P6: Contenedores con Docker (Introducción)

1. Descripción

El objetivo del ejercicio consiste en desplegar un balanceador de carga con dos servidores WordPress usando contenedores Docker.

2. Entorno de prácticas

En estas prácticas se empleará el software de virtualización VIRTUALBOX para simular los equipos GNU/Linux sobre los que se realizarán las pruebas.

3. Imágenes a utilizar

Se proporcionan scripts de instalación tanto para GNU/Linux como para Windows. Windows es bastante inestable con algunas configuraciones de la Máquina Virtual que se usarán durante la realización de las prácticas. Por ello, se recomienda encarecidamente usar Linux como sistema base.

- Script GNU/Linux: ejercicio-docker.sh (desde línea de comandos)

```
01 alumno@pc: $ sh ejercicio-docker.sh
```

- MS Windows: ejercicio-docker.ps1 (desde cmd)

```
01 Powershell.exe -executionpolicy bypass -file ejercicio-docker.ps1
```

Notas:

- Se pedirá un identificador (sin espacios) para poder reutilizar las versiones personalizadas de las imágenes creadas. Se recomienda usar como identificativo CDA_<numero_del_grupo>_<INICIALES>.
- En ambos scripts la variable $DIR_BASE específica donde se descargarán las imágenes y se crearán las MVs.
- Por defecto en GNU/Linux será en $HOME/CDA2425 y en Windows en C:/CDA2425.
- Puede modificarse antes de lanzar los scripts para hacer la instalación de las imágenes en otro directorio más conveniente (disco externo, etc)
- Si se hace desde el script anterior, se pueden arrancar las instancias VIRTUALBOX desde el interfaz gráfico de VirtualBOX o desde la línea de comandos con VBoxManage startvm <nombre MV>_<id>

4. Credenciales de acceso

La distribución Linux incluida en la MV tiene dados de alta dos usuarios con las siguientes credenciales y permisos:

login	password	permisos
root	purple	root
usuario	purple	permite sudo

5. Entorno desplegado

El entorno desplegado para la realización de las prácticas está formado por una máquinas virtuales sobre la que se desplegarán los diferentes servicios virtualizados en contenedores Docker.

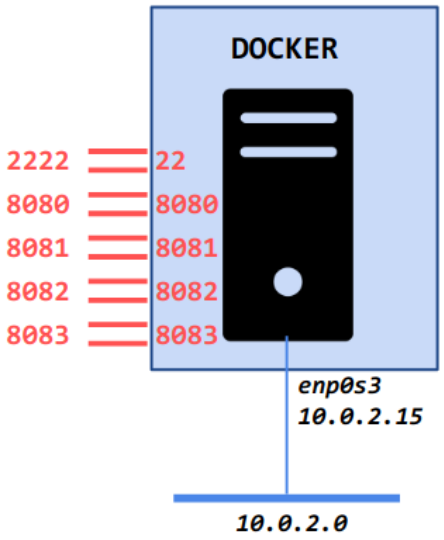

Fig 1. Entorno de trabajo.

- DOCKER tiene la dirección IP privada 10.0.2.15 y será el encargado desplegar los contenedores que ejecuten los diferentes servicios (balanceador, wordpress, base de datos). Adicionalmente, tiene redireccionados los puertos 8080 a 8085 del anfitrión para las pruebas desde el navegador Web del anfitrión y el puerto 2222 para conexiones SSH (22).

6. Manejo básico de contenedores Docker

6.1. Imágenes y contenedores.

- Lanzar Docker hello-world y comprobar las imágenes y contenedores disponibles.

```
usuario@docker:~ docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest:
sha256:c79d06dfdfd3d3eb04cafd0dc2bacab0992ebc243e083cabe208bac4dd7759
e0
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working
correctly.
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker
Hub.
(amd64)
3. The Docker daemon created a new container from that image which
runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which
sent it
to your terminal.

To try something more ambitious, you can run an Ubuntu container
with:
$ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

- Listar las imágenes descargadas

