Universidade do Minho
Escola de Engenharia

MiEI - Ano letivo 2020/2021

Virtualização de Redes

TP1

14 de março de 2021

Filipe Miguel Teixeira Freitas
Guimarães - A85308

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/.

- `docker pull alpine`

```
wtv@wtv-pc ~ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
ba3557a56b15: Pull complete
Digest: sha256:a75afd8b57e7f34e4dad8d65e2c7ba2e1975c795ce1ee22fa34f8cf46f96a3be
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

- `docker run --name myalpine`
  `-p 9999:8668`
  `-v /home/wtv:/home/usr/docker_dir/`
  `-it alpine:latest`

```
wtv@wtv-pc ~ docker run --name myalpine -p 9999:8668 -v /home/wtv:/home/usr/docker_dir/ -it alpine:latest
/ # cd home/usr/docker_dir/
/home/usr/docker_dir # l
```

2 How to create the volume “my-volume-1”?

- `docker volume create`

```
wtv@wtv-pc ~ docker volume create my-volume-1
my-volume-1
wtv@wtv-pc ~ docker volume ls
DRIVER    VOLUME NAME
local     my-volume-1
```

(a) **What is the volume mountpoint?**

(b) **Which driver is being used?**

Consegue-se verificar qual é o moutpoint e a driver usada correndo o comando

- `docker volume inspect my-volume-1`

e verificando que se encontra em */var/lib/docker/volumes/my-volume-1/_data* com a *driver* local como se pode verificar na seguinte figura.

```
wtv@wtv-pc ~ docker volume inspect my-volume-1
[
    {
        "CreatedAt": "2021-03-10T15:48:56Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/my-volume-1/_data",
        "Name": "my-volume-1",
        "Options": {},
        "Scope": "local"
    }
]
```

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

```
wtv@wtv-pc ~ docker run --name ubuntu1 -it ubuntu:latest /bin/bash
root@cf53ad0d3c76:/#

wtv@wtv-pc ~ docker run --name ubuntu2 -it ubuntu:latest /bin/bash
root@d0106346d3ae:/#
```

(a) **Is it possible to inspect the bridge network and find their IP addresses?**

Ao correr o comando *docker inspect bridge* conseguem-se ver ambos os containers criados bem como os respetivos endereços de IP.

(b) **Is it possible for the containers to communicate among themselves using their names?**

Não tendo uma *network* prtilhada não é possivel comunicarem pelos respetivos nomes.

```
root@cf53ad0d3c76:/# ping ubuntu2
ping: ubuntu2: Name or service not known
```

(c) **And with their IPs?**

Como por defeito são conectados em *bridge* é possível comunicarem usando os respetivos IP's.

```
root@cf53ad0d3c76:/# 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=0.374 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.157 ms
^C
--- 172.17.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1018ms
rtt min/avg/max/mdev = 0.157/0.265/0.374/0.108 ms
```

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

Para criar uma network procede-se como visto na seguinte imagem.

```
wtv@wtv-pc ~ docker network create my-network-1
c0335fe85c88f8cd8c8ee516d21ececbedd26de87cbffe364fb65cb96d2e2bb6
```

(a) **How to attach two containers to that network?**

Para adicionar uma determinada rede a um container coloca-se o nome da rede na *flag –network*.

```
wtv@wtv-pc ~ docker run -dit --name ubuntu-1 --network my-network-1 ubuntu:latest /bin/bash
ba4274006d4162115954278a161ec6fc3be87e13b613da4de43d322e629e8240
wtv@wtv-pc ~ docker run -dit --name ubuntu-2 --network my-network-1 ubuntu:latest /bin/bash
7638465104ca82dfbb7775d8ce5ecf018f9e54815992d0df1605da9c273a0762
```

(b) **Which relevant parameter is possible to found about that network?**

Ao correr o comando *docker network inspect* à rede previamente criada consegue-se ver quais containers estão associados à mesma, bem como os respetivos endereços IP.

```
wtv@wtv-pc ~ docker network inspect my-network-1
[
    {
        "Name": "my-network-1",
        "Id": "c0335fe85c88f8cd8c8ee516d21ececbedd26de87cbffe364fb65cb96d2e2bb6",
        "Created": "2021-03-13T17:49:31.367717337Z",
        "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": {
            "7638465104ca82dfbb7775d8ce5ecf018f9e54815992d0df1605da9c273a0762": {
                "Name": "ubuntu-2",
                "EndpointID": "053c334778b35daa4172a9124b10d30ffc7a7158cdd9f908612943d4acf499de",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "ba4274006d4162115954278a161ec6fc3be87e13b613da4de43d322e629e8240": {
                "Name": "ubuntu-1",
                "EndpointID": "9127901868c6c68fc58959ed286393a7f96ffde4bfcb39eb2cf0674bfecf2913",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

