

WEB ACADEMY

UFAM

MÓDULO: Contêineres com Docker

QUEM SOU EU?

Linnik Maciel

Bacharel em Engenharia de Software pelo IComp/UFAM (2022);
Pós graduado em Banco de Dados pela Fametro (2023);
Trabalha na Bemol Digital desde 2020 (como engenheiro de software desde 2022)



CONTATOS

E-mail

linnik.souza123@gmail.com

LinkedIn

<https://br.linkedin.com/in/jocelinnik>

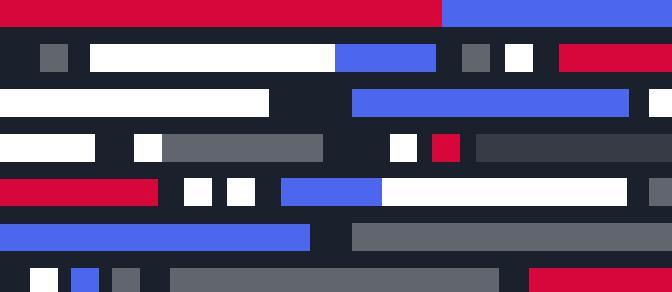
Github

<https://github.com/jocelinnik>



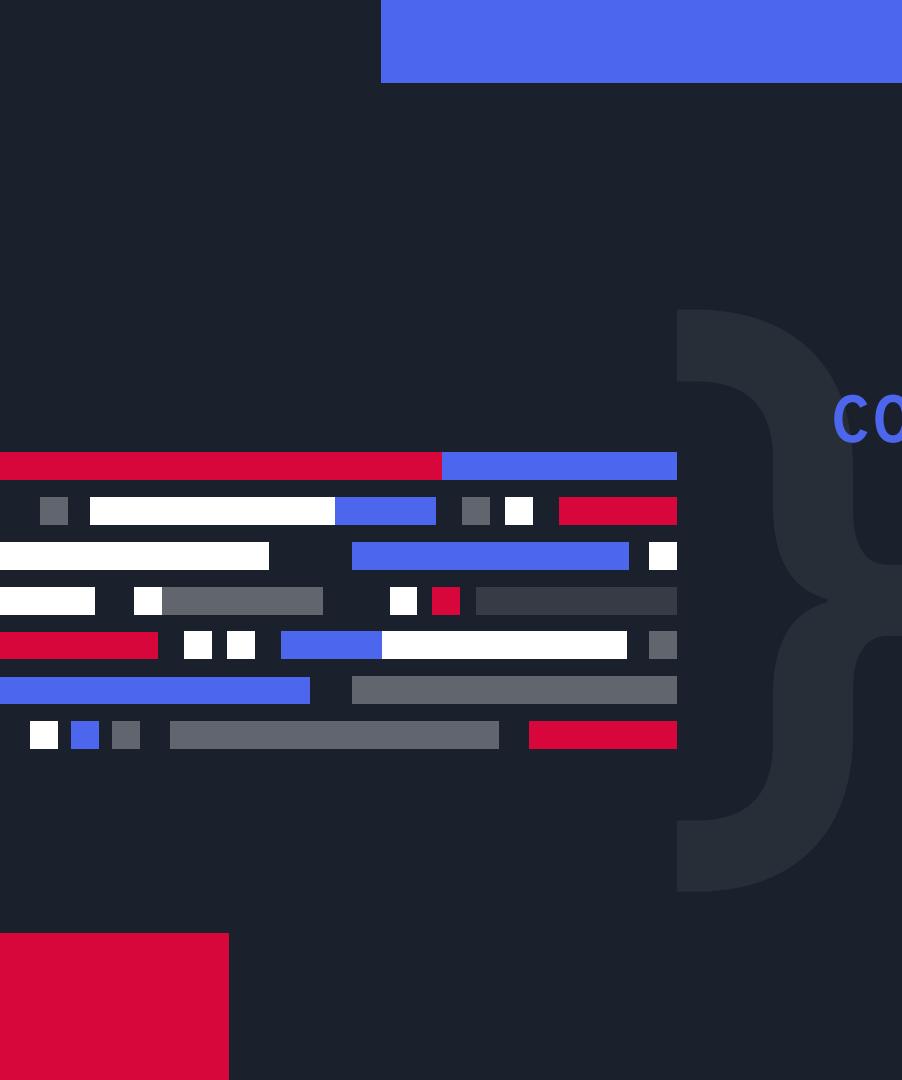
O INÍCIO





COMO FAZIAM OS POVOS DA MESOPOTÂMIA?

Há alguns anos, as empresas implantavam seus sistemas de informação em servidores físicos, presentes na própria sede da empresa. Esse tipo de implantação (on-premise) exigia das empresas profissionais especializados em infraestrutura para gerenciar esses servidores.



COMO FAZIAM OS POVOS DA MESOPOTÂMIA?

Em alguns casos, houveram necessidades de implantar diferentes sistemas dentro do mesmo servidor. Isso não é exatamente um problema quando temos um servidor com bastante capacidade de processamento (CPU e memória RAM), mas há outros problemas que podem surgir.

APLICAÇÕES NO SERVIDOR



APP 1

Sistema web de
e-commerce
Executa na porta 8000



APP 2

Sistema web de controle de
estoque
Executa na porta 8000



CHEGAMOS NO RENASCIMENTO

Para resolver esses conflitos que podem ocorrer entre aplicações dentro do servidor, a galera começou a fazer uso de **virtualizações**. Virtualização nada mais é do que executar um outro sistema operacional em cima de um outro instalado fisicamente em uma máquina.



APLICAÇÕES NO SERVIDOR



APP 1

DEPENDÊNCIAS 1

SO VIRTUAL 1



APP 2

DEPENDÊNCIAS 2

SO VIRTUAL 2

HYPERVERISOR

SO

SERVIDOR



PROBLEMA RESOLVIDO?

Aparentemente nosso problema de execução de múltiplas aplicações foi resolvido, mas a virtualização tem um custo.

Mas qual é esse custo?

Uso de recursos

Estamos executando múltiplos SO's em uma mesma máquina, e isso pode ser custoso para o servidor.

Complexidade

A virtualização pode se tornar complexa à medida que a quantidade de aplicações cresce.



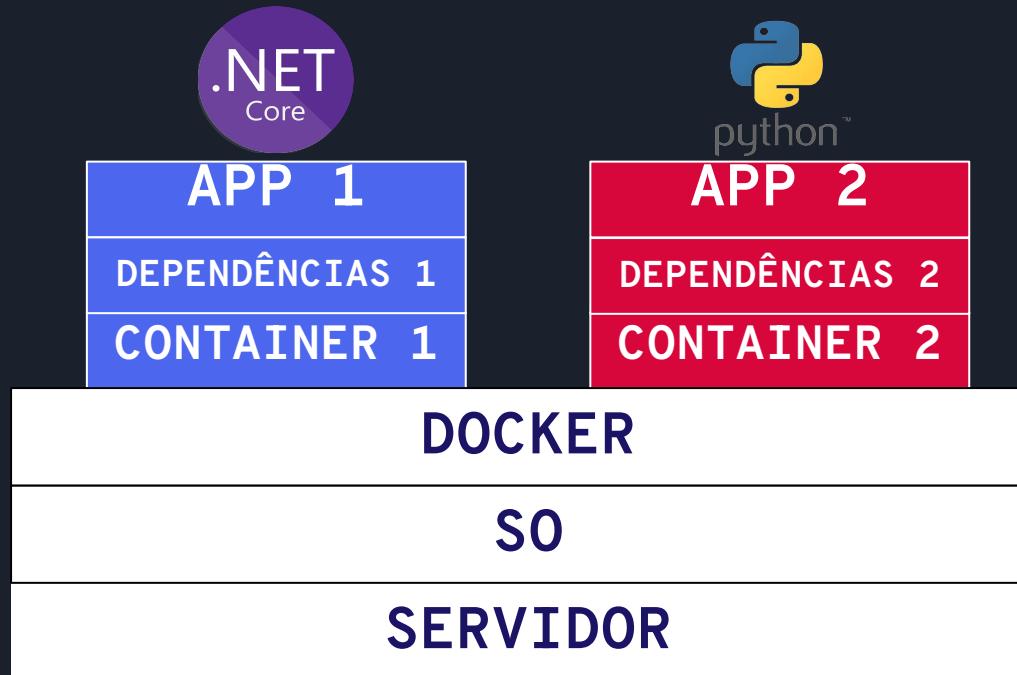
CONTÊINERIZAÇÃO

Com a conteinerização de aplicações, temos uma alternativa **leve** de executar diferentes aplicações de maneira **isolada**, proporcionando a flexibilidade de executar nossas aplicações em diferentes servidores sem modificar configurações.

A ferramenta que mais se destaca nesse cenário é o **Docker**.



APLICAÇÕES NO SERVIDOR



Isolado

O Docker faz uso de **namespaces** para isolar os recursos usados para cada contêiner.

Leve

Os contêineres são tratados como processos no servidor, consumindo menos recursos do que SO's virtualizados.

PID

Isolamento de processos

NET

Controle de interface de rede

IPC

Controle de recursos de
InterProcess Communication

MNT

Gestão de pontos de montagem

UTS

Isolar recursos de kernel
(UNIX Timesharing System)

COMEÇANDO COM O DOCKER





HORA DE EXERCÍCIO

Vamos botar um contêiner para executar que irá exibir a mensagem “Hello World” no terminal. Execute o seguinte comando:

```
$ docker run hello-world
```

