Lesson 05 Demo 02

Creating User-Defined Bridge Network

Objective: To create a user-defined bridge network for better control, isolation, and communication between containers within a customized network environment

Tools required: Docker

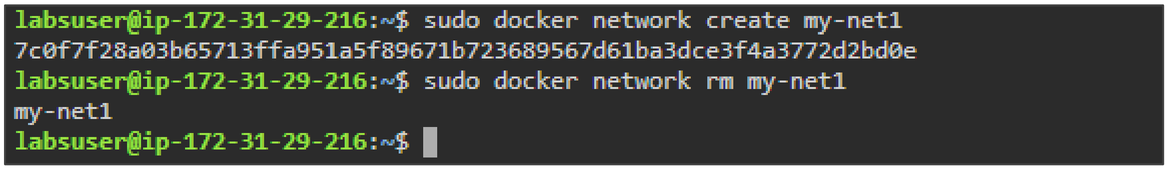
Prerequisites: None

Steps to be followed:
1. Create and delete user-defined network
2. Connect a ngnix container to the my-net network
3. Connect the running my-nginx container to the my-net network
4. Inspect the container and disconnect it from the network

Step 1: Create and delete user-defined network

1.1 Use the following commands to create a user network:

sudo docker network create my-net1
sudo docker network rm my-net1

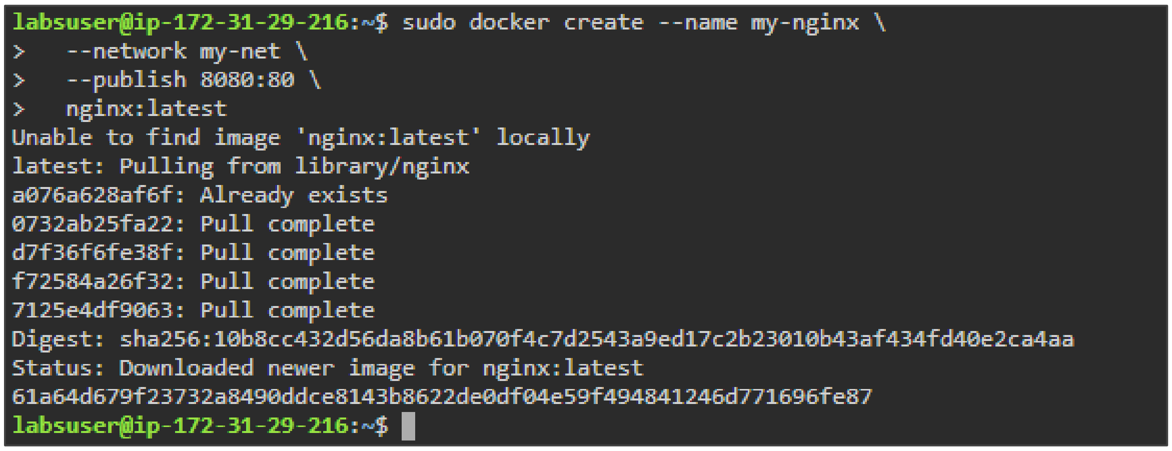


```
labsuser@ip-172-31-29-216:~$ sudo docker network create my-net1
7c0f7f28a03b65713ffa951a5f89671b723689567d61ba3dce3f4a3772d2bd0e
labsuser@ip-172-31-29-216:~$ sudo docker network rm my-net1
my-net1
labsuser@ip-172-31-29-216:~$ 
```

Step 2: Connect a nginx container to the my-net network

2.1 Use the following commands to create a new Docker container named **my-nginx**, attach it to the **my-net** network, and map port 8080 on the host to port 80 in the container:

sudo docker create --name my-nginx \
--network my-net \
--publish 8080:80 \
nginx:latest

