

Roteiro Palestra Docker

Fernando Anselmo
fernando.anselmo74@gmail.com

28 de junho de 2020

Comandos Iniciais

```
$ docker version
$ docker images
$ docker ps -a
```

1 Caso MySQL

MySQL é um sistema de gerenciamento de banco de dados, que utiliza a linguagem SQL. Atualmente é um dos SGBD mais populares.

```
$ docker pull mysql
$ docker run --name mybanco -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 -d mysql
```

Nas próximas vezes:

```
$ docker start mybanco
$ docker exec -it mybanco sh -c 'exec mysql -u root -p'
$ docker exec -it mybanco bash
root@b38dfbb9c50d:/# mysql -u root -p
mysql> select @@version;
$ docker stop mybanco
```

2 Caso Pentaho

Pentaho é um software de código aberto para inteligência empresarial, desenvolvido em Java. A solução cobre as áreas de ETL, reporting, OLAP e mineração de dados.

```
$ docker pull wmarinho/pentaho
$ docker run --name pentaho-server -p 8080:8080 -d wmarinho/pentaho
```

Acessar a imagem:

```
$ docker exec -it pentaho-server bash
```

Podemos ver aonde está o Pentaho:

```
# echo $PENTAHO_HOME
```

Ou podemos acessá-lo pelo navegador no endereço: <http://localhost:8080/pentaho/Login>.

Usuário: admin. Senha: password.

3 Caso PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional de objetos (ORDBMS) com ênfase em extensibilidade e conformidade com padrões. Lida com cargas de trabalho que variam de pequenos aplicativos a grandes aplicativos voltados para a Internet (ou para armazenamento de dados) com muitos usuários simultâneos.

```
$ docker pull postgres
$ docker run --name postbanco -e POSTGRES_PASSWORD=postgres -d -p 5432:5432
postgres
$ docker start postbanco
$ docker stop postbanco
```

Docker Compose

Compose é uma ferramenta para definir e executar aplicativos Docker com vários contêineres.

```
$ docker-compose build
$ docker-compose rm
$ docker-compose (up | down)
$ docker-compose (start | stop)
$ docker-compose run -d
$ docker-compose exec
$ docker-compose logs (-f)
```

4 Caso PostgreSQL Compose

Arquivo: docker-compose.yml

```
1 version: '2'
2 services:
3   db:
4     image: postgres
5     restart: always
6     environment:
7       POSTGRES_PASSWORD: postgres
8       POSTGRES_USER: postgres
9     ports:
10      - 5432:5432
11   adminer:
12     image: adminer
13     restart: always
14     ports:
15      - 8080:8080
16   client:
17     image: postgres
18     depends_on:
19      - db
20     command: psql -U postgres --password -h db
21   db-legacy:
22     image: postgres:9.5
23     restart: always
24     environment:
25       POSTGRES_PASSWORD: postgres
26       POSTGRES_USER: postgres
27     ports:
28      - 5532:5432
```

Levantando os contêineres:

```
$ docker-compose up
```

Nova aba do terminal na mesma pasta: Ctrl + Shift + T \$ docker-compose ps

```
$ docker-compose run client
```

```
# create database teste;
```

```
# \connect teste;
```

```
# create table base(id serial not null, nome varchar(50), primary key (id));
```

```
# \d
```

```
# \dS+
```

```
# \d base
```

```
# \q
```

```
$ docker exec -it palestra_db-legacy_1 psql -U postgres --password
```

Encerrar:

```
$ docker-compose down
```

5 Caso NGINX

Nginx é um servidor web rápido, leve, e com inúmeras possibilidades de configuração para melhor performance. Criar a seguinte estrutura de arquivos:

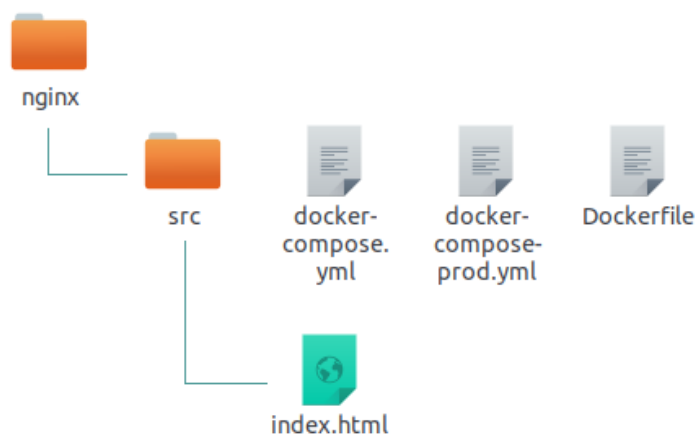


Figura 1: *Estrutura dos Arquivos*

1. Arquivo "src/index.html"

```
1 <html>
2   <body>
3     <h1>Hello World</h1>
4   </body>
5 </html>
```

2. Arquivo "Dockerfile"

```
1 FROM nginx
2 COPY src /usr/share/nginx/html
```

3. Arquivo "docker-compose.yml"

```
1 version: '2'
```

```

2 services:
3   app:
4     build: .
5     image: app:1.0.0
6     volumes:
7       - ./src:/usr/share/nginx/html
8     ports:
9       - 8080:80

