

2021/04/26 - AULA 05.1

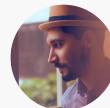
Laboratório de Sistemas Computacionais Complexos

<https://uclab.xyz/sistemas-complexos-2021-aula05-1>



Renato Cordeiro Ferreira
renatocf@ime.usp.br

João Francisco Daniel
joaofran@ime.usp.br



Alfredo Goldman
gold@ime.usp.br

Thatiane de Oliveira Rosa
thatiane@ime.usp.br



Em caso de dúvidas

Acessem www.slido.com com #complexos

ou



Agenda

Tema da aula:

Docker & Docker-Compose

1. Containerização
2. Primeiras interações com *Containers*
3. Criando os próprios *containers*
4. Gerenciando múltiplos *containers*

Containerização

Containerização

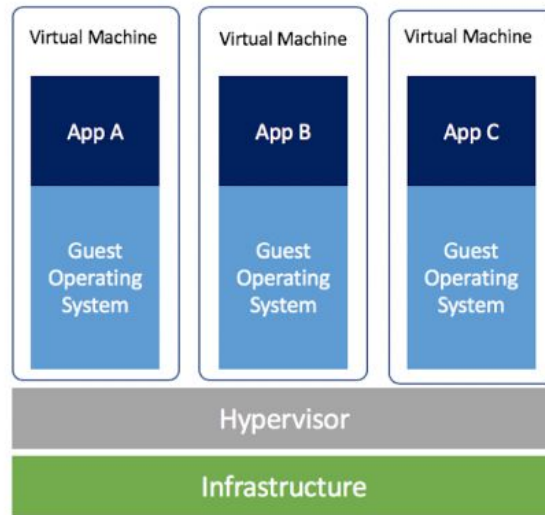
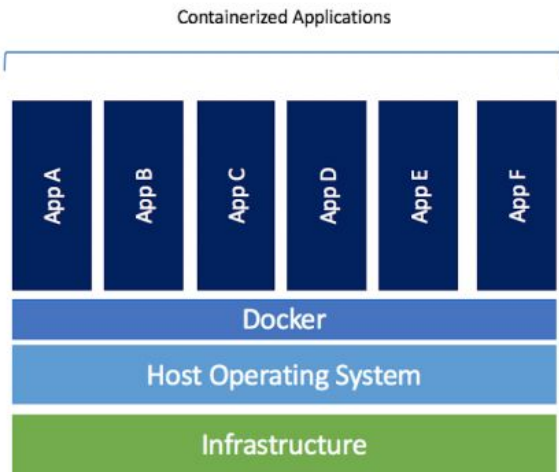
O que é?

Containerização é uma técnica de **virtualização**, semelhante ao uso de máquinas virtuais (VMs), caracterizada pela pequena camada de indireções.

1. Isolamento da infraestrutura física
2. Garantia de reprodutibilidade
3. Auxílio para desenvolvimento, entrega e execução de programas

Containerização

Docker vs. VM:



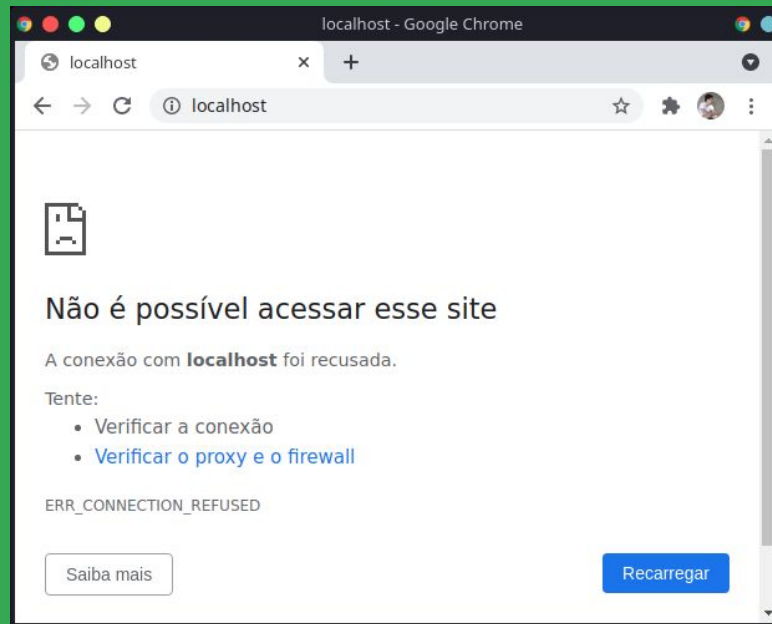
Primeiras interações com *Containers*

Primeiras interações com *Containers*

Executando o primeiro *container*:

