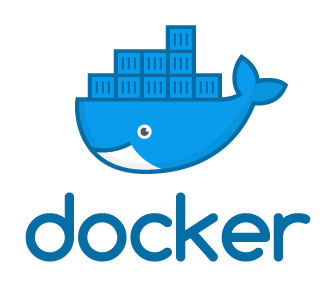
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  | **iModBot** | |  |  |
|  |  |  |  |  |  |

**Docker**

**Guia para iniciantes**



**Elaborado por:**

Nelson Henriques – 2190514

**Orientado por:**

Carlos Neves

Luís Conde

**Índice**

[1. Introdução 1](#_Toc75971718)

[1.1. Objetivos 1](#_Toc75971719)

[2. Terminologia Docker 2](#_Toc75971720)

[2.1. Images 2](#_Toc75971721)

[2.2. Containers 2](#_Toc75971722)

[2.3. Volumes 2](#_Toc75971723)

[2.4. Networks 2](#_Toc75971724)

[3. Instalar o software Docker e Docker-Compose 3](#_Toc75971725)

[3.1. Docker 3](#_Toc75971726)

[3.2. Docker-Compose 4](#_Toc75971727)

[4. Comandos básicos Docker 5](#_Toc75971728)

[5. Comandos básicos Docker-Compose 11](#_Toc75971729)

# Introdução

O Docker é uma ferramenta utilizada para tornar mais fácil criar, implementar e executar aplicações utilizando *containers*. Os *containers* permitem que o desenvolvedor empacote a aplicação bem como todas as partes de que precisa, como bibliotecas e outras dependências, e implemente-o como um *package*. Assim o desenvolvedor pode ter certeza de que a aplicação será executada em qualquer outra máquina Linux, independentemente de quaisquer configurações personalizadas que a máquina possa ter, que podem ser diferentes da máquina usada para escrever e testar o código.

De certa forma, o Docker é idêntico a uma máquina virtual. Mas, ao contrário de uma máquina virtual, em vez de criar um sistema operativo virtual completo, o Docker permite que as aplicações usem o mesmo *kernel* Linux do sistema em que estão a ser executadas, requer apenas que as aplicações sejam enviadas com coisas que ainda não estão a ser executadas no host. Isto possibilita um aumento significativo no desempenho e reduz o tamanho da aplicação.

Compose é uma ferramenta utilizada para definir e executar aplicações Docker compostas por vários containers. O Compose utiliza arquivos YAML para configurar os serviços da aplicação. Então, com um único comando cria e inicia todos os serviços da configuração.

## Objetivos

Pretende-se com este tutorial fornecer os conhecimentos básicos para a utilização do *software Docker e Docker-Compose.*

Ao longo do tutorial vai-se abordar os seguintes temas:

• Terminologia Docker;

• Instalar o software Docker e Docker-Compose;

• Comandos básicos Docker.

• Comandos básicos Docker-Compose.

(Este tutorial é destinado ao Docker e Docker-Compose em um sistema Linux 20.04)

# Terminologia Docker

## Images

As *images* do docker são a "receita" para um *container*. As *images* contém todas as definições de como inicializar o ambiente Linux. Normalmente, uma imagem Docker existe para cumprir especificamente uma tarefa. Por exemplo, uma imagem definiria o seu servidor web e outra imagem definiria a database subjacente.

## Containers

Os *containers* são instanciações de imagens. Eles são uma forma de imagem. Compare-a com a programação orientada a objetos, então sua classe seria uma imagem e uma instância da classe, um *container*. Outra comparação seria com contentores reais num cargueiro. Estes tem uma aparência idêntica por fora, mas por dentro são/podem ser completamente diferentes. Alguns factos interessantes sobre *containers*:

* Os *containers* não são persistentes. Assim que não estiverem a ser utilizados eles são desligados, ou até mesmo eliminados.
* Um *container* só vive enquanto um processo é executado dentro dele.
* Os *containers* podem incluir dados, mas estes também não são persistentes. Se quiser torná-los persistentes, pode-se utilizar *volumes*.
* Para comunicar entre *containers*, usa-se os mesmos protocolos que se usariam para comunicar entre computadores (por exemplo, TCP / IP).

