4 - Redes en Docker - ejercicios

Trabajar con redes docker

1.Vamos a crear dos redes de ese tipo (BRIDGE) con los siguientes datos:

1. Red1

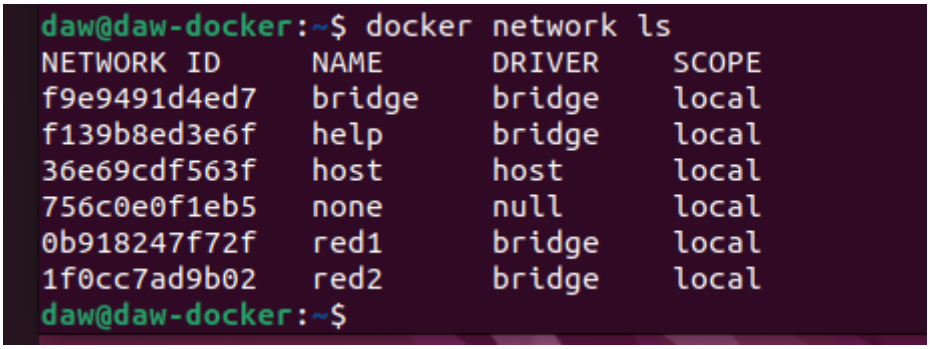
- Nombre: red1
- Dirección de red: 172.28.0.0
- Máscara de red: 255.255.0.0
- Gateway: 172.28.0.1

```
daw@daw-docker: ~$docker network create -d bridge --
subnet=172.28.0/16 --gateway=172.28.0.1 red1
```

2.Red2

- Nombre: red2
- Es resto de los datos será proporcionados automáticamente por Docker.

```
daw@daw-docker:~$ docker network create red2
```



2.Poner en ejecución un contenedor de la imagen ubuntu:20.04 que tenga como hostnamehost1 , como IP 172.28.0.10 y que esté conectado a la red1. Lo llamaremos u1.

```
daw@daw-docker:~$ docker run -it --name u1 --network red1 --ip
172.28.0.10 --hostname host1 ubuntu:20.04
```

3. Entrar en ese contenedor e instalar la aplicación ping (apt update && apt install inetutils-ping).

```
root@host1:/# apt update
root@host1:/# apt install inetutils-ping
```

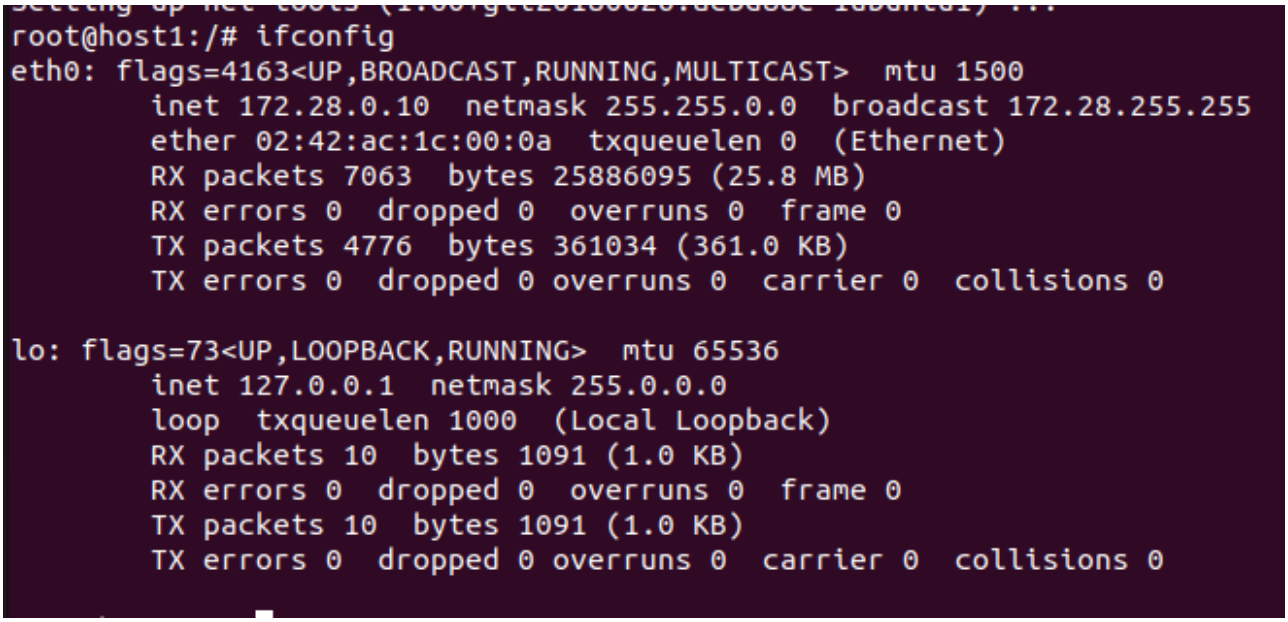
4. Poner en ejecución un contenedor de la imagen ubuntu:20.04 que tenga como hostname host2 y que esté conectado a la red2. En este caso será docker el que le de una IP correspondiente a esa red. Lo llamaremos u2 .

```
daw@daw-docker:~$ docker run -it --name u2 --network red2 --
hostname host2 ubuntu:20.04
```