1. **nginx** é um servidor HTTP
2. serve na porta 80 por padrão

```
docker container run \  
  --name my-nginx \  
  nginx:alpine
```

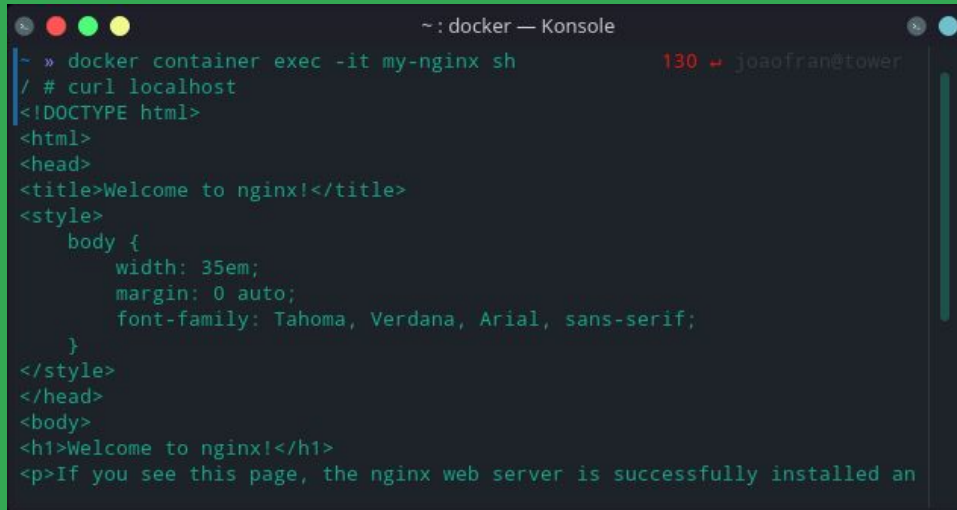


Primeiras interações com *Containers*

Acessando o mesmo *container*:

1. em outra janela do terminal
2. acessando um sh dentro do
nosso *container*

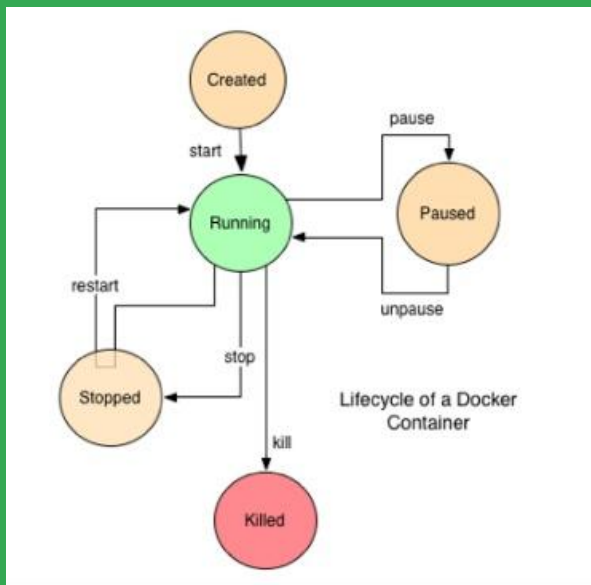
```
docker container exec \  
-it my-nginx \  
sh
```



```
~: docker — Konsole  
~ » docker container exec -it my-nginx sh  
/ # curl localhost  
<!DOCTYPE html>  
<html>  
<head>  
<title>Welcome to nginx!</title>  
<style>  
  body {  
    width: 35em;  
    margin: 0 auto;  
    font-family: Tahoma, Verdana, Arial, sans-serif;  
  }  
</style>  
</head>  
<body>  
<h1>Welcome to nginx!</h1>  
<p>If you see this page, the nginx web server is successfully installed an
```