SAÍDA ESPERADA

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:c2e23624975516c7e27b1b25be3682a8c6c4c0cea011b791ce98aa423b5040a0
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://hub.docker.com/
```

For more examples and ideas, visit:

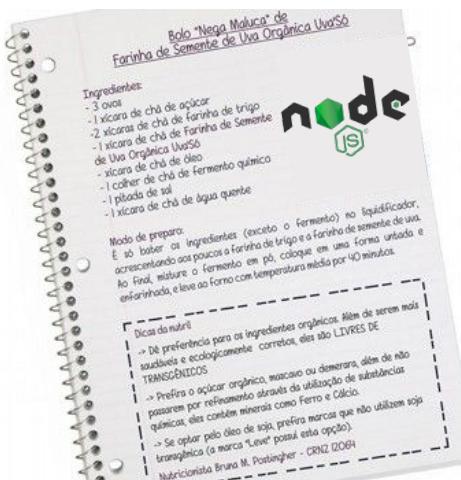
```
https://docs.docker.com/get-started/
```

O QUE ACONTECEU AQUI?

Pedimos para o Docker executar um contêiner baseado na imagem **hello-world**, que imprimiu a mensagem do slide anterior.

Mas o que é uma imagem?

IMAGENS



Imagens no Docker são um conjunto de instruções passadas para o Docker criar um contêiner, como uma receita de bolo.

A partir dessas imagens, podemos criar vários contêineres que serão únicos entre si (por conta dos **namespaces**).

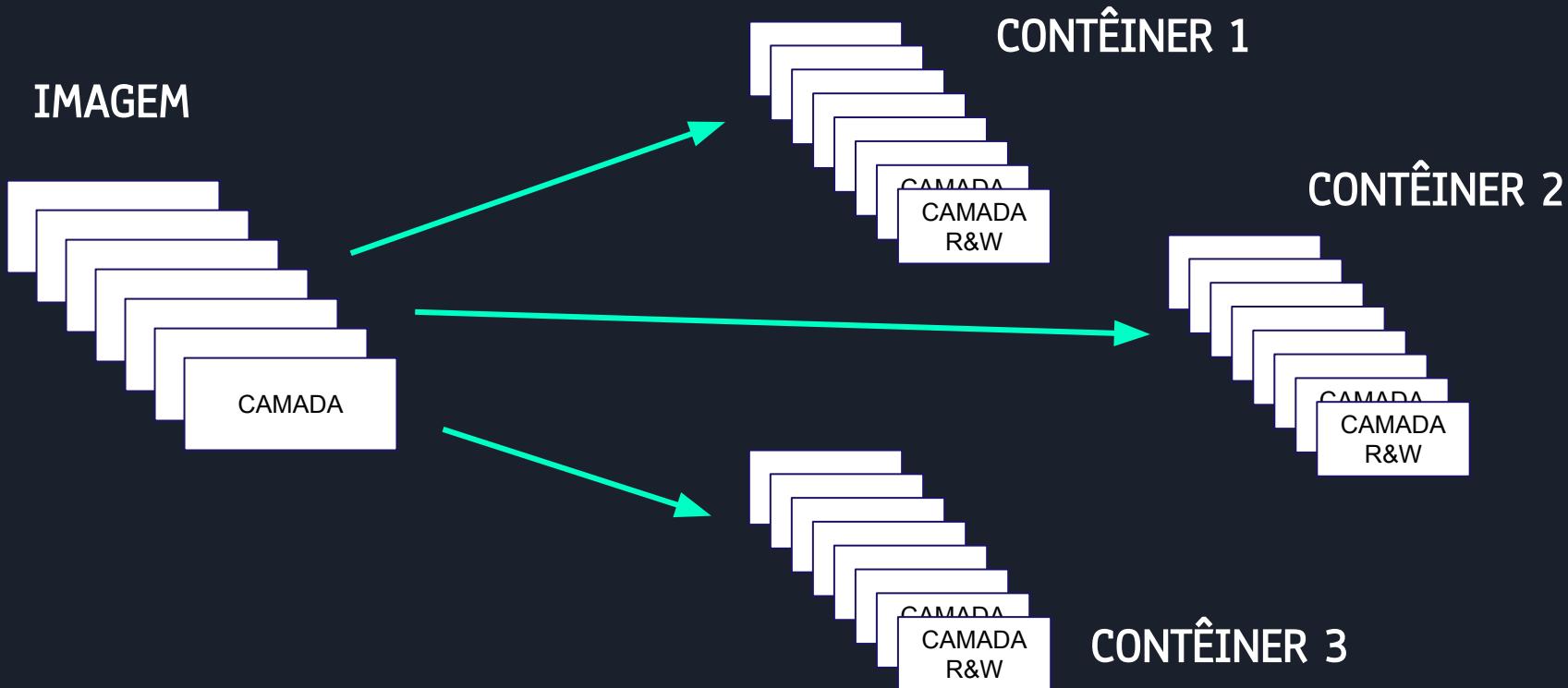


FORMAÇÃO DAS IMAGENS

Tal qual os ogros e as cebolas, as imagens são compostas por **camadas**, onde cada camada descreve um passo para criação de um contêiner.

Sendo os contêineres criados a partir das imagens, podemos dizer que os contêineres são compostos pelas mesmas camadas da imagem, com uma camada a mais de **leitura e escrita**.

CRIAÇÃO DE CONTÊINERES

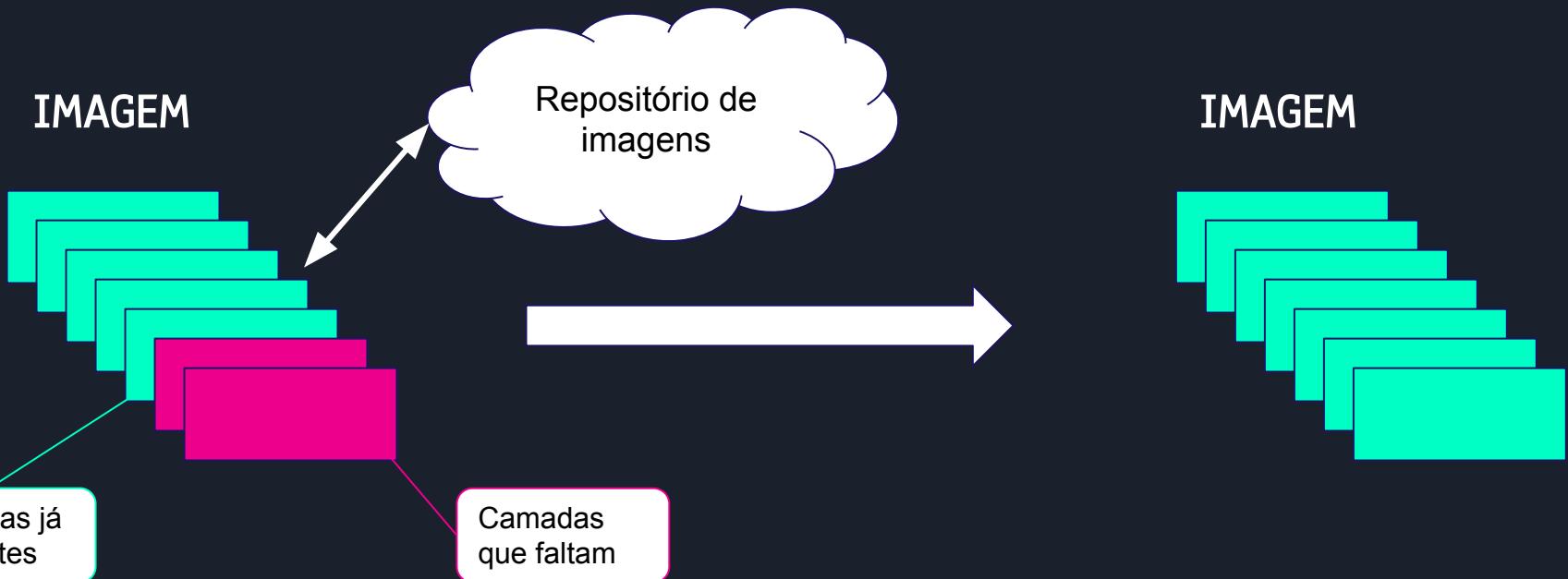




FORMAÇÃO DAS IMAGENS

O Docker também é inteligente o suficiente para identificar as camadas de uma imagem que já existem dentro do servidor. Assim, ele só vai no repositório baixar as camadas que faltam para formar a imagem completa.

BAIXANDO IMAGENS



DOCKER HUB

É o repositório público de imagens do Docker. É neste local que o Docker baixa as imagens para executar nossos contêineres.

<https://hub.docker.com/>

Docker Hub





HORA DE EXERCÍCIO

Vamos explorar mais os comandos do Docker e a criação de contêineres. Execute os seguintes comandos:

```
$ docker pull ubuntu  
$ docker run ubuntu  
$ docker ps
```

O que o Docker mostrou no console?

COMANDOS – CONTÊINERES



docker run

Cria e executa um novo
contêiner a partir de uma
imagem

`docker ps`

Exibe todos os contêineres
executando

```
docker ps -a
```

Exibe todos os contêineres
(parados e executando)

```
docker ps -q
```

Exibe o ID de todos os contêineres

`docker stop`

Para a execução de um
contêiner

`docker start`

Inicia a execução de um contêiner parado

docker stats

Exibe as estatísticas de um contêiner

`docker rm`

Apaga um contêiner

COMANDOS – IMAGENS



docker pull

Baixa uma imagem do
repositório de imagens

docker images

Exibe todas as imagens
baixadas do repositório

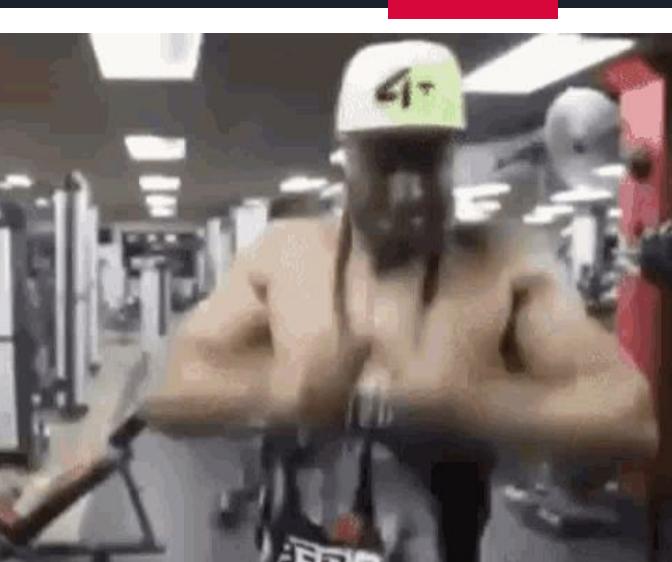
docker rmi

Apaga uma imagem

DANDO NOMES AOS CONTÊINERES

Por padrão, o Docker dá um nome aleatório para os contêineres que são criados, mas podemos dar nomes pré definidos durante a criação dos contêineres:

```
$ docker run --name ubuntu-c  
ubuntu sleep 1d
```

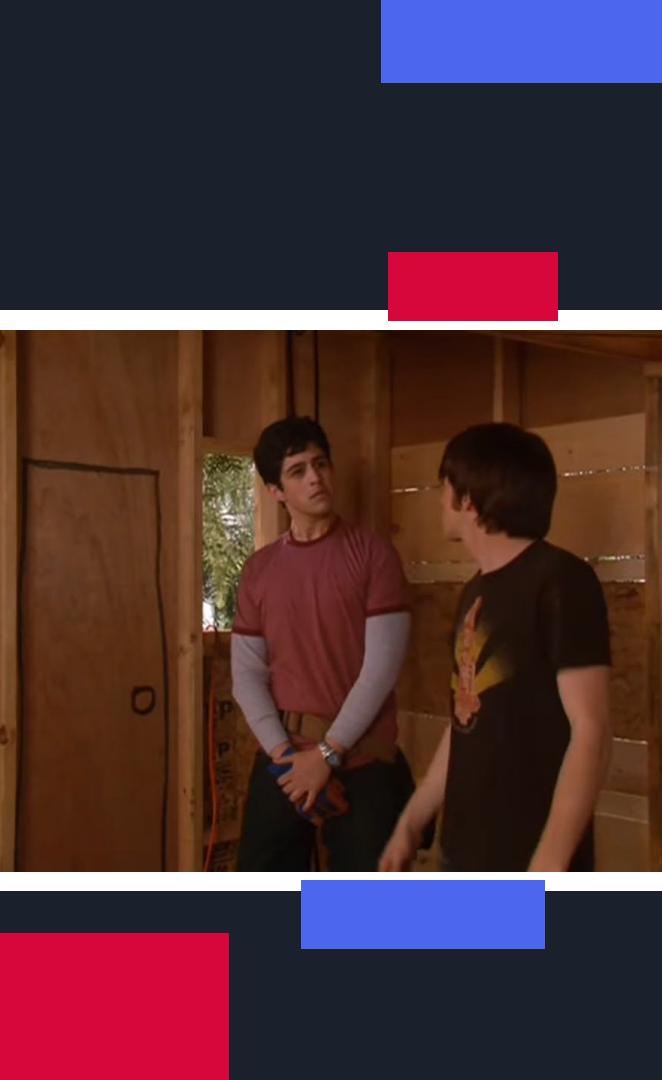


HORA DE EXERCÍCIO

Vamos executar um servidor HTTP dentro de um contêiner usando o **Nginx**:

```
$ docker pull nginx  
$ docker run --name srvhttp nginx
```

Acesse o endereço <http://localhost> e veja a página inicial do Nginx. Ou não?

A photograph of two young boys standing in a room with wooden walls. One boy is wearing a red t-shirt and the other is wearing a black t-shirt with a graphic on it. They appear to be looking at something off-camera.

MAPEAMENTO DE PORTAS

Os contêineres expõem portas de rede mas, por padrão, essas portas só são acessíveis internamente. Para acessá-las fora do contêiner, precisamos mapear as portas expostas:

```
$ docker run -d --name srvhttp -p  
8000:80 nginx
```

PÁGINA INICIAL DO NGINX

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



ATENÇÃO!

Ao baixar imagens do Docker em uma quantidade muito grande, pode ocorrer de você ser impedido de baixar novas imagens. Isso se dá pelo fato de termos um limite diário de vezes em que podemos baixar imagens, que pode ser atingido rapidamente quando se tem muitas máquinas realizando esta operação na mesma rede (laboratórios, escritórios etc).

ERRO OCORRIDO

```
▼ 248 $ echo 'DOCKER_OPTS="$DOCKER_OPTS --registry-mirror=https://mirror.gcr.io"'  
249 DOCKER_OPTS="$DOCKER_OPTS --registry-mirror=https://mirror.gcr.io"  
▼ 250 $ sudo service docker restart  
▼ 251 $ docker pull ${PACKAGE_BUILD_IMAGE}  
252 Using default tag: latest  
253 latest: Pulling from amreo/rpmbuild-centos7  
254  
255 Status: Downloaded newer image for amreo/rpmbuild-centos7:latest  
256 $ docker pull mongo:latest  
257 latest: Pulling from library/mongo  
258  
259 library/mongo  
260 toomanyrequests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit  
261 The command "docker pull mongo:latest" failed and exited with 1 during .  
262  
263 Your build has been stopped.
```

COMO RESOLVER?

O próprio texto do erro sugere que seja criado um perfil no Docker Hub, e que suas credenciais sejam autenticadas na interface CLI do Docker.

É possível criar uma conta gratuita no Docker Hub.

CRIAÇÃO DA CONTA

The image shows a screenshot of a web browser displaying the Docker Hub sign-up page. The URL in the address bar is <https://hub.docker.com>. The page has a blue header with the text "CRIAÇÃO DA CONTA" in large white letters. Below the header, the main content area features a large blue background with white text: "Build and Ship any Application Anywhere" and "Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications." On the right side, there is a "Get Started Today for Free" section with fields for "Username" (containing "jocelinink2"), "Email" (containing "linnik.souza123@gmail.com"), and "Password". There are also two checkboxes: one for "Send me occasional product updates and announcements" (unchecked) and one for "I agree to the [Subscription Service Agreement](#), [Privacy Policy](#), and [Data Processing Terms](#)" (checked). A reCAPTCHA field is present with the text "Não sou um robô" and a checkbox. At the bottom is a large blue "Sign Up" button.

Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

Get Started Today for Free

Already have an account? [Sign In](#)

Username
jocelinink2

Email
linnik.souza123@gmail.com

Password
••••••••

Send me occasional product updates and announcements.

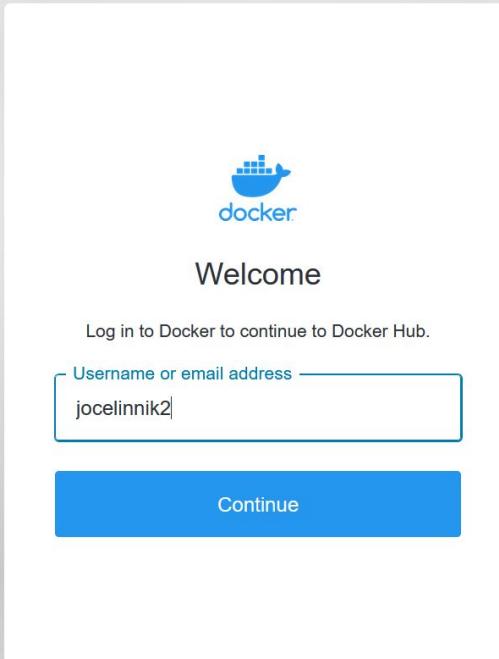
I agree to the [Subscription Service Agreement](#), [Privacy Policy](#), and [Data Processing Terms](#).

Não sou um robô 
reCAPTCHA
Privacidade • Termos

Sign Up

LOGIN

https://login.docker.com/u/login/identifier?state=hKFo2SBXZWR0b2JXLViREhmLVhCemw1LW11



The image shows a web browser window displaying the Docker login page. The URL in the address bar is https://login.docker.com/u/login/identifier?state=hKFo2SBXZWR0b2JXLViREhmLVhCemw1LW11. The page itself has a light gray background with a white central login form. At the top of the form is the Docker logo, which consists of a blue stylized ship icon above the word "docker". Below the logo, the word "Welcome" is centered in a large, dark font. Underneath "Welcome", a smaller text says "Log in to Docker to continue to Docker Hub.". A text input field is present, labeled "Username or email address" with a placeholder "jocelinnik2" entered. A large blue "Continue" button is at the bottom of the form.

Username or email address

jocelinnik2

Continue

LOGIN

A screenshot of a web browser displaying the Docker login page. The URL in the address bar is `https://login.docker.com/u/login/password?state=hKFo2SBXZWR0b2JXLViREhmLVhCemw1LW1`. The page features a large Docker logo at the top, followed by the text "Enter Your Password". Below this, instructions read "Enter your password for Docker to continue to Docker Hub". A user input field contains the text "jocelinnik2" with an "Edit" link to its right. Below it is a password input field containing masked text "••••••••••••" with an "Eye" icon to its right. At the bottom of the form are links for "Forgot password?" and a large blue "Continue" button.

Enter Your Password

Enter your password for Docker to continue to
Docker Hub

jocelinnik2 [Edit](#)

>Password [Eye](#)

[Forgot password?](#)

[Continue](#)

CRIAÇÃO DA CONTA

Será sugerido contratar um plano pago para o perfil criado, mas podemos continuar na versão gratuita.

Ainda teremos um limite de utilização das imagens, mas estará diretamente associado ao nosso perfil.

ESCOLHA DO PLANO

Personal	Pro	Team	Business
<p>Includes the Docker essentials and is ideal for individual developers, education, open source communities, and small businesses.</p> <p>\$0</p> <ul style="list-style-type: none">• Docker Desktop ⓘ• Unlimited public repositories• Docker Engine + Kubernetes ⓘ• Limited image pulls per day• Unlimited scoped tokens ⓘ	<p>Extends the Docker capabilities and includes pro tools for individual developers who want to accelerate their productivity.</p> <p>\$5 /month</p> <p>← Everything in Personal, plus:</p> <ul style="list-style-type: none">• Docker Desktop ⓘ• Unlimited private repositories• 5,000 image pulls per day• 5 concurrent builds ⓘ• 300 vulnerability scans	<p>Ideal for teams and includes capabilities for enhanced collaboration, productivity and security.</p> <p>\$9 /user/month max 100 users minimum 5 seats</p> <p>← Everything in Pro, plus:</p> <ul style="list-style-type: none">• Docker Desktop ⓘ• Unlimited teams• 15 concurrent builds ⓘ• Unlimited vulnerability scans• Add users in bulk• Audit logs ⓘ	<p>Ideal for medium and large businesses who need centralized management and advanced security capabilities.</p> <p>\$24 /user/month only available with an annual subscription</p> <p>← Everything in Team, plus:</p> <ul style="list-style-type: none">• Hardened Docker Desktop• Enhanced Container Isolation• Settings management• Centralized management• Image Access Management• Registry Access Management• Single Sign-On (SSO)• SCIM user provisioning• VDI support• Purchase via invoice• Volume Pricing Available• Company management

CRIAÇÃO DA CONTA

Vai ser enviado para o e-mail cadastrado um link de verificação da conta criada.

Após a verificação, temos nossa conta no Docker Hub criada e habilitada.

VERIFICAÇÃO DA CONTA

A screenshot of a web browser displaying the Docker Hub verification email page. The URL in the address bar is <https://hub.docker.com/verify-email>. The page content includes a message to check the inbox for verification instructions, a purple banner about Kubernetes development, a Docker Hub navigation bar with 'Upgrade' and user profile, and a large envelope icon with the text 'Please verify your email address' and explanatory text about the verification process.

Please check your inbox to verify the email associated with this account. You won't be able to create a repository or configure your Docker Hub without verifying your email address.

Want faster and simpler Kubernetes development? Test out [Telepresence for Docker](#) today.

dockerhub Search Docker Hub [Explore](#) [Repositories](#) [Organizations](#) [Help](#) [Upgrade](#)  jocelinnik2



Please verify your email address

Great! You're almost there. Before you can create a repository or configure Docker Hub, you'll need to verify your email address.

We've sent a verification email to
linnik.souza123@gmail.com.

CONTA VERIFICADA

The screenshot shows the Docker Hub homepage (<https://hub.docker.com>) with a purple header banner that reads "Want faster and simpler Kubernetes development? Test out [Telepresence for Docker](#) today." Below the banner, the Docker Hub navigation bar includes links for "Explore", "Repositories", "Organizations", "Help", and an "Upgrade" button. A user profile for "jocelinnik2" is visible on the right. The main content area features a large blue box with the text "Welcome to Docker" and "Download the desktop application". It includes a "Download for Windows" button and text indicating availability for Mac and Linux. At the bottom, there are three cards: "Create a Repository", "Docker Hub Basics", and "Language-Specific Guides".

Want faster and simpler Kubernetes development? Test out [Telepresence for Docker](#) today.

dockerhub Search Docker Hub

Explore Repositories Organizations Help ▾ Upgrade jocelinnik2

Welcome to Docker

Download the desktop application

[Download for Windows](#)

Also available for [Mac](#) and [Linux](#)

Create a Repository
Push container images to a repository on Docker Hub.

Docker Hub Basics
Watch the guide on how to create and push your first image into a Docker Hub repository.

Language-Specific Guides
Learn how to containerize language-specific applications using Docker.

AUTENTICAÇÃO DA CONTA

Agora devemos autenticar a interface CLI do Docker com nossa conta criada, para isso, utilizamos o comando **docker login**. Confirme sua senha e a autenticação será feita com sucesso.

