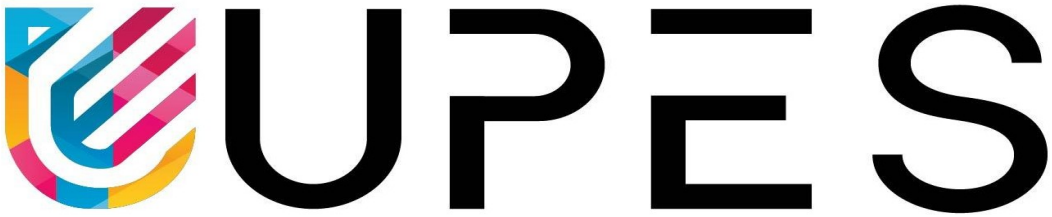


UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

APPLICATION CONTAINERIZATION AND ORCHESTRATION LAB

COURSE: B.Tech CSE (Devops)

INSTRUCTOR: DR.Hitesh Kumar Sharma

UNDERGRAD: Priyanshu Rai
SAP ID: 500096900

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.

```
priyanshurai@MacBook-Air ~ % docker network create backend-network
2ea295cdfdf2ed84b32d313c32da53aac311ecc551916908defa81715fd80f75f
priyanshurai@MacBook-Air ~ %
```

Task: Connect To Network

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

```
priyanshurai@MacBook-Air ~ % docker run -d --name=redis --net=backend-network redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
e886f0f47ef5: Already exists
457c001fd3ab: Pull complete
6631c396a680: Pull complete
5b3bbf47b73b: Pull complete
243ce8b1e9f3: Pull complete
4f4fb700ef54: Pull complete
7d9a2aec0fee: Pull complete
Digest: sha256:b68c6efe2c5f2d7d7d14a2749f66d6d81645ec0cacb92572b2fb7d5c42c82031
Status: Downloaded newer image for redis:latest
47167899eecfe08f3013081ebab1488bff0f2e58448c85f7fd24a4e3c70fe3be
```

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.

```
priyanshurai@MacBook-Air ~ % docker run --net=backend-network alpine ping -c1 redis
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
579b34f0a95b: 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=0.151 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.151/0.151/0.151 ms
priyanshurai@MacBook-Air ~ %
```

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.

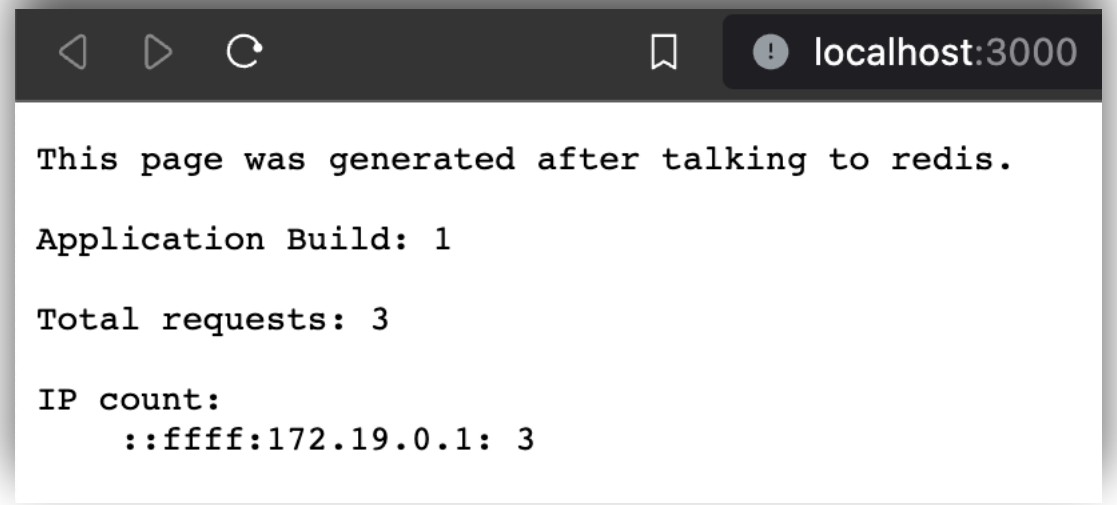
```
priyanshurai@MacBook-Air ~ % docker network create frontend-network
0465d38f11bacd57e3ff22fb1758c28a3b1ca1e31f2f55ec3fd2b512b7d05169
priyanshurai@MacBook-Air ~ %
```

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

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

```
priyanshurai@MacBook-Air ~ % docker network connect frontend-network redis
priyanshurai@MacBook-Air ~ % 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
Image docker.io/katacoda/redis-node-docker-example:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information at https://docs.docker.
com/registry/spec/deprecated-schema-v1/
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
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
0b0fa23ce82b5908266a156641976e042b58a822ef60b215b80e2ea7eedc208d
priyanshurai@MacBook-Air ~ %
```

You can test it using `curl docker:3000`




```
priyanshurai@MacBook-Air ~ % curl docker:3000
curl: (6) Could not resolve host: docker
priyanshurai@MacBook-Air ~ %
```

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*.

```
priyanshurai@MacBook-Air ~ % docker network create frontend-network2
c92cdca6580c20fa38d70c816583d9fb852618e7062339471fa6fc7741a8dc07
priyanshurai@MacBook-Air ~ %
```

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

```
priyanshurai@MacBook-Air ~ % docker network connect --alias db frontend-network2 redis
priyanshurai@MacBook-Air ~ % 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=0.280 ms

--- db ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.280/0.280/0.280 ms
priyanshurai@MacBook-Air ~ %
```

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.

```
priyanshurai@MacBook-Air ~ % docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
ea295cdfdf2e   backend-network     bridge    local
3677b264074e   bridge              bridge    local
0465d38f11ba   frontend-network    bridge    local
c92cdca6580c   frontend-network2   bridge    local
a41aaa5fb7b8   host                host      local
32d7a7d06010   none                null      local
priyanshurai@MacBook-Air ~ %
```

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

```
priyanshurai@MacBook-Air ~ % docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "0465d38f11bacd57e3ff22fb1758c28a3b1ca1e31f2f55ec3fd2b512b7d05169",
        "Created": "2023-10-06T07:10:12.18559759Z",
        "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": {
            "0b0fa23ce82b5908266a156641976e042b58a822ef60b215b80e2ea7eedc208d": {
                "Name": "strange_goodall",
                "EndpointID": "6889594d4af5f0e67afac15950b9c8a40a491ca4a3d33358ef7685f318118fb2",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            },
            "47167899eecfe08f3013081ebab1488bff0f2e58448c85f7fd24a4e3c70fe3be": {
                "Name": "redis",
                "EndpointID": "c7fc45812e7046ddf9677192fc449786f655cf6c97a3270fa9af4259a9898552",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
priyanshurai@MacBook-Air ~ %
```

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

```
priyanshurai@MacBook-Air ~ % docker network disconnect frontend-network redis
Error response from daemon: container 47167899eecfe08f3013081ebab1488bff0f2e58448c85f7fd24a4e3c70fe3be is not connected to network frontend-network
priyanshurai@MacBook-Air ~ %
```