EXPERIMENT 4

AIM: Working with Docker Network

Steps to Complete:

Step 1 - Create Network

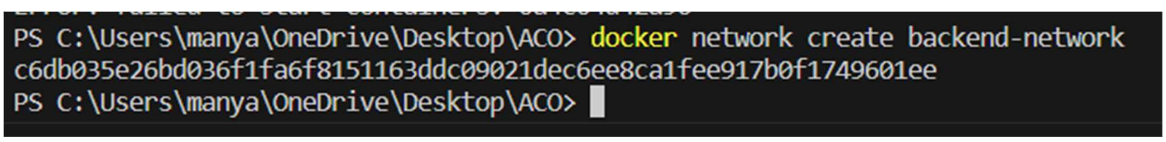
The first step is to create a network using the CLI. This network will allow us to attach multiple containers which will be able to discover each other.

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
```

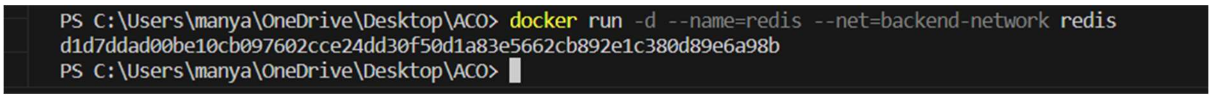


```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network create backend-network
c6db035e26bd036f1fa6f8151163ddc09021dec6ee8ca1fee917b0f1749601ee
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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
```



```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker run -d --name=redis --net=backend-network redis
d1d7ddad00be10cb097602cce24dd30f50d1a83e5662cb892e1c380d89e6a98b
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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

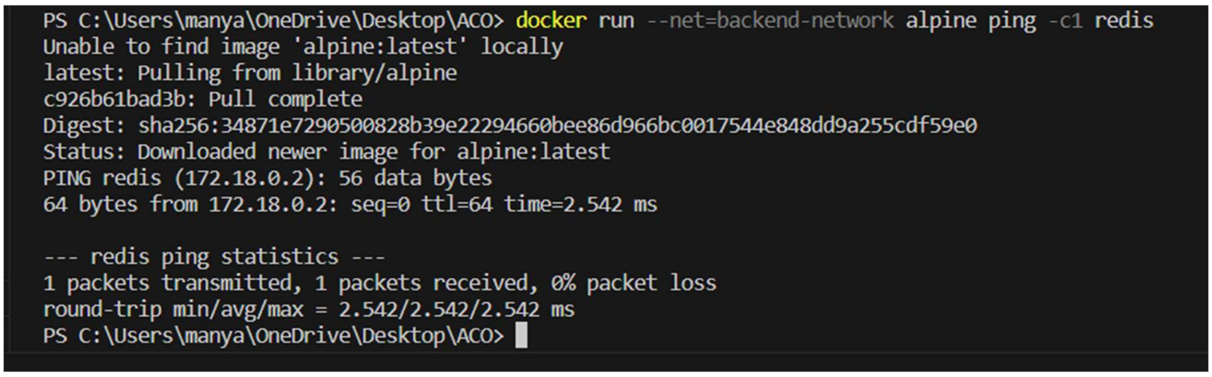
Step 2 - Network Communication

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
```



```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker run --net=backend-network alpine ping -c1 redis
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
c926b61bad3b: Pull complete
Digest: sha256:34871e7290500828b39e22294660bee86d966bc0017544e848dd9a255cdf59e0
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=2.542 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 2.542/2.542/2.542 ms
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

Step 3 - Connect Two Containers

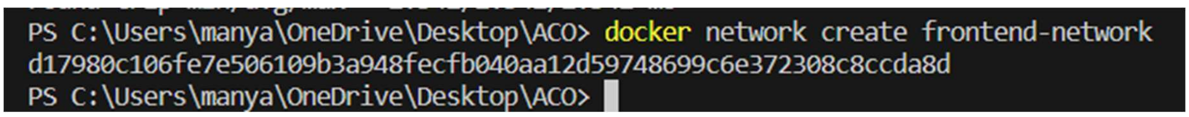
Docker supports multiple networks and containers being attached to more than one network at a time.

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.