(c) **Can the containers ping one another using their name?**

Desta vez os containers já conseguem fazer ping entre si usando o nome como se pode comprovar pela imagem.

```
wtv@wtv-pc ~ docker attach ubuntu-1
root@ba4274006d41:/# ping ubuntu-2
PING ubuntu-2 (172.18.0.3) 56(84) bytes of data.
64 bytes from ubuntu-2.my-network-1 (172.18.0.3): icmp_seq=1 ttl=64 time=0.191 ms
64 bytes from ubuntu-2.my-network-1 (172.18.0.3): icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from ubuntu-2.my-network-1 (172.18.0.3): icmp_seq=3 ttl=64 time=0.121 ms
^C
--- ubuntu-2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.121/0.144/0.191/0.033 ms
root@ba4274006d41:/#
```

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?

Com o comando *docker network ls* conseguimos ver que por defeito o docker tem 3 redes ativas. A *bridge* que é a *network driver* por defeito, a *host* que serve para comunicação entre o container e o *host* e *null* para a não existência de *network* num container.

```
wtv@wtv-pc ~ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
e7dc59e19e72   bridge    bridge    local
615aa60a1dea   host      host      local
04bd295db161   none      null      local
```

Ao fazer *docker network inspect bridge* conseguimos perceber quais containers estão a usar a bridge e qual é o endereço atribuido a este. Neste caso estão a correr os containers com nome test-net-1 e test-net-2 com os endereços de IP *127.17.0.2* e *127.17.0.3*, respetivamente.

```
wtv@wtv-pc ~ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "e7dc59e19e727996a140739e6d51fa5d648a6c32ceb9daa2f64f5f506f9f6d2a",
        "Created": "2021-03-10T14:03:07.573610894Z",
        "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": {
            "6b91a4f9e9a52cf9b31f4530e6e97689c530702f768f059e20a3dd17456962dc": {
                "Name": "test-net-2",
                "EndpointID": "8d9cb868cb368b5c79ad0717b291e1d981b02f41da1941aacf2ebfca2ceb0bc1",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "e42b77d25742eec3b435083fe22577e01a8b8ab01485188e7dd943ca92a15eeb": {
                "Name": "test-net-1",
                "EndpointID": "c25900c5f54870798aecfdc8399feffae776e2e8fe5f3508db9a974625b294a1",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
```

Por fim, com o comando *docker volume ls* conseguimos ver os volumes ativos bem como a respetiva *driver*. Neste caso existe um volume ativo com o nome *test-vol* e *driver* local.

```
wtv@wtv-pc ~ docker volume ls

DRIVER    VOLUME NAME
local     test-vol
```

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

(a) **These should be in the same docker network;**

(b) **These should share the same docker volume;**

(c) **One of the services should also have one bind mount;**

(d) **One of the services should have their 9999 port exposed in the 8888 host port;**

```yaml
version: '3'
services:
  service1:
    image: ubuntu:latest
    stdin_open: true # -i
    tty: true        # -t
    ports:
      - "9999:8888"
    volumes:
      - new-volume
      - /home/wtv/bind:/home/bind
    networks:
      - net
  service2:
    image: ubuntu:latest
    stdin_open: true # -i
    tty: true        # -t
    volumes:
      - new-volume
    networks:
      - net
volumes:
  new-volume:
networks:
  net:
```

Corre-se o *docker-compose* fazendo *docker-compose up*.

```
wtv@wtv-pc ~/universidade/2semestre/VR/Praticas/VR_TP1 main docker-compose up
Creating vr_tp1_service2_1 ... done
Creating vr_tp1_service1_1 ... done
Attaching to vr_tp1_service1_1, vr_tp1_service2_1
```

Verifica-se que estão os dois serviços a executar. Também consegue-se ver que o serviço 1 tem uma porta interna a ser partilhada para fora.

```
wtv@wtv-pc ~ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS         PORTS                    NAMES
a580d55b18cd   ubuntu:latest  "/bin/bash"   3 minutes ago   Up 3 minutes   0.0.0.0:9999->8888/tcp   vr_tp1_service1_1
1ec22ac172fd   ubuntu:latest  "/bin/bash"   3 minutes ago   Up 3 minutes                            vr_tp1_service2_1
```

Pode-se fazer *docker attach* para aceder à *bash*.

```
wtv@wtv-pc ~ docker attach vr_tp1_service1_1
root@a580d55b18cd:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  new-volume  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@a580d55b18cd:/#
```

Pode-se verificar ainda que tem uma diretoria montada em */home/bind*.

```
root@a580d55b18cd:/# cd home/
root@a580d55b18cd:/home# l
bind/
```

(e) **Using the inspect and ls commands, what conclusions can you take about their results?**

Consegue-se ver a rede criada.

```
wtv@wtv-pc ~ docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
b1332a3cb97a   bridge       bridge    local
615aa60a1dea   host         host      local
04bd295db161   none         null      local
fce867d6ccb3   vr_tp1_net   bridge    local
```

