

# Virtualização de Redes

14 de Março de 2021

## **Trabalho Prático 1**

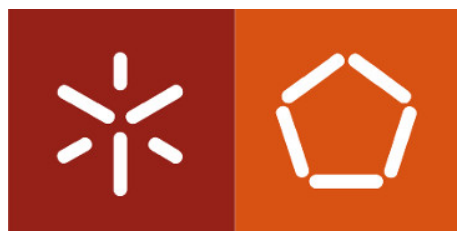
a83899

André Morais

---

## **Docker**

---



Mestrado Integrado em Engenharia Informática  
Universidade do Minho

# Conteúdo

<b>1</b>	<b>Questão 1</b>	<b>2</b>
<b>2</b>	<b>Questão 2</b>	<b>2</b>
2.1	a) & b) . . . . .	2
<b>3</b>	<b>Questão 3</b>	<b>3</b>
3.1	a) . . . . .	3
3.2	b) . . . . .	4
3.3	c) . . . . .	4
<b>4</b>	<b>Questão 4</b>	<b>4</b>
4.1	a) . . . . .	4
4.2	b) . . . . .	4
4.3	c) . . . . .	5
<b>5</b>	<b>Questão 5</b>	<b>6</b>
<b>6</b>	<b>Questão 6</b>	<b>7</b>
<b>7</b>	<b>Questão 7</b>	<b>8</b>
<b>8</b>	<b>Questão 8</b>	<b>9</b>

# 1 Questão 1

O comando dá o nome de alpine que foi criado, mapeando a diretoria */home/morais/docker\_dir* do host para a diretoria */home/internal\_dir* dentro do container. O **ash** corresponde ao interpretador de comandos default do Sistema Operativo

```
sudo docker run -it --name alpine -v \
/home/morais/docker_dir:/home/internal_dir \
-p 8668:9999 alpine:3.12.4 /bin/ash
```

```
morais@morais-VirtualBox:~$ sudo docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
7c2df4b686b4   alpine:3.12.4  "/bin/ash"              28 seconds ago Up 5 seconds  0.0.0.0:8668->9999/tcp             alpine
```

Figura 1: Uso do comando docker -ps a

# 2 Questão 2

Para criar um volume usou-se o seguinte comando:

```
morais@morais-VirtualBox:~$ sudo docker volume create my-volume-1
my-volume-1
morais@morais-VirtualBox:~$ sudo docker volume ls
DRIVER      VOLUME NAME
local       morais_httpd-vol
local       my-volume-1
```

Figura 2: Como criar um volume

## 2.1 a) & b)

Para saber o mountpoint e o driver executei o comando que se segue:

```
morais@morais-VirtualBox:~$ sudo docker volume inspect my-volume-1
[
  {
    "CreatedAt": "2021-03-12T15:50:21Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/my-volume-1/_data",
    "Name": "my-volume-1",
    "Options": {},
    "Scope": "local"
  }
]
```

Figura 3: Características de um volume

Como se pode verificar, o **mountpoint** é */var/lib/docker/volumes/my-volume-1/\_data* e o **driver** é *local*