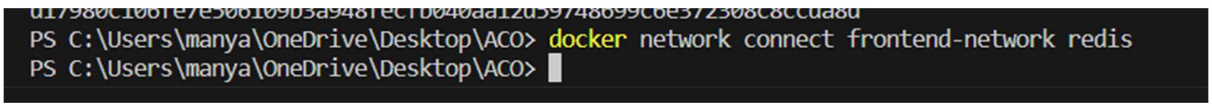
```
docker network create frontend-network
```



```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network create frontend-network
d17980c106fe7e506109b3a948fecfb040aa12d59748699c6e372308c8ccda8d
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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

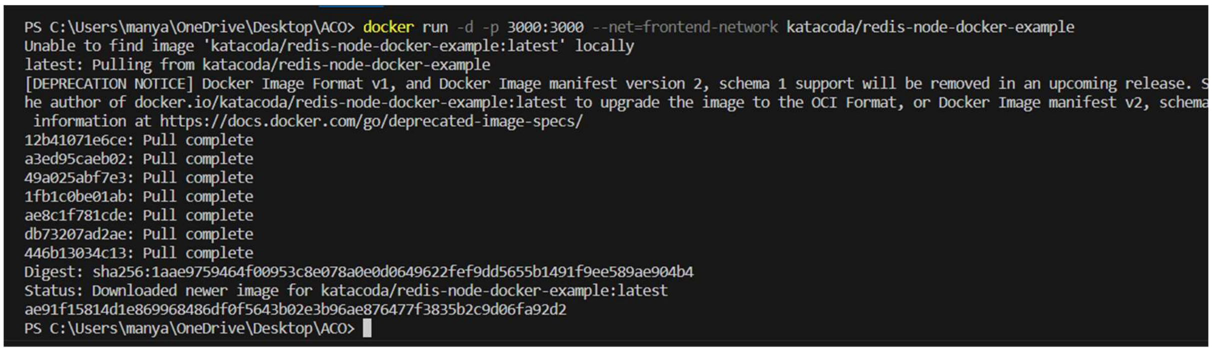
```
docker network connect frontend-network redis
```



```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network connect frontend-network redis
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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
```



```
PS C:\Users\manya\OneDrive\Desktop\ACO> 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. S
he author of docker.io/katacoda/redis-node-docker-example:latest to upgrade the image to the OCI Format, or Docker Image manifest v2, schema
 information at https://docs.docker.com/go/deprecated-image-specs/
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
ae91f15814d1e869968486df0f5643b02e3b96ae876477f3835b2c9d06fa92d2
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

Step 4 - Create Aliases

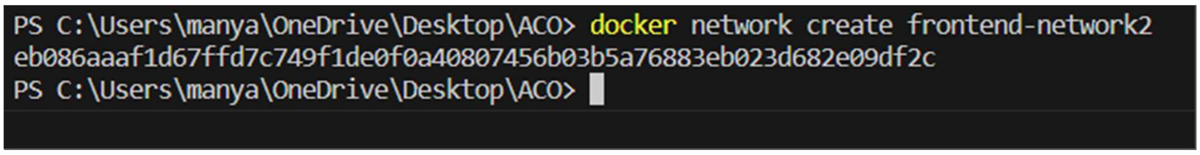
Links are still supported when using *docker network* and provide a way to define an Alias to the container name. This will give the container an extra DNS entry name and way to be discovered. When using --link the embedded DNS will guarantee that localised lookup result only on that container where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

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

Connect Container with Alias

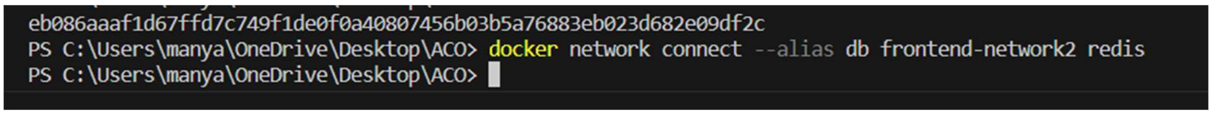
The following command will connect our Redis instance to the frontend-network with the alias of *db*.

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



```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network create frontend-network2
eb086aaaf1d67ffd7c749f1de0f0a40807456b03b5a76883eb023d682e09df2c
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

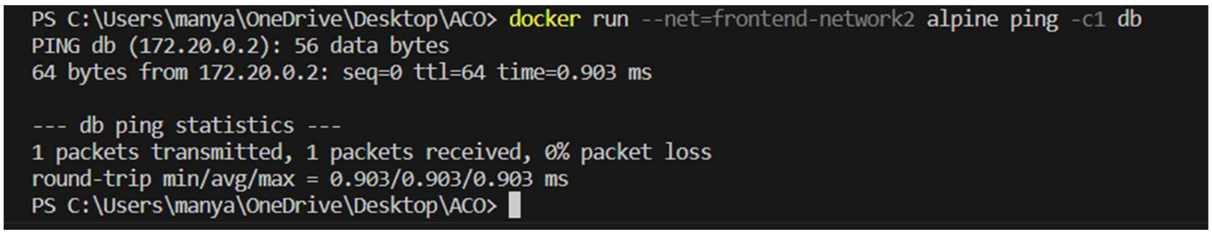
```
docker network connect --alias db frontend-network2 redis
```