## Volumes

Os *volumes* podem ser usados como camada de dados subjacente. Estes podem ser utilizados por vários *containers*. Todos os dados guardados num volume podem ser acedidos por qualquer *container* conectado a ele.

## Networks

O Docker vem com as suas próprias capacidades de *networking*. O nome de um *container* é seu nome de host. A maneira mais fácil de experimentar isso é ter dois *containers* em execução e executar o comando para verificar o ping em um *container* de dentro de outro *container*. O Docker Compose irá gerar sua própria sub-rede.

# Instalar o software Docker e Docker-Compose

## Docker

Iremos instalar utilizando o repositório oficial do Docker.

1. Atualize as *packages* do apt e adicione novas para o mesmo poder utilizar o repositório por HTTPS:

sudo apt-get update

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg \

lsb-release

1. Adicione a chave GPG do Docker.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

1. Usa o seguinte comando para criar o repositório local.

echo \

"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

1. Instale o Docker.

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

1. Verifique se o Docker foi instalado corretamente..

docker --version

deverá obter algo semelhante á Figura 1



Figura - Versão do Docker.

## Docker-Compose

1. Descarregue a versão mais recente e estável do Docker-Compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

1. Dê permissão de execução ao binário descarregado.

sudo chmod +x /usr/local/bin/docker-compose

1. Verifique se o Docker-Compose foi instalado corretamente.

docker-compose --version

Deverá obter algo semelhante á Figura 1



Figura - Versão do Docker-Compose.

# Comandos básicos Docker

1. Iremos começar correndo o clássico “Hello World”.

docker run hello-world

Deverás obter a seguinte resposta (Figura 3).

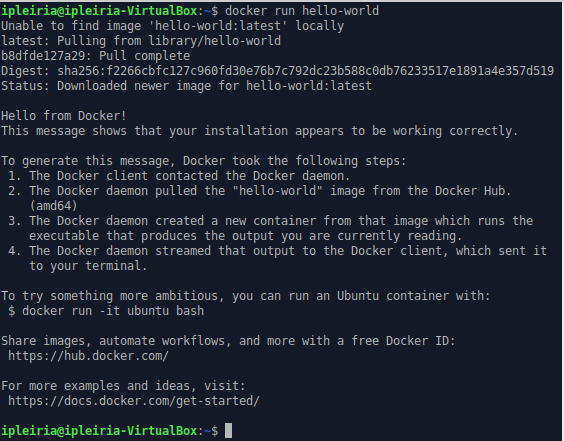


Figura - Output de "Hello World"

A *output* já descreve muito bem o que acontece ao executar este comando.

Iniciou-se um *container*, cujo a imagem foi descarregada do Docker hub, de seguida executou-o e enviou a resposta para o terminal.

1. Vê todos os containers que estão em execução ou foram executados.

docker ps -a

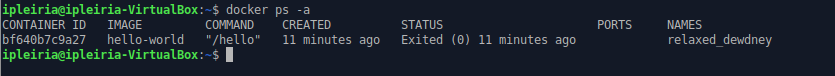


Figura - containers que estão em execução ou foram executados

Os “Names” são gerados aleatoriamente, podemos referir-nos aos *containers* pelos “Names” ou pelo “container id” (Figura 4)

1. Execute novamente o “Hello World” mas desta vez iremos atribuir um nome, assim não dependemos de um nome gerado aleatoriamente.

docker run --name My-HW hello-world

1. Execute novamente o comando do passo nº2 e verifique que o nome é igual ao que foi atribuído no comando anterior (Figura 5).

