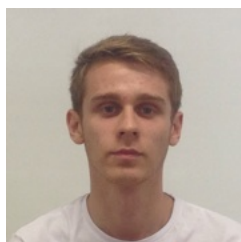


Universidade do Minho

TP1 - Docker

Trabalho Prático de Virtualização de Redes

Mestrado Integrado em Engenharia Informática



João Amorim A74806

14 de Março de 2021

1 Introdução

No âmbito da Unidade Curricular de Virtualização de Redes, foi pedido que estudássemos o TP1 fornecido, com o objetivo de aprender o básico sobre virtualização com Docker. Ao longo do enunciado, são apresentados vários passos para cada secção de modo a que não só aprendamos a teoria, mas para que nos familiarizemos com o uso das ferramentas do Docker. No final são apresentados vários exercícios, sendo o objetivo deste relatório o de apresentar as respostas a estes mesmos exercícios.

Com este trabalho, espera-se que tenham sido obtidos conhecimentos como a criação, remoção e gestão de containers, criação de imagens do Docker, utilização do docker-compose e criação dos Dockerfiles e finalmente a criação de builds automáticas.

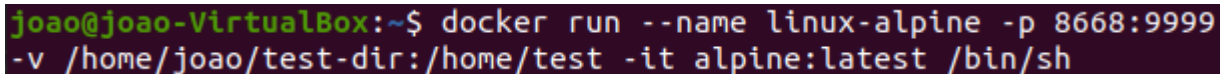
2 Docker - Practical Exercises

Nesta secção, serão apresentadas as respostas/informações relativas aos exercícios do TP1.

2.1 How to create a container from a Linux alpine that has the container 9999 port mapped in the host 8668. It should also have a bind mount, mounting the container/home/internal_dir in the hosts /home/user/docker_dir/.

Para criar o container pedido utilizamos o seguinte comando:

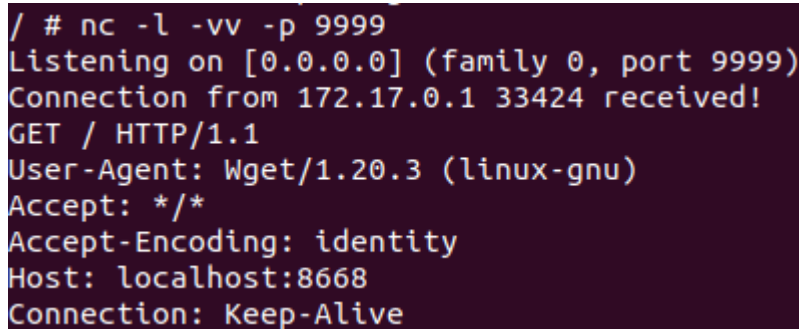
```
$ docker run --name linux-alpine -p 8668:9999 -v /home/joao/test-dir:/home/test -it alpine:latest /bin/sh
```

A terminal window with a dark background and light-colored text. The prompt is 'joao@joao-VirtualBox:~\$'. The command entered is 'docker run --name linux-alpine -p 8668:9999 -v /home/joao/test-dir:/home/test -it alpine:latest /bin/sh'.

```
joao@joao-VirtualBox:~$ docker run --name linux-alpine -p 8668:9999 -v /home/joao/test-dir:/home/test -it alpine:latest /bin/sh
```

Figure 1: Criação do container

A flag `--name` para atribuir o nome ao container, `-p` para o mapeamento dos ports, `-v` para a bind mount e `-it` para o import e build da imagem. O comando `run`, em conjunto com `/bin/sh`, vai criar o container e abrir a shell. Para verificar se o mapeamento dos ports foi feito corretamente, tendo em conta a diferença de comandos no Linux Alpine, corremos na shell (depois da instalação do netcat) do container e no host os seguintes comandos:

A terminal window with a dark background and light-colored text. The prompt is '/ #'. The command entered is 'nc -l -vv -p 9999'. The output shows a connection from 172.17.0.1 on port 33424, with HTTP headers: GET / HTTP/1.1, User-Agent: Wget/1.20.3 (linux-gnu), Accept: */*, Accept-Encoding: identity, Host: localhost:8668, and Connection: Keep-Alive.

```
/ # nc -l -vv -p 9999
Listening on [0.0.0.0] (family 0, port 9999)
Connection from 172.17.0.1 33424 received!
GET / HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: localhost:8668
Connection: Keep-Alive
```

