

# FIAP GRADUAÇÃO

# SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

Prof<sup>a</sup>. Aparecida Castello Rosa  
[profaparecida.rosa@fiap.com.br](mailto:profaparecida.rosa@fiap.com.br)

# Agenda

- Continuando com Docker
- Explorar o Docker por meio de um exemplo.
- Publicando a aplicação no Docker hub.
- Persistindo os dados no DB.
- Introdução a Volumes.

# Objetivos

- Continuando com Docker – Parte 3
- Explorar o Docker por meio de um exemplo e Containerizando a aplicação.
- Publicando a aplicação no Docker hub.
- Persistindo os dados no DB.
- Introdução a Volumes.

# Docker

Trabalhando com um Exemplo

# I Docker

- Exemplo get started
- [https://docs.docker.com/get-started/02\\_our\\_app/](https://docs.docker.com/get-started/02_our_app/)
- Criar a pasta exemplo e clonar o projeto
- `git clone https://github.com/docker/getting-started-app.git`

# Docker

```
Windows PowerShell
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/WindowsPowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo> mkdir exemplo

Diretório: C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo

Mode                LastWriteTime         Length Name
----                -
d-----          10/05/2024    09:55         exemplo

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo> |
```

```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo> ls

Diretório: C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo

Mode                LastWriteTime         Length Name
----                -
d-----          10/05/2024    09:55         exemplo

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo> cd .\exemplo\
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> |
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> git clone https://github.com/docker/getting-started-app.git
Cloning into 'getting-started-app'...
remote: Enumerating objects: 75, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 75 (delta 0), reused 2 (delta 0), pack-reused 71
Receiving objects: 100% (75/75), 1.81 MiB | 2.31 MiB/s, done.
Resolving deltas: 100% (14/14), done.
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> |
```

```
Windows PowerShell
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> ls

Diretório:
C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo

Mode                LastWriteTime         Length Name
----                -
d-----          10/05/2024    09:59             getting-started-app

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> |
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo> cd .\getting-started-app\
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> ls

Diretório: C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app

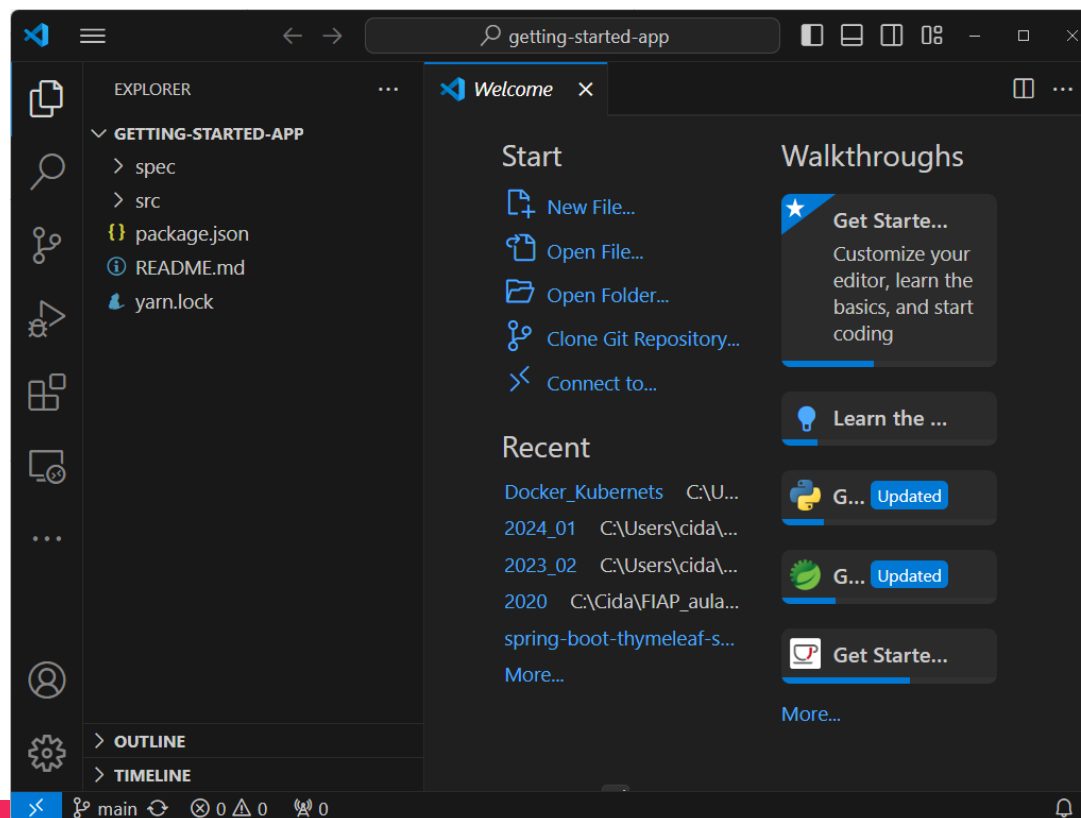
Mode                LastWriteTime         Length Name
----                -
d-----          10/05/2024    09:59             spec
d-----          10/05/2024    09:59             src
-a-----          10/05/2024    09:59         681 package.json
-a-----          10/05/2024    09:59         273 README.md
-a-----          10/05/2024    09:59       150541 yarn.lock

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```



# Docker – VS Code

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> code .  
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```



# I Docker - Dockerfile

**Dockerfile** é um meio que utilizamos para criar nossas próprias imagens, ou seja, ele serve como a receita para construir um **container**.

**Dockerfile** é um arquivo texto que contém todas as especificações e instruções para construir **imagens** Docker.

O **Dockerfile** então é uma forma de criar imagens para execução de **containers** Docker, ou seja, ele serve como uma receita para construir um **container**, permitindo definir um ambiente personalizado e próprio para as nossas aplicações, como, por exemplo, a imagem que vamos utilizar, aplicativos que precisam ser instalados, comandos que serão executados, os volumes que serão montados, etc.

É um **makefile** para criação de **containers**, e nele passamos todas as instruções para a criação do nosso **container**.

Para criar o **Dockerfile** utilizamos um editor de texto (bloco de notas, notepad++, vim do Linux) ou editor de código (VS Code, IntelliJ).

O **Dockerfile** fica na raiz do projeto.

# I Docker - Dockerfile

A estrutura do arquivo **Dockerfile** é composta por instruções, em ordem de execução. As principais instruções do **Dockerfile** são:

**FROM:** é obrigatório e informa qual *imagem* será utilizada como ponto de partida.

**RUN:** utilizado para executar comandos durante o **build** responsável pela montagem da *imagem*.

**CMD:** só deve ser inserido uma vez no **Dockerfile**, pois executa apenas na criação do **container**.

**EXPOSE:** informa quais serão as *portas* liberadas ao criar o container.

**COPY:** copia arquivos e pastas locais.

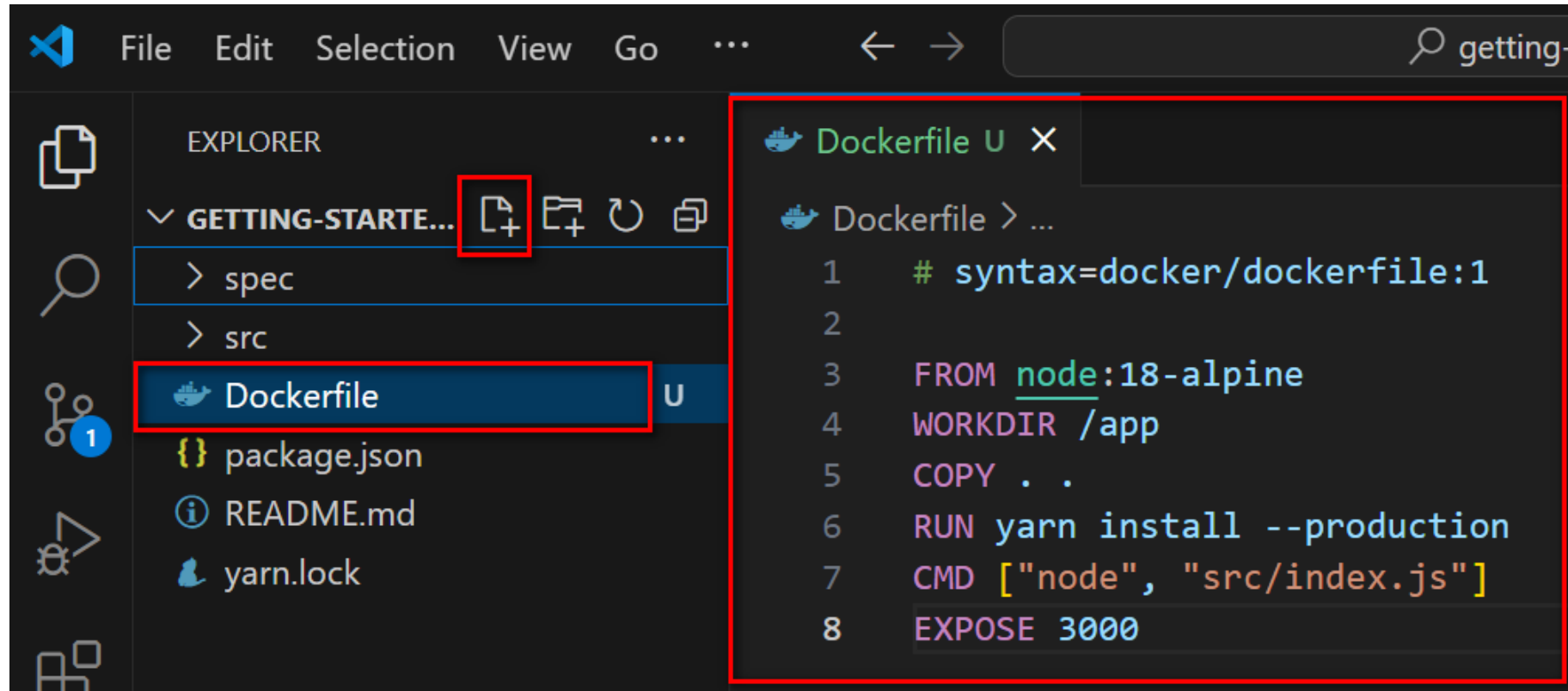
**ADD:** copia arquivos e pastas remotas ou compactadas.

**VOLUME:** informa um ponto de montagem e disponibiliza uma pasta entre o **container** e o **host**.

**WORKDIR:** define uma pasta no **container** onde serão executados os comandos.

**USER:** define o usuário para execução de comandos.

# Docker – getting-started



**FROM:** imagem base

**WORKDIR:** diretório da aplicação

**EXPOSE:** porta da aplicação

**COPY:** quais arquivos precisam ser copiados

# I Docker – *Build Image*

## *Build da Image*

Para criar a imagem do **container** utilizando o **dockerfile** criado, executamos o comando `docker build .`


