Vitualização de Redes
Trabalho Prático 1

-

Mestrado em Engenharia Informática
Universidade do Minho
Relatório

Projecto Individual

Pg42845 Moisés Manuel Borba Roriz Ramires

15 de março de 2021

Conteúdo

Capítulo 1

Perguntas

1.1 Pergunta 1

Para cirar o containercom as especeficações pedidas utilizamos o seguinte comando :
docker run -it –name test-port -p 8668:9999 -v /home/user/docker_dir/:/home/internal_dir ubuntu:latest/bin/bash
De seguida criamos um ficheiro no host:

Figura 1.1: Criação de ficheiro

E podemos verificar que este também se encontra no container:

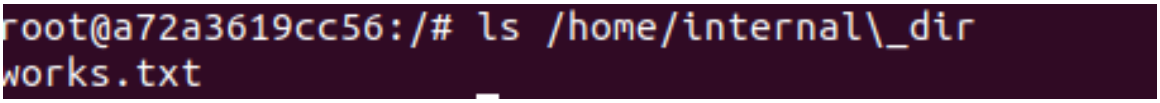

Figura 1.2: Verificação do ficheiro

1.2 Pergunta 2

1.2.1 a)

Para a criação do volume corremos o seguinte comando:
docker volume create my-volume-1

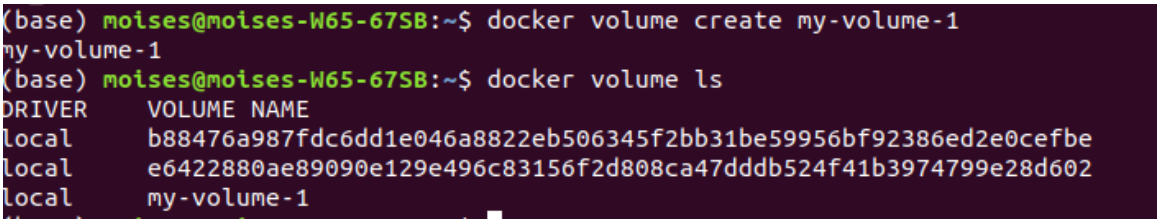

Figura 1.3: Criação do volume

1.2.2 b)

Para inespecionar o mountpoint e a Driver a ser utilizada utilizamos o seguinte comando:
docker volume inspect my-volume-1

```
(base) moises@moises-W65-67SB:~$ docker volume inspect my-volume-1
[
    {
        "CreatedAt": "2021-03-11T17:43:02Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/snap/docker/common/var-lib-docker/volumes/my-volume-1/_data",
        "Name": "my-volume-1",
        "Options": {},
        "Scope": "local"
    }
]
```

Figura 1.4: Inspect do volume

Podemos verificar que a Driver é a local e o Mountpoint é a directoria da imagem.

1.3 Pergunta 3

Para a criação dos containers utilizamos os seguinte comandos :
docker run -it –name first ubuntu:latest /bin/bash

```
(base) moises@moises-W65-67SB:~$ docker run -it --name first ubuntu:latest /bin/
bash
root@4da151f85ef9:/#
```

Figura 1.5: Criação do container first

docker run -it –name second ubuntu:latest /bin/bash

```
(base) moises@moises-W65-67SB:~$ docker run -it --name second ubuntu:latest /bin
/bash
root@15ccda88814c:/#
```

Figura 1.6: Criação do container second

1.3.1 a)

Para inspecionar a bridge utilizamos o seguinte comando:
docker network inspect bridge

```
docker network inspect bridge
```

Figura 1.7: Comando inspect bridge

```
"15ccda88814ce8d06149b3ba20ad0cfdfa07d52ff081958a1a5a65a5ee31032a": {
    "Name": "second",
    "EndpointID": "c66b5735b38cf24af182aee35edcd3130c028393721034209f5b4b5aeaad4b79",
    "MacAddress": "02:42:ac:11:00:05",
    "IPv4Address": "172.17.0.5/16",
    "IPv6Address": ""
},
"4da151f85ef9e773321d561257834b8b0b3b5c0d7bada2602a1006d8477c8cfd": {
    "Name": "first",
    "EndpointID": "e816d7e59fb110bb9b8c619ed2241551cf3aa518d5f9f43aa97e4557a609101c",
    "MacAddress": "02:42:ac:11:00:04",
    "IPv4Address": "172.17.0.4/16",
    "IPv6Address": ""
}
```


Figura 1.8: Resultado inspect bridge

E podemos verificar que é possível obter os IPs dos containers.