```
$ docker login  
--username=<USERNAME>
```

`docker login`

Realiza o login da interface
CLI com o Docker Hub

AUTENTICAÇÃO

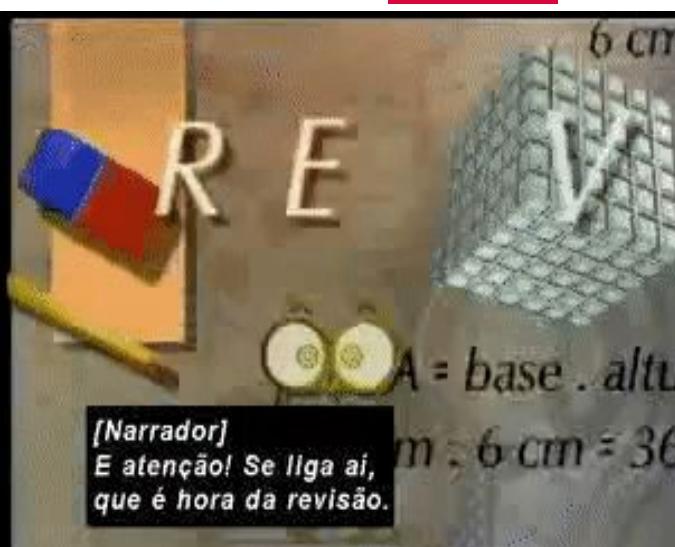
```
C:\workspace\docker\web-academy-docker>docker login --username=jocelinnik2  
Password:  
Login Succeeded
```

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at <https://docs.docker.com>

```
C:\workspace\docker\web-academy-docker>
```

RECAPITULANDO

- Virtualização X contêinerização
- Namespaces
- Contêiner
- Imagem
- Comandos
- Nomeação de contêineres
- Mapeamento de portas





QUESTIONÁRIO

Responda o seguinte questionário no Colabweb:

<https://colabweb.ufam.edu.br/mod/quiz/view.php?id=52128>

CUSTOMIZANDO UMA IMAGEM



CONSTRUINDO IMAGENS

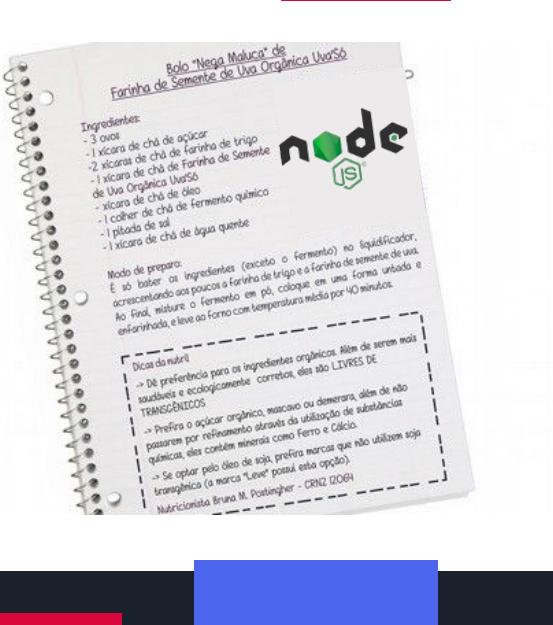
Não teria muita utilidade apenas utilizar imagens prontas sem podermos criar nossas próprias imagens de acordo com a nossa necessidade.

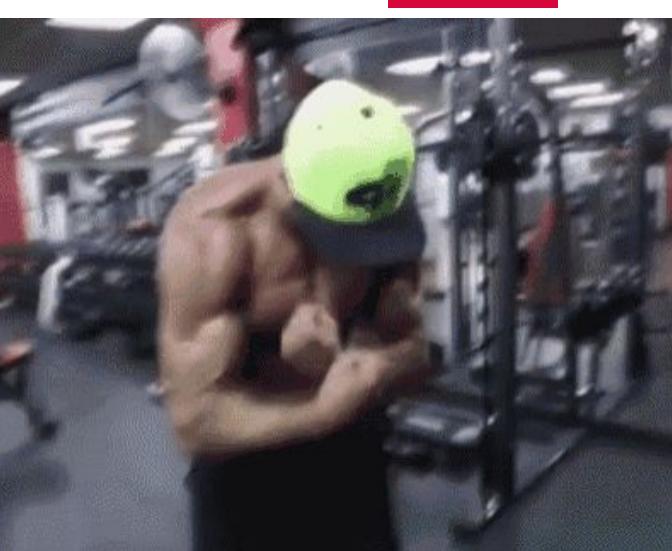
A partir de agora, vamos criar nossas imagens.

DOCKERFILE

Lembra que nossas imagens são como receitas de bolo que descrevem os passos para criar nossos contêineres?

Esses passos estão descritos em um arquivo chamado **Dockerfile**. Nele dizemos o que o Docker tem que fazer para criar os contêineres.





HORA DE EXERCÍCIO

Vamos criar uma aplicação web bem simples com **Node.js** que irá imprimir uma mensagem na página inicial. Siga os passos a seguir para iniciar um projeto Node.js:

```
$ mkdir webacademy-node  
$ cd webacademy-node  
$ npm init -y
```

INDEX.JS

```
const http = require("http");

const servidor = http.createServer((req, res) => {
    res.end("<h1>Use Docker. Gostoso demais.</h1>");
});

servidor.listen(
    4567,
    () => console.log("SERVIDOR RODANDO VIOLENTAMENTE NA PORTA 4567.")
);
```

DOCKERFILE

```
FROM node:18.16-slim
```

```
WORKDIR /home/app
```

```
COPY . .
```

```
EXPOSE 4567
```

```
CMD ["npm", "start"]
```

```
$ docker build -t  
webacademy/webacademy-node .
```

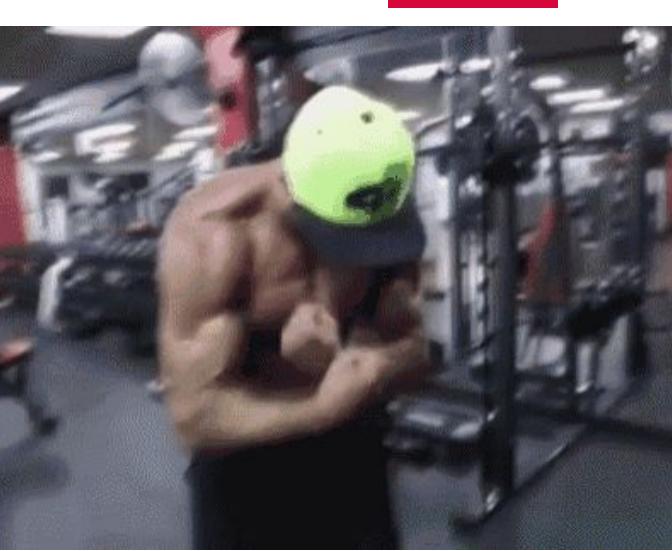
```
$ docker run -d --name  
webacademy-node -p 4567:4567  
webacademy/webacademy-node
```

docker build

Constrói uma imagem a partir
de um arquivo Dockerfile

PÁGINA INICIAL DA APP

Use Docker. Gostoso demais.



HORA DE EXERCÍCIO

Vamos recriar o nosso contêiner que executa um servidor HTTP com o Nginx, mas dessa vez criando uma imagem baseada no ubuntu e com as seguintes características:

- O servidor deve executar dentro do contêiner na porta 7000
- A página inicial deve imprimir a mensagem “Servidor Linux executando Nginx dentro de um contêiner Docker!”

CONFIGURAÇÕES DO NGINX

```
server {  
    listen 7000 default_server;  
    listen [::]:7000 default_server;  
  
    root /usr/share/nginx/html;  
    index index.html index.htm;  
  
    server_name _;  
    location / {  
        try_files $uri $uri/ =404;  
    }  
}
```

DOCKERFILE