```
docker build -t <nome da imagem> .
```

Estamos usando o diretório corrente, representado pelo caractere ".", para indicar o **path** (caminho) do arquivo **dockerfile**, mas não precisamos estar no mesmo diretório, para isso, basta passar o **path** do diretório onde o arquivo se encontra.

**Lembre apenas que é o *path* do diretório e não do arquivo.**

A flag `-t` é a tag do nome da nossa aplicação

```
$ docker build -t getting-started .
```



# Docker – Build Image

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker build -t getting-started .
[+] Building 28.1s (11/11) FINISHED                                docker:default
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build definition from Dockerfile               0.0s
=> => transferring dockerfile: 188B                               0.0s
=> resolve image config for docker.io/docker/dockerfile:1       2.5s
=> docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f281 1.4s
=> => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635 0.0s
=> => sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfd 8.40kB / 8.40kB 0.0s
=> => sha256:b5f3b260a9678e1d83d2fce86eeddf79420b79147eaba2a25986f4713 482B / 482B 0.0s
=> => sha256:68ebc061390d9a7d6e194f9d58309c754a53cb8b4e3b0d8959392 1.26kB / 1.26kB 0.0s
=> => sha256:96918c57e42509b97f10c074d80672ecdbd3bb7dcd38c1bd959 11.98MB / 11.98MB 1.2s
=> => extracting sha256:96918c57e42509b97f10c074d80672ecdbd3bb7dcd38c1bd95960cf291 0.2s
=> [internal] load metadata for docker.io/library/node:18-alpine 1.6s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:4837c2ac8998cf172f5892fb45f2 5.0s
=> => resolve docker.io/library/node:18-alpine@sha256:4837c2ac8998cf172f5892fb45f2 0.0s
=> => sha256:4abc20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14f 3.41MB / 3.41MB 0.4s
=> => sha256:45a0166cf96b2a4f328191f78f73e68e0e340450a962ff6fc34 39.82MB / 39.82MB 1.8s
=> => sha256:832e0dc1fe41d061d47d41e00abf6a9dab0c399d69bae854ef1bf 1.38MB / 1.38MB 0.7s
=> => sha256:4837c2ac8998cf172f5892fb45f229c328e4824c43c8506f8ba9c 1.43kB / 1.43kB 0.0s
=> => sha256:267ef0e247celace38e92d80c5c0ebb0aa2ce34e8ba28998f8f1a 1.16kB / 1.16kB 0.0s
=> => sha256:1835bef2bac85b1699df958a9ee9d867ec68551e192317c0a36f5 7.21kB / 7.21kB 0.0s
=> => extracting sha256:4abc20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e6 0.2s
=> => sha256:8ae971f79f99381da4a83f2cb63aa502fb847cc81a2f270326753f628 454B / 454B 0.8s
=> => extracting sha256:45a0166cf96b2a4f328191f78f73e68e0e340450a962ff6fc34013111c 1.6s
=> => extracting sha256:832e0dc1fe41d061d47d41e00abf6a9dab0c399d69bae854ef1bffe197 0.0s
=> => extracting sha256:8ae971f79f99381da4a83f2cb63aa502fb847cc81a2f270326753f6289 0.0s
=> [internal] load build context                                0.3s
=> => transferring context: 6.57MB                                0.3s
=> [2/4] WORKDIR /app                                          1.9s
=> [3/4] COPY . .                                              0.1s
=> [4/4] RUN yarn install --production                        14.0s
=> exporting to image                                          1.1s
=> => exporting layers                                          1.1s
=> => writing image sha256:1aff6656102e27b008b9244d101d96d82581d4a9eddea5c3af4def4 0.0s
=> => naming to docker.io/library/getting-started              0.0s
```

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```



# Docker

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
getting-started     latest         1aff6656102e   6 minutes ago  219MB
postgres            14-alpine      8258e2afe6e4   4 months ago   239MB
postgres            latest         a20f35f462a4   5 months ago   425MB
dpage/pgadmin4      latest         da73a5b9ac16   5 months ago   535MB
mysql               8.0            96bc8cf3633b   6 months ago   582MB
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

Executar o container a partir da imagem criada:

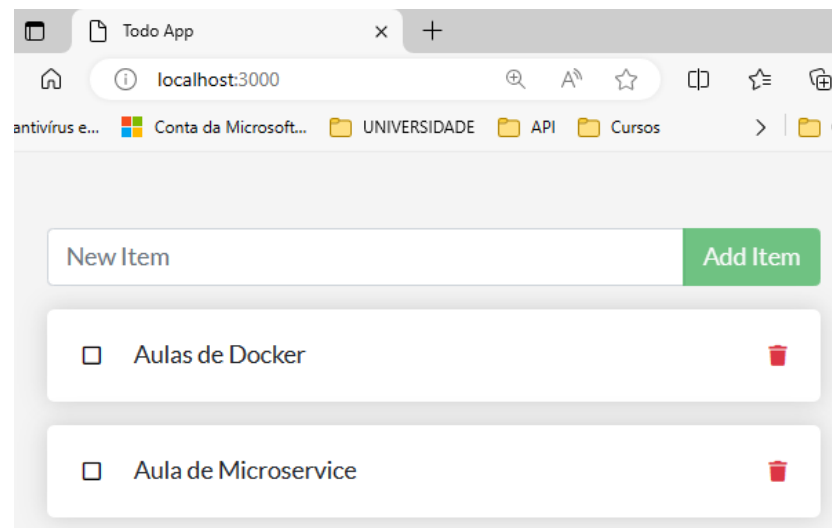
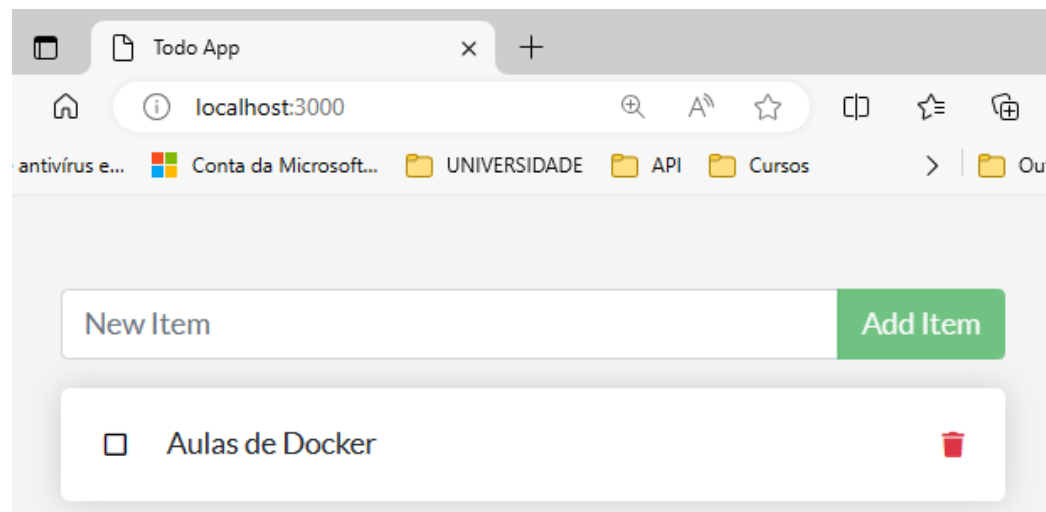
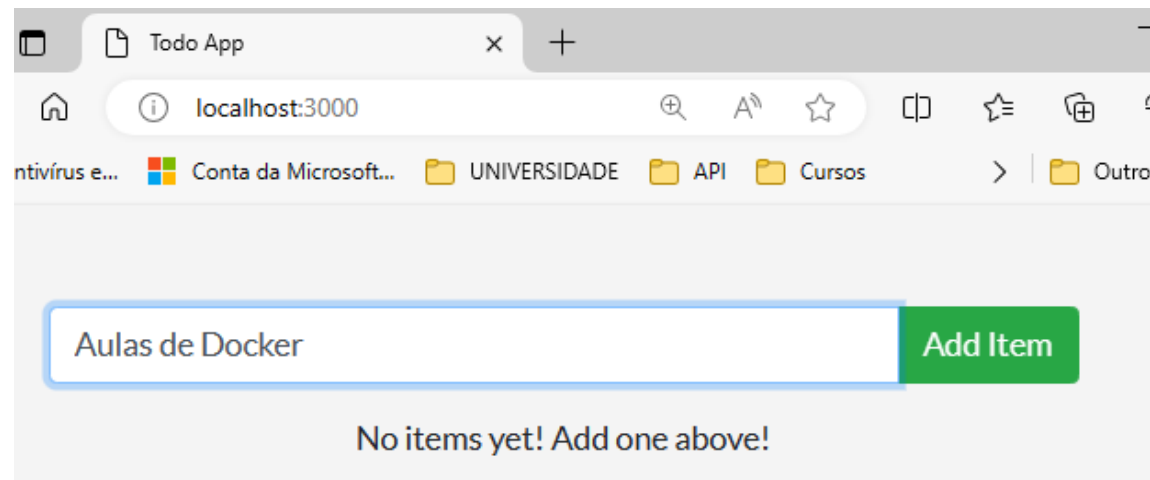
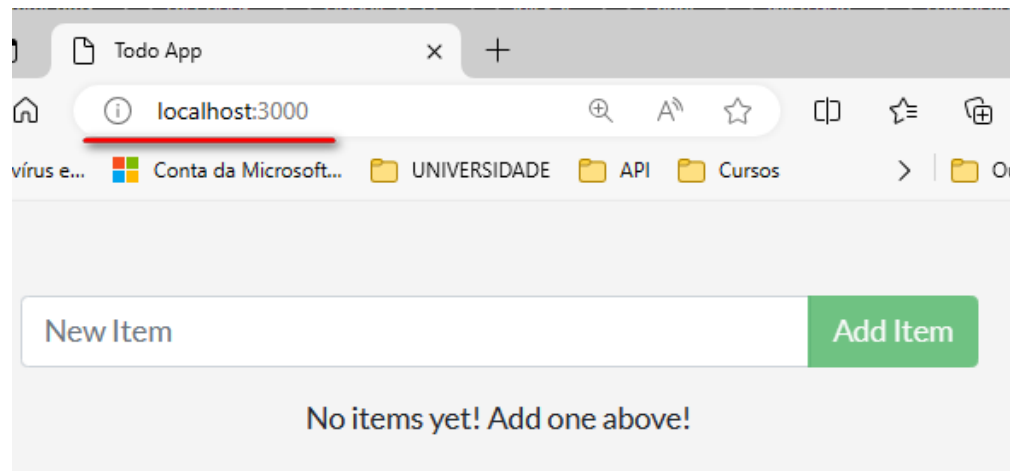
```
$ docker run -dp 3000:3000 getting-started:latest
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker run -dp 3000:3000 getting-started:latest
4f25488c0eabedfec9bb2b67f2ae9cbc46d371ef668e5b7cd401370dce3b6402
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS          NAMES
4f25488c0eab   getting-started:latest             "docker-entrypoint.s...  About a minute ago Up About a minute
0.0.0.0:3000->3000/tcp   gracious_roentgen
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app>
```

## AULA 25 – Docker – parte 3

# Testar no navegador





Os dados estão no **container**. Se removermos o **container** o que acontece?