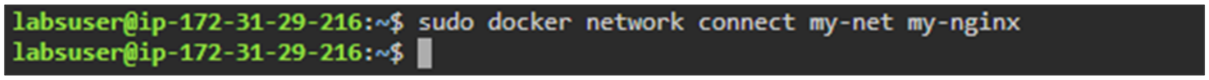


```
labsuser@ip-172-31-29-216:~$ sudo docker create --name my-nginx \
>   --network my-net \
>   --publish 8080:80 \
>   nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
a076a628af6f: Already exists
0732ab25fa22: Pull complete
d7f36f6fe38f: Pull complete
f72584a26f32: Pull complete
7125e4df9063: Pull complete
Digest: sha256:10b8cc432d56da8b61b070f4c7d2543a9ed17c2b23010b43af434fd40e2ca4aa
Status: Downloaded newer image for nginx:latest
61a64d679f23732a8490ddce8143b8622de0df04e59f494841246d771696fe87
labsuser@ip-172-31-29-216:~$
```

Step 3: Connect the running my-nginx container to the my-net network

3.1 Use the following command to connect the running **my-nginx** container to the existing **my-net** network:

sudo docker network connect my-net my-nginx

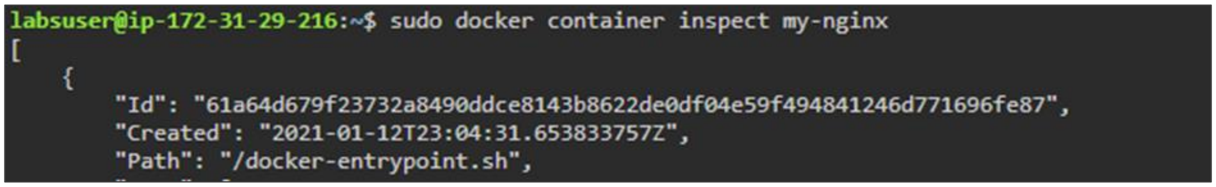


```
labsuser@ip-172-31-29-216:~$ sudo docker network connect my-net my-nginx
labsuser@ip-172-31-29-216:~$
```

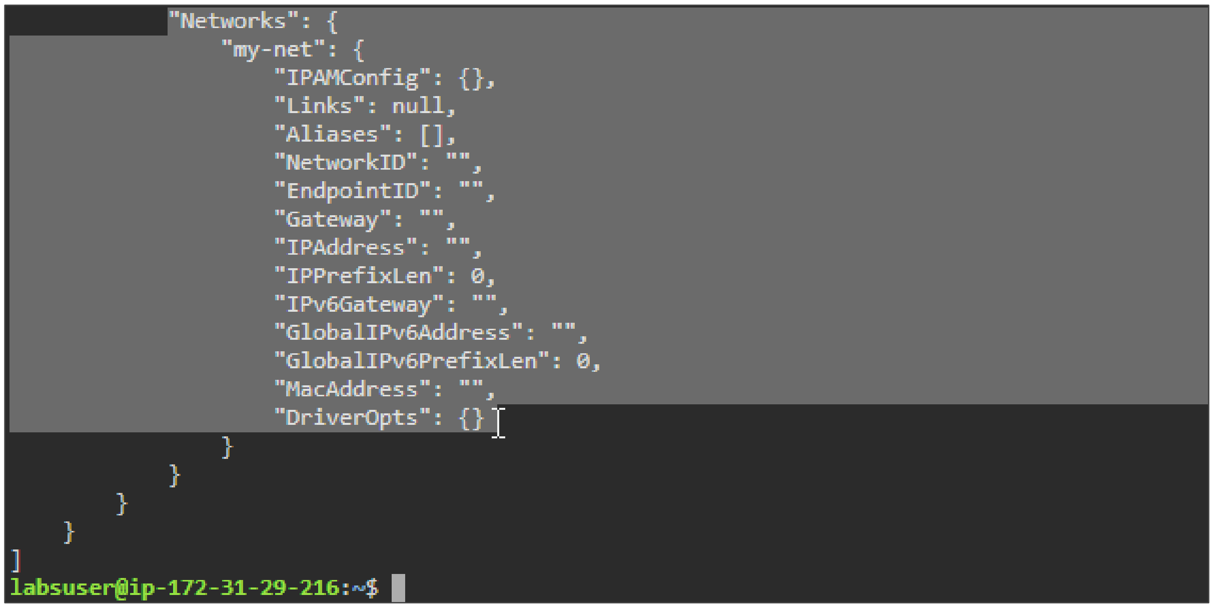
Step 4: Inspect the my-nginx container and check the networks

