> Tarea realizada por Manuel Macho Calvín

Ejercicio - redes

Tarea 1:

Crear una red bridge `redbd`.

```
docker network create redbd
docker network inspect redbd
```

```
manuelmc09@cliente:~$ docker network create redbd
84792e0307df0a51e41569287cbd35c6d30d5cb52f6437a3c852dff566440b75
manuelmc09@cliente:~$ docker network inspect redbd
[
    {
        "Name": "redbd",
        "Id": "84792e0307df0a51e41569287cbd35c6d30d5cb52f6437a3c852dff566440b75",
        "Created": "2022-01-20T17:32:26.970142243+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Tarea 2:

Crea un contenedor con una imagen de `mariaDB` que estará en la red `redbd` . Este contenedor se ejecutará en segundo plano, y será accesible a través del puerto 3306. (Es necesario definir la contraseña del usuario `root` y un volumen de datos persistente).

Pantallazos donde se vea el contenedor creado y en ejecución.

Primero creamos el contenedor:

```
docker run -d --name sql_mariadb -v data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -p 81:3306 -network redbd mariadb
```

Comprobamos que está en ejecución:

```
docker ps -a
```

```
manuelmc09@cliente:~$ docker run -d --name sql_mariadb -v data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -p 81:3306 --
network redbd mariadb
14abfa0a43293d2f8158a17460d8eb15fc361fefa70cabbea86f811a22803b0e
manuelmc09@cliente:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
  NAMES
14abfa0a4329   mariadb   "docker-entrypoint.s…"   8 seconds ago   Up 5 seconds   0.0.0.0:81->3306/tcp, :::81->3306/tcp
  sql_mariadb
```

Tarea 3.

Pantallazos donde se vea el contenedor creado y en ejecución.

Creamos un contenedor con `Adminer` que se pueda conectar al contenedor de la BD.

```
docker run -d --rm --network redbd -e PMA_ARBIRARY=1 -p 80:80 adminer
```

Comprobamos que está en ejecución:

```
docker ps
```

```
manuelmc09@cliente:~$ docker run -d --rm --network redbd -e PMA_ARBITRARY=1 -p 80:80 adminer
79505170a5e12bc7bb7305daf30a66d34caa05662da70bd7e8f898a7fd896f08
manuelmc09@cliente:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
        NAMES
79505170a5e1   adminer   "entrypoint.sh docke…"   7 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 808
0/tcp   jovial_boyd
14abfa0a4329   mariadb   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:81->3306/tcp, :::81->3306/tcp
        sql_mariadb
```

Pantallazo donde se vea el acceso a la BD a través de la interfaz web de `Adminer` .

Pantallazo donde se entre a la consola del servidor web en modo texto y se compruebe que se ha creado la BD.

Tarea 4:

Comprobar que el contenedor `Adminer` puede conectar con el contenedor `mysql` abriendo un navegador web y accediendo a la URL: **http://localhost:8080** .

Borrar los contenedores, la red y los volúmenes utilizados.

Primero paramos el contenedor en ejecución y luego lo eliminamos. (Aquí podríamos optar por eliminar el contenedor y su volumen al mismo tiempo con la opción: `docker rm -v sql_mariadb`)

```
docker stop sql_mariadb
docker rm sql_mariadb
```

Seguidamente eliminaremos el volumen. Podremos ver la lista de volúmenes existentes y elegir el que pertenece al contenedor.

```
docker volume ls
docker rm data
```

Eliminaremos la red bridge creada:

```
docker network rm redbd
```