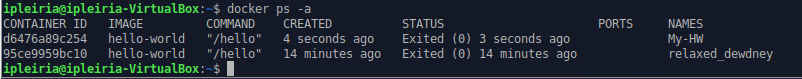


Figura - container com o nome customizado

1. Agora vamos executar um container que contem o ubunto (como este container não está localmente o Docker ira descarrega-lo do Docker Hub)e ao mesmo tempo iremos atribuir um nome customizado

docker run -it --name my-linux-container ubuntu bash

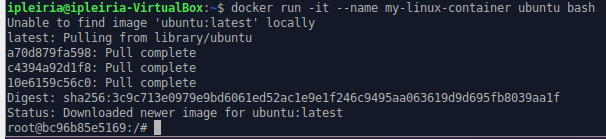


Figura - ubunto container

Agora está dentro do container ubunto (Figura 6) onde pode executar qualquer comando Linux isto é bastante útil quando estamos num sistema windows ou macOS, para sair basta escrever “exit”.

1. Mais uma vez ao executar o comando do passo nº2 para ver o estado de todos os containers executados ou em execução ate ao momento (Figura 7).

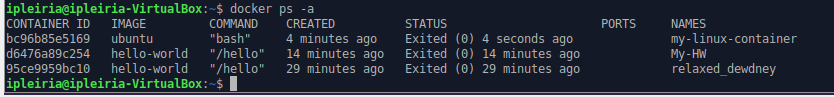


Figura - Containers executados ou em execução

1. Verifique quais as imagens que tem guardados localmente.

docker images

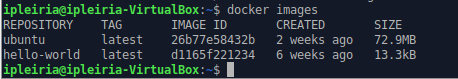


Figura - docker images

Este comando pode ser bastante útil pois permite saber todas as informações principais das “images” (Figura 8), como estão guardados localmente a sua execução da próxima vez será instantânea.

1. Uma vez iniciado um container com um nome pré-definido pelo utilizador o mesmo já não pode ser iniciado com o mesmo nome, por isso podemos executar o seguinte comando de modo a limpar o “histórico” de containers executados (Figura 9).

docker rm $(docker ps -a -f status=exited -q)

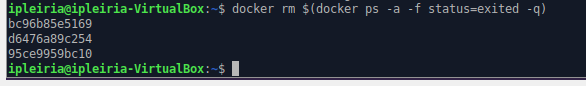


Figura - output

1. Verifique que a lista está limpa correndo o comando do passo nº2.
2. Inicie um container e ao mesmo tempo conecte-o a uma pasta local, para isso crie uma pasta no seu sistema operativo local e guarde o seu caminho, substituindo-o no comando abaixo.

docker run -it --name my-linux-container --rm -v [diretoria local] ubuntu bash

* -it (interactive container) – entra no mesmo assim que inicia.
* --rm – assim que termina elimina-o da lista.
* -v – comando para conectar a diretoria local sendo que a pasta a partilhar tem que anteceder “:”. Ex:

docker run -it --name my-linux-container --rm -v /home/ipleiria/Desktop:/meus\_dados ubuntu bash

1. Verifique que a pasta foi partilhada executando o comando ls dentro do *container.*

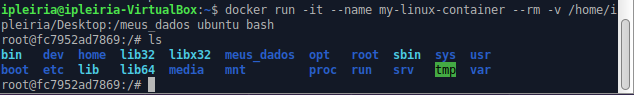


Figura - pasta partilhada

Como pode observar a pasta foi partilhada corretamente (Figura 10).

1. Execute um comando no *container* ubuntu, para isso, abra um novo terminal (não feche o atual), e utilize o seguinte código.

docker exec -it my-linux-container echo "Hello World"

Com este código irá imprimir um Hello World (Figura 11), sendo que depois do nome do *container* vem o comando a executar no mesmo.