```
FROM ubuntu:latest

RUN apt-get update -y
RUN apt-get install -y nginx

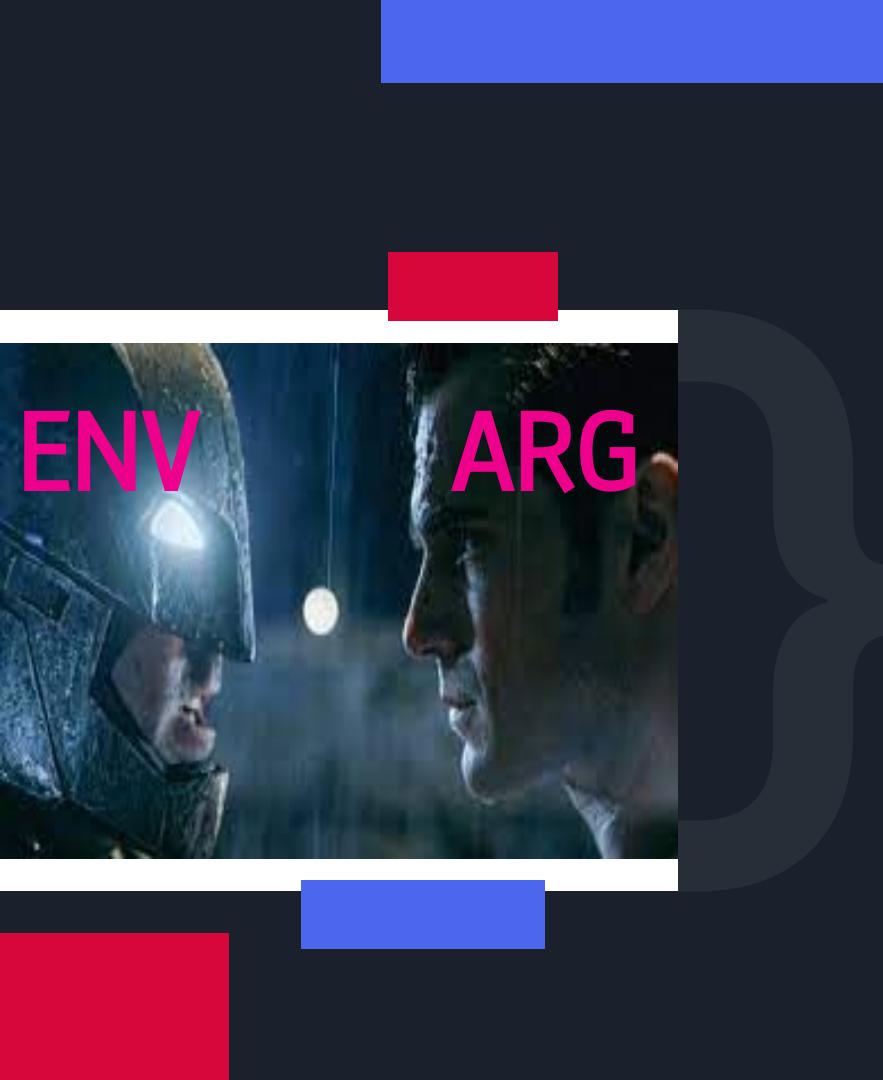
COPY default /etc/nginx/sites-available/default
COPY index.html /usr/share/nginx/html

EXPOSE 7000

CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

PÁGINA INICIAL DO NGINX

Servidor Linux executando Nginx dentro de um contêiner Docker!



ARG vs ENV

No Docker temos dois conceitos parecidos, mas utilizados em momentos diferentes do ciclo de vida as imagens e contêineres:

- ARG: configurações utilizadas em tempo de construção das imagens
- ENV: configurações utilizadas em tempo de criação dos contêineres

DOCKERFILE

```
FROM node:18.16-slim
```

```
WORKDIR /home/app
```

```
ARG PORTA_ARG=4567
```

```
ENV PORTA ${PORTA_ARG}
```

```
COPY . .
```

```
EXPOSE ${PORTA_ARG}
```

```
CMD ["npm", "start"]
```

```
$ docker build -t  
webacademy/webacademy-node:02  
--build-arg PORTA_ARG=4567 .
```

ADENTRANDO EM UM CONTÊINER

Podemos entrar em um contêiner para navegarmos em sua estrutura de diretórios pelo terminal. Para isso executamos o comando **docker exec** passando o ID ou nome do contêiner e o comando de inicialização do terminal TTY.

`docker exec`

Executa um comando dentro de
um contêiner

ACESSANDO O CONTEÎNER

```
$ docker exec -it webacademy-node /bin/bash
```

```
C:\workspace\docker\web-academy-docker\webacademy-nginx>docker exec -it webacademy-node /bin/bash
root@f5003d72b8b5:/home/app# ls
Dockerfile index.js package.json
root@f5003d72b8b5:/home/app# █
```

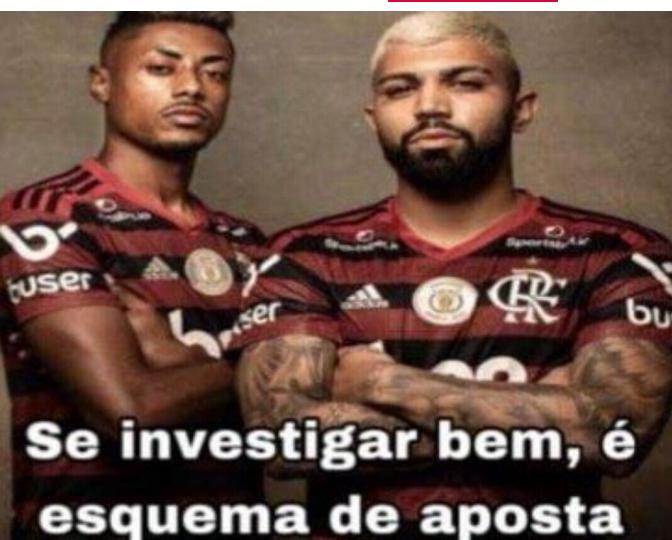
INSPEÇÃO

Podemos inspecionar um contêiner para verificar informações sobre mesmo, como informações de rede, variáveis de ambiente etc:

```
$ docker inspect <CONTEINER>
```

Também é possível inspecionar imagens com a mesma finalidade:

```
$ docker image inspect <IMAGEM>
```



docker inspect

Exibe todas as informações
sobre um contêiner

docker image

inspect

Exibe todas as informações
sobre uma imagem

INSPECIONANDO O CONTÊINER

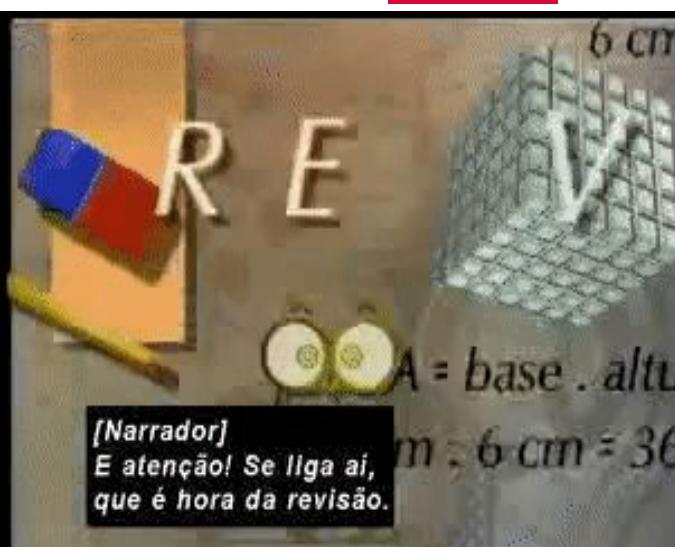
```
$ docker inspect webacademy-node
[
  {
    "Id": "f5003d72b8b5a8cfc910e6ffe457d6d5ebafabd1c4418a119ed2a905f45d1e5",
    "Created": "2023-06-20T03:19:02.2982769Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
      "npm",
      "start"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 4464,
      "ExitCode": 0,
      "Error": ""
    }
  }
]
```

INSPECIONANDO A IMAGEM

```
$ docker image inspect webacademy/webacademy-node:02
[
  {
    "Id": "sha256:5b4898b1123d19fee6b124a779cd66f75370acdf14d04c8a1e856d3c93df4780",
    "RepoTags": [
      "webacademy/webacademy-node:02"
    ],
    "RepoDigests": [],
    "Parent": "",
    "Comment": "buildkit.dockerfile.v0",
    "Created": "2023-06-20T02:46:06.864087Z",
    "Container": "",
    "ContainerConfig": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": null,
      "Cmd": null,
      "Image": "",
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": null
    }
  }
]
```

RECAPITULANDO

- Dockerfile
- Construção de imagens
- ARG X ENV
- Docker exec
- Inspeção



QUESTIONÁRIO

Responda o seguinte questionário no Colabweb:

<https://colabweb.ufam.edu.br/mod/quiz/view.php?id=52130>



VOLUMES



VOLUMES

Os volumes são diretórios localizados fora do contêiner, nos dando a capacidade de manipular arquivos dentro dos contêineres e realizar backups desses conteúdos.

Isso pode ser interessante pois, uma vez que um contêiner é apagado, todo o seu conteúdo vai junto.



TIPOS DE VOLUMES

BIND MOUNT

Associa um diretório do servidor com o contêiner

VOLUME

Armazenamento gerenciado pelo Docker

TMPFS MOUNT

Armazenamento temporário de arquivos na memória do servidor

EXEMPLO

Vamos criar um contêiner que irá executar um servidor MySQL. Execute o seguinte comando:

```
$ docker run -d --name  
bdcontainer -p 3306:3306 -e  
MYSQL_ROOT_PASSWORD=pass2023  
mysql
```

EXEMPLO

Com o contêiner no ar, vamos executar o CLI do MySQL para fazermos algumas operações no servidor:

```
$ docker exec -it bdcontainer  
/usr/bin/mysql -u root -p
```

SAÍDA ESPERADA

```
C:\workspace\docker\web-academy-docker>docker exec -it bdcontainer /usr/bin/mysql -u root -p  
Enter password:
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 9
```

```
Server version: 8.0.33 MySQL Community Server - GPL
```

```
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> |
```

SQL PARA EXECUTAR

```
CREATE SCHEMA IF NOT EXISTS webacademy;
```

```
USE webacademy;
```

```
CREATE TABLE IF NOT EXISTS webacademy_alunos (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(180) NOT NULL
);
```

SAÍDA ESPERADA

```
mysql> CREATE SCHEMA IF NOT EXISTS webacademy;  
Query OK, 1 row affected (0.02 sec)
```

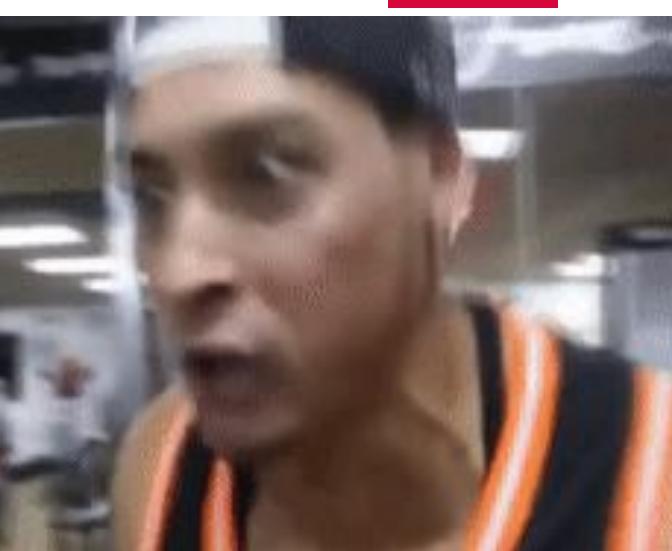
```
mysql> USE webacademy;  
Database changed  
mysql> █
```

```
mysql> CREATE TABLE IF NOT EXISTS webacademy_alunos (  
    -> id BIGINT PRIMARY KEY AUTO_INCREMENT,  
    -> nome VARCHAR(180) NOT NULL  
    -> );  
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> █
```

```
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| webacademy |  
+-----+  
5 rows in set (0.00 sec)
```