Figure 2: Verificação do mapeamento dos ports no container

```
joao@joao-VirtualBox:~$ wget localhost:8668
--2021-03-13 23:18:25-- http://localhost:8668/
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:8668... connected.
HTTP request sent, awaiting response...
```

Figure 3: Verificação do mapeamento dos ports no host

2.2 How to create the volume "my-volume-1"?

Com o comando:

```
docker volume create my-volume-1
```

```
joao@joao-VirtualBox:~$ docker volume inspect my-volume-1
[
  {
    "CreatedAt": "2021-03-13T20:14:35Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/my-volume-1/_data",
    "Name": "my-volume-1",
    "Options": {},
    "Scope": "local"
  }
]
```

Figure 4: Comando para inspeção do volume.

2.2.1 What is the volume mountpoint?

De acordo com a imagem, depois de efetuado o comando,

```
docker volume inspect my-volume-1
```

o mountpoint é:

```
/var/lib/docker/volumes/my-volume-1/_data
```

2.2.2 Which driver is being used?

O driver que está a ser usado é o local.

2.3 Create two basic containers (They should not be attached to any manually created network).

Containers criados com:

```
$docker run -it --name test-net-1 ubuntu:latest /bin/bash
$docker run -it --name test-net-2 ubuntu:latest /bin/bash
```

2.3.1 Is it possible to inspect the bridge network and find their IP addresses??

Depois de criados os containers, se usarmos o comando

```
docker network inspect bridge
```

verificamos que é possível descobrir os seus IP's.



```
"Name": "test-net-2",
"EndpointID": "5627cf31d6db8855991ea99
8",
"MacAddress": "02:42:ac:11:00:03",
"IPv4Address": "172.17.0.3/16",
"IPv6Address": ""

e65bc3bf584b2b529036dd03099e3b554e78b676d

"Name": "test-net-1",
"EndpointID": "e1f4a9839fb7ab164d6c8c5
e",
"MacAddress": "02:42:ac:11:00:02",
"IPv4Address": "172.17.0.2/16",
"IPv6Address": ""
```

Figure 5: Bridge Network

2.3.2 Is it possible for the containers to communicate among themselves using their names?

Não. Foi efetuado um ping do container test-net-1 para o test-net-2 e obteve-se o seguinte:

```
root@08194acb140b:/# ping test-net-2
ping: test-net-2: Name or service not known
```

Figure 6: Ping para o container test-net-2 através do nome

2.3.3 And with their IPs?

Sim. Foi efetuado um ping do container test-net-1 para o IP do container test-net-2, nomeadamente 172.17.0.3 e obteu-se o seguinte:

```
root@08194acb140b:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.078 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 172.17.0.3: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 172.17.0.3: icmp_seq=6 ttl=64 time=0.059 ms
```

Figure 7: Ping para o container test-net-2 através do IP

2.4 What is the command to create the network “my-network-1”?

O comando é:

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

2.4.1 How to attach two containers to that network?

Para ligar os dois container à rede, executamos os seguintes comandos:

```
$ docker run -it --name test-user-net-1 --network my-network-1 ubuntu:latest /bin/
$ docker run -it --name test-user-net-2 --network my-network-1 ubuntu:latest /bin/
```