Figura - comando executado

1. Crie um Docker image, para isso crie uma nova pasta e atribua-lhe o nome “My-Docker-Image”, dentro dessa pasta crie um ficheiro com o nome “Dockerfile” (certifique-se que não tem um tipo de ficheiro ex: .txt), abra-o com um editor de texto, neste momento deverá ter algo semelhante á Figura 12.

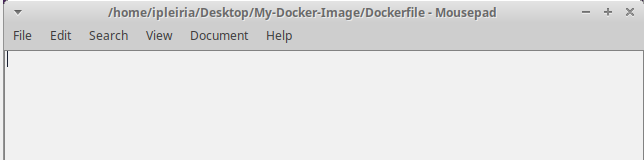


Figura - ficheiro Dockerfile

1. No ficheiro escreva as seguintes linhas de código (Figura 13), guarda e fecha o ficheiro.

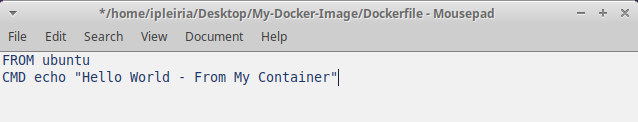


Figura - código a escrever

Existe uma grande variedade de “Keywords” para criar imagens Docker estas podem ser encontradas aqui: <https://docs.docker.com/engine/reference/builder/>.

1. No terminal vá á diretoria criada (esta diretoria varia dependendo de onde criou a pasta).

cd /home/ipleiria/Desktop/My-Docker-Image

1. Construa a imagem.

docker build -t my-ubuntu-image .

* -t – atribui um nome á imagem.

Deverá obter algo semelhante á Figura 14.

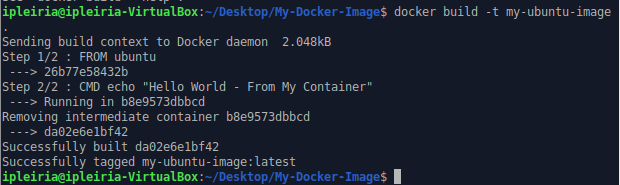


Figura - imagem criada

1. Verifique que a imagem foi criada (Figura 15).

docker images

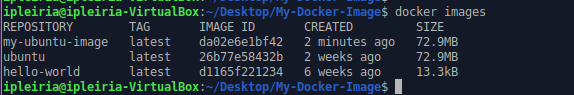


Figura - lista de imagens locais

1. Corra a imagem criada.

docker run my-ubuntu-image



Figura - output do meu container

Como pode observar ao executar o container ele escreveu no terminal o que foi escrito no Dockerfile (Figura 15).

Ex:

Pretende iniciar um ubuntu Docker já com o python3 instalado, para isso no ficheiro criado anteriormente basta trocar o “CMD echo "Hello World - From My Container"” por “RUN apt-get upgrade && apt-get update && apt-get install -y python3” e contruir novamente a imagem, assim terá uma imagem ubunto com o python3 já instalado.

Pode também guardar o estado do seu container em uma nova imagem através do seguinte comando:

docker commit $CONTAINER\_ID image name

# Comandos básicos Docker-Compose

1. Descarregue o conteúdo do repositório, este será o exemplo utilizado para o tutorial.

(<https://github.com/jakewright/tutorials/tree/master/docker/02-docker-compose>)

Nessa pasta irás encontrar 4 ficheiros, “product”, “Website”, “readme.md” e “Docker-compose.yml”, o ficheiro “Docker-compose.yml” é o ficheiro onde estão todas as definições da aplicação, que neste caso é um web site muito simples.

Este exemplo está pronto a ser executado mas antes disso iremos entender o que irá o ficheiro “Docker-compose.yml” fazer.

