

Ejercicio - redes

Tarea 1:

Tarea 2:

Tarea 3:

Tarea 4:

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 /home/usuario/data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=root -p 3306:3306 --network redbd mariadb
```

Comprobamos que está en ejecución:

```
docker ps
```

```
manuelmc09@cliente:~$ docker run -d --name sql_mariadb --network redbd -p 3306:3306 -v /home/usuario/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root mariadb
1d006db70976a906e249bfe3490926a6d127b25592e0e5f13a519fa60cf4123a
manuelmc09@cliente:~$ docker ps
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|--------------|---------|--------------------------|----------------|--------------|---|
| 1d006db70976 | mariadb | "docker-entrypoint.s..." | 12 seconds ago | Up 8 seconds | 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp |

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 --name c_adminer -p 8080:8080 --network redbd adminer
```

Comprobamos que está en ejecución:

```
docker ps
```

```
manuelmc09@cliente:~$ docker run -d --name c_adminer -p 8080:8080 --network redbd adminer
30557d84549f9d931405eee53be75549dd7169a37c43164e72f3696c4116b80e
manuelmc09@cliente:~$ docker ps
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|--------------|---------|--------------------------|---------------|--------------|---|
| 30557d84549f | adminer | "entrypoint.sh docke..." | 7 seconds ago | Up 4 seconds | 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp |
| 1d006db70976 | mariadb | "docker-entrypoint.s..." | 2 minutes ago | Up 2 minutes | 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp |

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>.

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

Login - Adminer

localhost:8080

Idioma: Español

Adminer 4.8.1

Login

| | |
|------------------------|------------------------|
| Motor de base de datos | MySQL |
| Servidor | db |
| Usuario | |
| Contraseña | root |
| Base de datos | contenedor_maria db |

Login ☐ Guardar contraseña

Accedemos a la BD del servidor en nuestro caso `sql_mariadb`

Login

| | |
|------------------------|-------------|
| Motor de base de datos | MySQL |
| Servidor | sql_mariadb |
| Usuario | root |
| Contraseña | |
| Base de datos | |

Login ☐ Guardar contraseña

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



Borrar los contenedores, la red y los volúmenes utilizados.

Primero paramos los contenedores 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 stop c_adminer
```

Seguidamente eliminaremos los contenedores

```
docker -v rm sql_mariadb
docker rm c_adminer
```

Eliminaremos la red bridge creada:

```
docker network rm redbd
```

```
manuelmc09@cliente:~$ docker stop sql_mariadb
sql_mariadb
manuelmc09@cliente:~$ docker stop c_adminer
c_adminer
manuelmc09@cliente:~$ docker -v rm sql_mariadb
sql_mariadb
manuelmc09@cliente:~$ docker rm c_adminer
c_adminer
manuelmc09@cliente:~$ docker network rm redbd
redbd
```