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
```

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
```

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
```

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
```

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

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

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`

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
```

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
```

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
```

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

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

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

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