1. Abra o ficheiro “Docker-compose.yml” com o seu editor preferido (Figura 17).

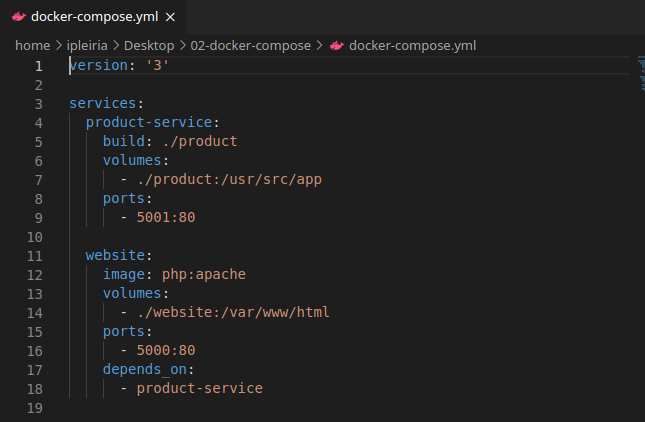


Figura - Docker-Compose.yml

Version: – Versão em que está a ser escrita o ficheiro (opcional a partir da versão 1.27).

services: – serviços a iniciar.

Product-service e website – nome dos serviços.

O restante são as dependências e definições dos serviços sendo que tem que ser adaptadas ao que pretende realizar, para mais informação consulte este link:

<https://docs.docker.com/compose/compose-file/>

1. Abra o terminal e vá até á diretoria descarregada.

cd /home/ipleiria/Desktop/02-docker-compose/

1. Agora irá “construir” a aplicação com base no ficheiro criado (Figura 18).

docker-compose build



Figura - construção terminada

1. Agora irá iniciar a aplicação criada (Figura 19).

docker-compose up

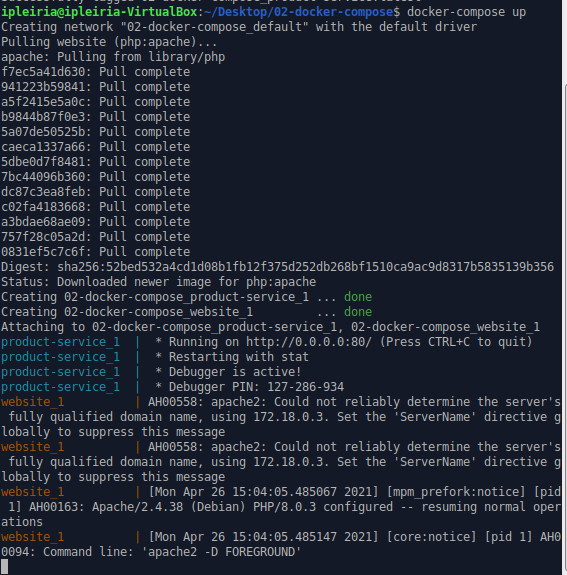


Figura - aplicação iniciada

1. Aceda a <localhost:5000> no seu browser, deverá aparecer algo semelhante á Figura 20.

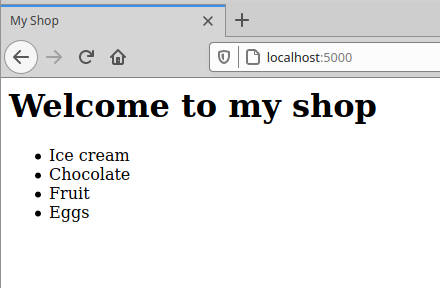


Figura - website exemplo

Caso dê um erro semelhante ao da Figura 21, corra o seguinte comando:

sudo chmod 755 website

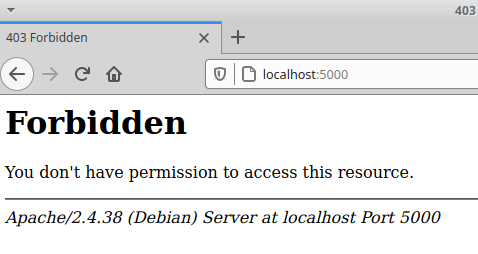


Figura - possivel erro