4.1 Use the following command to inspect and check the network:

sudo docker container inspect my-nginx



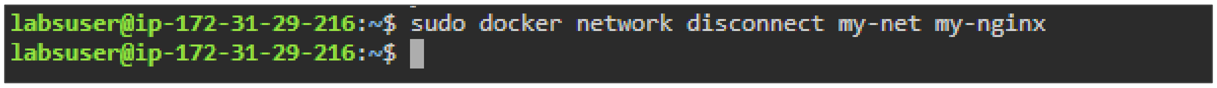
```
labsuser@ip-172-31-29-216:~$ sudo docker container inspect my-nginx
[
    {
        "Id": "61a64d679f23732a8490ddce8143b8622de0df04e59f494841246d771696fe87",
        "Created": "2021-01-12T23:04:31.653833757Z",
        "Path": "/docker-entrypoint.sh",
```



```
            "Networks": {
                "my-net": {
                    "IPAMConfig": {},
                    "Links": null,
                    "Aliases": [],
                    "NetworkID": "",
                    "EndpointID": "",
                    "Gateway": "",
                    "IPAddress": "",
                    "IPPrefixLen": 0,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "",
                    "DriverOpts": {}
                }
            }
        }
    }
]
labsuser@ip-172-31-29-216:~$
```

4.2 Use the following command to disconnect the container from the network:

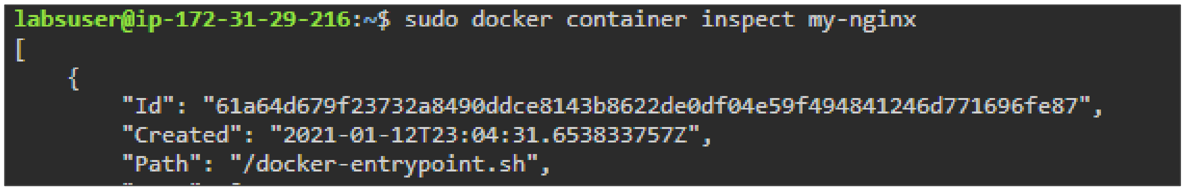
sudo docker network disconnect my-net my-nginx



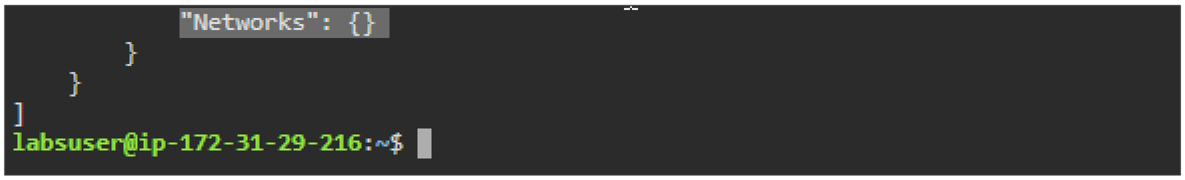
```
labsuser@ip-172-31-29-216:~$ sudo docker network disconnect my-net my-nginx
labsuser@ip-172-31-29-216:~$
```

4.3 Use the following command to inspect the container and check the network:

sudo docker container inspect my-nginx



```
labsuser@ip-172-31-29-216:~$ sudo docker container inspect my-nginx
[
    {
        "Id": "61a64d679f23732a8490ddce8143b8622de0df04e59f494841246d771696fe87",
        "Created": "2021-01-12T23:04:31.653833757Z",
        "Path": "/docker-entrypoint.sh",
```



```
             "Networks": {}
        }
    }
]
labsuser@ip-172-31-29-216:~$
```

By following these steps, you can successfully create a user-defined bridge network for better control, isolation, and communication between containers within a customized network environment.