03 Volumenes y redes

Hasta este punto, cuando nosotros creamos contenedores en base a imágenes y en el momento en que eliminamos el contenedor, esta información se pierde. Con el objetivo de hacer esta información persistente tenemos los volúmenes. La idea será indicarle al contenedor que la información que tiene que emplear estará y será almacenada en el volumen. En esta sección vamos a emplear una red que permita cominicar varios contenedores.

1. Tarea previa

Antes de nada, vamos a realizar una tarea de contenedores para repasar los conceptos vistos. Tendrá los siguientes puntos:

1. Montar la imagen de MariaDB con el tag jammy, publicar en el puerto 3306 del contenedor con el puerto 3306 de nuestro equipo, colocarle el nombre al contenedor de world-db (--name world-db) y definir las siguientes variables de entorno:

```
MARIADB_USER=example-user
MARIADB_PASSWORD=user-password
MARIADB_ROOT_PASSWORD=root-secret-password
MARIADB_DATABASE=world-db
```

2. Conectarse usando Table Plus o la extensión MySQL de Weijan Chen si estamos en VSCode a la base de datos con las credenciales del usuario (NO EL ROOT)
3. Conectarse a la base de datos world-db
4. Ejecutar el query de creación de tablas e inserción proporcionado en el archivo world.sql
5. Revisar que efectivamente tengamos la data

2. Solución

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name world-db `
>> -dp 3306:3306 `
>> -e MARIADB_USER=example-user `
>> -e MARIADB_PASSWORD=user-password `
>> -e MARIADB_ROOT_PASSWORD=root-secret-password `
>> -e MARIADB_DATABASE=world-db `
>> mariadb:jammy
529937e5de64377b418015fd37e2005fd6ba1ac896fbbf7d51dd00eeb116c56f

PS C:\Users\Carballeira\Documents\Docker> docker container ls
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS
PORTS                     NAMES
529937e5de64   mariadb:jammy   "docker-entrypoint.s…"   10 seconds ago   Up 9
seconds   0.0.0.0:3306->3306/tcp   world-db
```

El problema es que una vez conectados, si eliminamos el contenedor perderemos la información. Para evitar esto vamos a hacer uso de volumenes.

```
PS C:\Users\Carballeira\Documents\Docker> docker container rm -f 529
529
```

3. Creación de un volumen

Podemos crear un volumen para almacenar la información de nuestro contenedor de forma persistente de la siguiente manera.

```
PS C:\Users\Carballeira\Documents\Docker> docker volume create world-db
world-db
```

Para ver la información del contenedor podemos hacerlo de la siguiente manera.

```
PS C:\Users\Carballeira\Documents\Docker> docker volume inspect world-db
[
    {
        "CreatedAt": "2024-12-15T14:17:04Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/world-db/_data",
        "Name": "world-db",
        "Options": null,
        "Scope": "local"
    }
]
```

Si quisieramos crear un contenedor de mariadb que haga uso del volumen anteriormente creado tendríamos que ejecutar el comando siguiente que indica que el volumen `world-db` va a contener la información almacenada en `/var/lib/mysql`.

```
docker container run `
--name world-db `
-dp 3306:3306 `
-e MARIADB_USER=example-user `
-e MARIADB_PASSWORD=user-password `
-e MARIADB_ROOT_PASSWORD=root-secret-password `
-e MARIADB_DATABASE=world-db `
-v world-db:/var/lib/mysql `
mariadb:jammy
```

A mayores, en lugar de tableplus o la extensión MySQL de VSCode, vamos hacer uso de phpmyadmin para lo que vamos a crear un contenedor en el puerto 8090 para este servicio.

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name phpmyadmin `
>> -d `
>> -e PMA_ARBITRARY=1 `
>> -p 8090:80 `
>> phpmyadmin:5.2.0-apache
Unable to find image 'phpmyadmin:5.2.0-apache' locally
5.2.0-apache: Pulling from library/phpmyadmin
01b5b2efb836: Pulling fs layer
e98ea42e90a6: Pulling fs layer
d25586855c63: Pulling fs layer
139d4815e950: Pulling fs layer
1de46a46cfcd: Pulling fs layer
139d4815e950: Download complete
d22a5530765a: Download complete
0c7c6d5ce997: Download complete
dc59c4386191: Download complete
9eaa1559de60: Download complete
45244a9928d1: Download complete
9a420fd884ad: Download complete
0b85949bbffa: Download complete
96e5415e261a: Download complete
ed7aad4fc2fb: Download complete
1ff2782a8ae8: Download complete
9cc46e699e97: Download complete
9a8d67ebc9db: Download complete
Digest: sha256:7c8751488b72255047bdc38ea1e9bf6b33b163a3c15ae276b2ac3c1a633a53c6
Status: Downloaded newer image for phpmyadmin:5.2.0-apache
c6a54bd0069f69999a3dac43c7cbdab5eabeaf9a427a9189d60acdc0ce33e85f
```