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
NAMES
4f25488c0eab   getting-started:latest              "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:3000->3000/tcp
gracious_roentgen
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker rm 4f2
Error response from daemon: You cannot remove a running container 4f25488c0eabedfec9bb2b67f2ae9cbc46d371ef668e5b7cd401370dce3b6402. Stop the container before attempting removal or force remove
```

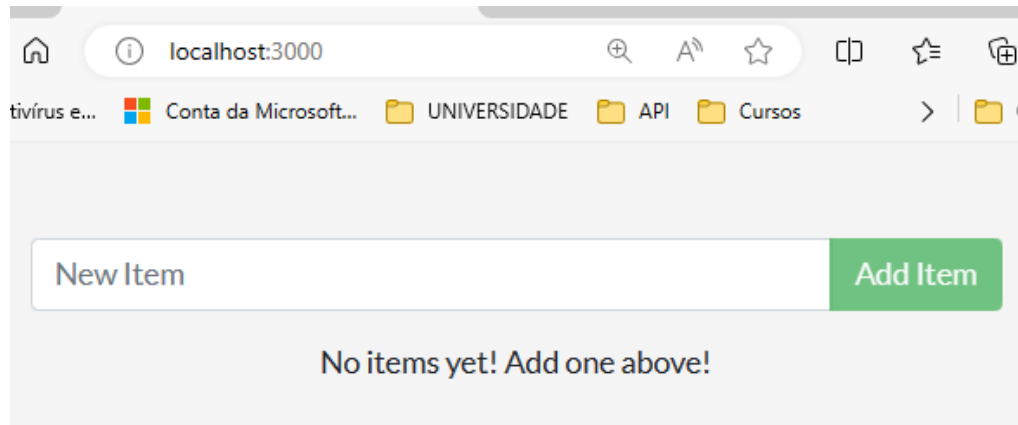
```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker rm -f 4f2
4f2
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

## AULA 25 – Docker – parte 3

# Docker -

```
Windows PowerShell
Windows PowerShell x + -
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
getting-started    latest            1aff6656102e      2 hours ago       219MB
postgres            14-alpine          8258e2afe6e4       4 months ago       239MB
postgres            latest             a20f35f462a4       5 months ago       425MB
dpage/pgadmin4      latest             da73a5b9ac16       5 months ago       535MB
mysql               8.0                96bc8cf3633b       6 months ago       582MB
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker run -dp 3000:3000 getting-started:latest
f7cd9b53ec45d850428d7438888c83e63e0be6edbdad1df68e42f24f664131815
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps
CONTAINER ID   IMAGE                  COMMAND              CREATED          STATUS              PORTS              NAMES
f7cd9b53ec45   getting-started:latest "docker-entrypoint.s... 8 seconds ago    Up 7 seconds        0.0.0.0:3000->3000/tcp   dreamy_johnson
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

# Testar aplicação



## E os dados? O que aconteceu?

Todo dado criado por um **container** é salvo no próprio **container**, quando o **container** é removido perdemos os dados.

No nosso exemplo, os dados estavam no **container** e não foram *persistidos*, ou seja, não foi especificado para salvar os dados.

Mais adiante vamos ver como persistimos os dados usando **volume**.

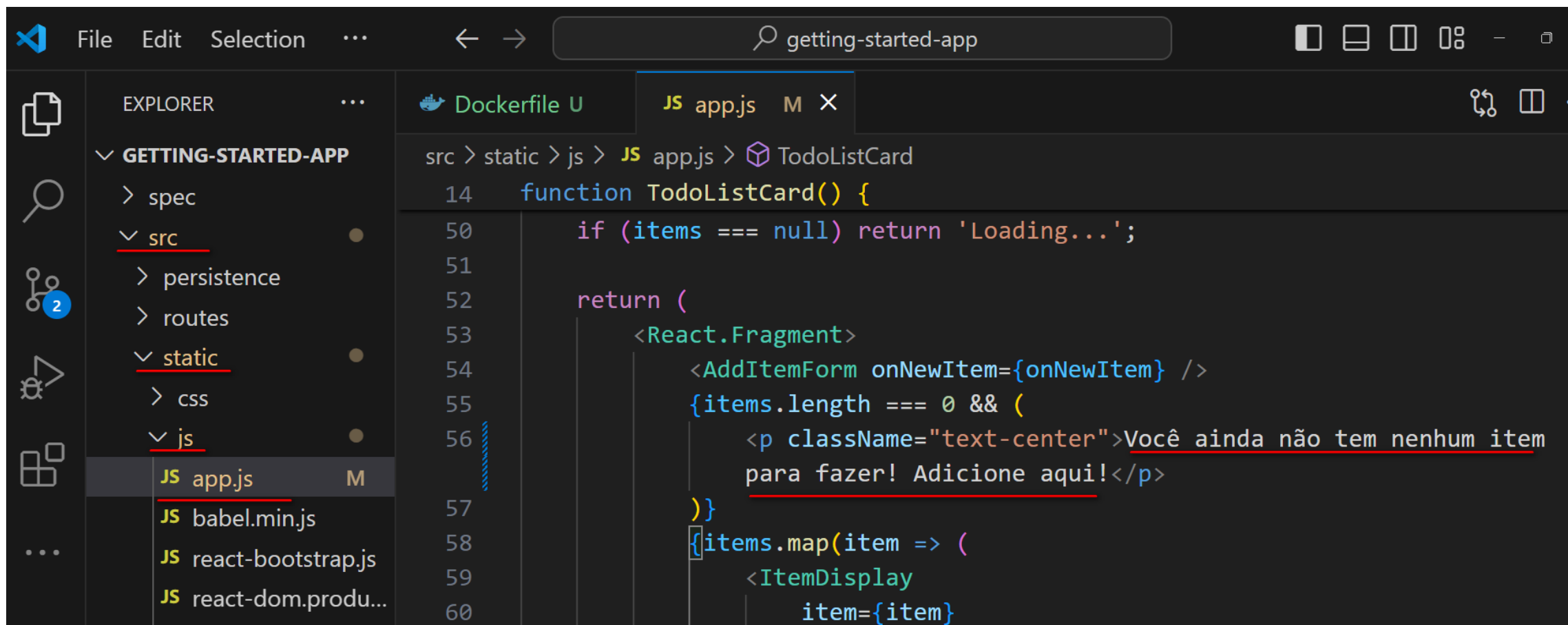
Vamos fazer uma alteração no código fonte da aplicação **TODO** do *getting-started*.

Em src/static/js/app.js, vamos editar o arquivo `app.js` para fazer uma alteração na linha 56. Vamos colocar o parágrafo em português.

Abrir o código fonte no VS Code e alterar o arquivo conforme exemplo a seguir:

```
- <p className="text-center">No items yet! Add one above!</p>  
+ <p className="text-center">Você ainda não tem nenhum item para fazer! Adicione aqui!</p>
```

Salvar o arquivo.



```

src > static > js > JS app.js > TodoListCard
14  function TodoListCard() {
50      if (items === null) return 'Loading...';
51
52      return (
53          <React.Fragment>
54              <AddItemForm onNewItem={onNewItem} />
55              {items.length === 0 && (
56                  <p className="text-center">Você ainda não tem nenhum item
                    para fazer! Adicione aqui!</p>
57              )}
58              {items.map(item => (
59                  <ItemDisplay
60                      item={item}

```

## AULA 25 – Docker – parte 3

## ***Build e versionamento da imagem***

Vamos "***buildar***" a ***imagem*** com a alteração que fizemos usando o comando `docker build` e definindo para **versão 2.0** da aplicação. Não esqueça do "." (ponto) ao final do comando.

Veja que não fizemos nenhuma alteração no ***Dockerfile***, somente no arquivo fonte da aplicação.

