SWAP: Preparación de las herramientas

David Cabezas Berrido

dxabezas@correo.ugr.es

6 de marzo de 2021

Índice

1. Objetivos

El objetivo de esta primera práctica es la puesta en marcha de dos máquinas virtuales idénticas que puedan conectarse a internet y entre sí. Así como la instalación y configuración de ciertos servicios y herramientas como son Apache, PHP, MySQL, SSH o CURL.

2. Creación de las máquinas e instalación del sistema

Comenzamos creando dos máquinas virtuales idénticas, seleccionamos Ubuntu-64b y la configuración recomendada (1 core, 1GB de RAM y 10GB de disco dinámicos).

Ambas máquinas vienen con un adaptador de red NAT por defecto, añadimos un segundo adaptador de red Host-only (`Settings -> Network -> Adapter 2 -> Host-only Adapter`) para que las máquinas puedan comunicarse entre sí. Si no tenemos ninguno, lo podemos crear en `File -> Host Network Manager -> Create`.

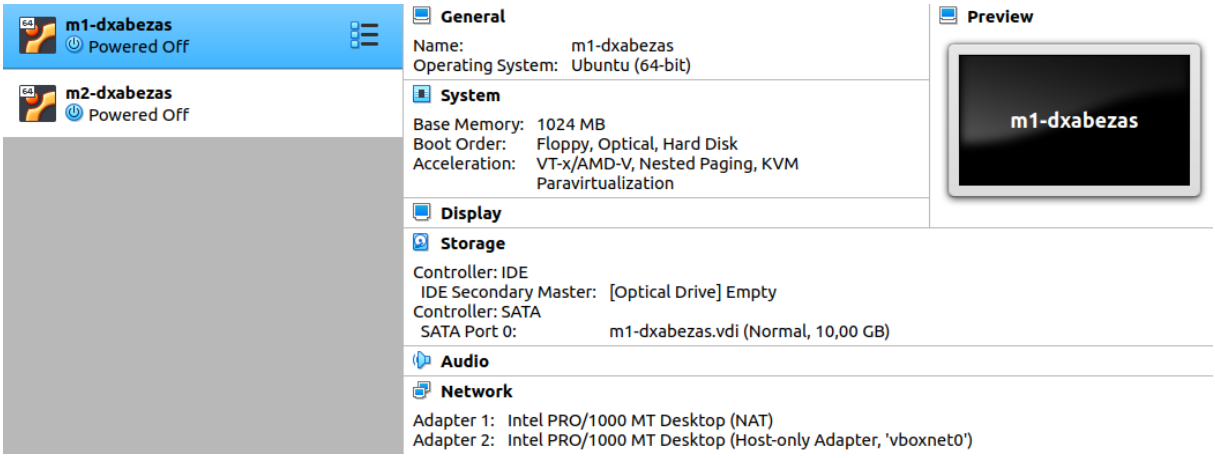


Figura 1: Resumen de la configuración de la máquina M1. Observamos que tiene los dos adaptadores de red que hemos comentado.

Seguidamente, instalamos Ubuntu Server 18.04 LTS en ambas máquinas. Podemos descargar la ISO de la página oficial. Creamos en las dos máquinas perfiles idénticos, con usuario **dxabezas** y constraseña **Swap1234**.

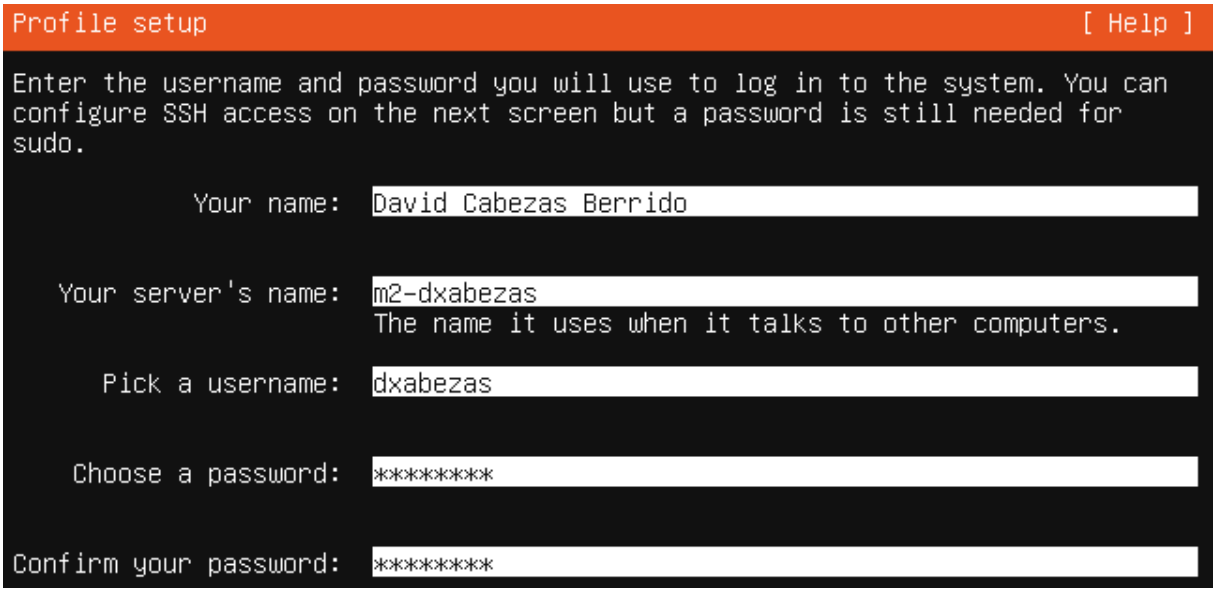


Figura 2: Creando el perfil en la máquina M2.