Si ahora nosostros accedemos a la URL localhost:8090 podremos ver que tendremos la web de phpmyadmin pero no vamos a poder conectarnos al contenedor que almacena la base de datos de mariadb. Para ello vamos a tener que hacer uso de redes de contenedores que se explican en la siguiente sección.

4. Redes de contenedores

enlace de interes

En este punto, podemos ver las redes creadas por defecto.

```
PS C:\Users\Carballeira\Documents\Docker> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
27470d59f292   bridge    bridge    local
0de5fed7fe42   host      host      local
8d2078019d88   none      null      local
```

Vamos a crear nuestra primera red de contenedores que va a ser utilizada por nuestros dos contenedores y va a permitir la conexión de ambos.

```
PS C:\Users\Carballeira\Documents\Docker> docker network create world-app
ec70090dda2b4bf496dc4b2fe60ba2bcb0b536f6d2587efe74779f5e3662aa79

PS C:\Users\Carballeira\Documents\Docker> docker network ls
NETWORK ID     NAME        DRIVER    SCOPE
27470d59f292   bridge      bridge    local
0de5fed7fe42   host        host      local
8d2078019d88   none        null      local
ec70090dda2b   world-app   bridge    local
```

Tendremos en este punto que añadir los contenedores a nuestra red e inspeccionar esta para poder verificar si han sido añadidos.

```
PS C:\Users\Carballeira\Documents\Docker> docker network connect world-app 18c
PS C:\Users\Carballeira\Documents\Docker> docker network connect world-app 46d
PS C:\Users\Carballeira\Documents\Docker> docker network inspect world-app
[
    {
        "Name": "world-app",
        "Id": "ec70090dda2b4bf496dc4b2fe60ba2bcb0b536f6d2587efe74779f5e3662aa79",
        "Created": "2024-12-15T15:16:24.258661095Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "18c87492a71623753233e4f285a5ba74e77cf1958c6e365e56a9cf5335e021b9": {
                "Name": "phpmyadmin",
                "EndpointID":
"81161324b8cf19cd8e04562522762944b319699fba42a0294163895d5fe7945f",
                "MacAddress": "02:42:ac:12:00:02",
```

```
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "46dd62dc13d33195f555c5f901f5a12f0dc2e694901f260afe24f4a0f93de4f5": {
                "Name": "world-db",
                "EndpointID":
"6296df0100ceb08ea89107c0d8e2b811ebcb387310fa4129968e822fdd76aea1",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

En este punto vamos a poder conectarnos con phpmyadmin a mi contenedor de base de datos.

Es importante saber que sería más sencillo que una vez creada la red se asignara el contenedor a la misma directamente de la siguiente manera.

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name phpmyadmin `
>> -dp 8090:80  `
>> --env PMA_ARBITRARY=1 `
>> --network world-app `
>> phpmyadmin:5.2.0-apache
a652e2b741f6a4f5b9badba794a93c1f2d10ab75a0d5dfd44bb9f9f51b5bb08b

PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> -dp 3306:3306 `
>> --name world-db `
>> --env MARIADB_USER=example-user `
>> --env MARIADB_PASSWORD=user-password `
>> --env MARIADB_ROOT_PASSWORD=root-secret-password `
>> --env MARIADB_DATABASE=world-db `
>> --volume world-db:/var/lib/mysql `
>> --network world-app `
>> mariadb:jammy
36e7c407f2599ec9b7803e311d986b5b1acd8741780cc2f493c43f41caffca0c

PS C:\Users\Carballeira\Documents\Docker> docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED
STATUS         PORTS                     NAMES
36e7c407f259   mariadb:jammy              "docker-entrypoint.s…"   4 seconds ago
Up 3 seconds   0.0.0.0:3306->3306/tcp   world-db
a652e2b741f6   phpmyadmin:5.2.0-apache    "/docker-entrypoint.…"   7 seconds ago
Up 7 seconds   0.0.0.0:8090->80/tcp     phpmyadmin
```