2.4.2 Which relevant parameter is possible to found about that network?

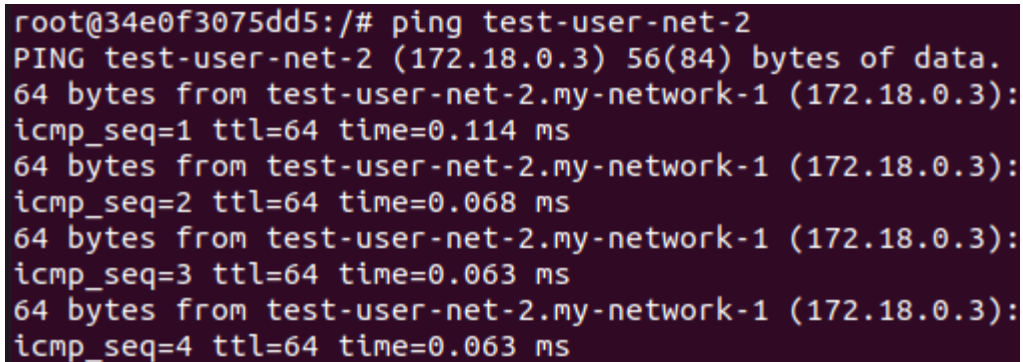
Se executarmos o comando

```
$ docker network inspect my-network-1
```

verificamos que foi criada uma conexão bridge, onde é possível observar a configuração dos dois containers, estando estes conectados à my-network-1.

2.4.3 Can the containers ping one another using their name?

Sim. Uma vez ligados à rede criada my-volume-1, se executarmos o ping, por nome, do primeiro container, test-user-net-1, para o segundo container, test-user-net-2, obtemos o seguinte:



```
root@34e0f3075dd5:/# ping test-user-net-2
PING test-user-net-2 (172.18.0.3) 56(84) bytes of data.
64 bytes from test-user-net-2.my-network-1 (172.18.0.3):
icmp_seq=1 ttl=64 time=0.114 ms
64 bytes from test-user-net-2.my-network-1 (172.18.0.3):
icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from test-user-net-2.my-network-1 (172.18.0.3):
icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from test-user-net-2.my-network-1 (172.18.0.3):
icmp_seq=4 ttl=64 time=0.063 ms
```

Figure 8: Ping entre os containers através do nome

2.5 Which were the results obtained from the commands `docker network ls`, `docker network inspect` and `docker volume ls` in section 1 of this document? What conclusions is possible to take from the result of these commands?

estes comandos permitem-nos obter várias informações. `Docker network ls`, diz-nos que redes temos disponíveis no docker, o `docker network inspect`, permite observar a configuração de uma determinada rede e que containers estão ligados a essa rede e finalmente o `docker volume ls` que nos mostra que volumes existem para persistência de dados.

No primeiro caso, antes de executar a secção 1, era possível observarmos 3 networks default, nomeadamente bridge, host e none, enquanto que depois de executados pontos desta mesma secção, é suposto observarmos mais uma rede chamada my-network-1. No segundo caso, não existem diferenças entre o início deste TP e o fim, mas foi sem dúvida uma ferramenta útil. No terceiro e último caso, inicialmente não existem quaisquer volumes criados,

sendo depois criados os volumes test-vol, que acaba por ser removido, e o new-test-vol.

2.6 Create a docker-compose that starts at least two services:

O ficheiro docker-compose.yml criado foi o seguinte:

```
version: '3'
services:
  ubuntu:
    image: ubuntu:latest
    ports:
      - "8888:9999"
    volumes:
      - test-vol:/home/test
    networks:
      - container-net
  alpine:
    image: alpine:latest
    volumes:
      - test-vol
    networks:
      - container-net
volumes:
  test-vol:
networks:
  container-net:
```

Com os comandos ls, podemos ver que os volumes, as networks e os containers foram criados. Para cada caso, foi adicionado automaticamente o nome da diretoria, algo que ainda não percebi se é suposto acontecer. No caso dos containers, com o comando docker ps -a, podemos ver que estes foram criados mas que não estão a correr.

2.7 Create a Dockerfile:

Dockerfile para a construção de uma imagem de um Apache Web Server.

```
FROM centos:latest
EXPOSE 80
RUN yum -y install httpd
VOLUME /usr/test
ENTRYPOINT ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

2.8 Create an automatic build. The docker file built in the previous exercise may be used.

```
hub.docker.com/r/jlpamorim/apachewebserver
docker pull jlpamorim/apachewebserver
```

3 Conclusão

Em suma, penso que este trabalho foi bastante esclarecedor. Esta primeira parte permitiu-me perceber o que é realmente a virtualização com Docker, quais as suas características, ferramentas e objetivos.

Uma vez que qualquer destas matérias me era desconhecida, penso que o conhecimento obtido foi bastante positivo.