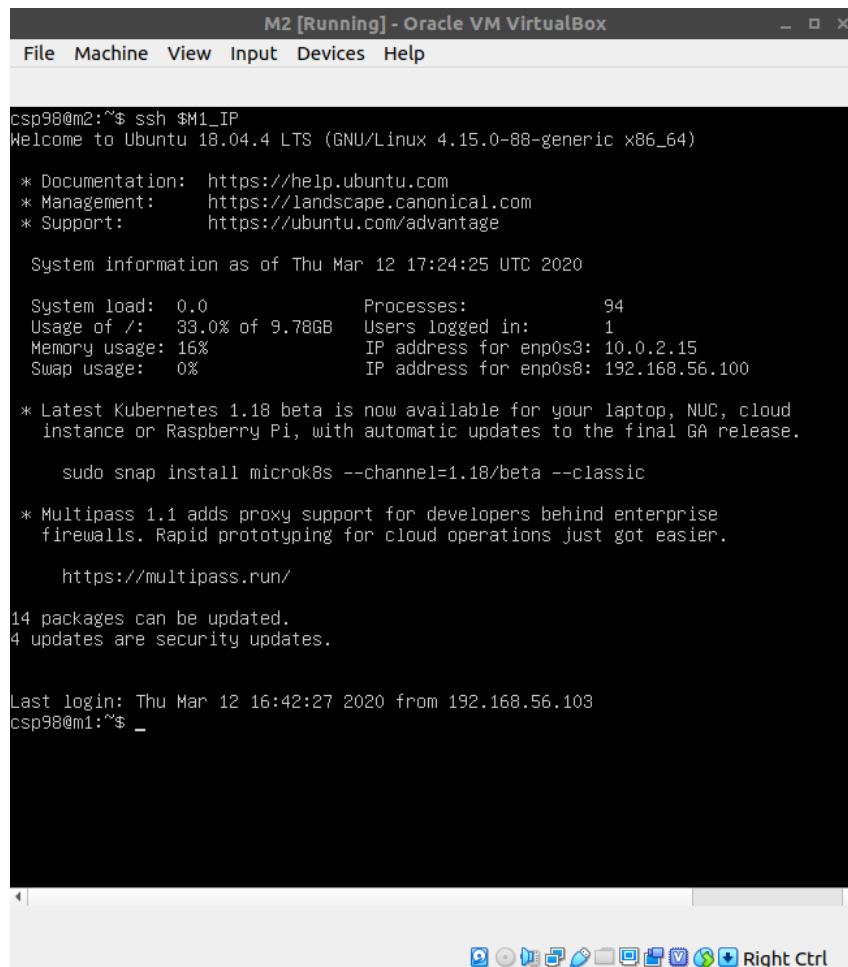
Pulsamos Enter tres veces.

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. Configuramos el clonado:

```
m1 > sudo nano /etc/crontab
```

Añadimos la siguiente línea:

```
00 */12 * * * csp98 scp -r /var/www/ 192.168.56.103:/var/
```

9. 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 un balanceador de carga (*nginx*).

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: **IMÁGENES DE AMBOS HTML**
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. **IMÁGENES DE AMBAS WEBS**
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;  
    }
```