```
eb086aaaf1d67ffd7c749f1de0f0a40807456b03b5a76883eb023d682e09df2c
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network connect --alias db frontend-network2 redis
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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
```



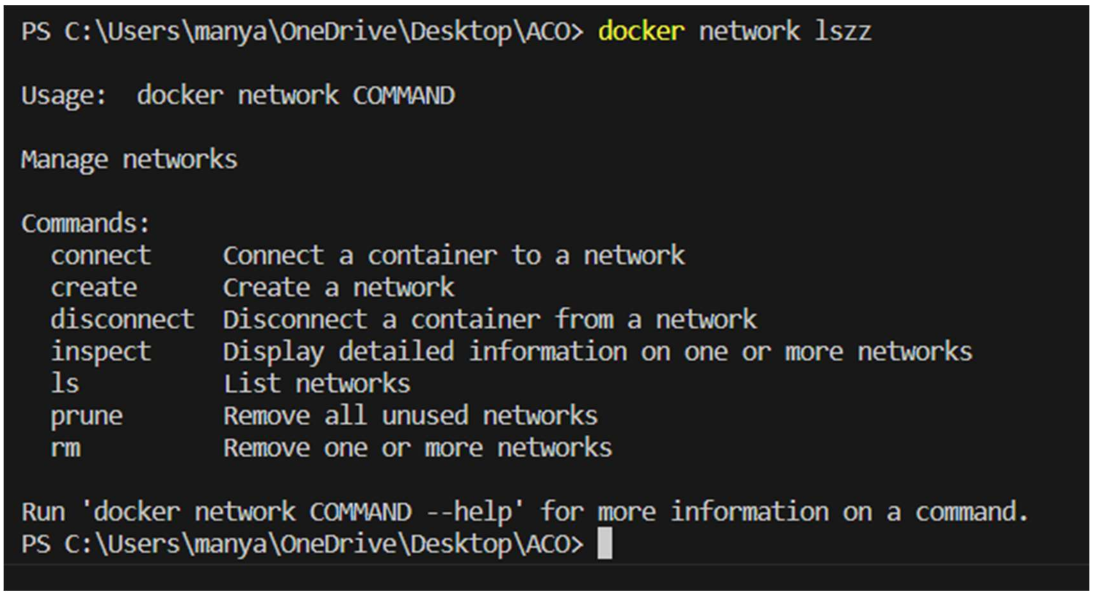
```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker run --net=frontend-network2 alpine ping -c1 db
PING db (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=0.903 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.903/0.903/0.903 ms
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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.

