Prácticas Docker

1. Configurar Puertos

- Usamos la imagen de MongoDB de la práctica anterior
- Vamos a averiguar los puertos por los que escucha. Podemos usar el comando inspect
- Por supuesto, lo más rápido es comprobar la documentación de Docker Hub para averiguarlo, donde además normalmente tenemos ejemplos

```
docker inspect --format='{{.Config.ExposedPorts}}' mongo
map[27017/tcp:{}]
```

- Comprobamos que lo hace por el puerto 27017
- Creamos un contenedor llamado mongo2 que escuche por el mismo puerto en el host

```
docker run -d -p 27017:27017 --name mongo2 mongo
```

- Comprobamos que funciona y los puertos por los que escucha

```
docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
ec0eacb01e02        mongo               "docker-
entrypoint.s…"   36 seconds ago      Up 35 seconds
0.0.0.0:27017->27017/tcp   mongo
```

- También podemos usar el comando PORT

```
docker port mongo2
27017/tcp -> 0.0.0.0:27017
```

- Ahora vamos a comprobar las redes que tenemos

```
docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
3d8689b8a3ea        bridge              bridge              local
81ce05a3ba16        host                host                local
```

```
17052d6bd175        none                null                local
```

- Comprobamos los contenedores que tiene la red Bridge. Al final , mas o menos tenemos nuestro contenedor mongo2, con la IP que se le ha asignado

```
docker network inspect bridge
….
…
..
"Containers": {

"01c80e50d517b8c0790d268064288fbdf8846716de7defff558a8ad3a90d197d": {
                "Name": "mongo2",
                "EndpointID":
"552052da215469f22370f6428731f38ae92a2ea8aaf954cbd8e73789e57dd898",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
```

- Podemos probar que llegamos al contenedor con un ping

```
ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.060 ms
```

- También aparece el rango de IPs que tiene para asignar

```
"IPAM": {
```

```
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
```

- Podemos comprobarlo también a través del contenedor
- Entre la información que nos aparece, debe estar la de la RED
-

```
docker inspect mongo2
….
….

"NetworkSettings": {
            "Bridge": "",
            "SandboxID":
"03ab27998f59c33800c481fd75f27fe8605ba90195109163d980608034eee770",
            "HairpinMode": false,
            "LinkLocalIPv6Add
```

- Si arrancamos otro contenedor Mongo, vamos a ver los datos que les indica. Recordemos que debemos poner otro puerto distinto para el host

```
docker run -d --name mongo3 -p 27018:27017 mongo
72b81e411c2b36deab5cd490dc2d2b65ea8712d8f27685e0083a761c68f37505
```

- Si inspeccionamos la red, debemos tener los dos contenedores, con sus direcciones IPs correspondientes

```
"Containers": {

"01c80e50d517b8c0790d268064288fbdf8846716de7defff558a8ad3a90d197d": {
                "Name": "mongo2",
                "EndpointID":
"552052da215469f22370f6428731f38ae92a2ea8aaf954cbd8e73789e57dd898",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },

"72b81e411c2b36deab5cd490dc2d2b65ea8712d8f27685e0083a761c68f37505": {
```

```
                "Name": "mongo3",
                "EndpointID":
"7fa5ebcd624205a2dd9205545a34a9e10920747d59e34411bd2cd74c08601cf1",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            }
```

- Vamos a probar que llegamos desde un contendor a otro
- Abrimos una bash contra mongo3

```
docker exec -it mongo3 bash
root@72b81e411c2b:/#
```

- Instalamos el ping

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

- Hacemos un ping contra mongo2. En mi caso es el 172.17.0.2, tal y como me ha indicado la red

```
ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.099 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.057 ms
```

- Como prueba final, vamos a conectarnos como cliente, desde mongo3 a la base de datos MongoDB que tenemos en el contenedor mongo2. Es decir, vamos a comprobar que los contenedores de una red concreta se pueden conectar entre sí sin problemas.
- Desde la Shell que hemos abierto en mongo3 ponemos el siguiente comando, que es la Shell de mongo

```
mongo --host 172.17.0.2 --port 27017
```

- Evidentemente, la IP y el puerto pertenecen a “mongo2”
- Debería conectarme sin problemas

```
mongo --host 172.17.0.2 --port 27017
MongoDB shell version v3.6.3
connecting to: mongodb://172.17.0.2:27017/
MongoDB server version: 3.6.3
Server has startup warnings:
2018-03-22T10:54:40.824+0000 I CONTROL  [initandlisten]
2018-03-22T10:54:40.824+0000 I CONTROL  [initandlisten] ** WARNING:
Access control is not enabled for the database.
```

```
2018-03-22T10:54:40.824+0000 I CONTROL  [initandlisten] **
Read and write access to data and configuration is unrestricted.
2018-03-22T10:54:40.824+0000 I CONTROL  [initandlisten]
2018-03-22T10:54:40.824+0000 I CONTROL  [initandlisten]
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten] **        We
suggest setting it to 'never'
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten]
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten] **        We
suggest setting it to 'never'
2018-03-22T10:54:40.825+0000 I CONTROL  [initandlisten]
>
```

- Ya estamos conectados. Podemos probar por ejemplo viendo las Bases de datos de MongoDB

```
show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

- Cerramos los dos contenedores
-