5. Ejercicio Bind Volumes y terminal interactiva

Vamos a descargar la imagen de apache y crear el contenedor.

```
PS C:\Users\Carballeira\Documents\Docker> docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
9bd25d4f7b77: Download complete
d7ad38c6dd97: Download complete
4f4fb700ef54: Download complete
bc0965b23a04: Download complete
79b49624e34b: Download complete
7d9f97915db2: Download complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest

PS C:\Users\Carballeira\Documents\Docker> docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    f4c5139eda46   5 months ago   220MB
```

Vamos a ejecutar el contenedor apache en modo interactivo y vamos a enlazar la web de la carpeta de Recursos ejemploWeb.

```
PS C:\Users\Carballeira\Documents\Docker\Recursos\ejemploWeb> docker container run
`
>> --name apache `
>> -dp 3000:80 `
>> -it `
>> -v ${PWD}:/usr/local/apache2/htdocs `
>> httpd
041c83835b7eea9b51195f55cefac384aee9c3daf5a0525c16754a4e4f3cabfa

PS C:\Users\Carballeira\Documents\Docker\Recursos\ejemploWeb> docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS
NAMES
041c83835b7e   httpd     "httpd-foreground"   5 seconds ago   Up 4 seconds
0.0.0.0:3000->80/tcp   apache
```

Nota: Por una parte, cuando usamos -it en conjunto, podremos interactuar con el contenedor en tiempo real, lo cual es útil cuando necesitas ejecutar comandos dentro del contenedor o depurar algo manualmente.

Nota2: En caso de sistemas unix, tendríamos que escribir: -v "$PWD":/usr/local/apache2/htdocs.

Tal y como podemos apreciar en la ruta /usr/local/apache2/htdocs del contenedor está el código de la web de ejemplo. Podremos verlo en la web.

```
PS C:\Users\Carballeira\Documents\Docker\Recursos\ejemploWeb> docker exec -it
apache bash
root@041c83835b7e:/usr/local/apache2# cd /usr/local/apache2/htdocs/

root@041c83835b7e:/usr/local/apache2/htdocs# ls -l
total 12
drwxrwxrwx 1 root root  4096 Dec 15 22:30 css
drwxrwxrwx 1 root root  4096 Dec 15 22:30 enunciado
drwxrwxrwx 1 root root  4096 Dec 15 22:30 img
-rwxrwxrwx 1 root root 10136 Dec 15 22:29 index.html

root@041c83835b7e:/usr/local/apache2/htdocs# head index.html
<!DOCTYPE html>
<html >
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link type="image/x-icon" rel="icon" href="img/logo.png" />
    <link type="text/css" rel="stylesheet" href="css/styles.css" />
    <title>Maquetación</title>
  </head>
  <body>
```

Tarea

Realiza los siguientes pasos:

1. Crea una red llamada tarea2 para el ejercicio en la que van a trabajar los contenedores.
2. Instala un servicio phpmyadmin y una base de datos mysql.
3. El contenido de la base de datos tiene que estar almacenado en un volumen llamado volumenTarea2.
4. Hacer un informe con los pasos explicados en markdown y una imagen de la conexión.