5. Entrar en ese contenedor e instalar la aplicación ping (apt update && apt install inetutils-ping).

```
root@host1:/# apt update
root@host1:/# apt install inetutils-ping
```

6.Pantallazo donde se vea la configuración de red del contenedor u1.



7.Pantallazo donde se vea la configuración de red del contenedor u2.

```
root@host2:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.0.2  netmask 255.255.0.0  broadcast 172.18.255.255
        ether 02:42:ac:12:00:02  txqueuelen 0  (Ethernet)
        RX packets 10382  bytes 26107298 (26.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7335  bytes 504763 (504.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 10  bytes 1091 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 1091 (1.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

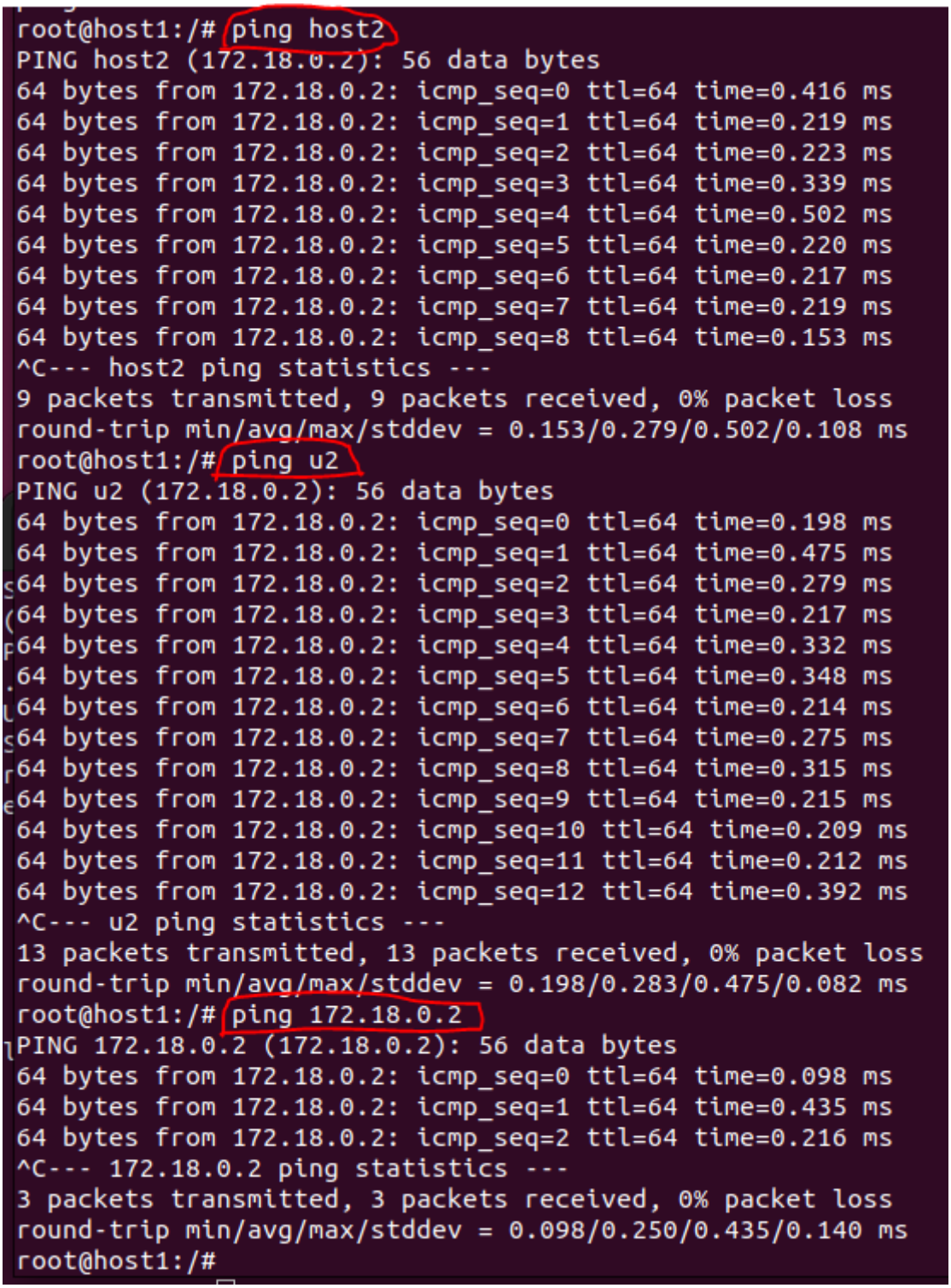
8.Pantallazo donde desde cualquiera de los dos contenedores se pueda ver que no podemos hacer ping al otro ni por ip ni por nombre.