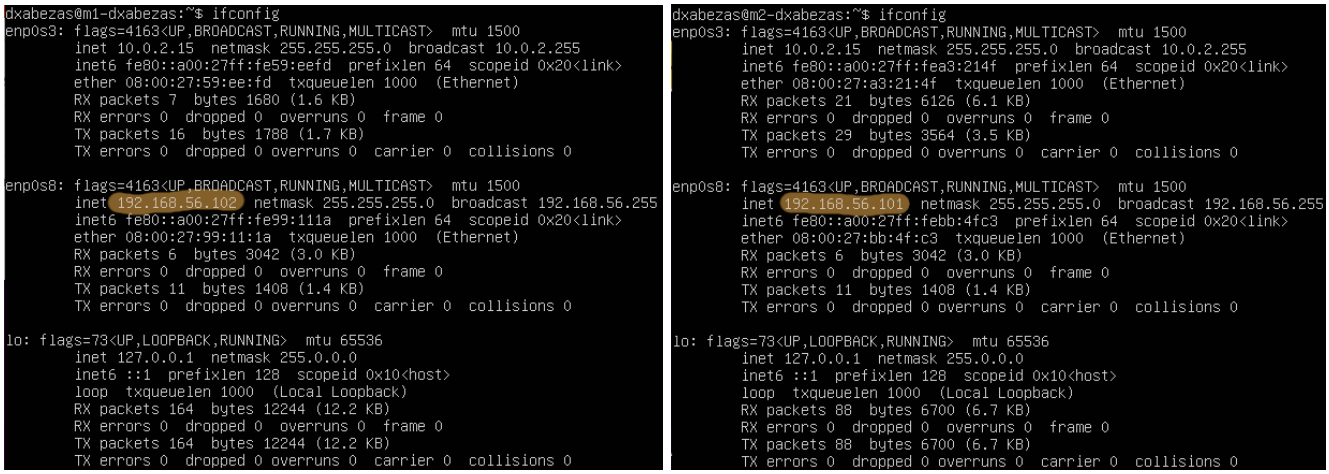
Durante la instalación escogemos siempre las opciones recomendadas, con la excepción de instalar OpenSSH.

3. SSH

Ya tenemos OpenSSH instalado en ambas máquinas. Si durante la instalación no lo hubiésemos marcado, tendríamos que ejecutar:

```
sudo apt-get install openssh-client
sudo apt-get install openssh-server
```

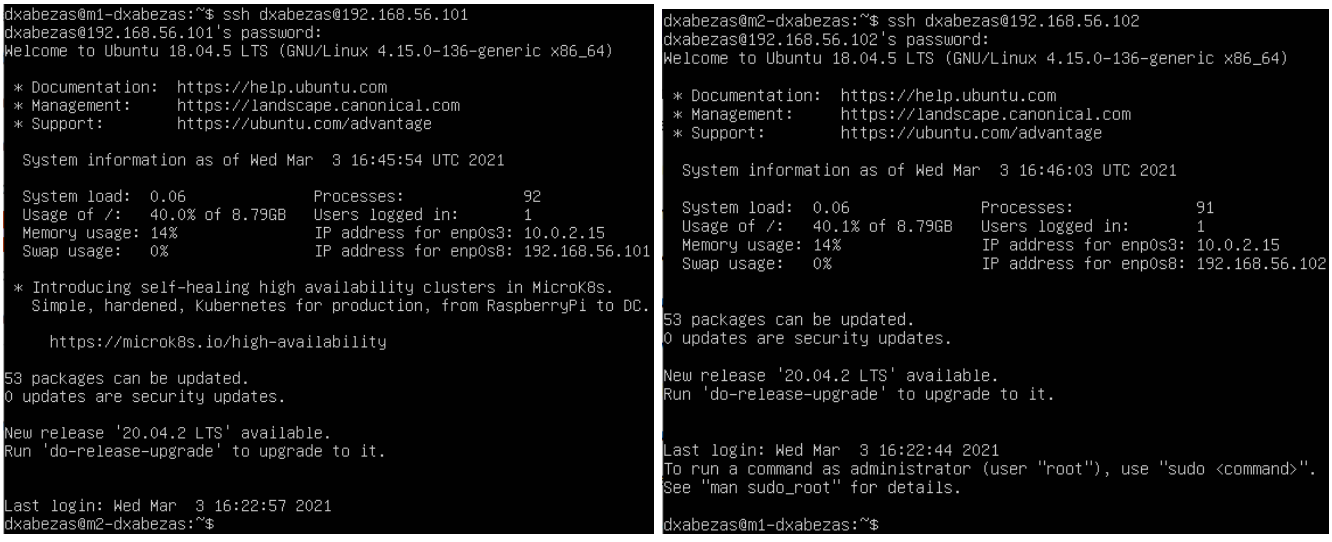
Con `ifconfig`, podemos ver la dirección IP de cada máquina:



(a) Máquina 1. (b) Máquina 2.

Figura 3: Direcciones IP de ambas máquinas.

Nos conectamos de una máquina a otra, ejecutando `ssh user@ip`.



(a) M1 se conecta a M2. (b) M2 se conecta a M1.

Figura 4: Cada máquina se conecta a la otra por SSH con el usuario que creamos (dxabezas, Swap1234).

NOTA: En lo que viene a continuación, la dirección IP de M1 es la que termina en 1 y la de M2 es la que termina

en 2. En la siguiente sección explico cómo las intercambio.