```

4. Arquivo "docker-compose-prod.yml"

```

1 version: '2'
2 services:
3   app:
4     build: .
5     image: app:1.0.0
6     ports:
7       - 80:80

```

As ações serão as seguintes:

```
$ docker-compose up --build
```

Acessar <http://localhost:8080>

Acessar outro terminal [CTRL+SHIFT+T]

```
$ docker-compose down
```

```
$ docker-compose -f docker-compose-prod.yml up --build
```

Acessar <http://localhost>

```
$ docker-compose down
```

6 Caso Django

Django é um framework para desenvolvimento rápido para web, escrito em Python, que utiliza o padrão model-template-view. Exemplo em: <https://gist.github.com/shudarshon/cf56741e6bcc26bedd4db236447e1654>:

```
$ docker-compose build
```

```
$ docker-compose up -d
```

```
$ docker-compose logs
```

```
$ docker inspect exdjango_postgres_1 | grep IP$ psql -h 172.17.0.3 -p 5432 -U postgres --password$ docker-compose down
```

7 Caso Raspberry Pi

Necessidades:

- Cabo crossover
- Fonte alimentação Raspberry (cabo mini USB)

Descobrir o Raspberry na rede:

1. Qual o prefixo do seu IP da Rede (na qual deve estar o Raspberry)?

```
$ ifconfig
```

2. Localizar o Raspberry no mesmo prefixo de IP (p.e. 192.168.10.x)

```
$ nmap -n -sP 192.168.10.255/24 (daqui para frente assumirei o IP do Raspberry como 192.168.10.2)
```

7.1 Instalar a Docker Machine

```
$ base=https://github.com/docker/machine/releases/download/v0.14.0 && curl -L  
$ base/docker-machine-$(uname -s)-$(uname -m) > /tmp/docker-machine && sudo install  
/tmp/docker-machine /usr/local/bin/docker-machine
```

Para copiar arquivo para o Raspberry:

```
$ scp machine.png pi@192.168.25.2:/home/pi/html
```

Tabela de Roteamento IP do Kernel

```
$ netstat -rn
```

Acessar o Raspberry:

```
$ ssh pi@192.168.10.2 (Senha: raspberry)
```

```
$ sudo nano /etc/os-release
```

Mudar o id: **ID=raspbian** para **ID=debian**

```
$ curl -sSL https://get.docker.com | sh
```

```
$ sudo usermod -aG docker pi
```

```
$ exit
```

7.2 Gerar as chaves: particular e pública

```
$ ssh-keygen -b 2048 -t rsa (key:id_rsa passphrase:raspberry)
```

```
$ cat ~/.ssh/id_rsa.pub | ssh -p 22 pi@192.168.10.2 'cat >>.ssh/authorized_keys'
```

7.3 Instalar o Docker no Raspberry

Comandos no Raspberry:

```
$ nano /etc/ssh/sshd_config
```

Parâmetro: **'#PasswordAuthentication yes'** para **'PasswordAuthentication no'**

```
$ sudo /etc/init.d/ssh restart
```

7.4 Criar a Docker Machine

```
$ docker-machine create --driver generic --generic-ip-address 192.168.10.2  
--generic-ssh-key ~/.ssh/id_rsa --generic-ssh-user pi --engine-storage-driver  
overlay2 pi-zero-1  
$ docker-machine ip pi-zero-1
```

7.5 Após Criada a Docker Machine

```
$ docker-machine env pi-zero-1
```

```
$ eval &(docker-machine env pi-zero-1)
```

Testar:

```
$ docker-machine ssh pi-zero-1
```

Criar uma pasta html/ e nela um arquivo index.html simples.

```
$ exit
```

```
$ docker run -d -p 80:80 --name nginx2 -v /home/pi/html:/var/www/html tobi312/rpi-nginx
```

Acessar: <http://192.168.10.2/>

7.6 Finalizar

```
$ ssh pi@192.168.10.2 (Senha: raspberry)
```

```
$ sudo halt
```