```
usuario@docker:~$ docker ps -a
```

```
REPOSITORY  TAG    IMAGE ID     CREATED      SIZE
hello-world latest 9c7a54a9a43c 6 months ago 13.3kB
```

- Listar los contenedores descargados

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e83ca3fd44d7 hello-world "/hello" 2 minutes ago Exited (0) 2 minutes
```

```
03  ago sharp_heisenberg
```

- Crear un contenedor Debian y ver la diferencias entre docker create, docker start, docker run y docker exec.
    1. Buscar y recuperar la imagen debian

```
01  usuario@docker:~$ docker search debian
```

```
01  NAME DESCRIPTION STARS OFFICIAL AUTOMATED
02  ubuntu Ubuntu is a Debian-based Linux operating sys... 15223 [OK]
03  debian Debian is a Linux distribution that's compos... 4495 [OK]
04  ...
05
06
```

```
01  usuario@docker:~$ docker pull debian
```

```
02  Using default tag: latest
03  latest: Pulling from library/debian
04  8457fd5474e7: Pull complete
05  Digest:
06  sha256:fab22df37377621693c68650b06680c0d8f7c6bf816ec92637944778db3ca2
07  c0
08  Status: Downloaded newer image for debian:latest
09  docker.io/library/debian:latest
```

2. Ejecutar el comando *echo* en el contenedor creado a partir de la imagen debian.

   Nota: Comprobar las diferencias entre (1) docker run [creación de contenedor y ejecución de comando inmediata] frente (2) docker create + docker start (con -a para vincular la consola del anfitrión al contenedor) [separa creación del contenedor y ejecución del comando]

```
01  usuario@docker:~$ docker run --name prueba1 debian echo "Hola CDA"
```

```
01  Hola CDA
```

```
01  usuario@docker:~$ docker create --name prueba2 debian echo "Hola CDA"
```

```
01  13f532fa149d65124a4eabfcddeba1e4dc9b7cfa80948fde6cf824493dcfcd98
```

```
usuario@docker:~$ docker ps -a
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS

PORTS NAMES

13f532fa149d debian "echo 'Hola CDA'" 10 seconds ago Created

prueba2

9f2ce863bccc debian "echo 'Hola CDA'" 20 seconds ago Exited (0)
18 seconds ago prueba1
e83ca3fd44d7 hello-world "/hello" 3 minutes ago Exited (0)
3 minutes ago sharp_heisenberg
```

```
usuario@docker:~$ docker start -a prueba2
```

```
Hola CDA
```

```
usuario@docker:~$ docker container prune
```

```
# elimina los contenedores parados
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
13f532fa149d65124a4eabfcddeba1e4dc9b7cfa80948fde6cf824493dcfcd98
9f2ce863bccc0c19a491bc199b3ef24e05607dcd14e9e9ea027430a67f179a44
e83ca3fd44d7a8dc8ddc8642d71be52ad02a636f2c4131daf92667d17ae06bcd
```

- Acceso interactivo (-it) a un contenedor Debian (se ejecuta el comando por defecto *bash*).
    1. Comprobar con *uname -a* que coincide el kernel de anfitrión y contenedores
    2. Acceso interactivo (con -it) en docker run (Nota: -d en docker run desvincula la consola del anfitrión, que posteriormente se vincula explíctamente con docker attach)
    3. Comprobar la diferencia entre salir con exit (terminal el comando por defecto *bash* y finaliza el contenedor) y salir con "secuencia de escape" [Contro]+pq

```
usuario@docker:~$ uname -a
```

```
Linux docker.cda.net 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1
(2023-05-12) x86_64 GNU/Linux
```

```
# ejecución en modo “attached”
usuario@docker:~$ docker run -it --name debian1 --hostname debian1
debian
```

```
root@debian1:/# hostname
```

```
root@debian1:/# uname -a
```

```
Linux debian1 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12)
root@debian1:/
# para salir pulsar [Control]+pq
```

```
# ejecución en modo “dettached (opción -d)”
usuario@docker:~$ docker run -it -d --name debian2 --hostname
debian2 debian
```

```
75836ba55a25357299a365acc8738dd9f2dbe7d666c54412b2055aa5e8edab62
```

```
# ver el estado de los contenedores.
usuario@docker:~$ docker ps
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
75836ba55a25 debian "bash" 3 seconds ago Up 2 seconds debian2

0b9527305db6 debian "bash" 50 seconds ago Up About a minute debian1
```

```
# conectar consola del anfitrion con debian2 (attach)
usuario@docker:~$ docker attach debian2
```

```
root@debian2:/
# para salir pulsar [Control]+pq
```

```
# Ejecutar comandos en debian2 (echo + bash [con acceso interactivo])
usuario@docker:~$ docker exec debianDOS echo "Hola otra vez"
```

```
Hola otra vez
exit
```