Ahora vamos a configurar SSH, para ello modificamos el fichero `/etc/ssh/sshd_config`, donde se encuentra la configuración del servicio `ssh-server`. `/etc/ssh/ssh_config` es para la configuración de `ssh-client`. Por ejemplo, podemos modificar el puerto del 22 (por defecto) al 2222 con la línea `Port 2222`. Para hacer efectivos los cambios, guardamos el archivo y reestauramos el servicio con

```
sudo service ssh restart
```

Ahora desde la otra máquina nos conectamos, pero debemos indicar el puerto explícitamente (con la opción `-p`) al no tratarse del por defecto.

```
dxabezas@m2-dxabezas:/etc/netplan$ ssh dxabezas@192.168.56.101
ssh: connect to host 192.168.56.101 port 22: Connection refused
dxabezas@m2-dxabezas:/etc/netplan$ ssh -p 2222 dxabezas@192.168.56.101
dxabezas@192.168.56.101's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)
```

Figura 5: La máquina 2 se intenta conectar a la máquina 1. El primer intento fracasa porque intenta conectarse al puerto por defecto.

Una buena práctica desde el punto de vista de la seguridad es deshabilitar el root login (`PermitRootLogin no`). De esta manera, nadie se puede conectar como root por SSH, tendrá que obtener permisos de superusuario una vez logueado en el sistema mediante sudo.

Finalmente, configuraremos el acceso sin contraseña mediante clave pública. Desde la máquina M2 generamos un par de claves con `ssh-keygen -t rsa`, nos da la opción de poner una passphrase por si tememos que alguien pueda acceder a nuestro dispositivo.

```
dxabezas@m2-dxabezas:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dxabezas/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dxabezas/.ssh/id_rsa.
Your public key has been saved in /home/dxabezas/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6qNcC9g56O/Zo2Iaz1COMHObi+wg4Mq4pRPITbOunyw dxabezas@m2-dxabezas
```

Figura 6: La máquina 2 genera un par de claves (pública y privada).

Ahora copiamos la clave pública en M1 ejecutando (desde M2)

```
ssh-copy-id -p 2222 dxabezas@192.168.56.101
```

nos pide la contraseña (Swap1234) y nos dice que se ha añadido 1 clave. Ahora nos podemos conectar con SSH sin poner sin que nos pida la contraseña, nos pedirá la passphrase en caso de haber puesto una.

Hemos elegido crear la pareja de claves en los archivos por defecto (`~/.ssh/id_rsa` y `~/.ssh/id_rsa.pub`). De haber seleccionado otros archivos, tendríamos que indicar con la opción `-i` el archivo de clave a copiar (pública) y a utilizar para autenticarse (privada).

4. Configuración de red

Ya tenemos los dos adaptadores de red (NAT y Host-only) creados. Cuando instalamos el sistema se configuran por defecto, pero podemos usar Netplan para realizar los cambios que deseemos.

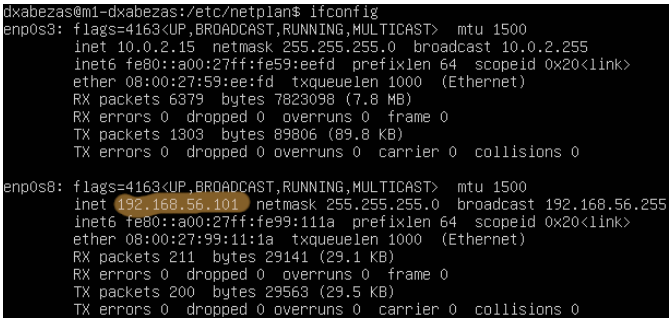
Para ello, modificamos el fichero `00-installer-config.yaml` en la carpeta `/etc/netplan`, donde ya encontramos la configuración por defecto del adaptador NAT (interfaz `enp0s3`) y del adaptador Host-only (interfaz `enp0s8`), que reciben direcciones IP via DHCP. Podemos consultar las direcciones asignadas con `ifconfig` (Figura 3).

```
network:
    version: 2
    ethernets:
        enp0s3:
            dhcp4: true
        enpos08:
            dhcp4: true
```

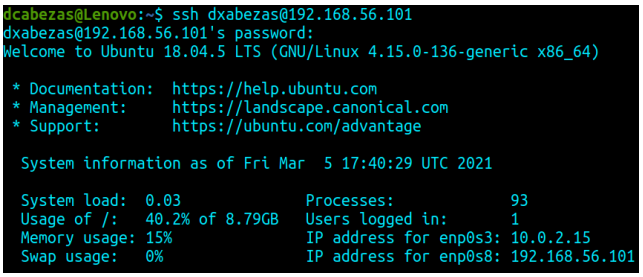
En la Figura 3 también observamos que las direcciones obtenidas están "cambiadas", la máquina 1 recibió `192.168.56.102` y la máquina 2 la `192.168.56.101`. Esto se debe a que realizamos primero la instalación de la M2, pero ahora podemos cambiarlas manualmente para tener direcciones más intuitivas. También en la Figura 3 observamos que se utiliza la máscara de red `255.255.255.0` (24 bits a 1 y 8 bits a 0), la mantenemos añadiendo `/24` tras la dirección. Editamos el archivo con `sudo nano 00-installer-config.yaml`. En la máquina 1 escribimos:

```
network:
    version: 2
    ethernets:
        enp0s3:
            dhcp4: true
        enpos08:
            dhcp4: no
            addresses: [192.168.56.101/24]
```

y en la M2 cambiamos a `192.168.56.102/24`. Para aplicar los cambios ejecutamos `sudo netplan apply`, podemos comprobarlos con ifconfig y conectarnos por ssh desde la máquina anfitriona o la otra máquina para comprobar que funciona. Si ya nos conectamos desde la máquina anfitriona antes de este cambio, habrá que borrar la fingerprint de known hosts primero.