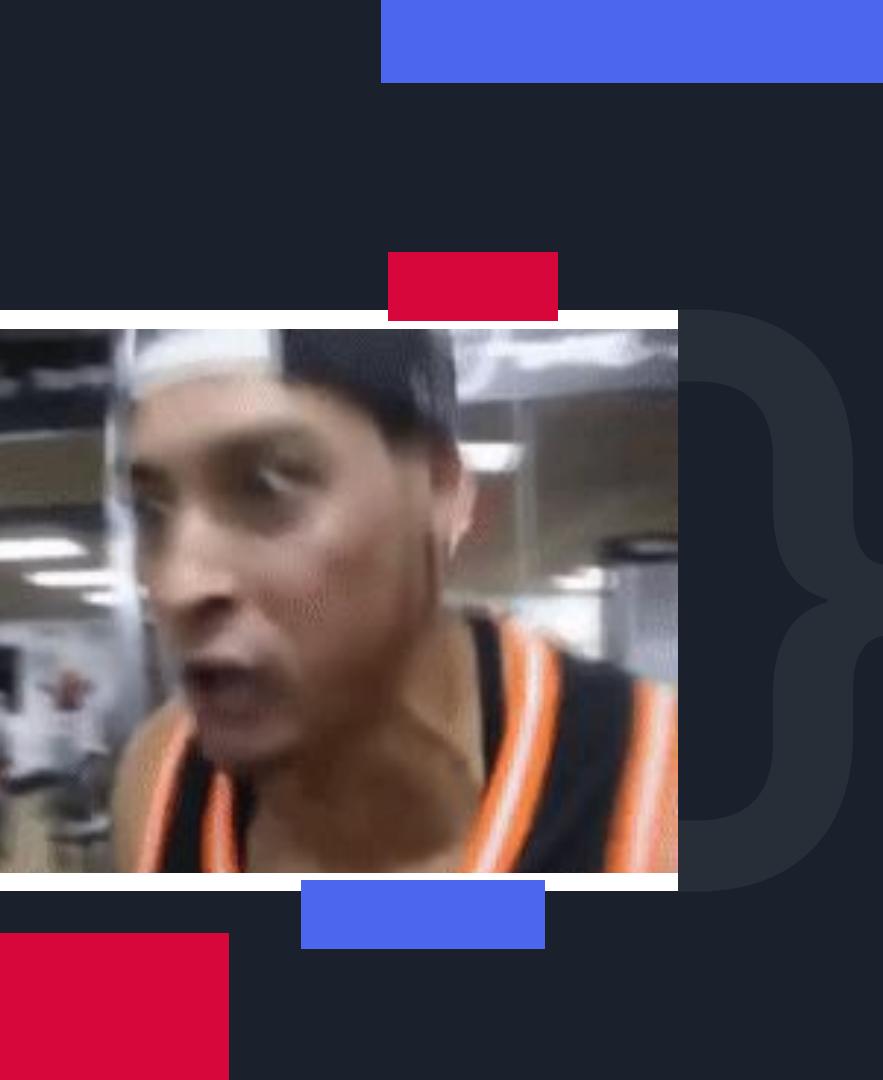
```
mysql> █
```



HORA DE EXERCÍCIO

Agora destrua o contêiner do MySQL e o recrie com as mesmas configurações. Entre no contêiner e busque pelo banco de dados **webacademy**.

Perceba que não temos mais o banco de dados que criamos, mas gostaríamos de mantê-lo mesmo destruindo o servidor. Para isso, vamos associar um volume para salvar os arquivos do MySQL que gerenciam os bancos de dados e demais configurações.



HORA DE EXERCÍCIO

Associando um diretório do servidor com um diretório do contêiner (bind mount), salvaremos um volume de arquivos através do Docker. Destrua qualquer contêiner do MySQL que estiver rodando, e vamos recriá-lo com a seguinte configuração:

```
$ docker run -d --name bdcontainer  
-p 3306:3306 -e  
MYSQL_ROOT_PASSWORD=pass2023  
-v <DIRETORIO_SRV>:/var/lib/mysql mysql
```

COMANDO EXECUTADO

```
C:\workspace\docker\web-academy-docker>docker run -d --name bdcontainer -p 3306:3306 -e MYSQL_ROOT_PASSWORD=pass2023 -v C:\workspace\docker\web-academy-docker\webacademy-volumes\.docker:/var/lib/mysql mysql  
a114069656d2e1a2b55ff47927a048c6e6efd45991a61b2371d49227a98db9e4
```

```
C:\workspace\docker\web-academy-docker>docker inspect bdcontainer
```

```
"Mounts": [  
  {  
    "Type": "bind",  
    "Source": "C:\\workspace\\docker\\web-academy-docker\\webacademy-volumes\\.docker",  
    "Destination": "/var/lib/mysql",  
    "Mode": "",  
    "RW": true,  
    "Propagation": "rprivate"  
  }  
,
```

EXEMPLO

A abordagem do **bind mount** funcionou bem, mas tem uma desvantagem: estamos dependendo diretamente da hierarquia de diretórios do servidor. Caso tenhamos que executar nosso contêiner em um outro servidor, teremos que modificar o comando.

Para evitar isso, faremos uso dos **volumes** propriamente ditos do Docker.

docker volume

Conjunto de comandos de
manipulação de volumes

`docker volume`

`ls`

Exibe todos os volumes existentes

`docker volume`

`create`

Cria um novo volume

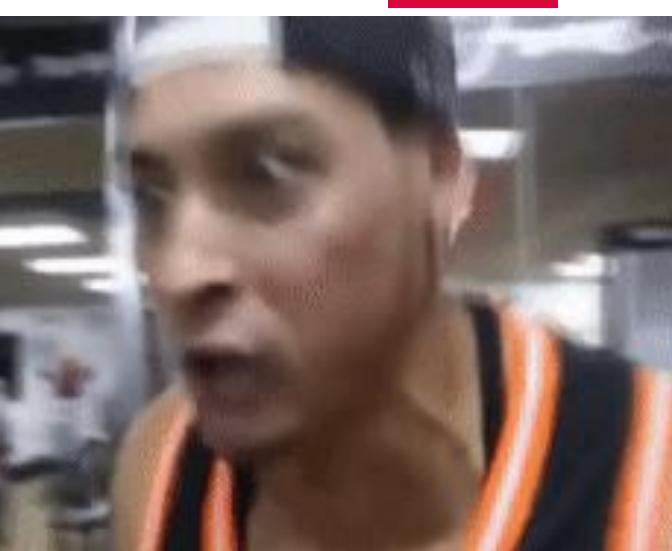
`docker volume inspect`

Inspeciona um volume
existente

`docker volume`

`rm`

Apaga um volume existente



HORA DE EXERCÍCIO

Vamos modificar a criação do nosso contêiner para usar um volume gerenciado pelo Docker. Antes disso, vamos criar um novo volume:

```
$ docker volume create  
vol_bdcontainer
```

Após a criação, vamos recriar o nosso contêiner MySQL.

COMANDO EXECUTADO

```
C:\workspace\docker\web-academy-docker>docker volume create vol_bdcontainer  
vol_bdcontainer
```

```
C:\workspace\docker\web-academy-docker>docker volume ls
```

DRIVER	VOLUME NAME
local	meu-volume
local	mysql-phpmyadmin_dbdata
local	vol_bdcontainer
local	volume-servidor-jupyter

```
C:\workspace\docker\web-academy-docker>docker run -d --name bdcontainer -p 3306:3306 -e MYSQL_ROOT_PASSWORD=pass2023 -v vol_bdcontainer  
:/var/lib/mysql mysql
```

```
3d26b6fd7d87121b399ccdf92f6e3c622f489b6c77c728e0b0f604655a7fb4f
```

```
C:\workspace\docker\web-academy-docker>docker inspect bdcontainer
```

```
[{"Mounts": [ { "Type": "volume", "Name": "vol_bdcontainer", "Source": "/var/lib/docker/volumes/vol_bdcontainer/_data", "Destination": "/var/lib/mysql", "Driver": "local", "Mode": "z", "RW": true, "Propagation": "" } ],
```

TMPFS MOUNT

Como o próprio nome sugere, esse tipo de associação fica armazenado temporariamente na memória RAM do servidor e, quando o contêiner é destruído, esse armazenamento é apagado. Pode não parecer muito útil, mas abre algumas possibilidades principalmente para testes em ambientes de validação.

RECAPITULANDO

- Volumes
- Bind mount
- Tmpfs mount
- Criação de volumes
- Utilização de volumes





QUESTIONÁRIO

Responda o seguinte questionário no Colabweb:

<https://colabweb.ufam.edu.br/mod/quiz/view.php?id=52131>

GERENCIAMENTO DE REDES



REDES

Lembrando do conceito de namespaces do Docker, os contêineres possuem suas interfaces de rede isoladas, sendo necessária uma forma de criar uma **ponte** entre eles para que haja a possibilidade de comunicação entre eles.

Vamos criar um exemplo em que dois contêineres ubuntu tentam se comunicar através do comando ping no terminal.



CRIAÇÃO DOS CONTÊINERES

```
FROM ubuntu:latest
```

```
RUN apt-get update -y
```

```
RUN apt-get install -y iputils-ping
```

```
CMD ["sleep", "1d"]
```

```
$ docker run -d --name webacademy-redes-1 --network rede_webacademy  
webacademy-redes
```

```
$ docker run -d --name webacademy-redes-2 --network rede_webacademy  
webacademy-redes
```

COMUNICAÇÃO DE REDE

Vamos tentar uma comunicação do contêiner **webacademy-redes-1** com o **webacademy-redes-2**. Para isso, vamos pegar o IP do segundo contêiner e executar o comando **ping**. Devemos ter o resultado da chamada com sucesso.

