Prácticas Docker

1. Crear redes y asociarlas a contenedores

- Comprobamos las redes que tenemos en este momento:

```
docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3d8689b8a3ea        bridge              bridge              local
81ce05a3ba16        host                host                local
17052d6bd175        none                null                local
```

- Creamos una nueva red de tipo bridge

```
docker network create net1
8e83268b846d8f6937d13383afa1791bf30faccb1a3b1a02248736ff3ee14c1c


network ls
NETWORK ID          NAME                DRIVER              SCOPE
3d8689b8a3ea        bridge              bridge              local
81ce05a3ba16        host                host                local
8e83268b846d        net1                bridge              local
17052d6bd175        none                null                local
```

- La inspeccionamos. NOTA (puede que los valores, direcciones IP, etc sean distintas porque dependen de la configuración y del uso que hayáis hecho de Docker dentro de vuestro PC)

```
docker network inspect net1
[
    {
        "Name": "net1",
        "Id":
"8e83268b846d8f6937d13383afa1791bf30faccb1a3b1a02248736ff3ee14c1c",
        "Created": "2018-03-22T23:40:59.678173713+01:00",
        "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": {},
        "Options": {},
        "Labels": {}
    }
]
```

- Vemos que tiene el rango de direcciones “Subnet": "172.18.0.0/16",
- A nivel físico del sistema operativo ha debido crear una nueva conexión para la red. La que tiene el nombre de docker0 es la que corresponde a la red bridge predeterminada de Docker
- (NOTA. El comando “nmcli” es Red Hat-Fedora, en vuestro Linux puede variar. Podemos usar ifconfig o “ip addr” por ejemplo para ver la información)

```
nmcli con
NOMBRE           UUID                                  TIPO
DISPOSITIVO
br-8e83268b846d  edc8f727-36ec-472e-95c3-6eeacb28b657  bridge
br-8e83268b846d
docker0          5770a886-453c-4836-bef6-aaeb077788c8  bridge
docker0
static-ens37     daf9e2af-7ccb-420e-b42e-f1ffe41fc58d  802-3-ethernet
ens37
virbr0           2b4ecf59-343a-4820-bd2f-c557f9ccdefe  bridge
virbr0
```

- El nombre de dispositivo que se ha puesto a la nueva conexión coincide con el ID que Docker ha asignado a la red

```
docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3d8689b8a3ea        bridge              bridge              local
81ce05a3ba16        host                host                local
8e83268b846d        net1                bridge              local
17052d6bd175        none                null                local
```

- Creamos un contenedor “httpd” con esa red. Lo llamamos por ejemplo “apache1”

```
docker run -d -p 8080:80 --network net1 --name apache1
httpd
```

- Comprobamos que funciona y los puertos por los que escucha

```
docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED
STATUS              PORTS
NAMES
2291b272538b   httpd     "httpd-foreground"   About a
minute ago   Up About a minute   0.0.0.0:8080->80/tcp,
:::8080->80/tcp   apache1
```

- Comprobamos en la red que efectivamente se ha unido el contenedor a la red y podemos comprobar también la IP que le ha facilitado

```
docker network inspect net1
"Containers": {

"2291b272538b4d72e22a3211197d8d13e7c75fbb941237179052eb669
5bdd875": {
                "Name": "apache1",
                "EndpointID":
"8a2bc70c4e25a1b5daa69c0fe458a86d66e13848d1890a4608943d8df
d975485",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},,
```

- Vamos a comprobar ahora que cuando creamos una red personalizada, Docker genera un DNS interno que nos permite acceder a los contenedores con el nombre que le pusimos con la opción –name al crearlos

- Si entramos dentro del contenedor “apache1” (tu debes usar el nombre que has usado para crearlo)

```
docker exec -it apache1 bash
```

- Instalamos ping

```
apt-get update
apt-get install -y iputils-ping
```

- Si probamos a realizar un ping a “apache1” (y recordemos que no hemos configurado nada en la red para poner ese nombre). Vemos que funciona bien y localiza la IP

```
ping apache1


PING apache1 (172.18.0.2) 56(84) bytes of data.
64 bytes from e909eff322a2 (172.18.0.2): icmp_seq=1 ttl=64
time=0.055 ms
64 bytes from e909eff322a2 (172.18.0.2): icmp_seq=2 ttl=64
time=0.044 ms
```

- Desde otro terminal, vamos a crear un segundo contenedor en esa red

```
docker run -d -p 8081:80 --name apache2 --network net1
httpd
f68d918b0911858f29e9d992639cb28399e741b0d75ccc643458c30201
35a3d9


docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED
STATUS          PORTS
NAMES
f68d918b0911   httpd     "httpd-foreground"   21 seconds
ago   Up 19 seconds   0.0.0.0:8081->80/tcp, :::8081-
>80/tcp   apache2
2291b272538b   httpd     "httpd-foreground"   6 minutes
ago    Up 6 minutes    0.0.0.0:8080->80/tcp, :::8080-
>80/tcp   apache1
```

- Volvemos a “apache1” y desde la bash en la que estamos hacemos un ping a apache2