(a) La IP de M1 ha cambiado.



(b) Nos conectamos a M1 desde el anfitrión con la nueva IP.

Figura 7: Conexión por SSH desde el anfitrión a M1 con la nueva IP.

5. Servicios LAMP

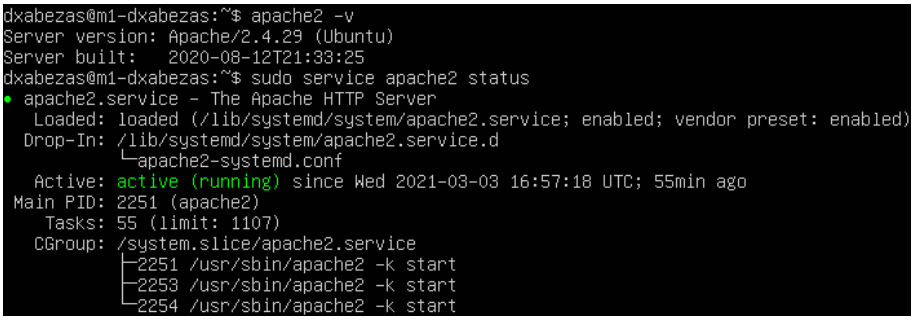
5.1. Apache2

Puesto que no marcamos los servicios LAMP durante la instalación, debemos instalarlos manualmente. Empezamos con Apache2, en ambas máquinas ejecutamos:

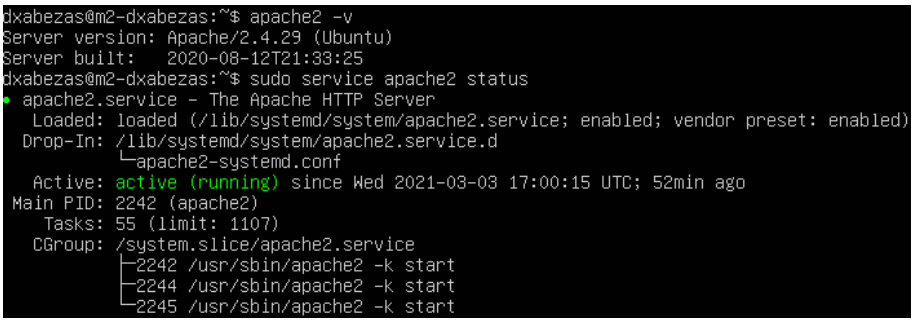
```
sudo apt install -y apache2
```

Para comprobar la versión que hemos instalado usamos `apache2 -v`, y para comprobar que esté en ejecución,

```
sudo service apache2 status
```



(a) Máquina 1.



(b) Máquina 2.

Figura 8: Comprobamos la versión de Apache2 y que esté activo.

En la máquina 1 cambiamos el puerto de escucha al 8000 escribiendo `Listen 8000` en el fichero `/etc/apache2/ports.conf`. En el fichero avisa de que modifiquemos también `/etc/apache2/sites-enabled/000-default.conf`, donde cambiamos la línea

`<VirtualHost *:80>`

por `<VirtualHost *:8000>`.

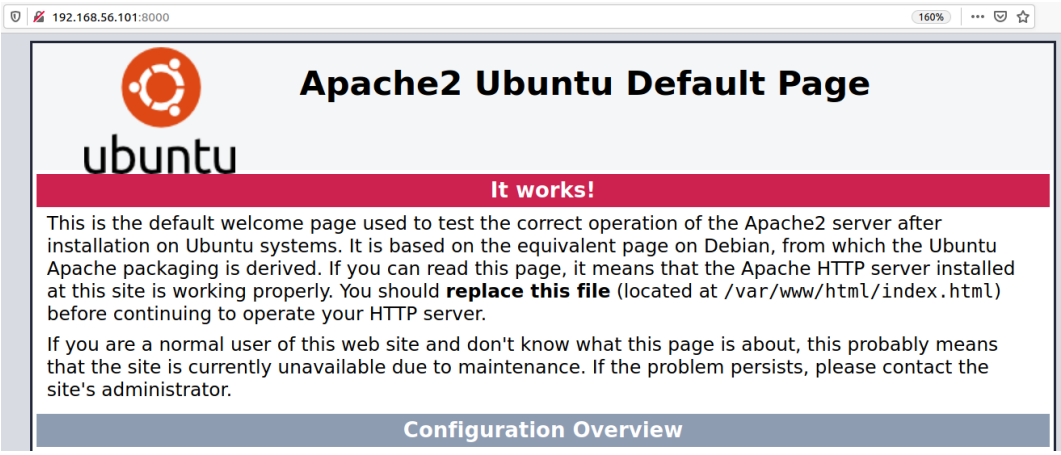
Ejecutamos `sudo apache2ctl configtest` para comprobar que está todo bien. Nos da un warning diciendo que le pongamos un nombre de dominio al servidor, ya que actualmente se está usando (`127.0.1.1`). Lo cambiamos añadiendo la línea