COMUNICAÇÃO DE REDE

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker inspect webacademy-redes-2
```

```
MacAddress : 02:42:dc:11:00:05 , c9:b0:0c:5d:42:00  
"Networks": {  
    "bridge": {  
        "IPAMConfig": null,  
        "Links": null,  
        "Aliases": null,  
        "NetworkID": "b456c53ba2c7d1558c478f",  
        "EndpointID": "dbc15046775e9ebd6e51c",  
        "Gateway": "172.17.0.1",  
        "IPAddress": "172.17.0.3",  
        "IPPrefixLen": 16,  
        "IPv6Gateway": "",  
        "GlobalIPv6Address": ""  
    }  
}
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker exec -it webacademy-redes-1 /bin/bash  
root@351f59a63029:/# 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=0.409 ms  
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.043 ms  
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.072 ms  
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.072 ms  
64 bytes from 172.17.0.3: icmp_seq=5 ttl=64 time=0.050 ms  
64 bytes from 172.17.0.3: icmp_seq=6 ttl=64 time=0.117 ms  
64 bytes from 172.17.0.3: icmp_seq=7 ttl=64 time=0.051 ms  
64 bytes from 172.17.0.3: icmp_seq=8 ttl=64 time=0.127 ms  
64 bytes from 172.17.0.3: icmp_seq=9 ttl=64 time=0.153 ms  
64 bytes from 172.17.0.3: icmp_seq=10 ttl=64 time=0.149 ms  
^C  
--- 172.17.0.3 ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 9400ms  
rtt min/avg/max/mdev = 0.043/0.124/0.409/0.102 ms  
root@351f59a63029:/#
```

COMUNICAÇÃO DE REDE

Essa abordagem funcionou, mas tem um problema: os IP's são gerados de forma aleatória nos contêineres. Dessa forma, o IP utilizado hoje poderá não ser o mesmo em uma nova tentativa de comunicação no futuro. Para isso, devemos utilizar a **resolução de DNS** para a comunicação, através dos nomes dos contêineres.

UÉ !?

```
root@351f59a63029:/# ping webacademy-redes-2  
ping: webacademy-redes-2: Name or service not known  
root@351f59a63029:/#
```



Aparentemente o comando ping não conseguiu encontrar o servidor através do nome. Isso se dá porque não definimos um **túnel** de comunicação de rede entre os contêineres. Para isso vamos criar esses túneis através do comando **docker network**.

docker network

Conjunto de comandos de
manipulação de rede

docker network

ls

Exibe todos os túneis de
redes existentes

docker network

create

Cria um novo túnel de rede

docker network

inspect

Inspeciona um túnel de rede
existente

docker network

rm

Apaga um túnel de rede
existente



HORA DE EXERCÍCIO

Vamos criar um novo túnel de rede chamado **rede_webacademy** que faz uso do driver **bridge**. Após a sua criação, iremos refazer nossos contêineres do Ubuntu conectando o nosso túnel de rede aos contêineres:

```
$ docker network create -d bridge  
rede_webacademy
```

Mais pra frente falaremos sobre o que é um **driver**.

COMUNICAÇÃO DE REDE

```
$ docker network create -d bridge rede_webacademy
```

```
$ docker run -d --name webacademy-redes-1 --network rede_webacademy  
webacademy-redes
```

```
$ docker run -d --name webacademy-redes-2 --network rede_webacademy  
webacademy-redes
```

COMUNICAÇÃO DE REDE

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker network create -d bridge rede_webacademy  
a0ee301657fe50caae9a7d53732815d42781ee728d98871582778a9552582673
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
1e204b2264a2	banco-dados_default	bridge	local
b456c53ba2c7	bridge	bridge	local
a297144746a9	code-bank_default	bridge	local
bbb74e15a55b	codebank_default	bridge	local
d013ca436f06	host	host	local
34c5f429a4ea	minha-bridge	bridge	local
babf51731b9c	none	null	local
d7719b40a26b	rede-farmacia-bd_default	bridge	local
a0ee301657fe	rede_webacademy	bridge	local
c7b5851efd70	sfmc-kafka-ui_default	bridge	local
1e0933a93f56	simulador_default	bridge	local

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker run -d --name webacademy-redes-1 --network rede_webacademy webacademy-redes  
444c44d5f7ece76a968450ec6c7daad8e84a0acd22d0fcc746a8a24918d28e3b
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker run -d --name webacademy-redes-2 --network rede_webacademy webacademy-redes  
5489965cf8f43714301ac5d6d869a50fcc1519dc1a247f1b718067c100898d10
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>
```

COMUNICAÇÃO DE REDE

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker exec -it webacademy-redes-1 /bin/bash
root@444c44d5f7ec:/# ping webacademy-redes-2
PING webacademy-redes-2 (172.25.0.3) 56(84) bytes of data.
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=1 ttl=64 time=0.193 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=2 ttl=64 time=0.247 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=3 ttl=64 time=0.090 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=4 ttl=64 time=0.365 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=5 ttl=64 time=0.102 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=6 ttl=64 time=0.189 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=7 ttl=64 time=0.064 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=8 ttl=64 time=0.081 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=9 ttl=64 time=0.070 ms
64 bytes from webacademy-redes-2.rede_webacademy (172.25.0.3): icmp_seq=10 ttl=64 time=0.133 ms
^C
--- webacademy-redes-2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9162ms
rtt min/avg/max/mdev = 0.064/0.153/0.365/0.091 ms
root@444c44d5f7ec:/# █
```

DRIVER

Um driver define a forma de conexão entre contêineres.

Há três drivers de rede disponíveis no Docker: **bridge**, **host** e **null**.



TIPOS DE DRIVER

BRIDGE

Define um túnel de comunicação entre contêineres

HOST

Define um túnel de comunicação de um contêiner com o servidor

NULL

Não define nenhum túnel de comunicação, impedindo comunicação externa

RECAPITULANDO

- Comunicação entre contêineres
- Criação de redes
- Utilização de redes
- Driver bridge
- Driver host
- Driver null





QUESTIONÁRIO

Responda o seguinte questionário no Colabweb:

<https://colabweb.ufam.edu.br/mod/quiz/view.php?id=52132>

COORDENAÇÃO DE CONTÊINERES





HORA DE EXERCÍCIO

Vamos criar dois contêineres, um rodando um servidor MySQL e o outro rodando um servidor com o cliente PHPMyAdmin. O contêiner do PHPMyAdmin deve acessar o servidor MySQL para gerenciar os bancos de dados e executar scripts SQL. Além disso, vamos salvar os arquivos de configuração do servidor MySQL para backup.

Após isso, acesse a interface do PHPMyAdmin no navegador.

COMANDOS

```
$ docker run -d --name bdcontainer -e MYSQL_ROOT_PASSWORD=pass2023  
-v vol_bdcontainer:/var/lib/mysql --network rede_webacademy -p  
3306:3306 mysql
```

```
$ docker run -d --name bdguicontainer -e PMA_HOST=bdcontainer -e  
PMA_HOST=3306 -e PMA_USER=root -e PMA_PASSWORD=pass2023 --network  
rede_webacademy -p 8080:80 phpmyadmin
```

EXECUÇÃO DOS CONTEÍNERES

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker run -d --name bdcontainer -e MYSQL_ROOT_PASSWORD=pass2023 -v vol_bdcontainer:/var/lib/mysql --network rede_webacademy -p 3306:3306 mysql  
598d493f337576ccb6692817a233a24983070479625baf0c916a99d7541caaf
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker run -d --name bdguicontainer -e PMA_HOST=bdcontainer -e PMA_PORT=3306 -e PMA_USER=root -e PMA_PASSWORD=pass2023 --network rede_webacademy -p 8080:80 phpmyadmin  
5c1d1adb4169da49d7131d36904363eb655424f94cf8421fae46204fcfb852a1
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>docker ps  
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES  
5c1d1adb4169        phpmyadmin        "/docker-entrypoint..."   21 seconds ago     Up 19 seconds      0.0.0.0:8080->80/tcp          bdguicontainer  
598d493f3375        mysql              "docker-entrypoint.s..."  2 minutes ago      Up 2 minutes       0.0.0.0:3306->3306/tcp, 33060/tcp   bdcontainer
```

```
C:\workspace\docker\web-academy-docker\webacademy-redes>
```

PÁGINA INICIAL

phpMyAdmin

Servidor: bdcontainer:3306

Bancos de dados SQL Status Contas de usuário Exportar Importar Configurações Log binário Mais

Novo Recente Favoritos

information_schema mysql performance_schema sys

Configurações gerais

Collation de conexão do servidor MySQL: utf8mb4_unicode_ci Mais configurações

Configurações do Tema

Idioma (Language) Português (Brasil) - Portuguese (Brazil)

Tema pmahomme Ver todos

Console

Configurações gerais

Servidor de banco de dados

- Servidor: bdcontainer via TCP/IP
- Tipo de servidor: MySQL
- Conexão do servidor: O SSL não está sendo usado
- Versão do servidor: 8.0.33 - MySQL Community Server - GPL
- Versão de protocolo: 10
- Usuário: root@172.25.0.3
- Charset do servidor: UTF-8 Unicode (utf8mb4)

Servidor web

- Apache/2.4.56 (Debian)
- Versão do cliente de banco de dados: libmysql - mysqld 8.1.18
- Extensão do PHP: mysqli curl mbstring sodium
- Versão do PHP: 8.1.18

EXECUÇÃO DOS CONTÊINERES

Conseguimos executar nossos contêineres, mas enfrentamos algumas dificuldades:

- Execução em ordem dos contêineres
- Comandos muito extensos
- Lembrar de todas as configurações dos contêineres



COORDENAÇÃO

O Docker possui uma ferramenta que nos permite coordenar a execução de múltiplos contêineres, facilitando a execução de serviços que pertencem a um mesmo contexto. Com ela, podemos criar e remover contêineres, túneis de rede e gerenciar configurações e volumes de uma maneira fácil.

Essa ferramenta é o **docker-compose** (ou **docker compose** a partir da versão 2).

docker compose

Conjunto de comandos de
coordenação de contêineres

docker compose

up

Inicia a criação dos
serviços, redes e execuções
dos contêineres

docker compose down

Inicia a remoção dos serviços
e redes

docker compose

ps

Exibe todos os contêineres em execução

docker compose restart

Reinicia os serviços

DOCKER-COMPOSE.YML

Todas as configurações dos serviços, redes, uso de volumes e demais configurações ficam descritas em um arquivo chamado docker-compose.yml. É nesse arquivo que o Docker lê as configurações e inicia os serviços.

DOCKER-COMPOSE .YML

```
services:          networks:          volumes:  
  service_1:       rede_1:           vol_1:  
    ...             driver: bridge     ...  
  service_2:       rede_2:           vol_2:  
    ...             driver: null      ...
```

SERVICES

Na seção **services** descrevemos todos os contêineres que iremos executar, configurando imagem base, construção através do Dockerfile, mapeamento de portas, volumes, redes e variáveis de ambiente utilizadas.