```
$ docker build -t getting-started:2.0 .
```

```

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker build -t getting-started:2.0 .
[+] Building 21.7s (11/11) FINISHED                                docker:default
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 188B                               0.0s
=> resolve image config for docker.io/docker/dockerfile:1       2.3s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 0.9s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:4837c2ac8998cf172f5892fb45f229c328e4824c43c8506f8ba9c7996d702430 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 10.06kB                               0.0s
=> CACHED [2/4] WORKDIR /app                                     0.0s
=> [3/4] COPY . .                                               0.1s
=> [4/4] RUN yarn install --production                          16.9s
=> exporting to image                                           1.2s
=> => exporting layers                                           1.2s
=> => writing image sha256:2255108c0459bfb40d3b75259eff261a9b1dc10f0106ede55365ede17594923b 0.0s
=> => naming to docker.io/library/getting-started:2.0          0.0s

```

#### What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```

PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |

```

Agora vamos listar as imagens.

```
$ docker image ls
```

```
Windows PowerShell
Windows PowerShell x + v
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
getting-started	2.0	2255108c0459	2 minutes ago	219MB
getting-started	latest	1aff6656102e	2 hours ago	219MB
postgres	14-alpine	8258e2afe6e4	4 months ago	239MB
postgres	latest	a20f35f462a4	5 months ago	425MB
dpage/pgadmin4	latest	da73a5b9ac16	5 months ago	535MB
mysql	8.0	96bc8cf3633b	6 months ago	582MB

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```



## Alterando a *tag* da imagem

Primeiro vamos remover o *container* anterior.

Queremos deixar a *imagem* gerada com as alterações como sendo a versão *latest*, mas já temos uma *imagem* com essa *tag*. Então vamos fazer trocar as *tags* entre as duas *imagens*.

Primeiro, vamos alterar a *tag* da versão *latest* para 1.0

```
$ docker image tag getting-started:latest getting-started:1.0
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker image tag getting-started:latest getting-started:1.0
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
getting-started	2.0	2255108c0459	10 minutes ago	219MB
getting-started	1.0	1aff6656102e	2 hours ago	219MB
getting-started	latest	1aff6656102e	2 hours ago	219MB
postgres	14-alpine	8258e2afe6e4	4 months ago	239MB
postgres	latest	a20f35f462a4	5 months ago	425MB
dpage/pgadmin4	latest	da73a5b9ac16	5 months ago	535MB
mysql	8.0	96bc8cf3633b	6 months ago	582MB

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

## AULA 25 – Docker – parte 3

Agora temos duas versões: *latest* e **1.0** com o mesmo *id* e, a versão **2.0**?

Vamos alterar a versão **2.0** para *latest*.

```
$ docker image tag getting-started:2.0 getting-started:latest
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker image tag getting-started:2.0 getting-started:latest
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
getting-started	2.0	2255108c0459	13 minutes ago	219MB
getting-started	latest	2255108c0459	13 minutes ago	219MB
getting-started	1.0	1aff6656102e	2 hours ago	219MB
postgres	14-alpine	8258e2afe6e4	4 months ago	239MB
postgres	latest	a20f35f462a4	5 months ago	425MB
dpage/pgadmin4	latest	da73a5b9ac16	5 months ago	535MB
mysql	8.0	96bc8cf3633b	6 months ago	582MB

Veja que agora temos a versão **2.0** e *latest* apontando para o mesmo *id*.

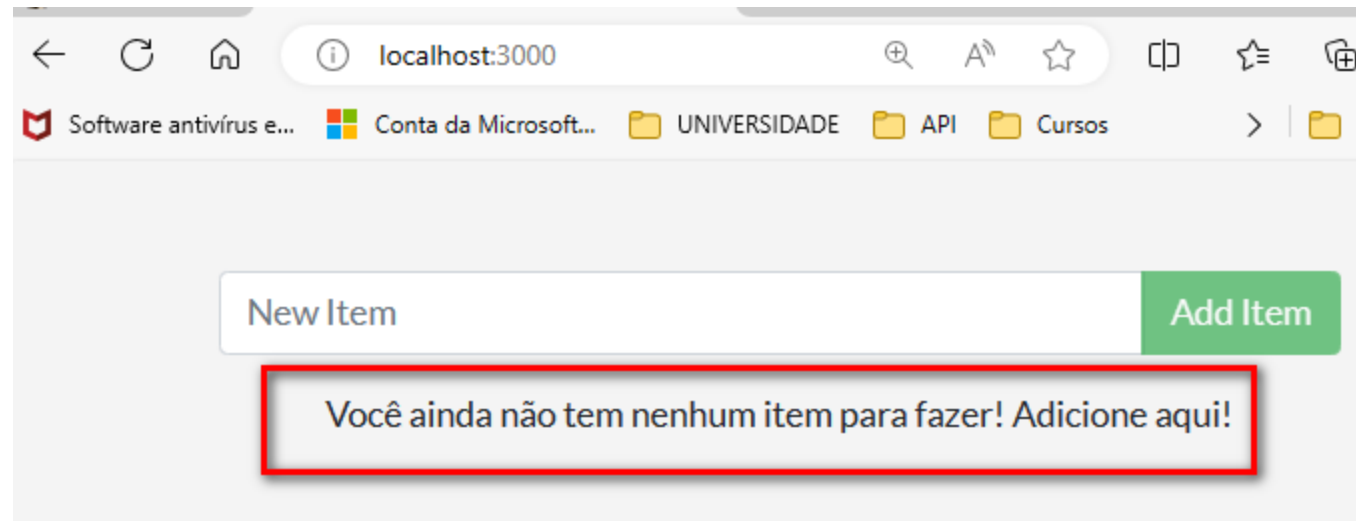
Vamos iniciar o **container** com a versão **latest** e testar se as alterações efetuadas no arquivo fonte foram atualizadas para a nova versão do **container**.

```
docker run -dp 3000:3000 getting-started:latest
```

```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
REPOSITORY    TAG          IMAGE ID          CREATED        SIZE
getting-started 2.0         2255108c0459     23 minutes ago  219MB
getting-started latest      2255108c0459     23 minutes ago  219MB
getting-started 1.0    1aff6656102e    3 hours ago      219MB
postgres      14-alpine   8258e2afe6e4     4 months ago   239MB
postgres      latest      a20f35f462a4     5 months ago   425MB
dpage/pgadmin4 latest      da73a5b9ac16     5 months ago   535MB
mysql         8.0         96bc8cf3633b     6 months ago   582MB
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app>
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker run -dp 3000:3000 getting-started:latest
bef154eacdbb91b62cdd0ce7d7d3ef23a60d7ef9726e863da25f6dd3d00ef1e7
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
bef154eacdbb  getting-started:latest  "docker-entrypoint.s..."  9 seconds ago  Up 8 seconds  0.0.0.0:3000->3000/tcp  adoring_clarke
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

## AULA 25 – Docker – parte 3

# I Testar a aplicação



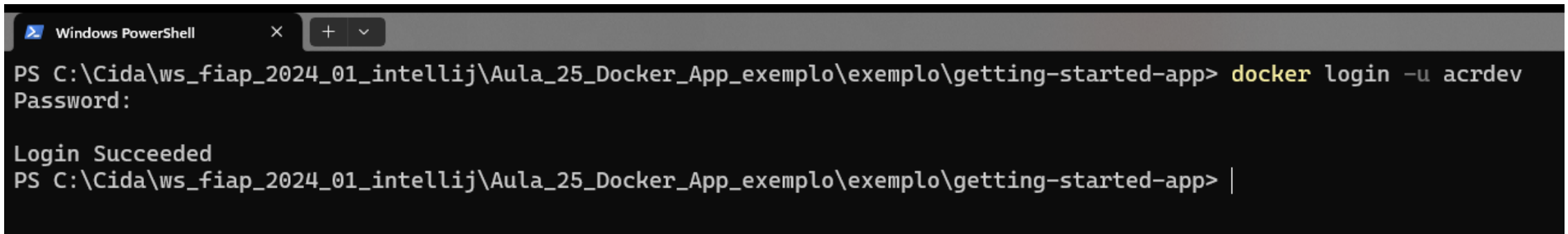
# Compartilhando a aplicação

Docker Hub

# ■ Push da Imagem para o Docker Hub

- Login no Docker Hub por linha de comando

```
$ docker login -u YOUR-USER-NAME
```

A screenshot of a Windows PowerShell terminal window. The title bar shows 'Windows PowerShell' with a close button and a tab. The terminal content shows a command prompt at 'PS C:\Cida\ws\_fiap\_2024\_01\_intellij\Aula\_25\_Docker\_App\_exemplo\exemplo\getting-started-app>' where the command 'docker login -u acrdev' has been entered. The prompt 'Password:' is shown, followed by 'Login Succeeded' and the prompt returning to the command line.

```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker login -u acrdev
Password:
Login Succeeded
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

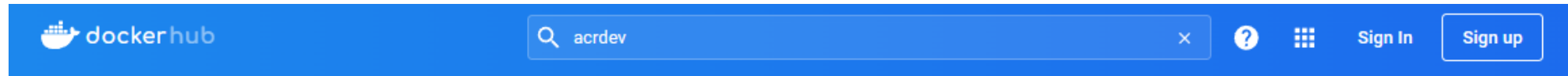
# Push da Imagem para o Docker Hub

```
Login Succeeded
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker tag getting-started acrdev/getting-started
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app>
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app>
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker push acrdev/getting-started
Using default tag: latest
The push refers to repository [docker.io/acrdev/getting-started]
9b96fe15fc09: Pushed
7f702bcca392: Pushed
037b708a86ee: Pushed
3fecce352908: Mounted from library/node
1da03ac21ec8: Pushed
926b35a05f1d: Pushed
d4fc045c9e3a: Pushed
latest: digest: sha256:21c4b97598d16ca34f59b1c92ad0f2a9cd254f834785e4eff90f0e57f84fb990 size: 1787
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<u>acrdev/getting-started</u>	latest	2255108c0459	23 hours ago	219MB
getting-started	2.0	2255108c0459	23 hours ago	219MB
getting-started	latest	2255108c0459	23 hours ago	219MB
getting-started	1.0	1aff6656102e	25 hours ago	219MB
postgres	14-alpine	8258e2afe6e4	4 months ago	239MB
postgres	latest	a20f35f462a4	5 months ago	425MB
dpage/pgadmin4	latest	da73a5b9ac16	5 months ago	535MB
mysql	8.0	96bc8cf3633b	6 months ago	582MB
openjdk	17-slim	37cb44321d04	2 years ago	408MB

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```

# Docker Hub



## Filters

1 - 1 of 1 result for **acrdev**.