`ServerName 192.168.56.101`

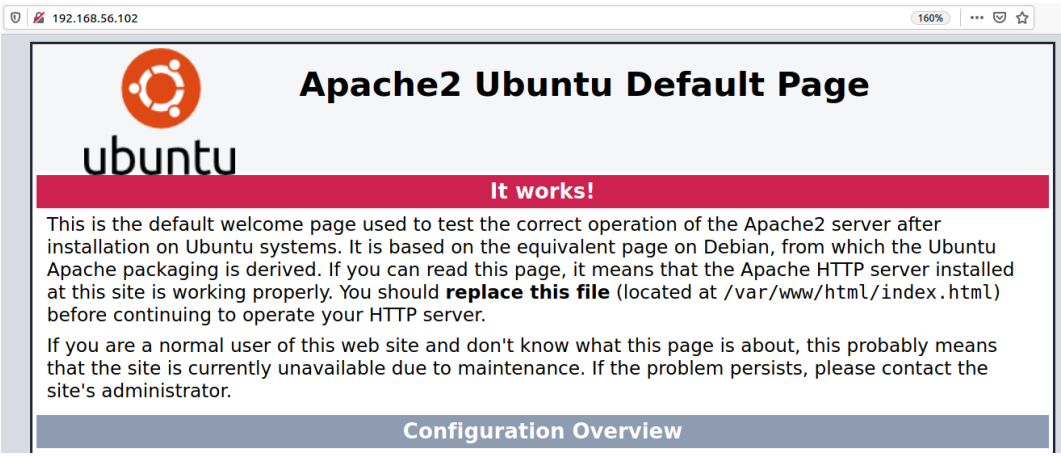
al fichero `/etc/apache2/apache2.conf`. También cambiamos el nombre a `192.168.56.102` en la M2. Ahora la comprobación sólo nos devuelve Syntax OK. Reestablecemos el servicio para que se apliquen los cambios.

`sudo systemctl restart apache2`

Desde la máquina anfitriona podemos visitar las direcciones `192.168.56.101:8000` (M1, hemos cambiado el puerto de escucha) y `192.168.56.102` (M2, puerto 80 por defecto). Comprobamos que Apache funciona correctamente, nos sirve el fichero `/var/www/html/index.html`. Se puede cambiar el directorio modificando `DocumentRoot` en `/etc/sites-available/000-default.conf`.



(a) M1, al puerto 8000.



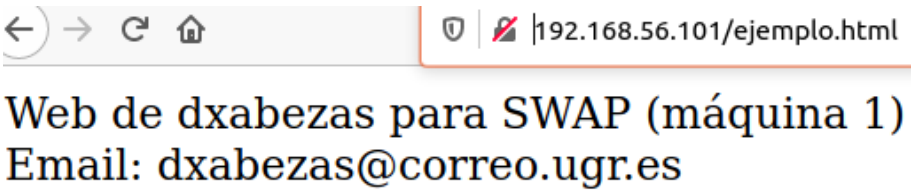
(b) M2, al puerto por efecto (80).

Figura 9: El servicio Apache funciona correctamente, hemos cambiado el puerto de escucha en M1.

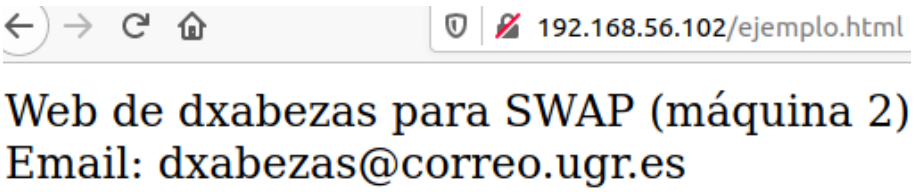
Creamos el fichero `/var/www/html/ejemplo.html` en ambas máquinas. Mostramos el de la máquina 1.

```
<html>
    <head>
        <meta charset="UTF-8">
    </head>
    <body>
        Web de dxabezas para SWAP (máquina 1) </br>
        Email: dxabezas@correo.ugr.es
    </body>
</html>
```

Desde el anfitrión visualizamos el fichero correctamente en ambas máquinas:



(a) Ejemplo de la máquina 1.



(b) Ejemplo de la máquina 2.

Figura 10: Archivos de ejemplo de ambas máquinas.

5.1.1. Redirección de puertos

En la máquina 1 atendemos peticiones desde el puerto 8000. Redireccionaremos las peticiones al puerto 80 para que se atiendan desde el 8000. Debemos hacer que Apache escuche en ambos puertos escribiendo `Listen 8000` y `Listen 80` en `/etc/apache2/ports.conf`.

A continuación seguimos las instrucciones de
https://stackoverflow.com/questions/8541182/apache-redirect-to-another-port.

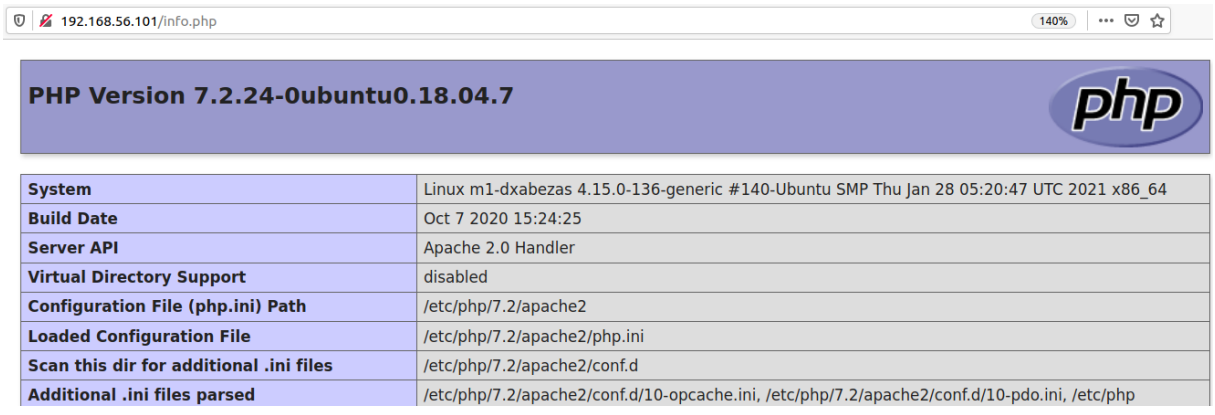
En `/etc/apache2/sites-enabled/000-default.conf`, añadimos el siguiente bloque:

```
<VirtualHost *:80>
    ProxyPreserveHost On
    ProxyRequests Off
    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
</VirtualHost>
```