- Comprobación de las conexiones a la red por defecto bridge (172.17.0.0/16) en anfitrion (interface docker0 con IP 172.17.0.1) y contenedores (requiere instalar los comandos ip y ping no incluidos en la imagen Docker)

```
usuario@docker:~$ ip address
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state
UP group default qlen 1000
    link/ether 08:00:27:11:11:11 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 scope global enp0s3
    valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP
group default
    link/ether 02:42:f1:42:3c:d6 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
    valid_lft forever preferred_lft forever
...
```

- Acceso interactivo al contenedor (-it) ejecutando un comando /bin/bash sobre el que se invocarán las pruebas (instalar paquetes con apt y comprobaciones con ip y ping)

```
usuario@docker:~$ docker exec -it debian1 /bin/bash
```

```
root@debianUNO:/# apt update
root@debianUNO:/# apt install --yes iproute2 iputils-ping
root@debianUNO:/# ip address
root@debianUNO:/# ping 172.17.0.1 # ping al anfitrion
root@debianUNO:/# ping 172.17.0.3 # ping a debian2
root@debianUNO:/# exit
exit
```

```
usuario@docker:~$ docker exec -it debian2 /bin/bash
```

```
root@debianDOS:/# apt update
root@debianDOS:/# apt install --yes iproute2 iputils-ping
root@debianDOS:/# ip address
root@debianDOS:/# ping 172.17.0.1 # ping al anfitrion
root@debianDOS:/# ping 172.17.0.2 # ping a debian2
root@debianDOS:/# exit
exit
```

- Parar los contenedores y eliminarlos.

```
usuario@docker:~$ docker stop debian1 debian2
usuario@docker:~$ docker ps -a
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
75836ba55a25 debian "bash" 9 minutes ago Exited (137) 8 seconds
ago debian2
0b9527305db6 debian "bash" 11 minutes ago Exited (137) 8 seconds
ago debian1
```

```
usuario@docker:~$ docker rm debian1 debian2
usuario@docker:~$ docker ps -a
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

```

6.2. Redirección de puertos (-p) y compartición de ficheros/directorios con volúmenes.

- Crear contenedores con redirecciones de puertos y un directorio compartido mediante un volumen Docker (**Importante**: el directorio compartido debe existir en el anfitrión antes de configurar el contenedor)

```
usuario@docker:~$ mkdir comun_html
usuario@docker:~$ docker run -it -d --name apache1 \
                        --hostname apache1 -d -p 8081:80 \
                        --volume $PWD/comun_html:/var/www/html debian

usuario@docker:~$ docker run -it -d --name apache2 \
                        --hostname apache2 -d -p 8082:80 \
                        --volume $PWD/comun_html:/var/www/html debian
```

Nota: El directorio del anfitrión comun_html se corresponderá con la ruta /var/www/html (directorio home por defecto del servidor web Apache) en ambos contenedores. Los dos contenedores (apache1 y apache2) tendrán acceso compartido a sus contenidos.