Best Match

### Products

- ☐ Images
- ☐ Extensions
- ☐ Plugins



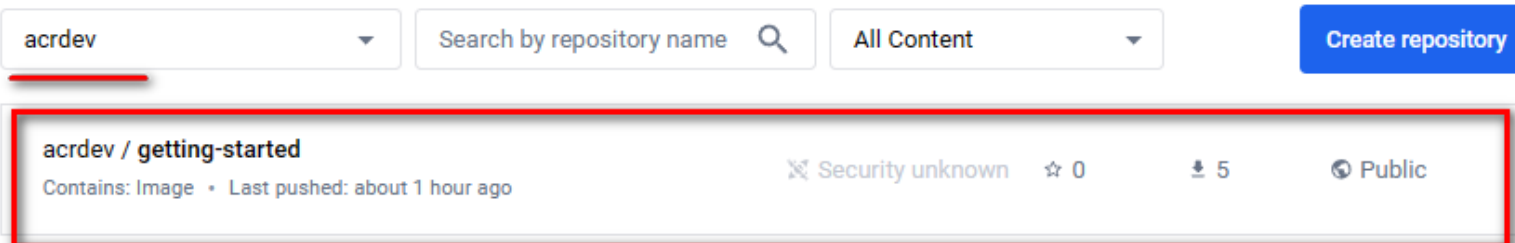
**acrdev/getting-started**

By [acrdev](#) • Updated an hour ago

Linux x86-64

0 • 0

Trusted Content





# Docker Hub

The screenshot shows the Docker Hub interface for the repository `acrdev/getting-started`. The repository is updated about 1 hour ago. A red box highlights the message "This repository does not have a description" with an "INCOMPLETE" status. Below this, it says "This repository does not have a category" also with an "INCOMPLETE" status. The "Tags" section shows one tag, `latest`, which is an image pulled and pushed an hour ago. The "Automated Builds" section explains how to connect GitHub or Bitbucket for automated builds and includes an "Upgrade" button. The "Repository overview" section is also marked as "INCOMPLETE" and includes an "Add overview" button.

dockerhub Explore **Repositories** Organizations Search Docker Hub ctrl+K ? A

acrdev / [Repositories](#) / [getting-started](#) / [General](#) Using 0 of 1 private repositories. [Get more](#)

**General** Tags Builds Collaborators Webhooks Settings

**acrdev/getting-started** Updated about 1 hour ago

This repository does not have a description [edit](#) **INCOMPLETE**

This repository does not have a category [edit](#) **INCOMPLETE**

**Docker commands** [Public View](#)

To push a new tag to this repository:

```
docker push acrdev/getting-started:tagname
```

**Tags**

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
<a href="#">latest</a>		Image	an hour ago	an hour ago

[See all](#)

**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

[Upgrade](#)

**Repository overview** **INCOMPLETE**

An overview describes what your image does and how to run it. It displays in [the public view of your repository](#) once you have pushed some content.

[Add overview](#)

# Docker Hub

docker hub Explore **Repositories** Organizations Search Docker Hub ctrl+K ? A

acrdev / [Repositories](#) / [getting-started](#) / [General](#) Using 0 of 1 private repositories. [Get more](#)

**General** Tags Builds Collaborators Webhooks Settings

**acrdev/getting-started** Updated about 1 hour ago

TODO List - Primeiro exemplo

Cancel Update

This repository does not have a category **INCOMPLETE**

**Docker commands** [Public View](#)

To push a new tag to this repository:

```
docker push acrdev/getting-started:tagname
```

**Tags**

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
<a href="#">latest</a>		Image	an hour ago	an hour ago

[See all](#)

**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

[Upgrade](#)

**Repository overview** **INCOMPLETE**

An overview describes what your image does and how to run it. It displays in [the public view of your repository](#) once you have pushed some content.

[Add overview](#)

# Docker Hub

dockerhub Explore **Repositories** Organizations

Search Docker Hub ctrl+K ? A

acrdev / [Repositories](#) / [getting-started](#) / [General](#) Using 0 of 1 private repositories. [Get more](#)

**General** Tags Builds Collaborators Webhooks Settings

**acrdev/getting-started**

Updated about 1 hour ago

TODO List - Primeiro exemplo

Categories  
API Management Cancel Update

Are you missing a category? [Give feedback](#) .

**Docker commands** Public View

To push a new tag to this repository:

```
docker push acrdev/getting-started:tagname
```

**Tags**

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	an hour ago	an hour ago

[See all](#)

**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) .

Upgrade

**Repository overview** INCOMPLETE



An overview describes what your image does and how to run it. It displays in [the public view of your repository](#) once you have pushed some content.

Add overview

# Docker Hub

### Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 latest		Image	an hour ago	an hour ago

[See all](#)

### Repository overview INCOMPLETE

Write

Preview

App TODO List - listar tarefas para fazer.  
Para executar a aplicação no navegador: `http://localhost:3000`

Cancel

Update

# Docker Hub


The screenshot shows the Docker Hub interface for the repository `acrdev/getting-started`. The page is divided into several sections:

- Header:** Includes the Docker Hub logo, navigation links (Explore, Repositories, Organizations), a search bar, and user profile information.
- Breadcrumbs:** `acrdev / Repositories / getting-started / General`
- Repository Overview:** Shows the repository name `acrdev/getting-started`, updated about 1 hour ago, and a description: "TODO List - Primeiro exemplo". It also has an "API MANAGEMENT" button.
- Docker commands:** Provides a command to push a new tag: `docker push acrdev/getting-started:tagname`. A red arrow points to the "Public View" button next to it.
- Tags:** A table showing the repository contains 1 tag(s). The table has columns for Tag, OS, Type, Pulled, and Pushed.
- Automated Builds:** A section explaining how to connect GitHub or Bitbucket for automated builds, with an "Upgrade" button.
- Repository overview:** A section with a description: "App TODO List - listar tarefas para fazer. Para executar a aplicação no navegador: `http://localhost:3000`" and an "Edit" button.

Tag	OS	Type	Pulled	Pushed
latest		Image	an hour ago	an hour ago

# Docker Hub

FIAP

 **dockerhub**

Explore Repositories Organizations


ctrl+K

?

⋮

A

[Explore](#) / [acrdev/getting-started](#)



**acrdev/getting-started** ☆0

By [acrdev](#) • Updated about 1 hour ago

TODO List - Primeiro exemplo

IMAGE

Manage Repository

↓ Pulls 5

**Overview** Tags

App TODO List - listar tarefas para fazer. Para executar a aplicação no navegador: <http://localhost:3000>

**Docker Pull Command**

```
docker pull acrdev/getting-started
```

Copy

**Baixando a nossa imagem**

# Docker Hub – Pull Image

The screenshot shows the Docker Hub homepage with a blue header. In the top right corner, the 'Sign In' and 'Sign up' buttons are highlighted with a red box. Below the header, a search bar contains the text 'acrdev', and a dropdown menu shows 'Community (1)' with the result 'acrdev/getting-started' highlighted by a red box. The main content area shows the 'acrdev/getting-started' image page. It includes a search bar, 'Sign In', and 'Sign up' buttons. Below the search bar, the breadcrumb 'Explore / acrdev/getting-started' is visible. The image details section shows a cube icon, the name 'acrdev/getting-started' with a star icon, and the text 'By acrdev · Updated about 2 hours ago'. Below this, it says 'TODO List - Primeiro exemplo' and 'IMAGE'. The 'Overview' tab is selected, showing the description 'App TODO List - listar tarefas para fazer. Para executar a aplicação no navegador: http://localhost:3000'. In the bottom right corner, a 'Docker Pull Command' box is highlighted with a red box, containing the command 'docker pull acrdev/getting-started' and a 'Copy' button.

dockerhub

Develop faster. Run anywhere.

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

Search: acrdev

Community (1)

acrdev/getting-started

Spotlight

dockerhub

Search Docker Hub

ctrl+K

Sign In

Sign up

Explore / acrdev/getting-started

acrdev/getting-started ☆0

By acrdev · Updated about 2 hours ago

TODO List - Primeiro exemplo

IMAGE

Overview Tags

App TODO List - listar tarefas para fazer. Para executar a aplicação no navegador: <http://localhost:3000>

Docker Pull Command

```
docker pull acrdev/getting-started
```

Copy



# Docker

```
Windows PowerShell
Windows PowerShell
PS C:\Users\cida> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
getting-started     2.0                2255108c0459       25 hours ago       219MB
getting-started     latest             2255108c0459       25 hours ago       219MB
getting-started     1.0                1aff6656102e       27 hours ago       219MB
postgres            14-alpine          8258e2afe6e4       4 months ago       239MB
postgres            latest             a20f35f462a4       5 months ago       425MB
dpage/pgadmin4      latest             da73a5b9ac16       5 months ago       535MB
mysql               8.0                96bc8cf3633b       6 months ago       582MB
openjdk             17-slim            37cb44321d04       2 years ago        408MB
PS C:\Users\cida> |
```

```
PS C:\Users\cida> docker run -dp 3000:3000 acrdev/getting-started:latest
Unable to find image 'acrdev/getting-started:latest' locally
latest: Pulling from acrdev/getting-started
Digest: sha256:21c4b97598d16ca34f59b1c92ad0f2a9cd254f834785e4eff90f0e57f84fb990
Status: Downloaded newer image for acrdev/getting-started:latest
9e993b1279f6e827a2fdaf46a3193f2c588e17b347bdabb1f126f47ec804f694
PS C:\Users\cida> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
9e993b1279f6	<u>acrdev/getting-started:latest</u>	"docker-entrypoint.s..."	7 seconds ago	<u>Up 6 seconds</u>	0.0.0.0:3000->3000/tcp

```
suspicious_mendel
PS C:\Users\cida> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
getting-started	2.0	2255108c0459	25 hours ago	219MB
getting-started	latest	2255108c0459	25 hours ago	219MB
acrdev/getting-started	latest	2255108c0459	25 hours ago	219MB
getting-started	1.0	1aff6656102e	27 hours ago	219MB
postgres	14-alpine	8258e2afe6e4	4 months ago	239MB
postgres	latest	a20f35f462a4	5 months ago	425MB
dpage/pgadmin4	latest	da73a5b9ac16	5 months ago	535MB
mysql	8.0	96bc8cf3633b	6 months ago	582MB
openjdk	17-slim	37cb44321d04	2 years ago	408MB

```
PS C:\Users\cida> |
```

Add Item

Você ainda não tem nenhum item para fazer! Adicione aqui!

**Persistindo os dados no DB**

## Volumes

Quando um **container** é executado, ele usa as diversas camadas de uma **imagem** para seu sistema de arquivos (**filesystem**).

Cada **container** também recebe seu próprio "espaço temporário" ("**scratch space**") para criar/atualizar/remover arquivos.

Quaisquer alterações não serão vistas em outro **container**, mesmo que estejam usando a mesma **imagem**.

Todo dado criado por um **container** é salvo no próprio **container**, quando o **container** é removido perdemos os dados.