```
root@host1:/# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
^C--- 172.18.0.2 ping statistics ---
9 packets transmitted, 0 packets received, 100% packet loss
```

```
root@host1:/# ping u2
ping: unknown host
```

9.Pantallazo donde se pueda comprobar que si conectamos el contenedor u1 a la red2 (con docker network connect), desde el contenedor u1, tenemos acceso al contenedor u2 mediante ping, tanto por nombre como por ip.

```
daw@daw-docker:~$ docker network connect red2 u1
```



Despliegue de Nextcloud + mariadb

1.Crea una red de tipo bridge.

```
daw@daw-docker:~$ docker network create redMariaNext
```

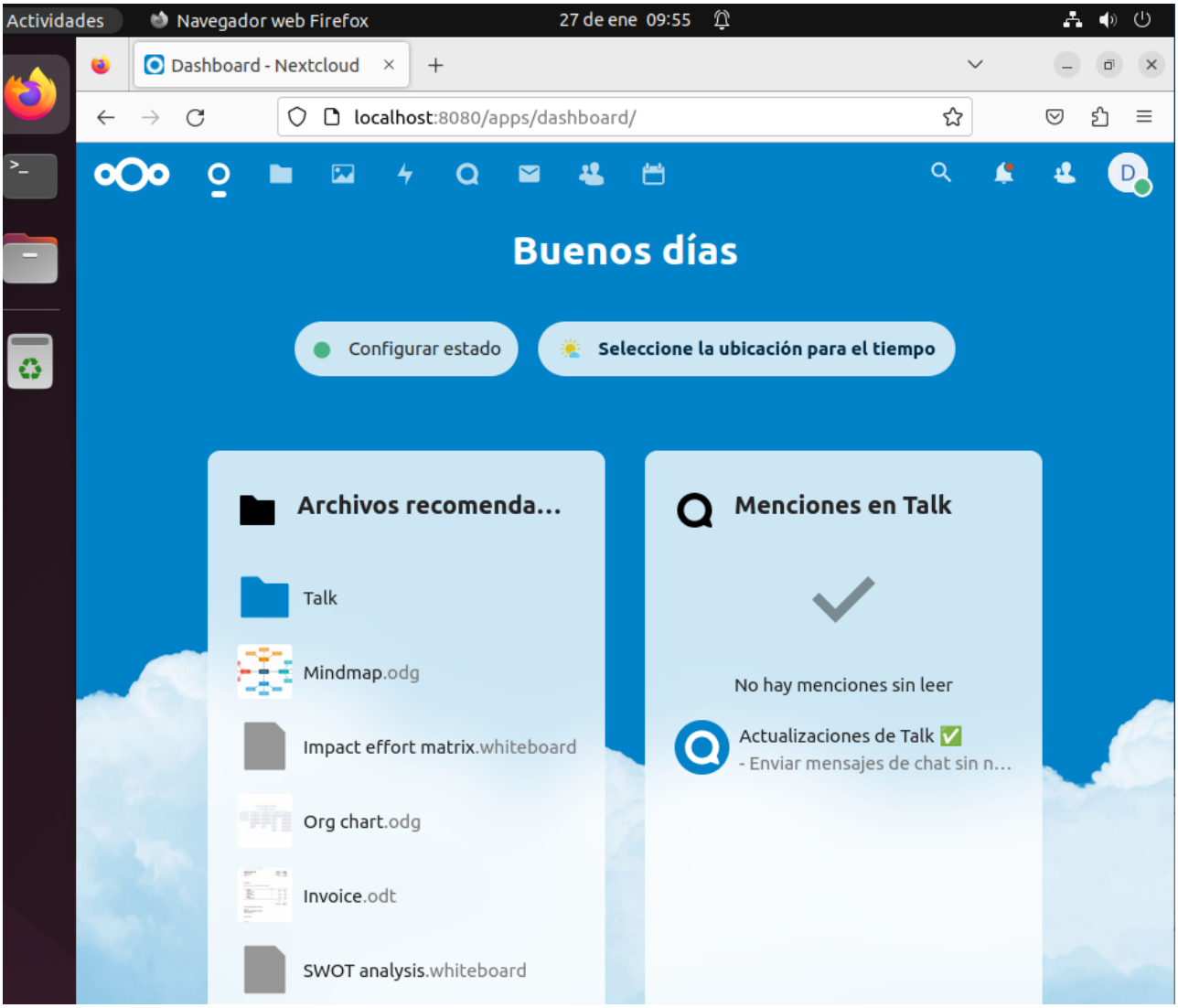
2. Crea el contenedor de la base de datos conectado a la red que has creado. La base de datos se debe configurar para crear una base de dato y un usuario. Además el contenedor debe utilizar almacenamiento (volúmenes o bind mount) para guardar la información. Puedes seguir la documentación de mariadb o la de PostgreSQL .