Bem como os container associados e os respetivos IP's.

```
wtv@wtv-pc ~ docker network inspect vr_tp1_net
[
    {
        "Name": "vr_tp1_net",
        "Id": "fce867d6ccb314cba61941cb368ab2174b47ad37c7f8c60fd141f514da80e3ad",
        "Created": "2021-03-13T21:53:13.789303864Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1ec22ac172fd24de29a234c1303becd6c704b8927b18f225a52e5f42eae2e055": {
                "Name": "vr_tp1_service2_1",
                "EndpointID": "4f12cd87c86a360d847af348acecdcca2b99de083d7c2374df25d8e26ec92d0e",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            },
            "a580d55b18cd47bed5ec2fa4ecc8bdd2b2c6f4eb868fa51786e82093ab19812a": {
                "Name": "vr_tp1_service1_1",
                "EndpointID": "698a70fa5978be8821ffb3b331e161ac4426b552e2d47feaa01b85839243e8b5",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "net",
            "com.docker.compose.project": "vr_tp1",
            "com.docker.compose.version": "1.28.5"
        }
    }
```

Com o comando *docker volume ls* conseguimos ver que o volume foi corretamente criado.

```
local     vr_tp1_new-volume
```

Por fim conseguimos com o *inspect* obter mais informações sobre o mesmo.

```
wtv@wtv-pc ~ docker volume inspect vr_tp1_new-volume
[
    {
        "CreatedAt": "2021-03-13T21:53:14Z",
        "Driver": "local",
        "Labels": {
            "com.docker.compose.project": "vr_tp1",
            "com.docker.compose.version": "1.28.5",
            "com.docker.compose.volume": "new-volume"
        },
        "Mountpoint": "/var/lib/docker/volumes/vr_tp1_new-volume/_data",
        "Name": "vr_tp1_new-volume",
        "Options": null,
        "Scope": "local"
    }
]
```

7 Create a Docker file that:

(a) **Have some tool installed. Any that is chosen by the student. Extra points for any application that is listening from outside requests (web server, for example);**

(b) **Has a volume, for later persistence;**

Para concretizar o que é pedido no enunciado decidi então recorrer ao *docker hub* e encontrar uma versão do node que pudesse facilmente instalar pacotes. Escolhi o alpine e fiz upgrade e instalei o vim.

Criei a pasta para armazenar os ficheiros necessários para executar um servidor *node.js* executei os respetivos comandos necessários para instalar as dependências.

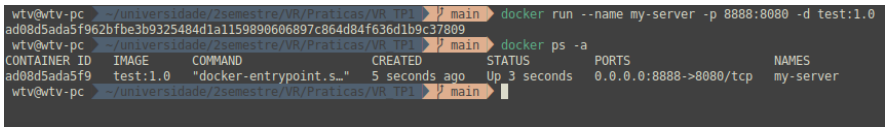
Por fim coloquei o comando *npm start* para ser executado no execução deste container.

```
1 FROM node:alpine
2
3 #Expor a porta onde vai estar instalado o servidor
```

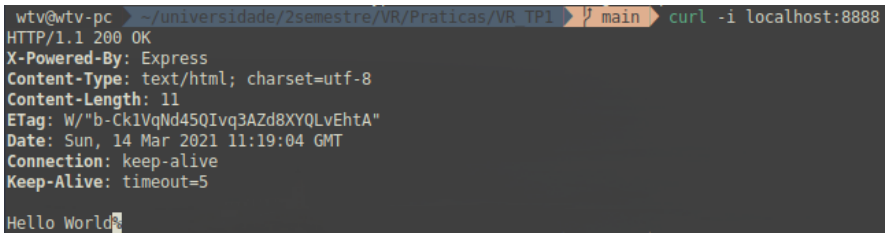
```
4  EXPOSE 8080
5
6  #Atualizar o Alpine
7  RUN apk -U upgrade
8  RUN apk add vim
9
10 #Criar a pasta para os ficheiros do servidor
11 RUN mkdir -p /home/site
12
13 #Instalar as dependencias do servidor
14 WORKDIR /home/site
15 COPY package*.json ./
16 RUN npm install
17
18 #Copiar os ficheiros para dentro do container
19 COPY . .
20
21 #Criar um volume para presistencia
22 VOLUME /home/site
23
24 #Comando a executar quando o container for executado
25 CMD [ "npm","start" ]
```

Para ”compilar”o Docker file executa-se comando *docker build . -t test:1.0*.

Consegue-se agora correr o container e para isso defini a porta 8888 para corresponder com a porta interna 8080.



Pode se testar a comunicação com o exterior usando o *curl*.



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

Repositório no GitHub:
https://github.com/filipeguimaraes/VR_TP1/
Docker hub:
https://hub.docker.com/r/filipeguimaraes99/vr_tp1