1.3.2 b)

Não, os container não consegeum comunicar através dos seus nomes.

```
root@15ccda88814c:/# ping first
ping: first: Name or service not known
```

Figura 1.9: ping name

```
root@4da151f85ef9:/# ping second
ping: second: Name or service not known
```

Figura 1.10: ping namr

1.3.3 c)

Sim,os container consegeum comunicar através dos IPs.

```
root@15ccda88814c:/# ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.
64 bytes from 172.17.0.4: icmp_seq=1 ttl=64 time=0.191 ms
64 bytes from 172.17.0.4: icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from 172.17.0.4: icmp_seq=3 ttl=64 time=0.121 ms
^C
--- 172.17.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
```

Figura 1.11: ping IP

```
root@4da151f85ef9:/# ping 172.17.0.5
PING 172.17.0.5 (172.17.0.5) 56(84) bytes of data.
64 bytes from 172.17.0.5: icmp_seq=1 ttl=64 time=0.128 ms
64 bytes from 172.17.0.5: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 172.17.0.5: icmp_seq=3 ttl=64 time=0.101 ms
^C
--- 172.17.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
```

Figura 1.12: ping IP

1.4 Pergunta 4

O comando utiliza é o seguinte:
docker network create my-network-1

```
(base) moises@moises-W65-67SB:~$ docker network create my-network-1
b7cc16f9f448d8f384afe26577d5039b3481cd8ab46f1e73819957ca7204ad45
```

Figura 1.13: Criação de network

E podemos verificar que esta foi criada:

```
(base) moises@moises-W65-67SB:~$ docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
56f96b241d69   bridge         bridge    local
4cd1f156a0ec   host           host      local
b7cc16f9f448   my-network-1   bridge    local
12b2bc8f4b23   none           null      local
```

Figura 1.14: Criação de network

1.4.1 a)

Utilizando os seguinte comandos:
docker run -dit –name netone –network my-network-1 ubuntu:latest /bin/bash docker run -dit –name nettwo –network my-network-1 ubuntu:latest /bin/bash

```
(base) moises@moises-W65-67SB:~$ docker run -dit --name netone --network my-network-1 ubuntu:latest /bin/bash
7af43d5ccfac3f5be30aaa4fd7ea8d4e14b44a8d2eb9b83c46bb6f772dc9d7a9
^[[A(base) moises@moises-W65-67SB:~$ docker run -dit --name nettwo --network my-network-1 ubuntu:latest /bin/bash
3f49f8afe86e1a1bf0840e40f1a296e4f69e3212c52161fb6afa00bf298231f4
```

Figura 1.15: Criação dos containers

1.4.2 b)

Ao fazer um inspect à network podemos observar vários dados, Id, Nome, Subnet, Driver, Gateway, containners ...

```
(base) moises@moises-W65-67SB:~$ docker network inspect my-network-1
[
    {
        "Name": "my-network-1",
        "Id": "b7cc16f9f448d8f384afe26577d5039b3481cd8ab46f1e73819957ca7204ad45",
        "Created": "2021-03-11T17:58:48.33666458Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3f49f8afe86e1a1bf0840e40f1a296e4f69e3212c52161fb6afa00bf298231f4": {
                "Name": "nettwo",
                "EndpointID": "4d3d0db56f396c616447ca974be0fe3643ccd10f34be6eb83948823988a31fe1",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "7af43d5ccfac3f5be30aaa4fd7ea8d4e14b44a8d2eb9b83c46bb6f772dc9d7a9": {
                "Name": "netone",
                "EndpointID": "e1068c02cb09db113b887086ba023bb9b9979c4a1b50e37d1103c5db747c38a3",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Figura 1.16: Inspect network

1.4.3 c)

Sim, como se pode verificar na figura:

```
root@7af43d5ccfac:/# ping nettwo
PING nettwo (172.18.0.3) 56(84) bytes of data.
64 bytes from nettwo.my-network-1 (172.18.0.3): icmp_seq=1 ttl=64 time=0.133 ms
64 bytes from nettwo.my-network-1 (172.18.0.3): icmp_seq=2 ttl=64 time=0.125 ms
64 bytes from nettwo.my-network-1 (172.18.0.3): icmp_seq=3 ttl=64 time=0.062 ms
^C
--- nettwo ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.062/0.106/0.133/0.031 ms
```

Figura 1.17: ping

```
root@3f49f8afe86e:/# ping netone
PING netone (172.18.0.2) 56(84) bytes of data.
64 bytes from netone.my-network-1 (172.18.0.2): icmp_seq=1 ttl=64 time=0.091 ms
64 bytes from netone.my-network-1 (172.18.0.2): icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from netone.my-network-1 (172.18.0.2): icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from netone.my-network-1 (172.18.0.2): icmp_seq=4 ttl=64 time=0.094 ms
^C
--- netone ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.071/0.092/0.114/0.015 ms
```

Figura 1.18: ping

1.5 Pergunta 5

De acordo com os comandos e os seus outputs, está tudo a funcionar.

1.6 Pergunta 6

Docker compose:

```yaml
version: '3.7'
services:
  service1:
    image: ubuntu
    tty: true
    ports:
      - "8080:80"
    volumes:
      - volume_t:/test
    networks:
      - container-net
  service2:
    image: ubuntu
    tty: true
    ports:
      - "8888:9999"
    volumes:
      - type: volume
        source: volume_t
        target: /test
      - type: bind
        source: ~/Desktop/VR/
        target: /test2
    networks:
      - container-net
volumes:
  volume_t:
networks:
  container-net:
```

Figura 1.19: docker-compose

Para correr o ficheiro corremos o comando:
docker-compose up -d

```
(base) moises@moises-W65-67SB:~/Desktop/VR$ docker-compose up -d
Creating vr_service1_1 ... done
Creating vr_service2_1 ... done
```

Figura 1.20: run docker compose

E podemos verificar que os containers se encontra activos:

```
(base) moises@moises-W65-67SB:~/Desktop/VR$ docker ps -a
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS                    NAMES
785284ea6879   ubuntu    "tail -F anything"   9 seconds ago   Up 6 seconds   0.0.0.0:8888->9999/tcp   vr_service2_1
4c3313290ab5   ubuntu    "tail -F anything"   9 seconds ago   Up 5 seconds   0.0.0.0:8080->80/tcp     vr_service1_1
```

Figura 1.21: Check containers

Podemos verificar a network:

```
(base) moises@moises-W65-67SB:~/Desktop/VR$ docker network inspect vr_container-net
[
    {
        "Name": "vr_container-net",
        "Id": "c28def8a5e3066eded87aaa8cbae74d94b4f01a9451998e6fe52284f64d8dabb",
        "Created": "2021-03-12T18:05:47.420267611Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.22.0.0/16",
                    "Gateway": "172.22.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "4c3313290ab5cbe687f1bc79c58b6f34782362a56f965900157ba21e74a4d201": {
                "Name": "vr_service1_1",
                "EndpointID": "b908e77172aca98e3663aaf70fae0f5df38dd2b73a0f4f36074c454647682c51",
                "MacAddress": "02:42:ac:16:00:03",
                "IPv4Address": "172.22.0.3/16",
                "IPv6Address": ""
            },
            "785284ea68797e34fc403970ed82d8c01b11989ae57142fef08511d0d7d7f08e": {
                "Name": "vr_service2_1",
                "EndpointID": "41e3c6fb73a24b96ad7ecb95f7befb82a9e30771a0e467063868bb3730d90077",
                "MacAddress": "02:42:ac:16:00:02",
                "IPv4Address": "172.22.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "container-net",
            "com.docker.compose.project": "vr",
            "com.docker.compose.version": "1.28.5"
        }
    }
]
```

Figura 1.22: run docker compose

E o volume:

```
(base) moises@moises-W65-67SB:~/Desktop/VR$ docker volume ls
DRIVER    VOLUME NAME
local     b88476a987fdc6dd1e046a8822eb506345f2bb31be59956bf92386ed2e0cefbe
local     e6422880ae89090e129e496c83156f2d808ca47dddb524f41b3974799e28d602
local     my-volume-1
local     vr_volume_t
```

Figura 1.23: volume ls

```
(base) moises@moises-W65-67SB:~/Desktop/VR$ docker volume inspect vr_volume_t
[
    {
        "CreatedAt": "2021-03-12T17:55:37Z",
        "Driver": "local",
        "Labels": {
            "com.docker.compose.project": "vr",
            "com.docker.compose.version": "1.28.5",
            "com.docker.compose.volume": "volume_t"
        },
        "Mountpoint": "/var/snap/docker/common/var-lib-docker/volumes/vr_volume_
t/_data",
        "Name": "vr_volume_t",
        "Options": null,
        "Scope": "local"
    }
]
```

Figura 1.24: volume inspect

1.7 Pergunta 7

Docker file:

```
FROM ubuntu:latest
EXPOSE 8888
RUN apt-get update && apt-get -y install cowsay
VOLUME /home/output
```

Figura 1.25: Docker file

Podemos começar o container:

```
(base) moises@moises-W65-67SB:~/Desktop/VR/dockerfile$ docker build . -t test
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 4dd97cefde62
Step 2/4 : EXPOSE 8888
 ---> Running in 2c14d19a7a85
Removing intermediate container 2c14d19a7a85
 ---> ea928ee14502
Step 3/4 : RUN apt-get update && apt-get -y install cowsay
 ---> Running in 057ea29d8c61
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [21.6 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [681 kB]
```

Figura 1.26: start docker build

E verificar que este se encontra funcional

```
(base) moises@moises-W65-67SB:~/Desktop/VR/dockerfile$ docker run --name dockerbuild -p 8888:8888 -it test
root@9982649a24b3:/# /usr/games/cowsay "hi"
 ____
< hi >
 ----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
root@9982649a24b3:/#
```

Figura 1.27: run container

1.8 Pergunta 8

URL: https://hub.docker.com/r/moisesramires/docker_test