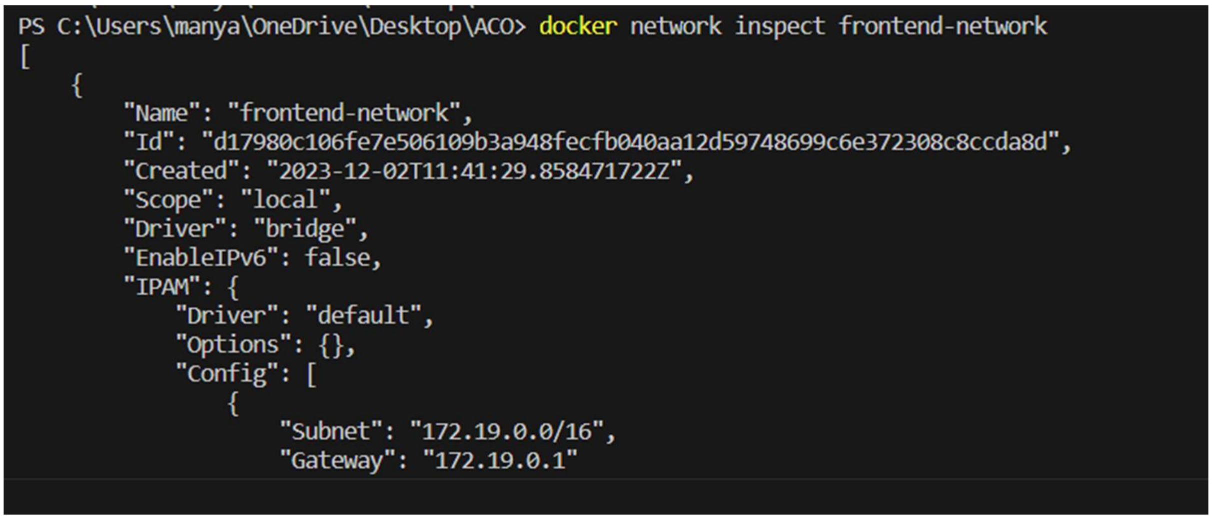
```
docker network lszz
```



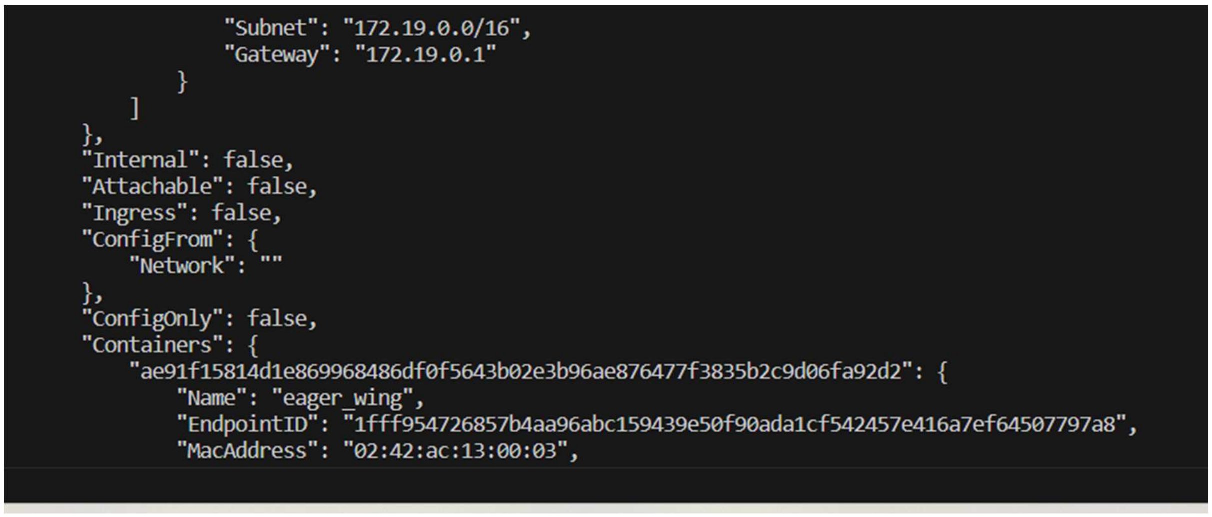
```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network lszz

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

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

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



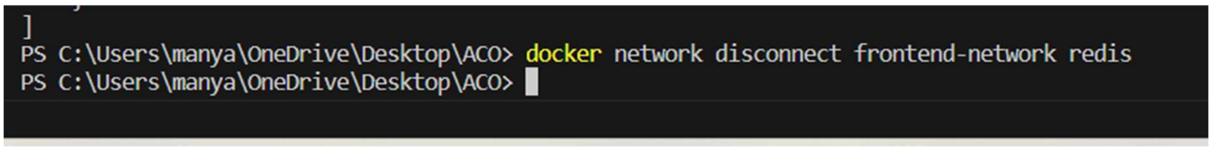
```
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "d17980c106fe7e506109b3a948fecfb040aa12d59748699c6e372308c8ccda8d",
        "Created": "2023-12-02T11:41:29.858471722Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
```