- Acceso interactivo con un /bin/bash e instalación de herramientas (Apache y PHP)

```
usuario@docker:~$ docker exec -it apache1 /bin/bash
```

```
root@apache1:~# apt update
root@apache1:~# apt install --yes apache2 libapache2-mod-php php
iproute2 iputils-ping
root@apache1:~# /etc/init.d/apache2 start
```

```
usuario@docker:~$ docker exec -it apache2 /bin/bash
```

```
root@apache2:~# apt update
root@apache2:~# apt install --yes apache2 libapache2-mod-php php
iproute2 iputils-ping
root@apache2:~# /etc/init.d/apache2 start
```

- Contenido PHP compartido (a crear en el directorio comun_html del anfitrión) [desde un terminal propio]

```
usuario@docker:~$ cd comun_html/
usuario@docker:~/comun_html$ ls -l
usuario@docker:~/comun_html$ sudo rm index.html
usuario@docker:~/comun_html$ sudo echo "Servido por: <?php echo
gethostname(); ?>" > index.php
```

- Prueba de funcionamiento: Desde el anfitrión abrir en un navegador la URL http://localhost:8081 (responderá apache1) y a continuación la URL http://localhost:8082 (responderá apache2) [ver configuración de redireccionamiento de puertos (-p)].
- Parar y eliminar contenedores creados

```
usuario@docker:~$ docker stop apache1 apache2 tercero
usuario@docker:~$ docker rm apache1 apache2 tercero
usuario@docker:~$ docker ps -a
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

```

6.3. Uso conjunto de contenedores y redes

- Conexión MariaBD y PHPMyAdmin
    1. La imagen de MariaBD asume que el directorio con los datos (/var/lib/mysql) sea proporcionado desde el anfitrión, bien con un volumen Docker (opción -v) o con bind mount (opción --mount). De esa forma, los datos almacenados no se perderán al detener el contenedor y estarán disponibles si se vuelve a arrancar. Además, MariaDB define una serie de variables de entorno (a establecer con -e). En concreto, la variable MARIADB_ROOT_PASSWORD se asume que portará la contraseña del usuario adminstrador de la BD (necesaria para conectarse con MariaDB).
    2. La imagen de PHPMyAdmin define una serie de variables de entorno (a establecer con -e). En concreto, la variable PMA_HOST se asume que portará el nombre DNS o dirección IP del servidor MySQL o MariaDB al que se conectará PHPMyAdmin para administrarlo.

```
usuario@docker:~$ mkdir datos_mariadb
usuario@docker:~$ docker network create red_mysql
```

```
395f4f3599e14935b2efdf0f89be0eb382827e66cbf0a04f96e53ec480974c5d
```

```
usuario@docker:~$ docker run -d --name servidor_bd \
                        --network red_mysql \
                        --volume $PWD/datos_mariadb:/var/lib/mysql \
                        -e MARIADB_ROOT_PASSWORD=purple mariadb:latest
```

```
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
...
Status: Downloaded newer image for mariadb:latest
```

```
usuario@docker:~$ docker run -d --name phpmyadmin \
                                  --network red_mysql \
                                   -e PMA_HOST=servidor_bd \
                                   -p 8080:80 phpmyadmin:latest
```

```
Unable to find image 'phpmyadmin:latest' locally
```

```
02 latest: Pulling from library/phpmyadmin
03 ...
04 Status: Downloaded newer image for phpmyadmin:latest
```

- Desde el anfitrión acceder con un navegador a la URL http://localhost:8080.
- Conectarse a PHPMyAdmin con usuario root y contraseña purple .
- Al terminar, parar y eliminar contenedores y redes utilizados

```
01 usuario@docker:~$ docker stop servidor_bd phpmyadmin
02 usuario@docker:~$ docker rm servidor_bd phpmyadmin
03 usuario@docker:~$ docker ps -a
04 usuario@docker:~$ docker network rm red_mysql
05 usuario@docker:~$ docker network ls
```

7. Ejercicios

Tarea 1 : Despliegue de un cluster de balanceo de carga para un servidor WordPress (con MariaDB).

