

Docker. Práctica 4

Modulo 1.....	1
Configuración de contenedores con variables de entorno	1
Configuración de un contenedor con la imagen mariadb	1
Accediendo a servidor de base de datos desde el exterior	2
Contenedor	3
Ejecución simple de contenedores	3
Domonio	4
HolaMundo	4
El "Hola Mundo" de docke.....	4
interactivo	5
Web.....	6
Modificación del contenido servidor por el servidor web	6

Modulo 1

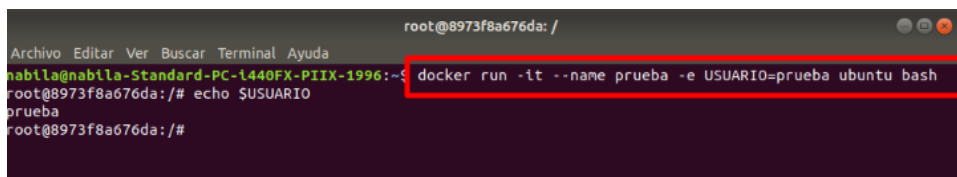
Configuración

Lleva a cabo al menos tres de los ejemplos mostrados en el módulo y documentalo en tu repositorio incluyendo capturas de pantalla.

Configuración de contenedores con variables de entorno

crear un contenedor que necesita alguna configuración específica, lo que vamos a hacer es crear variables de entorno en el contenedor, para que el proceso que inicializa el contenedor pueda realizar dicha configuración.

Para crear una variable de entorno al crear un contenedor usamos el flag `-e` o `--env`:



```
root@8973f8a676da: /
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-1440FX-PIIX-1996:~$ docker run -it --name prueba -e USUARIO=prueba ubuntu bash
root@8973f8a676da: /# echo $USUARIO
prueba
root@8973f8a676da: /#
```

Configuración de un contenedor con la imagen mariadb

En ocasiones es obligatorio el inicializar alguna variable de entorno para que el contenedor pueda ser ejecutado. Si miramos la documentación en Docker Hub de la imagen mariadb, observamos que podemos definir algunas variables de entorno para la creación y configuración del contenedor (por ejemplo: `MYSQL_DATABASE`, `MYSQL_USER`, `MYSQL_PASSWORD`,...). Pero hay una que la tenemos

que indicar de forma obligatoria, la contraseña del usuario root (MYSQL_ROOT_PASSWORD), por lo tanto:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw mariadb
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
b65bcf19d144: Pull complete
0370eb708844: Pull complete
d283e803fc62: Pull complete
1288fc1c0995: Pull complete
a2b92b7101b0: Pull complete
aaf748c1cd8e: Pull complete
a8abf8355411: Pull complete
426bf5c6de40: Pull complete
Digest: sha256:ec97b993b11423a5b60448c85e01e3f696aaf6dfbf6f83c12d03bb02563f134e
Status: Downloaded newer image for mariadb:latest
f75f8d76480b40fc71fc83b93a3bb88eaa2dba1b5dd902b34ffe0a3c1cde9038
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Podemos ver que se ha creado una variable de entorno:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
f75f8d76480b   mariadb   "docker-entrypoint.s..." 39 seconds ago Up 38 seconds 3306/tcp       some-mariadb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

ejecutar:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it some-mariadb env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=f75f8d76480b
TERM=xterm
MYSQL_ROOT_PASSWORD=my-secret-pw
GOSU_VERSION=1.14
LANG=C.UTF-8
MARIADB_VERSION=1:10.11.2+maria-ubu2204
HOME=/root
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para acceder

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it some-mariadb bash
root@f75f8d76480b:/#
```

Accediendo a servidor de base de datos desde el exterior

vamos a mapear los puertos para acceder desde el exterior a la base de datos:

Lo primero que vamos a hacer es eliminar el contenedor anterior:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker rm -f some-mariadb
some-mariadb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

vamos a crear otro contenedor, pero en esta ocasión vamos a mapear el puerto 3306 del anfitrión con el puerto 3306 del contenedor:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d -p 3306:3306 --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw mariadb
d13cfa52c8c326cb343928be8de23a77393c09a3377a2051450d7403ce5cd6
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Comprobamos

```
d13cfa52c8c326cb343928b0e8de23a77393c09a3377a7a2051450d7403ce5cd0
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." 36 seconds ago Up 34 seconds 0.0.0.0:3306->3306/tcp
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

desde nuestro equipo (donde hemos instalado un cliente de mysql) nos conectamos que tiene la ip 127.0.0.1 vamos a conectarnos a la base de datos (hay que tener instalado el cliente de mariadb):

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ mysql -u root -p -h 127.0.0.1
Enter password:
```

Contenedor

Ejecución simple de contenedores

Con el comando run vamos a crear un contenedor donde vamos a ejecutar un comando:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run ubuntu echo 'Hello word'
Hello word
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Comprobamos con este comando:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  NAMES                CREATED        STATUS        PORTS
247a6c5efcbd   ubuntu    "echo 'Hello word'"      pedantic_germain     40 seconds ago Exited (0) 38 seconds ago
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." some-mariadb         12 minutes ago Up 12 minutes 0.0.0.0:3
306->3306/tcp, :::3306->3306/tcp
8973f8a676da   ubuntu    "bash"                   prueba               24 minutes ago Exited (0) 22 minutes ago
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Con el comando docker images podemos visualizar las imágenes que ya tenemos descargadas en nuestro registro local:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dockerfile    latest   8fec96b2307f   6 days ago    615MB
wordpress     latest   8fec96b2307f   6 days ago    615MB
mariadb       latest   6e11fcfc66ad   7 days ago    401MB
nginx        latest   904b8cb13b93   7 days ago    142MB
ubuntu       latest   74f2314a03de   8 days ago    77.8MB
hello-world   latest   feb5d9fea6a5   17 months ago 13.3kB
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it ubuntu bash
root@db00a47b9856:/#
```