## Entendendo na prática

Vamos iniciar um **container** Ubuntu e criar um arquivo chamado `/data.txt` com um número gerado randomicamente entre 1 e 10000.

```
$ docker run -d ubuntu bash -c "shuf -i 1-10000 -n 1 -o /data.txt && tail -f /dev/null"
```

Estamos iniciando um `bash shell` que chama dois comandos (porque estamos utilizando `&&`). A primeira parte escolhe um único número aleatório e grava em `/data.txt`. O segundo comando é simplesmente observar um arquivo para manter o **container** funcionando.

```
Windows PowerShell
Windows PowerShell x + v
PS C:\Users\cida>
PS C:\Users\cida> docker run -d ubuntu bash -c "shuf -i 1-10000 -n 1 -o /data.txt && tail -f /dev/null"
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
49b384cc7b4a: Pull complete
Digest: sha256:3f85b7caad41a95462cf5b787d8a04604c8262cdcdf9a472b8c52ef83375fe15
Status: Downloaded newer image for ubuntu:latest
088c3986d0cac84728cdc7eb23955ac3f0fefcc0cf6382020c9a9f4beb212624
PS C:\Users\cida> |
```

## Executando o *container*

```
$ docker exec <container-id> cat /data.txt
```

```
PS C:\Users\cida>
PS C:\Users\cida> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
088c3986d0ca   ubuntu   "bash -c 'shuf -i 1-...' 43 seconds ago Up 42 seconds        angry_williams
PS C:\Users\cida> docker exec 088 cat /data.txt
4775
PS C:\Users\cida> |
```

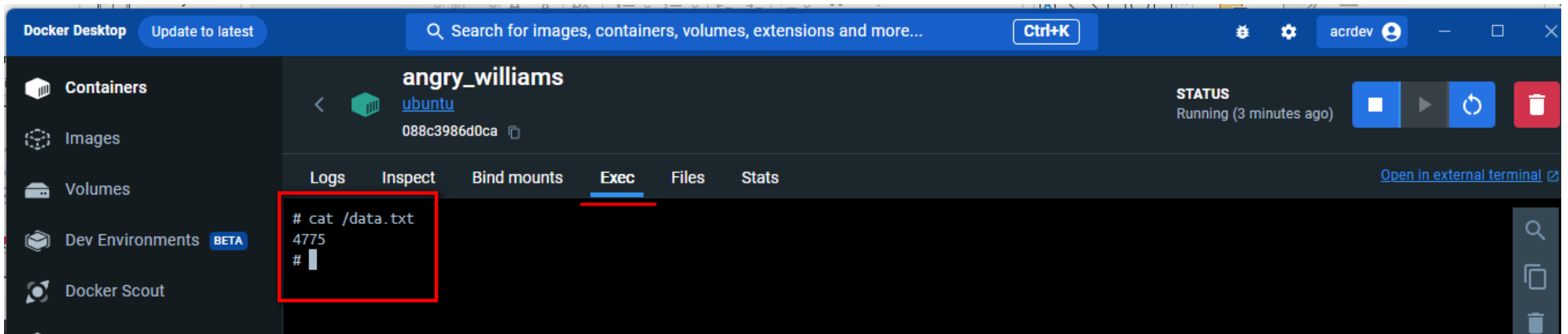
Docker Desktop interface showing container management. The 'Containers' tab is active, displaying a table of containers. The table includes columns for Name, Image, Status, CPU (%), Port(s), Last started, and Actions. Two containers are listed: 'suspicious\_mendel' (Exited) and 'angry\_williams' (Running). The 'angry\_williams' container is highlighted with a red box, and a context menu is open for it, showing options like 'View details', 'View image packages and CVEs', 'Copy docker run', 'Open in terminal', 'View files', 'Pause', and 'Restart'. A red arrow points to the 'Open in terminal' option.

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
suspicious_mendel 9e993b1279f6	acrdev/getting-started:late	Exited	0%	3000:3000	1 hour ago	[Play] [Three dots] [Trash]
angry_williams 088c3986d0ca	ubuntu	Running	0%		2 minutes ago	[Stop] [Three dots] [Trash]

- View details
- View image packages and CVEs
- Copy docker run
- Open in terminal
- View files
- Pause
- Restart

## AULA 25 – Docker – parte 3





# Docker

Agora vamos iniciar um outro **container** do Ubuntu com a mesma imagem e podemos ver que ele não tem o arquivo criado (data.txt).

```
$ docker run -it ubuntu ls /
```

```
PS C:\Users\cida>
PS C:\Users\cida> docker run -it ubuntu ls /
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
PS C:\Users\cida> |
```

Nesse caso, o comando lista os arquivos no diretório **root** do **container**. Não temos o arquivo `data.txt`. Isso acontece porque ele foi escrito para o **scratch space** do primeiro **container**.

Vamos remover o primeiro *container*.

```
$ docker rm -f <container-id>
```

```
PS C:\Users\cida>
PS C:\Users\cida> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
2bba66dd0c7e   ubuntu                             "ls /"                 2 minutes ago Exited (0) 2 minutes ago
088c3986d0ca   ubuntu                             "bash -c 'shuf -i 1-..." 9 minutes ago Up 9 minutes
angry_williams
9e993b1279f6   acrdev/getting-started:latest      "docker-entrypoint.s..." About an hour ago Exited (0) 24 minutes ago
suspicious_mendel
PS C:\Users\cida> docker rm -f 088
088
PS C:\Users\cida> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
2bba66dd0c7e   ubuntu                             "ls /"                 9 minutes ago Exited (0) 9 minutes ago
088c3986d0ca   ubuntu                             "bash -c 'shuf -i 1-..." 9 minutes ago Up 9 minutes
angry_williams
9e993b1279f6   acrdev/getting-started:latest      "docker-entrypoint.s..." About an hour ago Exited (0) 32 minutes ago
suspicious_mendel
PS C:\Users\cida> |
```

## Volumes

Os **volumes** fornecem a capacidade de conectar caminhos de sistema de arquivos (*filesystem path*) específicos do **container** de volta à máquina **host**.

Se “**montarmos**” (*mount*) um diretório no **container**, as alterações nesse diretório também serão refletidas na máquina **host**. Se montarmos o mesmo diretório nas reinicializações do **container**, veremos os mesmos arquivos.

**Volumes** são diretórios externos ao **container**, que são montados diretamente nele, e dessa forma, não seguem o padrão de camadas.

A principal função do volume é **persistir os dados**. Diferentemente do *filesystem* do **container** que é **volátil** e toda informação escrita nele é perdida quando deletamos o **container**, quando escrevemos em um **volume** os dados continuam lá, independentemente do estado do **container**.

Com **volumes** do Docker separamos os arquivos de dados que são gerados por um aplicativo ou banco de dados do restante do armazenamento do **container**, facilitando dessa forma a substituição ou atualização de um **container**.

Os **volumes** permitem que dados importantes existam fora do **container**, o que significa que podemos substituir um **container** sem perder os dados que ele criou.

Também podemos excluir um **container** sem excluir os dados que estão no **volume**, o que permite que os **container** sejam alterados ou atualizados sem perder dados do usuário.

Para usar esse recurso usamos o comando **Volume**, e, ele deve ser preparado antes que os **containers** que usam os dados sejam criados.

Existem dois tipos de arquivos associados a um aplicativo:

- 1- Os arquivos necessários para executar o aplicativo;
- 2- Os arquivos de dados que o aplicativo gera enquanto é executado.

# TODO App

Persistindo os dados

## TODO App

Por padrão, o **TODO App (Aplicativo de Tarefas)** armazena seus dados em um banco de dados SQLite em `/etc/todos/todo.db` no sistema de arquivos (*filesystem*) do **container**. O SQLite é um banco de dados relacional que armazena todos os dados em um único arquivo e é ideal para pequenas demonstrações.

Como o banco de dados é um arquivo único, podemos **persistir** esse arquivo no **host** e disponibilizá-lo para o próximo **container**, então ele poderá continuar de onde o último parou.

Ao criar um **volume** e anexá-lo (geralmente chamado de "*montagem*") ao diretório onde armazenamos os dados, podemos persistir os dados. À medida que o **container** grava no arquivo `todo.db`, ele persistirá os dados no **volume** do **host**.

O Docker gerencia o **volume**, incluindo o local de armazenamento em disco.

## Volume - Comandos principais

```
#Criar volume
$ docker volume create

# Listar os volume
$ docker volume ls

# Remover volume
docker rm -v <container-id>
docker volume rm <volume-name>

# Detalhes do volume
$ docker volume inspect <name>

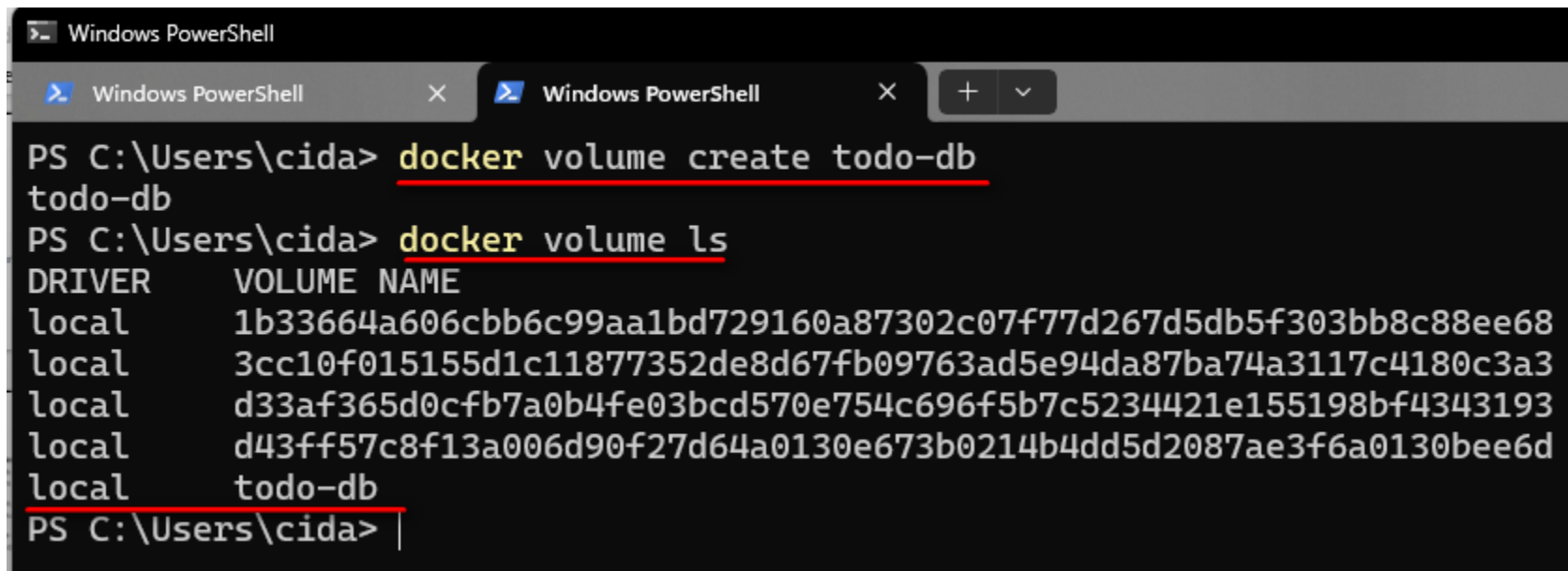
# Remover os volumes que não estão sendo utilizados (Cuidado com esse comando):
$ docker volume prune
```



Precisamos montar o **volume**.