La tarea consiste en realizar manualmente un despliegue de contenedores Docker siguiendo el esquema que se muestra en la siguiente figura:

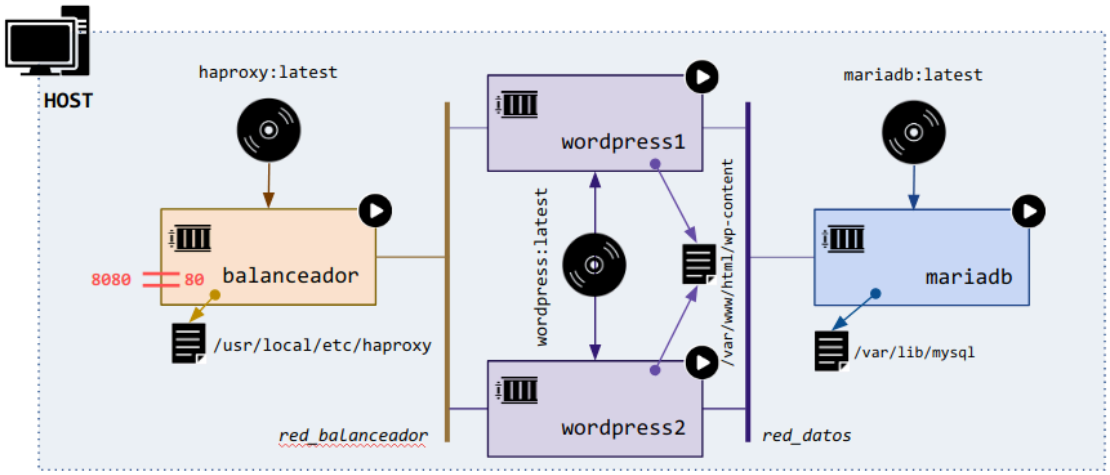

Fig 2. Entorno a desplegar.

Tal y como se puede observar en Fig 2, se deben desplegar un total de cuatro contenedores que ofrecen diferentes servicios.

- Un contenendor HAProxy haciendo de frontal (con el puerto 80 de ese contenedor redirigido al puerto 8080 del anfitrión para verificar el acceso)

- Dos contenedores con el gestor de contenidos Wordpress instalado sobre los que repartirá las peticiones el balanceador HAProxy
- Un contenedor con una base de datos MariaDB compartida por los dos Wordpress.

Tal y como se puede observar, *balanceador*, *wordpress1* y *wordpress2* deben pertenecer a la misma red ya que el *balanceador* debe poder distribubir las diferentes peticiones entre contenedores WordPress. Por otro lado, tanto wordpress1 como wordpress2 requieren acceso a la base de datos (mariadb). Esto implica que comparta red con el contenedor *mariadb*.

Pasos a realizar:

1. Crear en el anfitrión los directorios a usar con -v (volume mapping)
    a. Para /var/lib/mysql del contenedor MariaDB
    b. Para /var/www/html/wp-content de los contenedores Wordpress
    c. Para /usr/local/etc/haproxy del contenedor HAProxy
2. Crear el fichero de configuración de HAProxy (haproxy.cfg) en frontend ajustar bind a 0.0.0.0:80 ó *:80 en backend ajustar la dirección de los server usando los nombres de los contenedores Wordpress
3. Crear las redes necesarias
4. Crear los contenedores necesarios, ajustando las variables de entorno (con -e) según corresponda
5. Establecer las conexiones de red adicionales

Tarea 2 : Entregable (memoria)

1. Formato: PDF
2. Nombre: Practica6_(apellidos_nombre).pdf
3. Describir:
    - Los elementos del entorno que se han creado (redes y contenedores, redirecciones, volúmenes o bind mounts, etc), bien de forma textual, con esquemas o combinando ambas formas.
    - Los aspectos más relevantes de su configuración (redirecciones, mounts, etc).
    - Aportar los comandos utilizados, la salida que dan (si es relevante) y una breve descripción de lo que hace cada uno.
    - Añadir capturas del resultado final una vez se acceda mediante un

navegador al anfitrión con la URL http://localhost:8080

4. NOTA: No es necesario completar la instalación de los Wordpress basta con constatar que se muestra la página de configuración.