Primeiras interações com *Containers*

Encerrando e apagando o *container*:



```
docker container \  
stop my-nginx
```




```
docker container \  
rm my-nginx
```

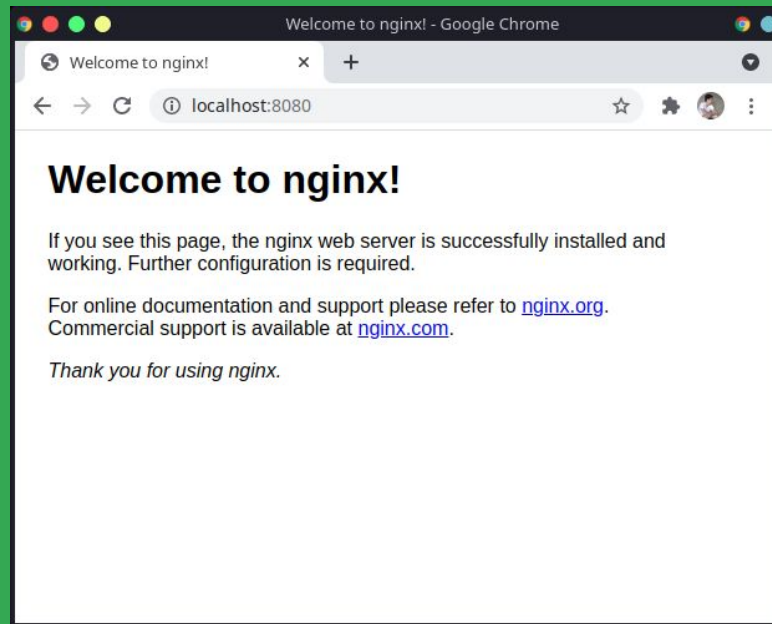
Primeiras interações com *Containers*

Vinculando portas:

1. *containers* têm suas próprias portas
2. é possível criar vínculos com a *host*



```
docker container run \  
  --name my-nginx \  
  --publish 8080:80 \  
  nginx:alpine
```

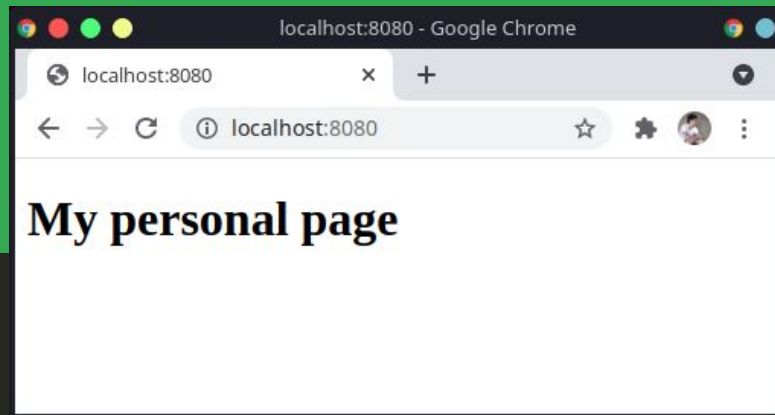


Primeiras interações com *Containers*

Vinculando arquivos:


1. *containers* têm isolamento de *filesystem*
2. também é possível vincular com a *host*

```
docker container run \
  --name my-nginx \
  -p 8080:80 \
  --volume $(pwd)/index.html:/usr/share/nginx/html/index.html \
  nginx:alpine
```



Primeiras interações com *Containers*

Outros comandos básicos e interessantes:



```
docker container ls [-a]
```



```
docker container run \  
  --name my-nginx \  
  --detach nginx:alpine
```

Criando os próprios *containers*

Criando os próprios *containers*

O que são *containers* na prática?

