EXPERIMENT 4

AIM: Working with Docker Network

Steps to Complete:

Step 1 - Create Network

The first step is to create a network using the CLI. This network will allow us to attach multiple containers which will be able to discover each other.

In this example, we're going to start by creating a *backend-network*. All containers attached to our backend will be on this network.

Task: Create Network

To start with we create the network with our predefined name.

```
docker network create backend-network
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network create backend-network
e99ba0ea3fcc757dc219ec28789ebd0236621fcb862d75be5f2ced4441fee20e

F:\dockerlab\ACO-LAB-2021-25>
```

Task: Connect To Network

When we launch new containers, we can use the *--net* attribute to assign which network they should be connected to.

```
docker run -d --name=redis --net=backend-network redis
```

```
F:\dockerlab\ACO-LAB-2021-25>docker run -d --name=redis --net=backend-network redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
a378f10b3218: Already exists
b266cd8112a6: Pull complete
7ba86e6448de: Pull complete
3aeb7c9e9a5f: Pull complete
de3be2a98bda: Pull complete
4f4fb700ef54: Pull complete
98e18d21aa3b: Pull complete
Digest: sha256:1f1bd4adf5dabf173b235ba373faef55f3ad53394791d1473763bf5a2181780d
Status: Downloaded newer image for redis:latest
66faf1318c3073d6d112221a2c56589fa8ceaae7f112694d0e13a4724bc7f495

F:\dockerlab\ACO-LAB-2021-25>
```

In the next step we'll explore the state of the network.

Step 2 - Network Communication

Unlike using links, *docker network* behave like traditional networks where nodes can be attached/detached.

Task: Explore

The first thing you'll notice is that Docker no longer assigns environment variables or updates the hosts file of containers. Explore using the following two commands and you'll notice it no longer mentions other containers.

```
docker run --net=backend-network  alpine ping -c1 redis
```

```
F:\dockerlab\ACO-LAB-2021-25>docker run --net=backend-network alpine ping -c1 redis
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:latest
PING redis (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.324 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.324/0.324/0.324 ms

F:\dockerlab\ACO-LAB-2021-25>
```

Step 3 - Connect Two Containers

Docker supports multiple networks and containers being attached to more than one network at a time.

For example, let's create a separate network with a Node.js application that communicates with our existing Redis instance.

Task

The first task is to create a new network in the same way.

```
docker network create frontend-network
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network create frontend-network
d82b3958af1d75168dfe426ee818e3e2ee0f02aca2277a21aeca184b44c5ff0f
```

When using the *connect* command it is possible to attach existing containers to the network.

```
docker network connect frontend-network redis
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network connect frontend-network redis

F:\dockerlab\ACO-LAB-2021-25>
```

When we launch the web server, given it's attached to the same network it will be able to communicate with our Redis instance.

```
docker run -d -p 3000:3000 --net=frontend-network  katacoda/redis-node-docker-example
```

You can test it using `curl docker:3000`

```
F:\dockerlab\ACO-LAB-2021-25>docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example
Unable to find image 'katacoda/redis-node-docker-example:latest' locally
latest: Pulling from katacoda/redis-node-docker-example
[DEPRECATION NOTICE] Docker Image Format v1, and Docker Image manifest version 2, schema 1 support will be removed in an
 upcoming release. Suggest the author of docker.io/katacoda/redis-node-docker-example:latest to upgrade the image to the
 OCI Format, or Docker Image manifest v2, schema 2. More information at https://docs.docker.com/go/deprecated-image-spec
s/
12b41071e6ce: Pull complete
a3ed95caeb02: Pull complete
49a025abf7e3: Pull complete
1fb1c0be01ab: Pull complete
ae8c1f781cde: Pull complete
db73207ad2ae: Pull complete
446b13034c13: Pull complete
Digest: sha256:1aae9759464f00953c8e078a0e0d0649622fef9dd5655b1491f9ee589ae904b4
Status: Downloaded newer image for katacoda/redis-node-docker-example:latest
95df4484589ed3df026be320b4edbd3aea9f408f2e27761665f7bfde09c37fee

F:\dockerlab\ACO-LAB-2021-25>
```

Step 4 - Create Aliases

Links are still supported when using *docker network* and provide a way to define an Alias to the container name. This will give the container an extra DNS entry name and way to be discovered. When using --link the embedded DNS will guarantee that localised lookup result only on that container where the --link is used.

The other approach is to provide an alias when connecting a container to a network.

Connect Container with Alias

The following command will connect our Redis instance to the frontend-network with the alias of *db*.

```
docker network create frontend-network2
```

```
docker network connect --alias db frontend-network2 redis
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network create frontend-network2
0a74ceb51f2e417d22b41df0d5c885ae2fd0c620ac296f97e6564b4b9652a0fc

F:\dockerlab\ACO-LAB-2021-25>docker network connect --alias db frontend-network2 redis

F:\dockerlab\ACO-LAB-2021-25>
```

When containers attempt to access a service via the name db, they will be given the IP address of our Redis container.

```
docker run --net=frontend-network2  alpine ping -c1 db
```

```
F:\dockerlab\ACO-LAB-2021-25>docker run --net=frontend-network2 alpine ping -c1 db
PING db (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=0.185 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.185/0.185/0.185 ms

F:\dockerlab\ACO-LAB-2021-25>
```

Step 5 - Disconnect Containers

With our networks created, we can use the CLI to explore the details.

The following command will list all the networks on our host.

```
docker network ls
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
e99ba0ea3fcc   backend-network     bridge    local
0aa23eda6efb   bridge              bridge    local
d82b3958af1d   frontend-network    bridge    local
0a74ceb51f2e   frontend-network2   bridge    local
e2e3fdb5bfe8   host                host      local
5adc6ba0e6d7   none                null      local
```

We can then explore the network to see which containers are attached and their IP addresses.

```
docker network inspect frontend-network
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "d82b3958af1d75168dfe426ee818e3e2ee0f02aca2277a21aeca184b44c5ff0f",
        "Created": "2023-10-26T19:08:40.368846909Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "66faf1318c3073d6d112221a2c56589fa8ceaae7f112694d0e13a4724bc7f495": {
                "Name": "redis",
                "EndpointID": "78e2335348251e12b31bc61862e448a1048ccaf30ad610662403a684c0bbafec",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            },
            "95df4484589ed3df026be320b4edbd3aea9f408f2e27761665f7bfde09c37fee": {
                "Name": "amazing_rhodes",
                "EndpointID": "cb45c3197ac059bc130dfb255e7d65b5cb6b02345a506482df7d0cfabedc5681",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]

F:\dockerlab\ACO-LAB-2021-25>
```

The following command disconnects the redis container from the *frontend-network*.

```
docker network disconnect frontend-network  redis
```

```
F:\dockerlab\ACO-LAB-2021-25>docker network disconnect frontend-network redis

F:\dockerlab\ACO-LAB-2021-25>
```