Create user-defined bridge networks (Exercise)

To try this out   Go To

https://labs.play-with-docker.com/

Click on Start (Big Green Button) Login using your Docker Credentials and Click on “**Add New Instance”**

In this example, we again start two `alpine` containers, but attach them to a user-defined network called `alpine-net` . These containers are not connected to the default `bridge` network at all. We then start a third `alpine` container which is connected to the `bridge` network but not connected to `alpine-net`, and a fourth `alpine` container which is connected to both networks.

1. Create the `alpine-net` network. You do not need the `--driver bridge` flag since it’s the default, but this example shows how to specify it.
2. `docker network create --driver bridge alpine-net`
3. `docker network ls`
4. `Do a Docker Inspect`

```
      $ docker inspect <networkID>
```

5. Create your four containers. Notice the `--network` flags. You can only connect to one network during the `docker run` command, so you need to use `docker network connect` afterward to connect `alpine4` to the `bridge` network as well.

```
    docker run -dit --name alpine1 --network alpine-net alpine ash
    docker run -dit --name alpine2 --network alpine-net alpine ash
    docker run -dit --name alpine3 alpine ash
    docker run -dit --name alpine4 --network alpine-net alpine ash
    docker network connect bridge alpine4
```

6. On user-defined networks like `alpine-net`, containers can not only communicate by IP address, but can also resolve a container name to an IP address. This capability is called **automatic service discovery**. Let's connect to `alpine1` and test this out. `alpine1` should be able to resolve `alpine2` and `alpine4` (and `alpine1`, itself) to IP addresses.

```
docker container attach alpine1
```

```
     # ping -c 2 alpine2
```

```
     << You should get a response >>
```

```
     # ping -c 2 alpine4
```

7. From `alpine1`, you should not be able to connect to `alpine3` at all, since it is not on the `alpine-net` network. Not only that, but you can't connect to `alpine3` from `alpine1` by its IP address either. Look back at the `docker network inspect` output for the `bridge` network and find `alpine3`'s IP address: `172.17.0.2` Try to ping it.

```
# ping -c 2 172.17.0.2

PING 172.17.0.2 (172.17.0.2): 56 data bytes

--- 172.17.0.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Detach from `alpine1` using detach sequence, `CTRL` + `p` `CTRL` + `q` (hold down `CTRL` and type `p` followed by `q`).

8. Remember that `alpine4` is connected to both the default `bridge` network and `alpine-net`. It should be able to reach all of the other containers. However, you will need to address `alpine3` by its IP address. Attach to it and run the tests.

```
$ docker container attach alpine4

# ping -c 2 alpine1

PING alpine1 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.074 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.082 ms

--- alpine1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.074/0.078/0.082 ms

# ping -c 2 alpine2

PING alpine2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.075 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.080 ms

--- alpine2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/0.077/0.080 ms

# ping -c 2 alpine3
ping: bad address 'alpine3'

# ping -c 2 172.17.0.2

PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.089 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.075 ms

--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/0.082/0.089 ms

# ping -c 2 alpine4
```

```
PING alpine4 (172.18.0.4): 56 data bytes
64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.033 ms
64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.064 ms

--- alpine4 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.033/0.048/0.064 ms
```

9.