Um **container** é um processo em execução, que roda as instruções definidas em um pacote executável chamado de **imagem**

1. nome da seção de fato: **Criando as próprias imagens**
2. uma imagem deve conter tudo o que for necessário para rodar a sua aplicação
 - a. *runtime*
 - b. ambiente
 - c. código-fonte
 - d. dependências externas
 - e. ...

Criando as próprias imagens

Criando as próprias imagens

Mais detalhes sobre imagens:

1. Nos comandos anteriores, “nginx:alpine” é a especificação de uma imagem
2. Imagens são definidas a partir de uma Dockerfile

Criando as próprias imagens

Retomando nosso exemplo anterior:

1. Dockerfile aceita comandos
2. Imagens são organizadas em camadas para facilitar a reutilização
3. Cada comando gera uma nova camada



```
FROM nginx:alpine  
COPY ./index.html /usr/share/nginx/html/index.html
```



```
docker image \  
  build . \  
  -t my-nginx-image
```



```
docker container run \  
  --name my-nginx-container \  
  -p 8080:80 \  
  -d my-nginx-image
```

Criando as próprias imagens

Mais comandos de Dockerfile:



```
FROM node:alpine

WORKDIR /usr/src/sorter

COPY numbers.txt sort.js ./

CMD node sort.js
```




```
docker image \
  build . \
  -t my-node-sorter
```



```
docker container run \
  -v $(pwd):/usr/src/sorter \
  -d my-node-sorter
```

Criando as próprias imagens

Outros comandos de Dockerfile interessantes:



```
ENV KEY=value      \  
    PORT=3000      \  
    HOST=0.0.0.0   \  
    NODE_ENV=development  
  
EXPOSE ${PORT}
```

Compartilhando imagens no Docker Hub

Compartilhando imagens no Docker Hub

Docker Hub:

Container registry oficial da Docker, é uma espécie de repositório de imagens

1. buscar imagens de base (ex: nginx, node, mongo, etc)
2. compartilhar as próprias imagens com mais pessoas

Compartilhando imagens no Docker Hub

Usando outras imagens:



```
FROM <imagem>:<tag>
```



```
docker container run <imagem>:<tag>
```



```
docker image pull <imagem>:<tag>
```

Compartilhando imagens no Docker Hub

Compartilhando imagens próprias:



```
docker login
```

```
docker image push <username>/<imagem>:<tag>
```


Gerenciando múltiplos *containers*

Gerenciando múltiplos *containers*

Docker-Compose:

Utilitário em cima do Docker que visa facilitar a gestão de *containers*

1. uso baseado em arquivo de configuração `docker-compose.yml`
2. faz a gestão dos *containers* por você
 - a. criar, rodar, parar e limpar
 - b. posicionar vários *containers* em uma mesma rede virtual
 - c. gerir volumes e cache
 - d. ...

Gerenciando múltiplos *containers*

```
version: '3.7'

services:
  back:
    image: jooaodanieel/complexos-docker-back
    ports:
      - 3001:3000

  front:
    image: jooaodanieel/complexos-docker-front
    depends_on:
      - back
    environment:
      - BACKEND_URL=http://localhost:3001
    ports:
      - 3000:3000
```

`docker-compose.yml`:

1. **dois *containers*: front e back**
2. **ports → `docker ... --publish`**
3. **environment**
4. **`depends_on` e ordem de criação**

Gerenciando múltiplos *containers*



```
docker-compose up
```



```
docker-compose down
```

Licença

Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: **Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença**

Mais detalhes sobre essa licença em: creativecommons.org/licenses/by-nc-sa/3.0/

Créditos

Imagens usadas nesta apresentação são provenientes de: [freepik.com](https://www.freepik.com)

Frequência



Senha do Estudante: **585fdi**

2021/04/26 - AULA 05.1

Laboratório de Sistemas Computacionais Complexos

<https://uclab.xyz/sistemas-complexos-2021-aula05-1>



Renato Cordeiro Ferreira
renatocf@ime.usp.br

João Francisco Daniel
joaofran@ime.usp.br



Alfredo Goldman
gold@ime.usp.br

Thatiane de Oliveira Rosa
thatiane@ime.usp.br