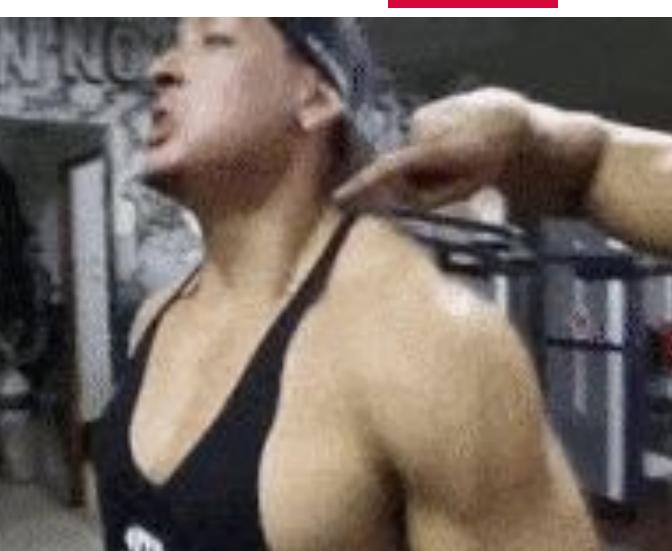
NETWORKS

Na seção **networks** descrevemos as configurações de rede utilizadas pelos contêineres, configurando os nomes e os tipos de **driver** que serão utilizados.

VOLUMES

Na seção **volumes** descrevemos as configurações de volumes utilizados pelos serviços, como nomes e tamanhos de disco que os volumes ocuparão.

Diferente dos serviços e redes, quando a coordenação é encerrada, os volumes criados não são apagados.



HORA DE EXERCÍCIO

Vamos recriar os nossos contêineres do MySQL e do PHPMyAdmin, agora utilizando a coordenação de contêineres para facilitar a execução dos serviços.

DOCKER-COMPOSE .YML

```
version: '3'

services:
  bdcontainer:
    container_name: bdcontainer
    image: mysql:latest
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_PASSWORD=pass2023
    volumes:
      - vol_mysql:/var/lib/mysql
    networks:
      - rede_bd

  bdguicontainer:
    container_name: bdguicontainer
    image: phpmyadmin:latest
    ports:
      - 8080:80
    environment:
      - PMA_HOST=bdcontainer
      - PMA_PORT=3306
      - PMA_USER=root
      - PMA_PASSWORD=pass2023
    depends_on:
      - bdcontainer
    networks:
      - rede_bd

networks:
  rede_bd:
    driver: bridge

volumes:
  vol_mysql:
```

EXECUÇÃO DOS SERVIÇOS

```
C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose up -d  
Creating bdcontainer ... done  
Creating bdguicontainer ... done
```

```
C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose ps
```

Name	Command	State	Ports
bdcontainer	docker-entrypoint.sh mysql	Up	0.0.0.0:3306->3306/tcp, 33060/tcp
bdguicontainer	/docker-entrypoint.sh apac ...	Up	0.0.0.0:8080->80/tcp

```
C:\workspace\docker\web-academy-docker\webacademy-docker>
```

PÁGINA INICIAL

phpMyAdmin

Servidor: bdcontainer:3306

Bancos de dados SQL Status Contas de usuário Exportar Importar Configurações Log binário Mais

Novo Recente Favoritos

information_schema mysql performance_schema sys

Configurações gerais

Collation de conexão do servidor MySQL: utf8mb4_unicode_ci Mais configurações

Configurações do Tema

Idioma (Language) Português (Brasil) - Portuguese (Brazil)

Tema pmahomme Ver todos

Console

Configurações gerais

Servidor de banco de dados

- Servidor: bdcontainer via TCP/IP
- Tipo de servidor: MySQL
- Conexão do servidor: O SSL não está sendo usado
- Versão do servidor: 8.0.33 - MySQL Community Server - GPL
- Versão de protocolo: 10
- Usuário: root@172.25.0.3
- Charset do servidor: UTF-8 Unicode (utf8mb4)

Servidor web

- Apache/2.4.56 (Debian)
- Versão do cliente de banco de dados: libmysql - mysqld 8.1.18
- Extensão do PHP: mysqli curl mbstring sodium
- Versão do PHP: 8.1.18

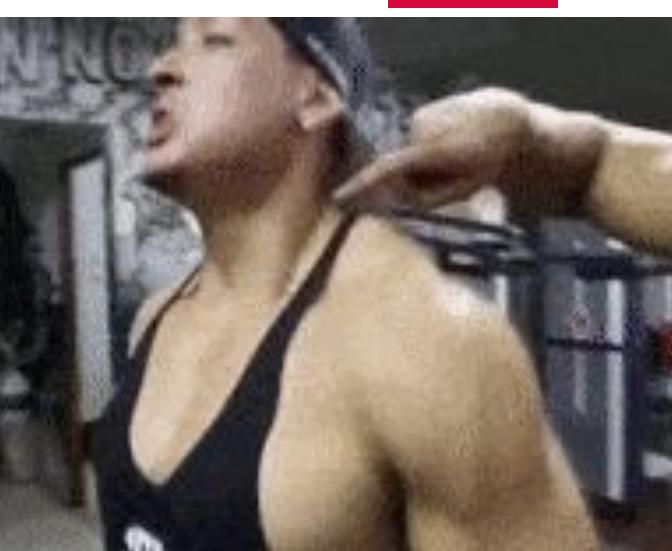
ENCERRAMENTO DOS SERVIÇOS

```
C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose up -d
Creating bdcontainer ... done
Creating bdguicontainer ... done

C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose ps
      Name            Command           State        Ports
-----  
bdcontainer     docker-entrypoint.sh mysqld    Up      0.0.0.0:3306->3306/tcp, 33060/tcp
bdguicontainer   /docker-entrypoint.sh apac ... Up      0.0.0.0:8080->80/tcp

C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose down
Stopping bdguicontainer ... done
Stopping bdcontainer    ... done
Removing bdguicontainer ... done
Removing bdcontainer    ... done
Removing network webacademy-dockercompose_rede_bd

C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose ps
      Name      Command      State      Ports
-----  
C:\workspace\docker\web-academy-docker\webacademy-docker>
```



HORA DE EXERCÍCIO

Vamos criar um arquivo Dockerfile que irá gerar uma imagem baseada no MySQL, mostrando a criação de serviços através de uma imagem customizada.

.DOCKER/MYSQL/DOCKERFILE

```
FROM mysql:latest
```

```
WORKDIR /var/lib/mysql
```

```
VOLUME [ "/var/lib/mysql" ]
```

```
EXPOSE 3306
```

. DOCKER/MYSQL/.ENV

MYSQL_ROOT_PASSWORD=pass2023

. DOCKER/PHPMYADMIN/.ENV

PMA_HOST=bdcontainer

PMA_PORT=3306

PMA_USER=root

PMA_PASSWORD=pass2023

DOCKER-COMPOSE .YML

```
version: '3'

services:
  bdcontainer:
    container_name: bdguicontainer
    image: phpmyadmin:latest
    ports:
      - 8080:80
    env_file:
      - ./docker/phpmyadmin/.env
    depends_on:
      - bdcontainer
    networks:
      - rede_bd
  bdguicontainer:
    container_name: bdguicontainer
    image: phpmyadmin:latest
    ports:
      - 8080:80
    env_file:
      - ./docker/phpmyadmin/.env
    depends_on:
      - bdcontainer
    networks:
      - rede_bd
networks:
  rede_bd:
    driver: bridge
volumes:
  vol_mysql:
```

EXECUÇÃO DOS SERVIÇOS

```
C:\workspace\docker\web-academy-docker\webacademy-dockercompose>docker-compose up -d
Creating network "webacademy-dockercompose_rede_bd" with driver "bridge"
Building bdcontainer
[+] Building 0.6s (6/6) FINISHED
=> [internal] load build definition from Dockerfile          0.1s
=> => transferring dockerfile: 126B                         0.0s
=> [internal] load .dockerignore                            0.1s
=> => transferring context: 2B                           0.0s
=> [internal] load metadata for docker.io/library/mysql:latest 0.0s
=> [1/2] FROM docker.io/library/mysql:latest                0.3s
=> [2/2] WORKDIR /var/lib/mysql                            0.0s
=> exporting to image                                     0.1s
=> => exporting layers                                    0.0s
=> => writing image sha256:d23d98515f22a76775c629ead3ee4cde98e893498433cc9ee7d667e3760a28f2 0.0s
=> => naming to docker.io/library/webacademy-dockercompose_bdcontainer                           0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
WARNING: Image for service bdcontainer was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Creating bdcontainer ... done
Creating bdguicontainer ... done

C:\workspace\docker\web-academy-docker\webacademy-dockercompose>
```

EXECUÇÃO DOS SERVIÇOS

```
C:\workspace\docker\web-academy-docker\webacademy-docker>docker-compose ps
```

Name	Command	State	Ports
bdcontainer	docker-entrypoint.sh mysqld	Up	0.0.0.0:3306->3306/tcp, 33060/tcp
bdguicontainer	/docker-entrypoint.sh apac ...	Up	0.0.0.0:8080->80/tcp

RECAPITULANDO

- Coordenação de contêineres
- Docker-compose.yml
- Comandos docker-compose
- Criação de serviços, redes e volumes
- Variáveis de ambiente
- Construção de imagens personalizadas





QUESTIONÁRIO

Responda o seguinte questionário no Colabweb:

<https://colabweb.ufam.edu.br/mod/quiz/view.php?id=52133>

CONSIDERAÇÕES FINAIS

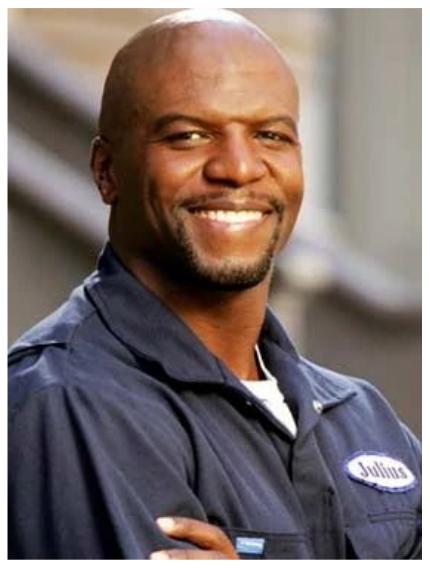


ÚLTIMOS DETALHES

Ainda há muitas ferramentas disponíveis no Docker que podem ser exploradas. Uma ótima fonte de consulta é a documentação oficial disponível em:

<https://docs.docker.com/>

OBRIGADO!



"Se eu não comprar nada o desconto é maior"
Julius