

## Join GitHub today

Dismiss

GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.

Sign up

Branch: master ▼

Find file

Copy path

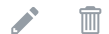
dgs19 / exercises / D\_S6\_L2\_Docker\_Network\_CLI\_commands\_ex.md

 mpoupouka checked\_az

1964352 on May 12

2 contributors 

Raw Blame History



103 lines (91 sloc) 3.11 KB

class: center, middle

# Section 6 - Docker Networking Basics

## 2 - Docker Network CLI commands - Exercise

### Exercise 1

Prerequisites:

- Docker clean environment (no container exists)
  1. Create a nginx container on the default network and expose port 8080 on the host.
  2. Create a user-defined bridge network `net1`.
  3. Connect the nginx container to the `net1` network and disconnect it from the default bridge network.
  4. Verify that nginx container is connected only to the net1 network.

5. Create an alpine container on the `net1` network, override the default CMD to ping the nginx container.
6. Finally, clean up the environment.

## Exercise 1 Solution

1. Create a nginx container on the default network and expose port 8080 on the host:

```
# docker container run --publish 8080:80 --name web_server -d nginx
b59cd0f04a1fba1e5adb3f2433b17f78c49adff9237eddb27c30c9313ace6790
```

2. Create a user-defined bridge network `net1` :

```
# docker network create net1
```

3. Connect the nginx container to the `net1` network and disconnect it from the default bridge network:

```
# docker network connect net1 web_server
# docker network disconnect bridge web_server
```

4. Verify that nginx container is connected only to the `net1` network:

```
# docker network inspect net1
[
  {
    "Name": "net1",
    ...
    "Containers": {
      "b59cd0f04a1fba1e5adb3f2433b17f78c49adff9237eddb27c30c9313ace6790": {
        "Name": "web_server",
        "EndpointID":
        "af70f727e811cc35d731621d0737e406d13d53168445a273165a6cd6abe0a835",
        "MacAddress": "02:42:ac:12:00:02",
```

```

        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    }
}
...

$ docker container inspect web_server
...
    "Networks": {
        "net1": {
...
            "Gateway": "172.18.0.1",
            "IPAddress": "172.18.0.2",

```

5. Create an alpine container on the `net1` network, override the default CMD to ping the nginx container:

```

# docker container run --network net1 alpine ping -c 3 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.078 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.204 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.274 ms

--- 172.18.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.078/0.185/0.274 ms

```

#### Note:

- What happens if we use the container name "web\_server" instead of the IP address for the ping command? Answer:
- Docker provides also a DNS service!

```

# docker container run --network net1 alpine ping -c3 web_server
PING web_server (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.070 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.196 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.255 ms

```

6. Finally, clean up the environment.

- Force delete all containers:

```
# docker rm -f $(docker ps -a -q)
```

- Delete the net1 virtual network:

```
# docker network rm net1
```