Ahora tenemos que añadir los módulos `proxy` y `proxy_http`. Finalmente, reiniciamos el servicio.

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo systemctl restart apache2
```

Ahora M1 es capaz de servir los archivos desde el puerto por defecto:



System	Linux m1-dxabezas 4.15.0-136-generic #140-Ubuntu SMP Thu Jan 28 05:20:47 UTC 2021 x86_64
Build Date	Oct 7 2020 15:24:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php

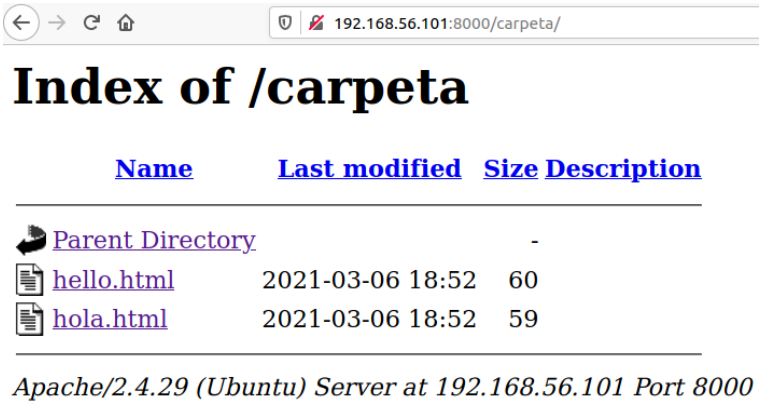
Figura 11: M1 sirve el fichero `/var/www/html/info.php` (que añadimos en la Sección 5.3) ante una petición al puerto 80, que redirige al 8000.

5.1.2. Directorios virtuales

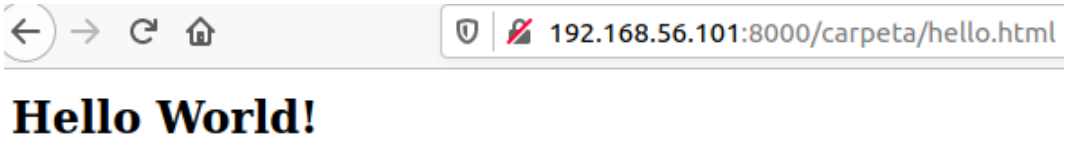
Serviremos un directorio en `home` de la máquina 1. Creamos dos ficheros: `/home/dxabezas/carpeta/hola.html`, `/home/dxabezas/carpeta/hello.html`, con los mensajes "Hola Mundo!" y "Hello World!". A continuación, creamos un link simbólico de `carpeta` en `/var/www/html`

```
sudo ln -s ~/carpeta /var/www/html/carpeta
```

Ahora Apache puede servir ambos archivos:



(a) Directorio `carpeta`.



(b) Archivo `/home/dxabezas/carpeta/hello.html`.

Figura 12: Gracias al enlace simbólico, Apache puede servir un subdirectorio de `home`.

5.2. MySQL

A continuación instalamos MySQL, tanto el servicio de servidor como de cliente