1 - Criar um volume.

```
$ docker volume create todo-db
```



```
Windows PowerShell
Windows PowerShell x Windows PowerShell x + v
PS C:\Users\cida> docker volume create todo-db
todo-db
PS C:\Users\cida> docker volume ls
DRIVER      VOLUME NAME
local       1b33664a606cbb6c99aa1bd729160a87302c07f77d267d5db5f303bb8c88ee68
local       3cc10f015155d1c11877352de8d67fb09763ad5e94da87ba74a3117c4180c3a3
local       d33af365d0cfb7a0b4fe03bcd570e754c696f5b7c5234421e155198bf4343193
local       d43ff57c8f13a006d90f27d64a0130e673b0214b4dd5d2087ae3f6a0130bee6d
local       todo-db
PS C:\Users\cida> |
```

2 - Parar e remover o **container** do **TODO App** com o comando `docker rm -f <id>`, porque ele ainda está em execução (ou não) sem usar o volume persistente.

```
$ docker ps
# ou
$ docker ps -a
```

```
PS C:\Users\cida> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS
PORTS          NAMES
857fc14aa29f   getting-started:latest             "docker-entrypoint.s..."  46 seconds ago Up 44 seconds
0.0.0.0:3000->3000/tcp   cranky_lichterman
PS C:\Users\cida>
PS C:\Users\cida> docker rm -f 857
857
PS C:\Users\cida> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS        PORTS          NAMES
PS C:\Users\cida> |
```

3 - Iniciar o **container** do **TODO App**, adicionando a opção `--mount` para especificar montagem de **volume**. Vamos dar um nome ao volume e montá-lo em `/etc/todos` no **container**, que captura todos os arquivos criados no caminho.

```
$ docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
```

Entendendo cada comando:

`--mount` - comando utilizado para montar **volumes**.

`type=volume` - Indica que o tipo é **volume**. Ainda existe o tipo **bind**, no qual, em vez de indicar um **volume**, indicamos um diretório como **source**.

`source=todo-db` - qual o **volume** que vamos montar.

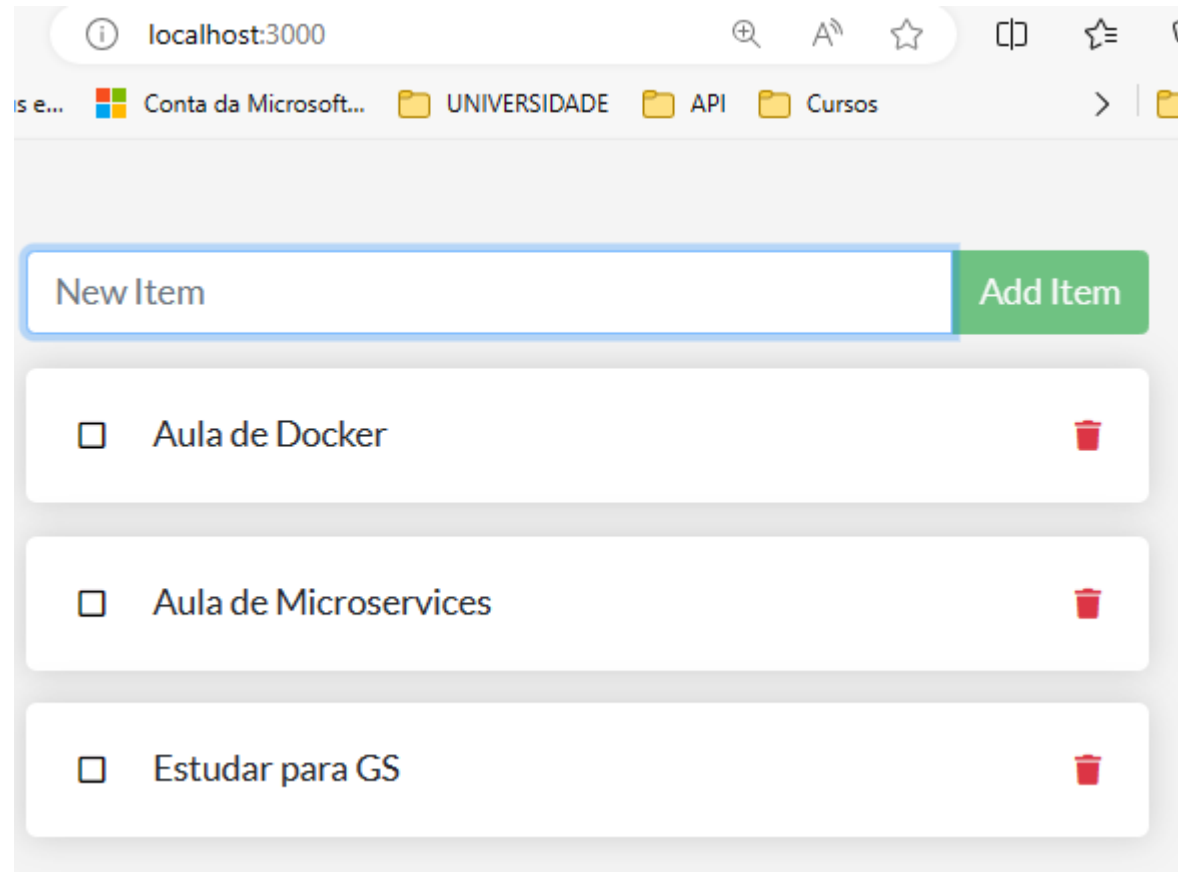
`target` - onde, no **container**, será montado esse **volume**.

```
Windows PowerShell
Windows PowerShell
PS C:\Users\cida> docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
27e56aba53a1c5c6b9fa4dde4b5562559b9f72473936e78fa1e9234010aec7ee
PS C:\Users\cida> |
```

```
PS C:\Users\cida> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS
27e56aba53a1   getting-started                    "docker-entrypoint.s... About a minute ago Up About a minute 0.0.0.0:3000
->3000/tcp     great_mclaren
PS C:\Users\cida>
PS C:\Users\cida> docker volume ls
DRIVER          VOLUME NAME
local          1b33664a606cbb6c99aa1bd729160a87302c07f77d267d5db5f303bb8c88ee68
local          3cc10f015155d1c11877352de8d67fb09763ad5e94da87ba74a3117c4180c3a3
local          d33af365d0cfb7a0b4fe03bcd570e754c696f5b7c5234421e155198bf4343193
local          d43ff57c8f13a006d90f27d64a0130e673b0214b4dd5d2087ae3f6a0130bee6d
local          todo-db
PS C:\Users\cida> |
```

# Docker

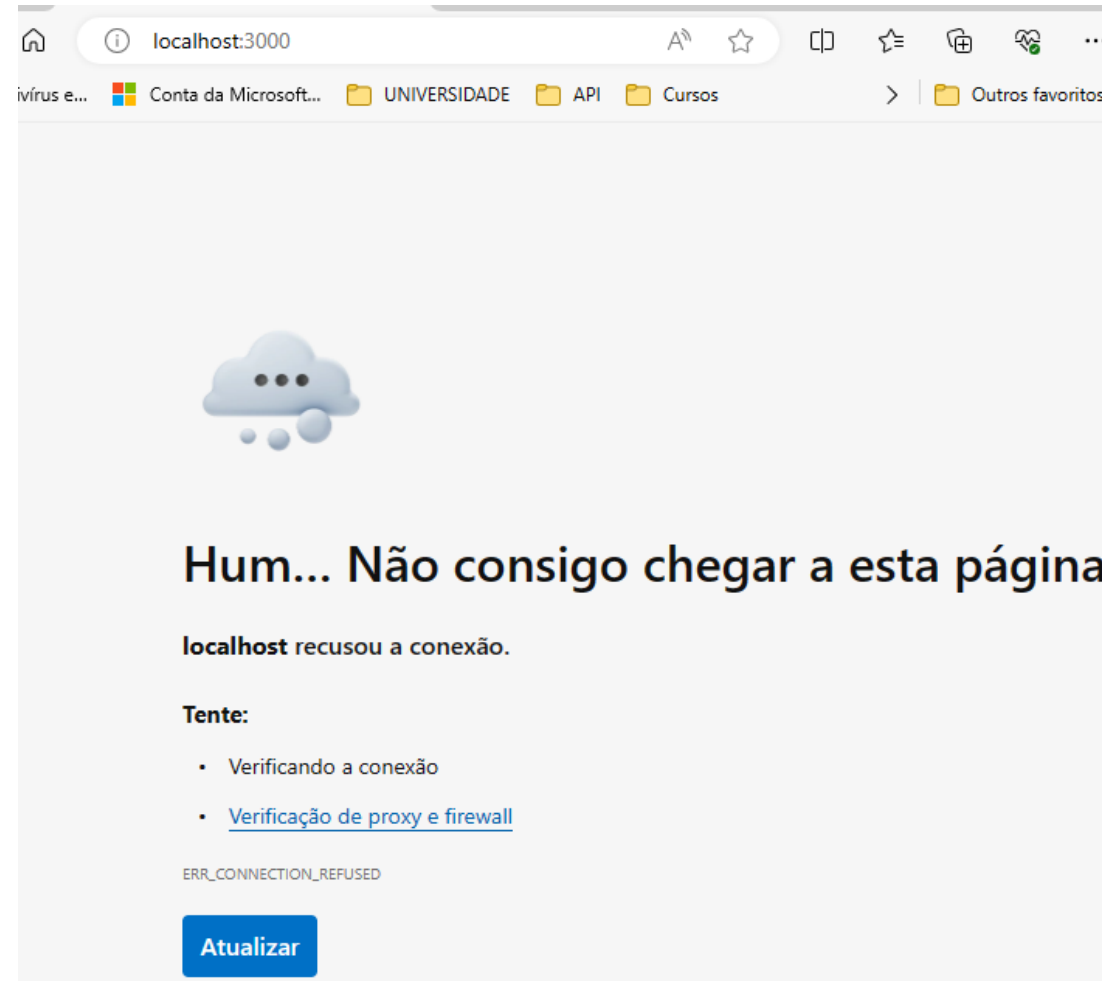
1 - Assim que o **container** for iniciado, vamos abrir o aplicativo (no navegador) e adicionar alguns itens à lista de tarefas.



