

## 1. Ziele

- Verständnis von bridge-Netzwerken

## 2. Aufgaben

- Definieren Sie das docker-Netzwerk 192.168.100.0/24 mit Gateway 192.168.100.1

```
docker network create \
--driver=bridge \
--subnet=192.168.100.0/24 \
--gateway=192.168.100.1 \
my_net
```

- Überprüfen Sie den Erfolg mit `docker network ls` und `docker network inspect`

```
vmadmin@lp-22-04:~$ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
5561ba6b7cc1	bridge	bridge	local
5a558bb27117	host	host	local
0ca9993f7d18	my_net	bridge	local
ef77f749ea9d4	mynet	bridge	local
8a532b648872	none	null	local

[illegible]

- Starten Sie 2 ubuntu-Container und ordnen Sie diese dem oben erstellten Netzwerk zu:  
Der erste Container soll seine IP-Adresse via DHCP erhalten. Der zweite soll die IP-Adresse 192.168.100.100 erhalten

The screenshot shows a terminal window with two panes. The left pane is titled 'root@7c524a72ceb5:/'. It contains the following commands and output:

```

vmsdmlng@ip: 22.84: # docker run -it --name ubuntu_2 --netw
ork_my_net --ip=192.168.100.100 ubuntu
root@7c524a72ceb5:/ #

```

The right pane is titled 'root@6d1e9c97234:/'. It contains the following commands and output:

```

vmsdmlng@ip: 22.84: # docker run -it --name ubuntu_1 --new
ork_my_net ubuntu
root@6d1e9c97234:/ #
vmsdmlng@ip: 22.84: # docker run -it --name ubuntu_3 --netw
ork_my_net ubuntu
root@6d1e9c97234:/ #

```

- Überprüfen Sie den Erfolg mit `docker network inspect`

```

def main() {
    // Create a new instance of the class
    var obj = new MyClass();

    // Call the method
    obj.myMethod();

    // Print the result
    Console.WriteLine(obj.myProperty);
}

// This is the class definition
public class MyClass {
    // Private field
    private int myProperty;

    // Public method
    public void myMethod() {
        // Set the value of the property
        myProperty = 42;
    }
}

```

- Installieren Sie in beiden Containern die Pakete `iputils-ping` und `net-tools` (für `ifconfig`)

a	p	t		u	p	d	a	t	e
---	---	---	--	---	---	---	---	---	---

```
apt install iputils-ping
```

- Überprüfen Sie, ob sich die beiden Container gegenseitig anpingen können.

```
root@CS24472ceb5:/# ping ubuntu1
PING ubuntu1 (192.168.100.2) 56(84) bytes of data:
0 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=1 ttl=64 time=0.389 ms
64 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=2 ttl=64 time=0.086 ms
64 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=3 ttl=64 time=0.173 ms
64 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=4 ttl=64 time=0.091 ms
64 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=5 ttl=64 time=0.078 ms
64 bytes from ubuntu1._my.net (192.168.100.2): icmp_seq=6 ttl=64 time=0.066 ms
```

[illegible]

- Stoppen und löschen Sie die Container, löschen Sie das Netzwerk

```
vmadmin@ip-22-04:~$ docker network rm my_net
my_net
vmadmin@ip-22-04:~$ docker network rm mynet
mynet
vmadmin@ip-22-04:~$ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
5561ba6b7cc1	bridge	bridge	local
5a58bb27117	host	host	local
8a532b048872	none	null	local

```
vmadmin@ip-22-04:~$
```

### 3. Hilfsmittel

<https://gbssg.gitlab.io/m347/docker-netzwerke/>

#### 4. Erwartete Resultate

## 17. Erwartete Resultate

Mit Screenshots dokumentiertes und kommentiertes Vorgehen

Zeit: 45 Minuten