### 3 Questão 3

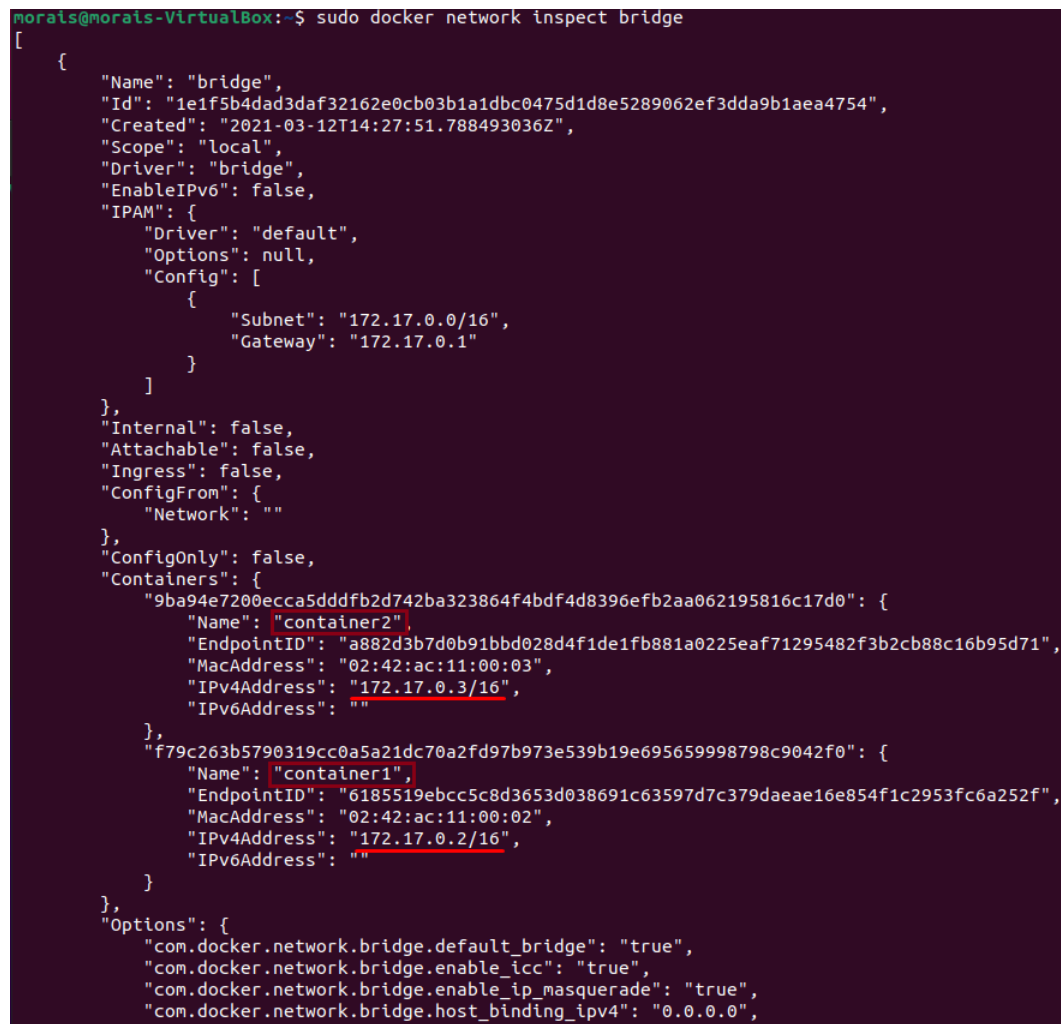
Foram inicialmente criados dois container:

```
sudo docker run -dit --name container1 ubuntu:latest /bin/bash
sudo docker run -dit --name container2 ubuntu:latest /bin/bash
```

A flag '-dit' permite que os dois containers estejam conectados entre uma bridge

#### 3.1 a)

Sim, é possível como podemos verificar na seguinte imagem:



```
Morais@Morais-VirtualBox:~$ sudo docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "1e1f5b4dad3daf32162e0cb03b1a1dbc0475d1d8e5289062ef3dda9b1aea4754",
    "Created": "2021-03-12T14:27:51.788493036Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "9ba94e7200ecca5ddd7b2d742ba323864f4bdf4d8396efb2aa062195816c17d0": {
        "Name": "container2",
        "EndpointID": "a882d3b7d0b91bbd028d4f1de1fb881a0225eaf71295482f3b2cb88c16b95d71",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
      },
      "f79c263b5790319cc0a5a21dc70a2fd97b973e539b19e695659998798c9042f0": {
        "Name": "container1",
        "EndpointID": "6185519ebcc5c8d3653d038691c63597d7c379daee16e854f1c2953fc6a252f",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0"
    }
  }
]
```

Figura 4: Docker network bridge inspect

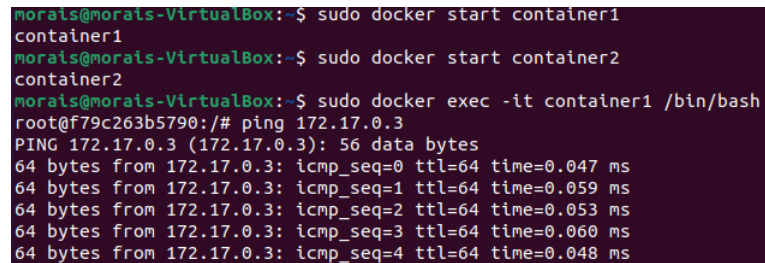
Pode-se verificar que os IPs dos containers associados as redes locais são *172.17.0.3/16* e *172.17.0.2/16*

### 3.2 b)

Não, não se pode comunicar entre eles por nome visto que não estão na mesma network

### 3.3 c)

Sim, é possível comunicar usando os IPs como está representado na figura seguinte:

A terminal window with a dark background and light-colored text. The text shows the execution of Docker commands to start two containers, container1 and container2. Then, container1 is exec'd into, and a ping command is run from inside container1 to 172.17.0.3. The output shows successful ping results with 56 data bytes and various TTL and time values.

```
morais@morais-VirtualBox:~$ sudo docker start container1
container1
morais@morais-VirtualBox:~$ sudo docker start container2
container2
morais@morais-VirtualBox:~$ sudo docker exec -it container1 /bin/bash
root@f79c263b5790:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: icmp_seq=0 ttl=64 time=0.047 ms
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.059 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.060 ms
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.048 ms
```

Figura 5: Ping entre dois containers

## 4 Questão 4

Para criar uma rede com o nome my-network-1 utilizou-se o comando

```
sudo docker network create my-network-1
```

### 4.1 a)

Já tendo um container criado, basta correr o seguinte comando para ligar o container à network

```
sudo docker network connect my-network-1 container1
sudo docker network connect my-network-1 container2
```

### 4.2 b)

Como é possível observar na figura, conseguimos descobrir o IP da subrede criada, como também o IP dos containers ligados à mesma. Também é possível descobrir qual o Driver que esta network está a usar, quando foi criada e o seu nome

```

morais@morais-VirtualBox:~$ sudo docker network inspect my-network-1
[
  {
    "Name": "my-network-1",
    "Id": "b41a0e5d511b30607c683f68b330bbe4f4831fc16abd12ec70d824b5f09161bc",
    "Created": "2021-03-12T16:51:18.812748873Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.19.0.0/16",
          "Gateway": "172.19.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "9ba94e720ecca5dddfb2d742ba323864f4bdf4d8396efb2aa062195816c17d0": {
        "Name": "container2",
        "EndpointID": "a3607de5c312120491dd253774a433ec3fa4fd682f69e1c74aa7f55c263da7b2",
        "MacAddress": "02:42:ac:13:00:03",
        "IPv4Address": "172.19.0.3/16",
        "IPv6Address": ""
      },
      "f79c263b5790319cc0a5a21dc70a2fd97b973e539b19e695659998798c9042f0": {
        "Name": "container1",
        "EndpointID": "e04310bbacee9ea8703f768034bfb858ef5d5c4f17c61aaaa0a6307c63d0ce5e",
        "MacAddress": "02:42:ac:13:00:02",
        "IPv4Address": "172.19.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]

```

Figura 6: Docker Network inspect

#### 4.3 c)

A comunicação entre os containers ligados à network criada, pode ser feita através dos nomes

```

root@f79c263b5790:/# ping container2
PING container2 (172.19.0.3): 56 data bytes
64 bytes from 172.19.0.3: icmp_seq=0 ttl=64 time=0.096 ms
64 bytes from 172.19.0.3: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 172.19.0.3: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 172.19.0.3: icmp_seq=3 ttl=64 time=0.059 ms

```

Figura 7: Ping entre containers pelo nome

## 5 Questão 5

```
morais@morais-VirtualBox:~$ sudo docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
1e1f5b4dad3d        bridge              bridge              local
f204aed3f2d2        host                host                local
c60eac5cf228        morais_container-net bridge              local
b41a0e5d511b        my-network-1        bridge              local
88ea357a97ae        none                null                local
```

Figura 8: Resultado do ls das networks

```
morais@morais-VirtualBox:~$ sudo docker network inspect my-network-1
[
  {
    "Name": "my-network-1",
    "Id": "b41a0e5d511b30607c683f68b330bbe4f4831fc16abd12ec70d824b5f09161bc",
    "Created": "2021-03-12T16:51:18.812748873Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.19.0.0/16",
          "Gateway": "172.19.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "9ba94e720ecca5dddfb2d742ba323864f4bdf4d8396efb2aa062195816c17d0": {
        "Name": "container2",
        "EndpointID": "a3607de5c312120491dd253774a433ec3fa4fd682f69e1c74aa7f55c263da7b2",
        "MacAddress": "02:42:ac:13:00:03",
        "IPv4Address": "172.19.0.3/16",
        "IPv6Address": ""
      },
      "f79c263b5790319cc0a5a21dc70a2fd97b973e539b19e695659998798c9042f0": {
        "Name": "container1",
        "EndpointID": "e04310bbacee9ea8703f768034bfb858ef5d5c4f17c61aaaa0a6307c63d0ce5e",
        "MacAddress": "02:42:ac:13:00:02",
        "IPv4Address": "172.19.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

Figura 9: Resultado do inspect da network

```

morais@morais-VirtualBox:~$ sudo docker volume ls
DRIVER      VOLUME NAME
local       morais_httpd-vol
local       my-volume-1

```

Figura 10: Resultado do ls dos volumes

## 6 Questão 6