# Docker

2 - Agora vamos parar e remover o **container** do aplicativo de **TODO**. Use Docker Desktop ou `docker ps` para obter o ID e depois `docker rm -f <id>` para removê-lo.

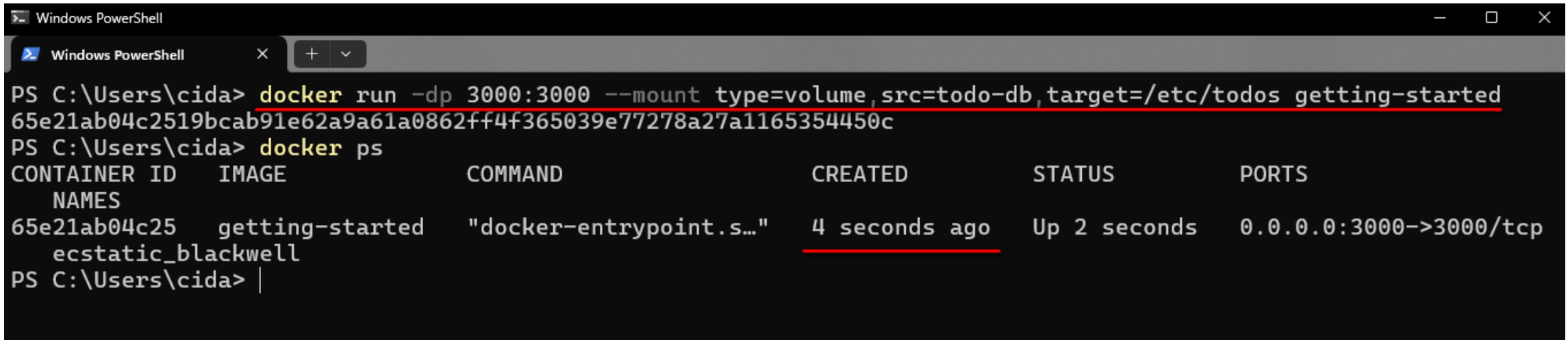
```
Windows PowerShell
PS C:\Users\cida> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
27e56aba53a1   getting-started "docker-entrypoint.s..." 6 minutes ago  Up 6 minutes  0.0.0.0:3000->3000/tcp
great_mclaren
PS C:\Users\cida> docker rm -f 27e56aba53a1
27e56aba53a1
PS C:\Users\cida> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Users\cida> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Users\cida> |
```



# Docker

3 - Vamos iniciar um novo **container** usando as etapas anteriores.

```
$ docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
```



```
Windows PowerShell
PS C:\Users\cida> docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
65e21ab04c2519bcab91e62a9a61a0862ff4f365039e77278a27a1165354450c
PS C:\Users\cida> docker ps
```

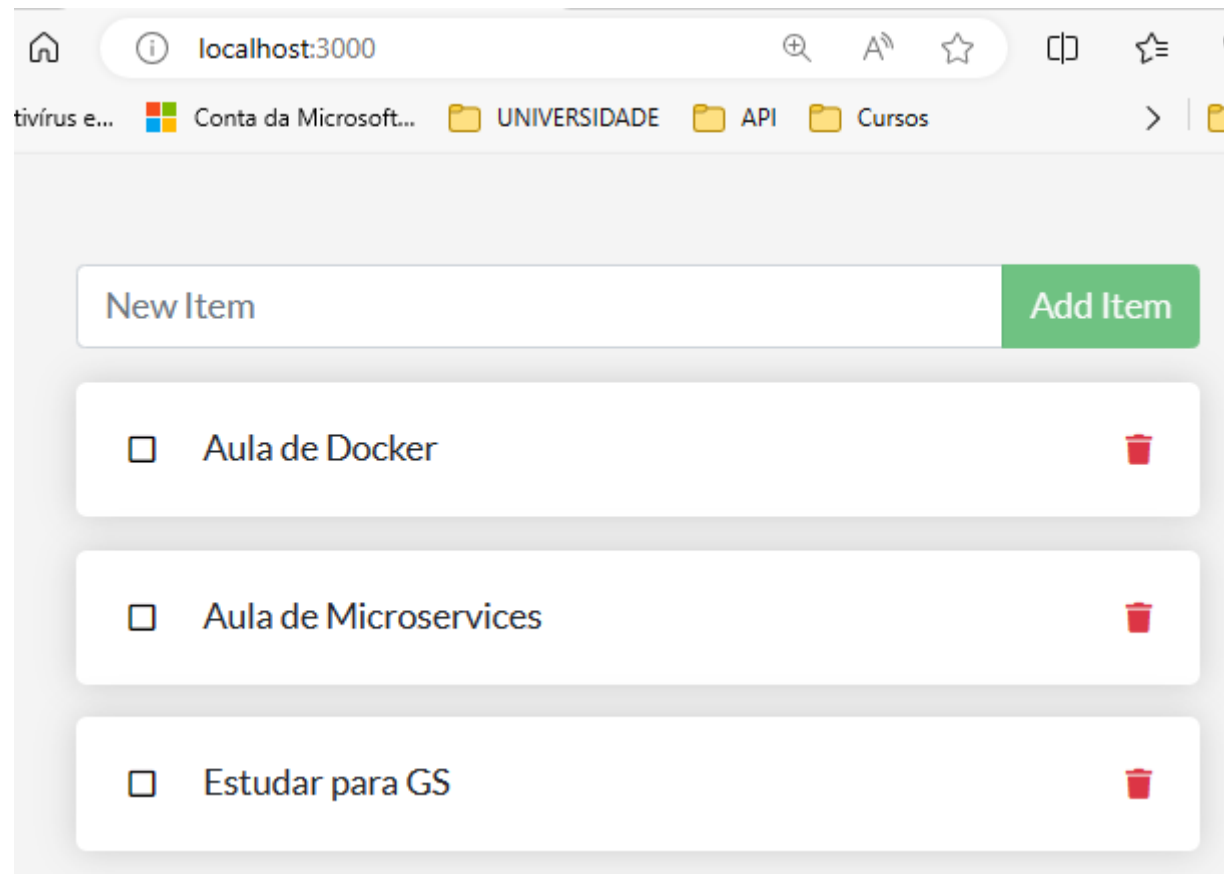
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
65e21ab04c25	getting-started	"docker-entrypoint.s..."	4 seconds ago	Up 2 seconds	0.0.0.0:3000->3000/tcp

```
ecstatic_blackwell
PS C:\Users\cida> |
```



# Docker

4 - Vamos abrir o aplicativo novamente. Você deverá ver os itens ainda em sua lista.



## **Criando Volume no Docker Desktop**

# I Docker Desktop

Para criar um volume utilizando o Docker Desktop.

- 1 - Selecione **Volumes** no Docker Desktop.
- 2 - Em **Volumes**, selecione **Create**.
- 3 - Especifique `todo-db` como o nome do **volume**, e então selecione **Create**.

Para parar e remover o *container* do aplicativo:

- 1 - Selecione **Containers** no Docker Desktop.
- 2 - Selecione **Delete** na coluna **Actions** do *container*.

Para iniciar o *container* do aplicativo de tarefas com o volume montado:

- 1 - Selecione a caixa de pesquisa na parte superior do Docker Desktop.
- 2 - Na janela de pesquisa, selecione a guia **Imagens**.
- 3 - Na caixa de pesquisa, especifique o nome do contêiner `getting-started`.

# Docker Desktop - continuação

## Dica

Use o filtro de pesquisa para filtrar imagens e mostrar apenas Imagens Locais.

4 - Selecione sua imagem e selecione **Run**.

5 - Selecione **Configurações opcionais**.

6 - Em **host Port**, especifique a porta, por exemplo, `3000`.

7 - Em **Host path**, especifique o nome do volume, `todo-db`.

8 - Em **Container path**, especifique `/etc/todos`.

9 - Selecione **Run**.

# Inspecionando o Volume

## Inspecionando o Volume

Muitos perguntam: “**Onde o Docker está armazenando meus dados quando uso um volume?**”

Para saber, podemos utilizar o comando `docker volume inspect <name>`.

Primeiro vamos listar os **Volumes**.

```
docker volume ls
```

Depois vamos inspecionar o **Volume** `todo-db`.

```
$ docker volume inspect todo-db
```

```
PS C:\Users\cida>
PS C:\Users\cida> docker volume ls
DRIVER      VOLUME NAME
local       1b33664a606cbb6c99aa1bd729160a87302c07f77d267d5db5f303bb8c88ee68
local       3cc10f015155d1c11877352de8d67fb09763ad5e94da87ba74a3117c4180c3a3
local       d33af365d0cfb7a0b4fe03bcd570e754c696f5b7c5234421e155198bf4343193
local       d43ff57c8f13a006d90f27d64a0130e673b0214b4dd5d2087ae3f6a0130bee6d
local       todo-db
PS C:\Users\cida> |
```

```
Windows PowerShell
Windows PowerShell x + v
PS C:\Users\cida> docker volume inspect todo-db
[
  {
    "CreatedAt": "2024-05-12T13:13:35Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
    "Name": "todo-db",
    "Options": null,
    "Scope": "local"
  }
]
PS C:\Users\cida> |
```

O **Mountpoint** (ponto de montagem) é o local real dos dados no disco. Observe que na maioria das máquinas, precisamos ter acesso **root** para acessar este diretório a partir do **host**.



```
PS C:\Users\cida>
PS C:\Users\cida> docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
```



**Copyright © 2024**  
**Prof<sup>a</sup>. Aparecida de Fátima Castello Rosa**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).