```
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "ae91f15814d1e869968486df0f5643b02e3b96ae876477f3835b2c9d06fa92d2": {
                "Name": "eager_wing",
                "EndpointID": "1fff954726857b4aa96abc159439e50f90ada1cf542457e416a7ef64507797a8",
                "MacAddress": "02:42:ac:13:00:03",
```

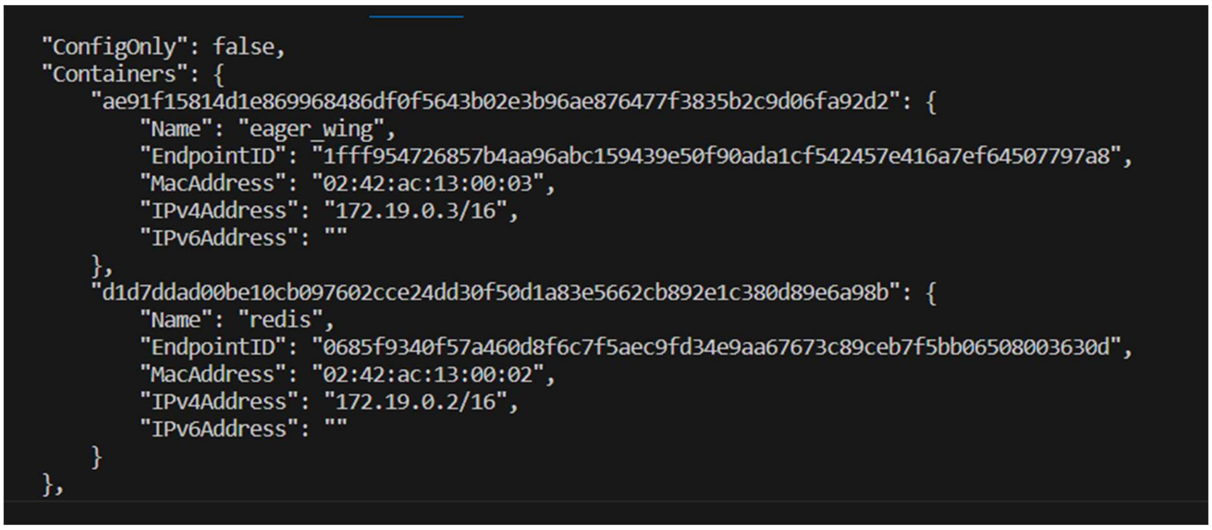
The following command disconnects the redis container from the *frontend-network*.

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



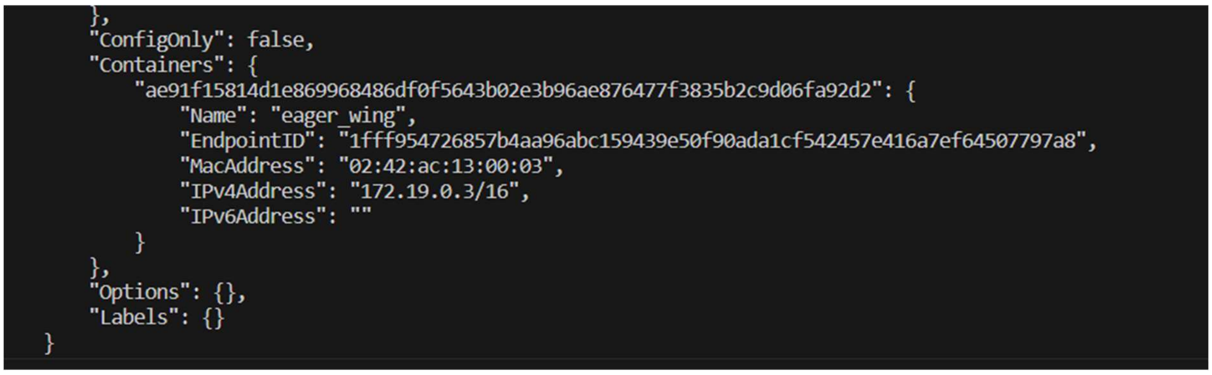
```
]
PS C:\Users\manya\OneDrive\Desktop\ACO> docker network disconnect frontend-network redis
PS C:\Users\manya\OneDrive\Desktop\ACO>
```

Before:



```
"ConfigOnly": false,
"Containers": {
    "ae91f15814d1e869968486df0f5643b02e3b96ae876477f3835b2c9d06fa92d2": {
        "Name": "eager_wing",
        "EndpointID": "1fff954726857b4aa96abc159439e50f90ada1cf542457e416a7ef64507797a8",
        "MacAddress": "02:42:ac:13:00:03",
        "IPv4Address": "172.19.0.3/16",
        "IPv6Address": ""
    },
    "d1d7ddad00be10cb097602cce24dd30f50d1a83e5662cb892e1c380d89e6a98b": {
        "Name": "redis",
        "EndpointID": "0685f9340f57a460d8f6c7f5aec9fd34e9aa67673c89ceb7f5bb06508003630d",
        "MacAddress": "02:42:ac:13:00:02",
        "IPv4Address": "172.19.0.2/16",
        "IPv6Address": ""
    }
},
```

After:



```
        },
        "ConfigOnly": false,
        "Containers": {
            "ae91f15814d1e869968486df0f5643b02e3b96ae876477f3835b2c9d06fa92d2": {
                "Name": "eager_wing",
                "EndpointID": "1fff954726857b4aa96abc159439e50f90ada1cf542457e416a7ef64507797a8",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
```