```

version: "3"

services:
  nginx:
    image: nginx:latest
    container_name: nginx
    volumes:
      - my_volume:/home
    networks:
      - container_net
    ports:
      - "8888:9999"
  mongo:
    image: mongo:latest
    container_name: mongo
    volumes:
      - my_volume:/home
      - ./home/morais/exemplo:/home/exemplo
    networks:
      - container_net
volumes:
  my_volume:
networks:
  container_net:

```

Figura 11: Ficheiro docker-compose.yml

Foi executado, depois da criação do ficheiro .yml, o comando *sudo docker-compose up*  
*-d*



```

morais@morais-VirtualBox:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED      STATUS        PORTS                               NAMES
34b35f21c08e       mongo:latest       "docker-entrypoint.s..." 3 minutes ago Up 3 minutes 27017/tcp                          mongo
40177a2ef53c       nginx:latest       "/docker-entrypoint..." 3 minutes ago Up 3 minutes 80/tcp, 0.0.0.0:8888->9999/tcp      nginx
9ba94e7200ec       ubuntu:latest      "/bin/bash"              2 hours ago   Exited (0) 2 minutes ago                               container2
f79c263b5790       ubuntu:latest      "/bin/bash"              2 hours ago   Exited (0) 2 minutes ago                               container1
7c2df4b686b4       alpine:3.12.4      "/bin/ash"               2 hours ago   Exited (137) 2 hours ago                               alpine

morais@morais-VirtualBox:~$ sudo docker volume ls
DRIVER      VOLUME NAME
local       c6c97f61572c3d272ad4ff36def63cad5b9f6f5b4c8007fe1d6e6f2731b9b9ab
local       dd1ac3fc02eb6d44a55470fa5822a7ef2c2ad517f20a77f28d98ea94c2344f66
local       morais_httpd-vol
local       morais_my_volume
local       my-volume-1

morais@morais-VirtualBox:~$ sudo docker network ls
NETWORK ID        NAME                DRIVER      SCOPE
1e1f5b4dad3d      bridge              bridge      local
f204aed3f2d2      host                host        local
c60eac5cf228      morais_container-net bridge      local
bef000dfdfa2      morais_container_net bridge      local
b41a0e5d511b      my-network-1        bridge      local
88ea357a97ae      none                null        local

```

Figura 12: Resultados

Na imagem em cima, podemos verificar que os dois novos serviços foram criados e encontram-se a correr. Também foi criada uma network e um volume comum, assim como a porta do nginx foi mapeada para a esperada.

## 7 Questão 7

```

FROM ubuntu:latest
EXPOSE 8888
RUN apt-get update && apt-get -y install netcat
VOLUME /home/output
ENTRYPOINT ["/bin/nc", "-k", "-l", "-p", "8888", "-vv"]

```

Figura 13: Dockerfile

Após criado o Dockerfile foi necessário fazer build do mesmo. Usei o comando *docker build -t netcat:0.1*

```

update-alternatives: warning: skip
nc) doesn't exist
Setting up netcat (1.206-1ubuntu1)
Processing triggers for libc-bin (2
Removing intermediate container 416
--> 06c67ae66716
Step 4/5 : VOLUME /home/output
--> Running in 7267f402359c
Removing intermediate container 726
--> 2ee8edc08a5b
Step 5/5 : ENTRYPOINT ["/bin/nc", "-
--> Running in 3418a40ae120
Removing intermediate container 341
--> 41d454564fa7
Successfully built 41d454564fa7
Successfully tagged netcat:0.1

```

Figura 14: Sucesso na instalação do nginx

Depois ter sido construído com sucesso, foi verificar se as imagens estavam criadas

```
morais@morais-VirtualBox:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
netcat        0.1       41d454564fa7   3 minutes ago  101MB
ubuntu        latest    4dd97cefde62   8 days ago    72.9MB
```

Figura 15: Resultado do comando docker images

De seguida, foi necessário testar e para isso criei um container

```
morais@morais-VirtualBox:~$ sudo docker run --name teste -p 8888:8888 -d netcat:0.1
46ab8b32373cfa0c1b359839060ed1999644a54a4c1d496796543c630d51f62c
```

Figura 16: Container Criado

```
morais@morais-VirtualBox:~$ sudo docker logs teste
Listening on 0.0.0.0 8888
```

Figura 17: docker logs teste

E por fim, ao fazer *wget localhost:8888* podemos verificar que a ligação foi feita com sucesso

```
morais@morais-VirtualBox:~$ wget localhost:8888
--2021-03-12 19:14:08-- http://localhost:8888/
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8888... connected.
HTTP request sent, awaiting response... █
```

Figura 18: Verificar a conexão

## 8 Questão 8

<https://hub.docker.com/r/demorales/vr-tp1>