- Debe funcionar perfectamente, ya que la red personalizada se encarga de gestionar los nombres de los contenedores. Y como hemos visto, nosotros no hemos tenido que configurar nada.

```
ping apache2
PING apache2 (172.18.0.3) 56(84) bytes of data.
64 bytes from mongo2.net1 (172.18.0.3): icmp_seq=1 ttl=64 time=0.150 ms
64 bytes from mongo2.net1 (172.18.0.3): icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from mongo2.net1 (172.18.0.3): icmp_seq=3 ttl=64 time=0.060 ms
```

- Ahora, vamos a crear una segunda red. Además, le vamos a poner dos características:
    - Que su subred sea la 172.30.0.0/16
    - Que los contenedores se les asocie una IP que comience a partir de la 172.30.10.0/24

```
docker network create net2  --subnet=172.30.0.0/16 --ip-range=172.30.10.0/24
31ed5d42621519c775729042167d62ff4b34cb76af13a4007168a57b182e8704
```

- Podemos comprobar haciendo un inspect a la red

```
docker network inspect net2
…..
…..
…..
"IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.30.0.0/16",
                    "IPRange": "172.30.10.0/24"
```

```
                }
            ]
        },
        "Internal": false,
….
….
```

- Vamos a crear un contendor mongo asociado a esa red. Le llamamos Apache3

```
docker run -d -p 8082:80 --name apache3 --network net2
httpd
424a257d62bbb49394db363a2927a437d6ff3fd83cd2262fcb6f9112a3
ae39de


docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED
STATUS          PORTS
NAMES
424a257d62bb   httpd     "httpd-foreground"   55 seconds
ago   Up 53 seconds   0.0.0.0:8082->80/tcp, :::8082-
>80/tcp   apache3
f68d918b0911   httpd     "httpd-foreground"   5 minutes
ago    Up 5 minutes    0.0.0.0:8081->80/tcp, :::8081-
>80/tcp   apache2
2291b272538b   httpd     "httpd-foreground"   11 minutes
ago   Up 11 minutes   0.0.0.0:8080->80/tcp, :::8080-
>80/tcp   apache1
```

- Comprobamos con inspect la red que tiene al contenedor. Mandamos la información a un fichero porque salen bastantes cosas

```
docker inspect apache3 > apache3.json
```

- Editamos el fichero y buscamos la zona de la red. Podemos ver que está asociado a la net2 y con la dirección IP adecuada, a partir de la 10

```
"Networks": {
                "net2": {
                    "IPAMConfig": null,
                    "Links": null,
```

```
                    "Aliases": [
                        "f03d25490e17"
                    ],
                    "NetworkID":
"31ed5d42621519c775729042167d62ff4b34cb76af13a4007168a57b1
82e8704",
                    "EndpointID":
"85f58410558a16e1523875874ff9942d655d4a30eeaea71ae5defb426
544e4f5",
                    "Gateway": "172.30.10.0",
                    "IPAddress": "172.30.10.1",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:1e:0a:01",
                    "DriverOpts": null
                }
```

- Si intentamos acceder desde la bash que tenemos abierta en “apache1” a “apache3” no la encuentra porque no están en la misma subred. Lo podemos probar haciendo un “ping”

- Por último, vamos a asociar este contenedor a la red “net1”

```
docker network connect net1 apache3
```

- Volvemos a hacer un inspect del contenedor. Podemos comprobar que ahora tiene asociadas la net1 y la net2 y que tiene una IP de cada una de ellas

```
"net1": {
                    "IPAMConfig": {},
                    "Links": null,
                    "Aliases": [
                        "f03d25490e17"
                    ],
                    "NetworkID":
"8e83268b846d8f6937d13383afa1791bf30faccb1a3b1a02248736ff3
ee14c1c",
```

```
                    "EndpointID":
"55ba7b5b56569f456b8bc8796467b89ac1018e4efcabd3230ce3306900a39981",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.4",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:12:00:04",
                    "DriverOpts": null
                },
"net2": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": [
                        "f03d25490e17"
                    ],
                    "NetworkID":
"31ed5d42621519c775729042167d62ff4b34cb76af13a4007168a57b182e8704",
                    "EndpointID":
"85f58410558a16e1523875874ff9942d655d4a30eeaea71ae5defb426544e4f5",
                    "Gateway": "172.30.10.0",
                    "IPAddress": "172.30.10.1",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:1e:0a:01",
                    "DriverOpts": null
                }
```


- Si volvemos a la bash que teníamos abierta en apache1, debemos poder acceder ahora a apache3

```
ping apache3
```

```
PING apache3 (172.18.0.4) 56(84) bytes of data.
64 bytes from mongo3.net1 (172.18.0.4): icmp_seq=1 ttl=64 time=0.178 ms
64 bytes from mongo3.net1 (172.18.0.4): icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from mongo3.net1 (172.18.0.4): icmp_seq=3 ttl=64 time=0.058 ms
```

- Desconectamos apache3 de la red “net1”

```
docker network disconnect net1 apache3
```