```
sudo apt install mysql-client mysql-server
```

Con `sudo systemctl status mysql.service`, comprobamos que el servicio está en marcha. Procedemos a la instalación segura con `sudo mysql_secure_installation`. Nos hace varias preguntas para configurar la seguridad, en ambas máquinas:

- No establecemos el plugin de validación de contraseñas, que obliga a que las contraseñas sean seguras.
- Elegimos **Swap1234** como contraseña para el usuario root de MySQL.
- Eliminamos el usuario anónimo para pruebas, compromete la seguridad de la base de datos una vez en producción.
- Deshabilitamos el remote root login, el root deberá loguearse desde localhost.
- Eliminamos la base de datos de test para la que todo el mundo tiene privilegios, compromete la seguridad en producción. También se eliminan los privilegios sobre esta BD en la tabla de privilegios.
- Recargamos la tabla de privilegios para hacer estos cambios efectivos.

Ya podemos acceder con `sudo mysql -u root -p=Swap1234`, o `-p` para que nos solicite la contraseña a continuación.

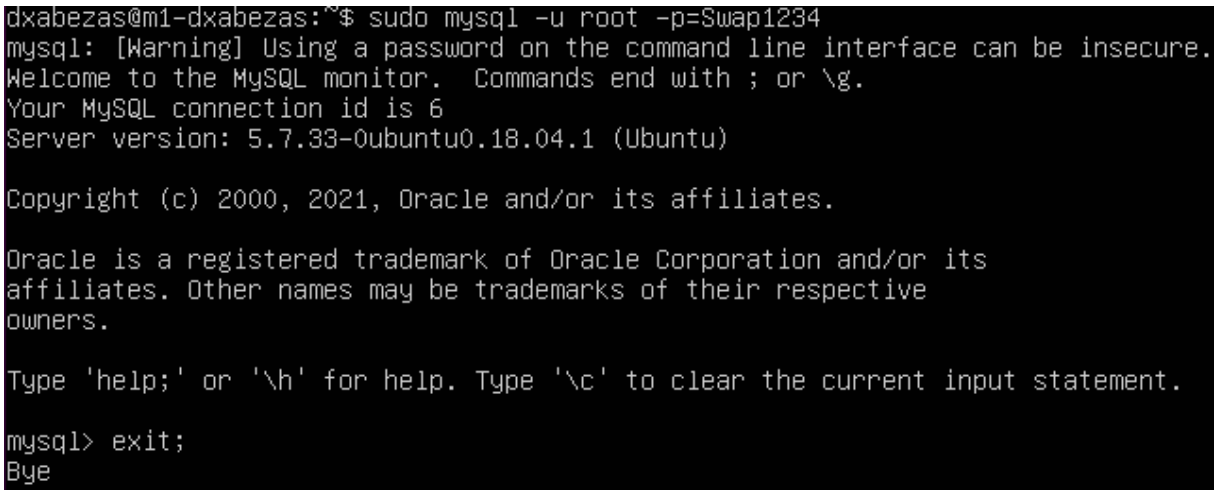


Figura 13: Acceso a MySQL en la máquina 1.

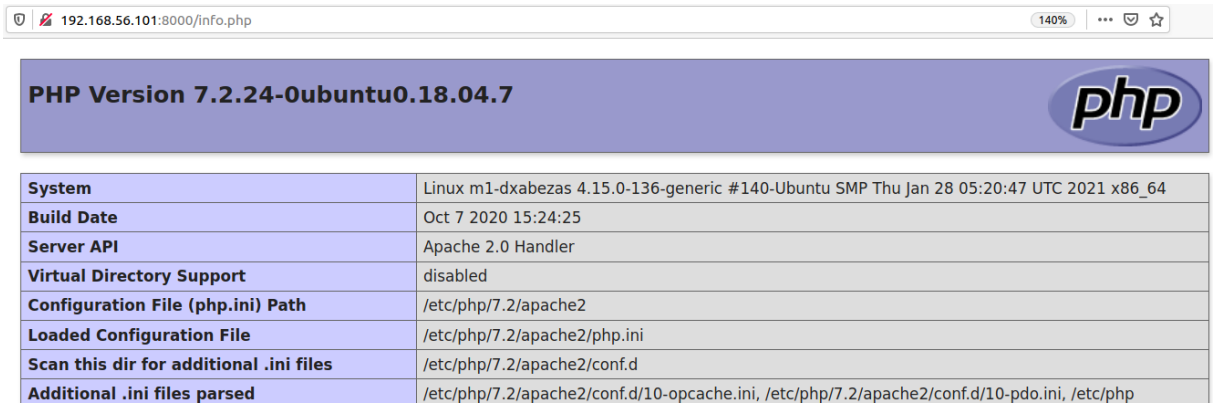
5.3. PHP

Instalamos PHP con `sudo apt install php`, automáticamente nos avisa de que va a instalar la versión 7.2 y de algunos paquetes que necesita como `php7.2-json` y `libapache2-mod-php7.2`.

Para comprobar que funciona, creamos un fichero `/var/www/html/info.php` con el contenido

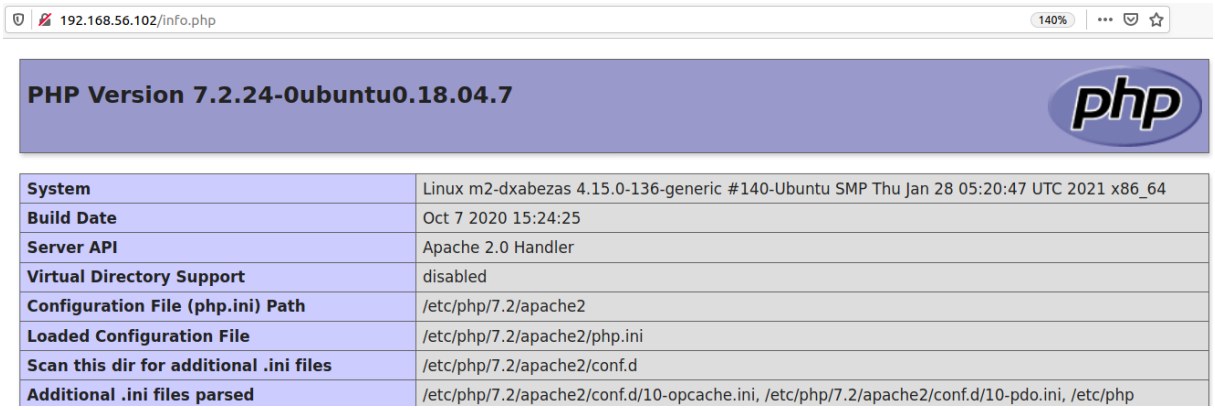
```
<?php
phpinfo();
?>
```

y comprobamos que se sirve correctamente desde la máquina anfitriona.



System	Linux m1-dxabezas 4.15.0-136-generic #140-Ubuntu SMP Thu Jan 28 05:20:47 UTC 2021 x86_64
Build Date	Oct 7 2020 15:24:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php

(a) Máquina 1, en la que cambiamos el puerto de HTML.



System	Linux m2-dxabezas 4.15.0-136-generic #140-Ubuntu SMP Thu Jan 28 05:20:47 UTC 2021 x86_64
Build Date	Oct 7 2020 15:24:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php

(b) Máquina 2.

Figura 14: Desde la máquina anfitriona, comprobamos que PHP funciona correctamente en ambas máquinas.

6. cURL

Instalamos cURL:

```
sudo apt install curl
```

nos dice en ambas máquinas que ya está instalado y en su última versión.

Como primera prueba, obtenemos fichero `/var/www/html/ejemplo.html` de la máquina 1 desde la 2:

```
dxabezas@m2-dxabezas:/var/www/html$ curl 192.168.56.101/ejemplo.html
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
Web de dxabezas para SWAP (máquina 1) </br>
Email: dxabezas@correo.ugr.es
</body>
</html>
```

Figura 15: Visualizamos el ejemplo de M1 en M2 mediante cURL.

Con la opción `-o`, podemos descargar el fichero en la máquina 2:

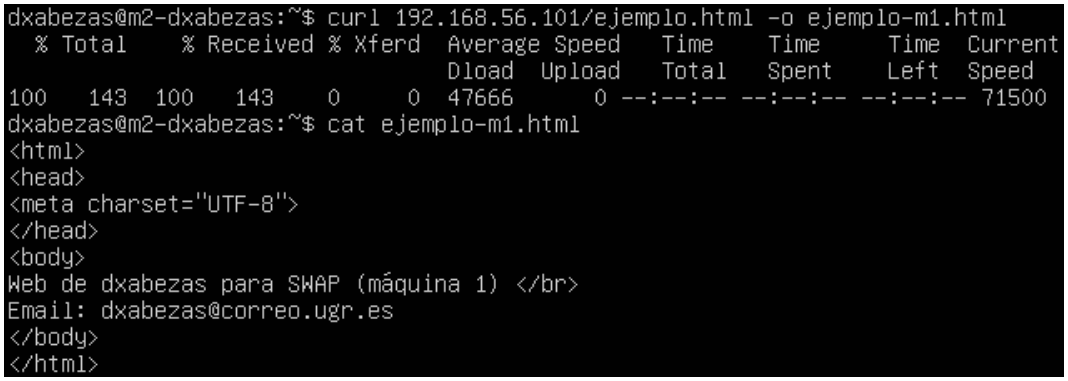


Figura 16: Descargamos el ejemplo de M1 en M2 mediante cURL.

La opción `-O` es similar, guarda el fichero con el mismo nombre con el que está subido. Esta vez hacemos la petición al puerto 8000 en lugar de al puerto por defecto (80).

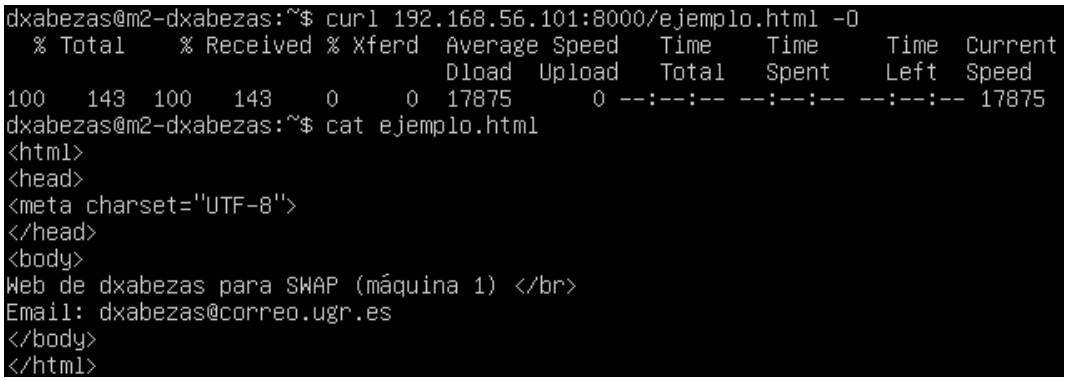


Figura 17: Descargamos el ejemplo de M1 en M2 mediante cURL.

Todas las peticiones que hemos hecho son GET (por defecto). Si quisiésemos realizar otra petición (como POST, PUT, COPY o DELETE), tendríamos que explicitarla con la opción `--request <peticion>`.

Por ejemplo, podríamos subir un JSON con el siguiente comando:

```
curl --location --request POST '<direccion>' --header 'Content-Type: application/json'
--data-raw '{nombre: "David", apellidos: "Cabezas Berrido"}'
```

6.1. Cookies

Para utilizar cookies con cURL, debemos conocer dos opciones. La primera es `-c` o `--cookie-jar`, con la que se indica el nombre de un archivo para almacenar las cookies. Por ejemplo:

```
curl -c cookie.txt https://www.google.com
```

Se crea el archivo `cookie.txt`:

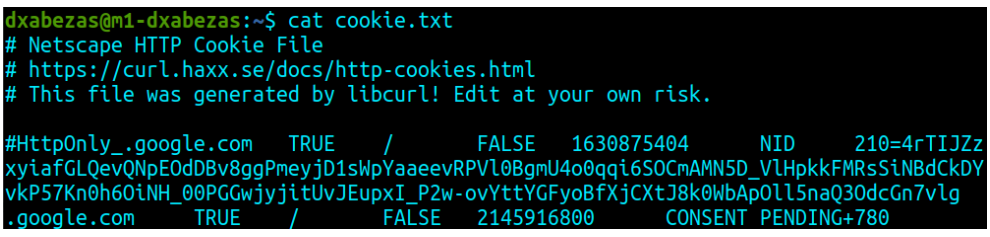


Figura 18: Se ha creado el archivo con la cookie.

La segunda opción es `-b` o `--cookie`, para enviar cookies. Por ejemplo:

```
curl -b cookie.txt https://www.google.com
```