```
daw@daw-docker:~$ docker run -d --name miMariadb
--network redMariaNext
-v /opt/mysql_base1:/var/lib/mysql
-e MYSQL_DATABASE=base1
-e MYSQL_USER=usuario1
-e MYSQL_PASSWORD=diego
-e MYSQL_ROOT_PASSWORD=diego
mariadb:10.5
```

3. A continuación, siguiendo la documentación de la imagen nextcloud , crea un contenedor conectado a la misma red, e indica las variables adecuadas para que se configure de forma adecuada y realice la conexión a la base de datos. El contenedor también debe ser persistente usando almacenamiento.

```
daw@daw-docker:~$ docker run -d -p 8080:80 --name nextMaria --
network redMariaNext
--link miMariadb:mariadb
-v nextcloud:/var/www/html  nextcloud
```

4. Accede a la aplicación usando un navegador web.



5.Pantallazo con la instrucción para crear el contenedor de la base de datos.

```
daw@daw-docker:~$ docker run -d --name miMariadb --network redMariaNext -v /opt/
mysql_base1:/var/lib/mysql -e MYSQL_DATABASE=base1 -e MYSQL_USER=usuario1 -e MYS
QL_PASSWORD=diego -e MYSQL_ROOT_PASSWORD=diego mariadb:10.5
Unable to find image 'mariadb:10.5' locally
10.5: Pulling from library/mariadb
846c0b181fff: Already exists
2279a7485340: Pull complete
30c7fe7ba3fd: Pull complete
6b43169afb5c: Pull complete
179c81612958: Pull complete
770ca0d9f1d8: Pull complete
fb7fe1d666e4: Pull complete
13c598845a6e: Pull complete
Digest: sha256:b16e4cbe3d507bff1e40542196854a8da16bd1831e2cc413d7166f9b698ec30f
Status: Downloaded newer image for mariadb:10.5
b1356f70616be88de575a5f2869db0b5a301ce3852f8dbbeb63cca8c1c16b1df
```

6.Pantallazo con la instrucción para crear el contenedor de la aplicación.

```
daw@daw-docker:~$ docker run -d -p 8080:80 --name nextMaria --network redMariaNe
xt --link miMariadb:mariadb -v nextcloud:/var/www/html  nextcloud
854ee3530fa5851d1015effa0466e5fa4105eb83aae22d1357cca5b989e0b38c
```

7.Pantallazo donde se ve el acceso a la aplicación desde un navegador web.