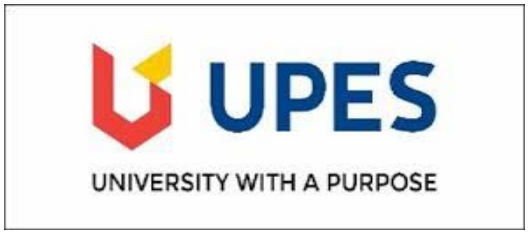

Lab Experiment – 3

To create Docker network and attach it with Containers for connectivity

NAME -Divyaansh Jain

ROLL NO – R171218040

SAP ID – 500067134

COURSE – B. Tech CSE- DevOps Batch 1

SUBJECT – Application Containerization

SEMESTER – 6th semester

Submitted To:

Dr. Hitesh Kumar Sir

To create Docker network and attach it with Containers for connectivity

- To create your own network, use the command “docker network create <network-name>”.

```
$ docker network create backend-network
b4e9e89d4311fa85646f77c34c48ed766be75acb948939ab43a85ffb47d3bda9
```

- To launch a container inside a network use option **--net=<network-name>** while launching a container.

```
$ docker run -d --name=redis --net=backend-network redis
c6a35ca1659bb09a1c3c2776c6c4ae31b40598f33555655eb1bf3002c094621a
```

- 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 env
  ```

  ```
  docker run --net=backend-network alpine cat /etc/hosts
  ```

  Instead, the way containers can communicate via an Embedded DNS Server in Docker. This DNS server is assigned to all containers via the IP 127.0.0.11 and set in the *resolv.conf* file.

```
$ docker run --net=backend-network alpine env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=a01a62f4c85b
HOME=/root
$ docker run --net=backend-network alpine cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.0.3      4abd7dc1e4f3
$ docker run --net=backend-network alpine cat /etc/resolv.conf
nameserver 127.0.0.11
options ndots:0
```

- When containers attempt to access other containers via a well-known name, such as Redis, the DNS server will return the IP address of the correct Container. In this case, the fully qualified name of Redis will be *redis.backend-network*.

```
$ docker run --net=backend-network alpine ping -c1 redis
PING redis (172.19.0.2): 56 data bytes
64 bytes from 172.19.0.2: seq=0 ttl=64 time=0.137 ms

--- redis ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.137/0.137/0.137 ms
```

- Create a network named “frontend-network” and use the connect command to attach existing containers to the network. Now, launch the web server, given it's attached to the same network it will be able to communicate with our Redis instance.

```
$ docker network create frontend-network
aaa21e0129a187605fc0ad1d0494c79ce8fa92fc741144cfad539878f1ce7ef0
$
$ docker network connect frontend-network redis
$
$ 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
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
68583cdf03077eb7e6a07119b6e891808bca152a446d7b7839740d8360b1b229
```

- Check using the curl command.

```
$ curl docker:3000
This page was generated after talking to redis.

Application Build: 1

Total requests: 1

IP count:
    ::ffff:172.17.0.56: 1
```

- To see all the existed network use command **docker network ls**.

```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b4e9e89d4311        backend-network     bridge              local
5442d01e32b3        bridge              bridge              local
aaa21e0129a1        frontend-network    bridge              local
6bca334fe9d9        frontend-network2   bridge              local
0c288d2ed25a        host                host                local
4d3221fe852b        none                null                local
```

- To see the network details of the container and check if it is inside the specified network use command **docker inspect**. This command gives all the details about the container.

```
$ docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "aaa21e0129a187605fc0ad1d0494c79ce8fa92fc741144cfad539878f1ce7ef0",
        "Created": "2021-02-05T06:02:41.849101806Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "68583cdf03077eb7e6a07119b6e891808bca152a446d7b7839740d8360b1b229": {
                "Name": "trusting_banach",
                "EndpointID": "96020bbfb615e7b7c0e6b1045863f5ec4f940389309a29eacd28359df987cb33",
                "MacAddress": "02:42:ac:14:00:03",
                "IPv4Address": "172.20.0.3/16",
                "IPv6Address": ""
```

- You can use the command disconnects the redis container from the *frontend-network*.

```
$ docker network disconnect frontend-network redis
$ docker network inspect frontend-network
[
    {
        "Name": "frontend-network",
        "Id": "aaa21e0129a187605fc0ad1d0494c79ce8fa92fc741144cfad539878f1ce7ef0",
        "Created": "2021-02-05T06:02:41.849101806Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "68583cdf03077eb7e6a07119b6e891808bca152a446d7b7839740d8360b1b229": {
                "Name": "trusting_banach",
                "EndpointID": "96020bbfb615e7b7c0e6b1045863f5ec4f940389309a29eacd28359df987cb33",
                "MacAddress": "02:42:ac:14:00:03",
                "IPv4Address": "172.20.0.3/16",
```