Para ver los contenedores que no se están ejecutando:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." 24 minutes ago Up 24 minutes 0.0.0.0:3306->3306/tcp
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para eliminar el contenedor podemos identificarlo con su id:

Ejecutamos el comando :

Vamos a comprobar que todo funciona creando nuestro primer contenedor desde la imagen hello-world:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run hello-world

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/

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$

```

interactivo

Ejecutando un contenedor interactivo

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor1 ubuntu bash
root@6098786c8364:/#

```

El contenedor se para cuando salimos de él. Para volver a conectarnos a él:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker start contenedor1
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$

```

Si el contenedor se está ejecutando podemos ejecutar comandos en él con el subcomando exec:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker start contenedor1
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec contenedor1 ls -al
total 56
drwxr-xr-x 1 root root 4096 Mar  9 17:50 .
drwxr-xr-x 1 root root 4096 Mar  9 17:50 ..
-rwxr-xr-x 1 root root  0 Mar  9 17:50 .dockerenv
lrwxrwxrwx 1 root root  7 Mar  1 02:03 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 18  2022 boot
drwxr-xr-x 5 root root 360 Mar  9 17:51 dev
drwxr-xr-x 1 root root 4096 Mar  9 17:50 etc
drwxr-xr-x 2 root root 4096 Apr 18  2022 home
lrwxrwxrwx 1 root root  7 Mar  1 02:03 lib -> usr/lib
lrwxrwxrwx 1 root root  9 Mar  1 02:03 lib32 -> usr/lib32
lrwxrwxrwx 1 root root  9 Mar  1 02:03 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Mar  1 02:03 libx32 -> usr/libx32
drwxr-xr-x 2 root root 4096 Mar  1 02:03 media
drwxr-xr-x 2 root root 4096 Mar  1 02:03 mnt
drwxr-xr-x 2 root root 4096 Mar  1 02:03 opt
dr-xr-xr-x 309 root root  0 Mar  9 17:51 proc
drwx----- 1 root root 4096 Mar  9 17:51 root
drwxr-xr-x 5 root root 4096 Mar  1 02:06 run
lrwxrwxrwx 1 root root  8 Mar  1 02:03/sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Mar  1 02:03 srv
dr-xr-xr-x 13 root root  0 Mar  9 17:51 sys
drwxrwxrwt 2 root root 4096 Mar  1 02:06 tmp
drwxr-xr-x 14 root root 4096 Mar  1 02:03 usr
drwxr-xr-x 11 root root 4096 Mar  1 02:06 var
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$

```

Con la orden docker restart reiniciamos el contenedor, lo paramos y lo iniciamos.

Para mostrar información de un contenedor ejecutamos docker inspect:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker inspect contenedor1
[
  {
    "Id": "6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca",
    "Created": "2023-03-09T17:50:50.266934461Z",
    "Path": "bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 9829,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-03-09T17:51:50.86388607Z",
      "FinishedAt": "2023-03-09T17:51:38.811513904Z"
    },
    "Image": "sha256:74f2314a03de34a0a2d552b805411fc9553a02ea71c1291b815b2f645f565683",
    "ResolvConfPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/hostname",
    "HostsPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/hosts",
    "LogPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca-json.log",
    "Name": "/contenedor1",
    "RestartCount": 0
  }
]
```

En realidad, todas las imágenes tienen definidas un proceso que se ejecuta, en concreto la imagen ubuntu tiene definida por defecto el proceso bash, por lo que podríamos haber ejecutado:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor3 ubuntu
root@73ff49e7b2ee:/#
```

Web

Creando un contenedor con un servidor web

Tenemos muchas imágenes en el registro público **docker hub**, por ejemplo podemos crear un servidor web con apache 2.4:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name my-apache-app -p 8080:80 httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
3f9582a2cbe7: Already exists
9423d69c3be7: Pull complete
d1f584c02b5d: Pull complete
758a20a64707: Pull complete
08507f82f391: Pull complete
Digest: sha256:76618ddd53f315a1436a56dc84ad57032e1b2123f2f6489ce9c575c4b280c4f4
Status: Downloaded newer image for httpd:2.4
1ecd9150e58d2338743d8d5ea538a15148289d1b73fabab18ed5d2d4cddb8b89
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para probarlo accede desde un navegador a <http://localhost:8080>



Para acceder al log del contenedor podemos ejecutar:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker logs my-apache-app
AH00558: httpd: Could not reliably determine the server's fully qualified domain
```

Modificación del contenido servidor por el servidor web

Accediendo de forma interactiva al contenedor y haciendo la modificación:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it my-apache-app bash
root@1ecd9150e58d:/usr/local/apache2#
```

Ejecutando directamente el comando de creación del fichero index.html en el contenedor:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec my-apache-app bash -c 'echo "<h1>Curso Docker</h1>" > /usr/local/apache2/htdocs/index.html'
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Accedemos al navegador:

