

FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

Prof^a. Aparecida Castello Rosa
profaparecida.rosa@fiap.com.br

I Agenda

- Continuando com Docker
- Persistindo os dados no DB.
- Introdução a Volumes.
- *Dockerizando* projeto produto-mvc

Objetivos

- Continuando com Docker – Parte 4
- Persistindo os dados no DB.
- Introdução a Volumes.
- Criar *Dockerfile* para projeto produto-mvc.

Docker

Trabalhando com um Exemplo

I Docker

- Exemplo get started
- 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

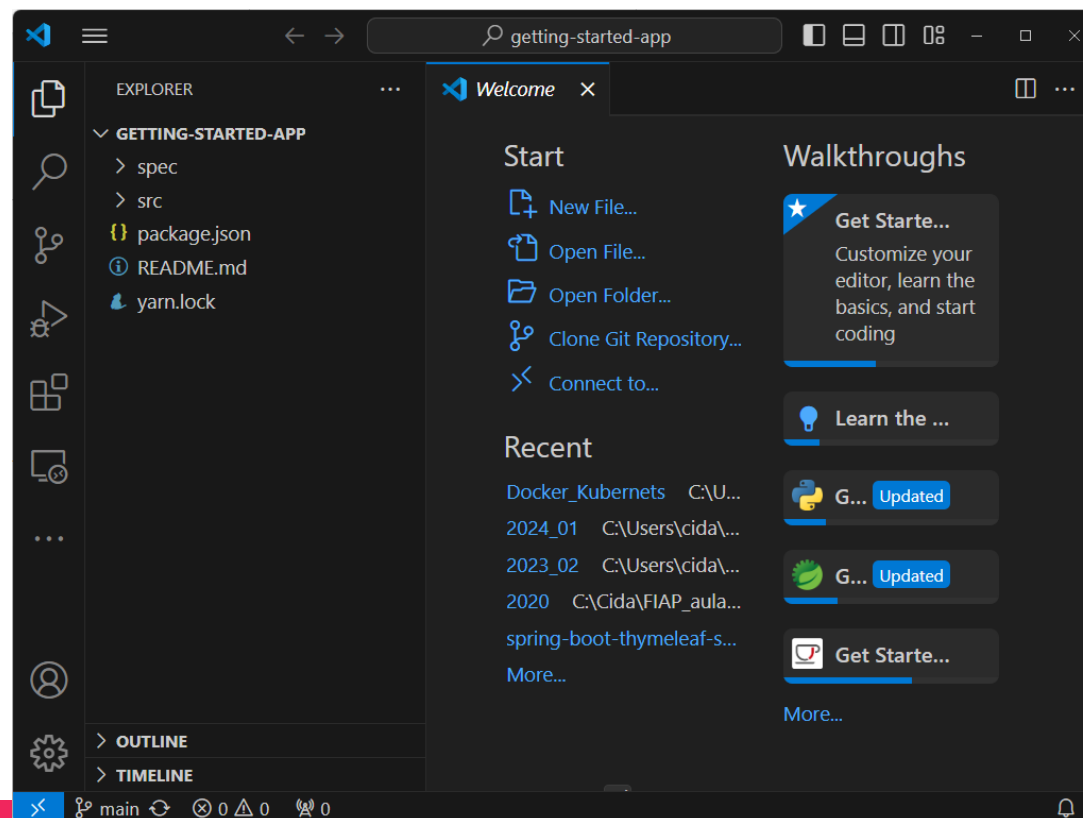
Diretório: C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app

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

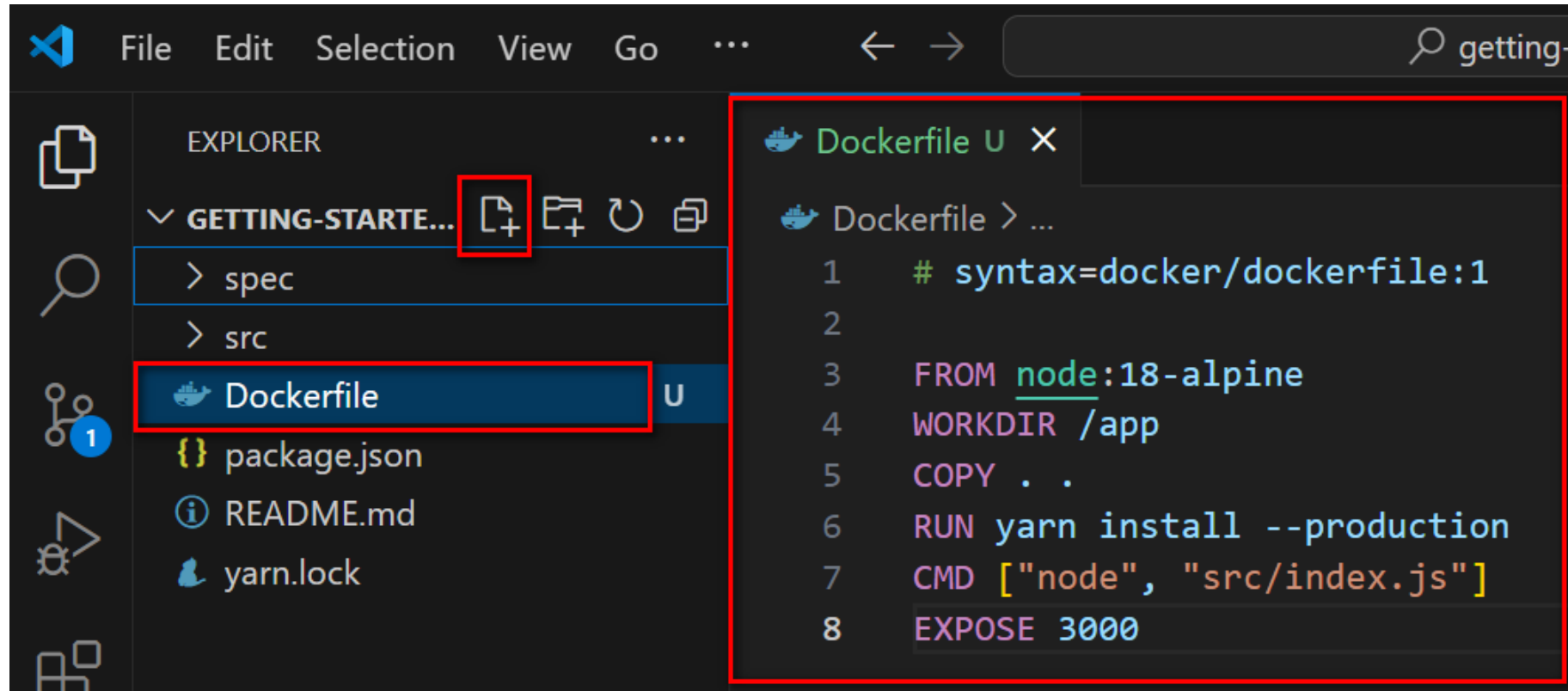
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_25_Docker_App_exemplo\exemplo\getting-started-app> |
```


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> |
```



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

```
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:

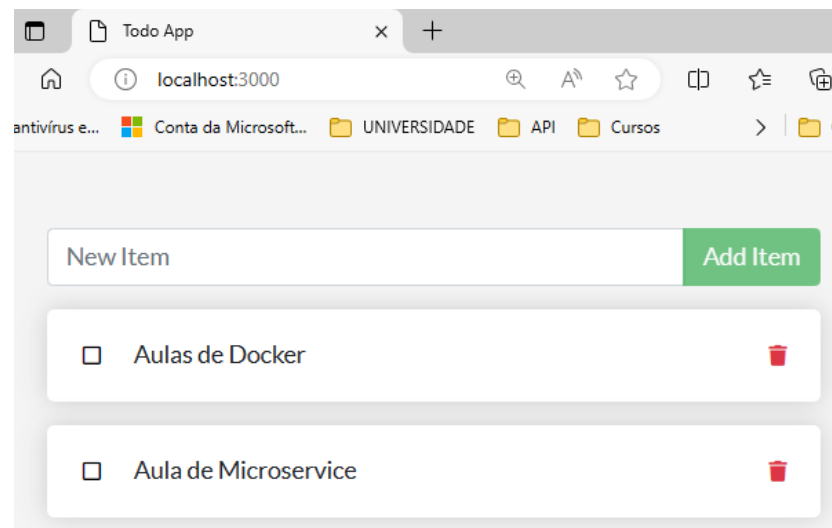
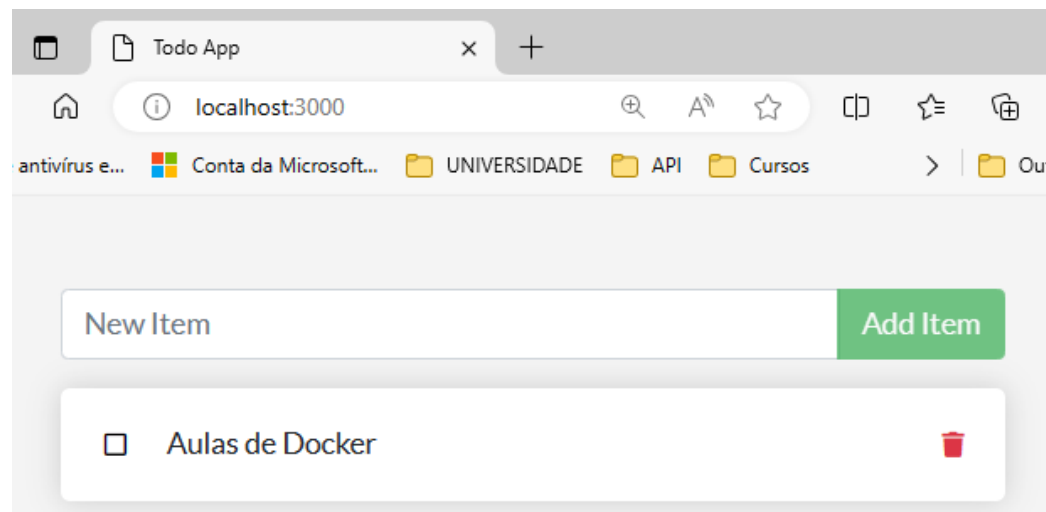
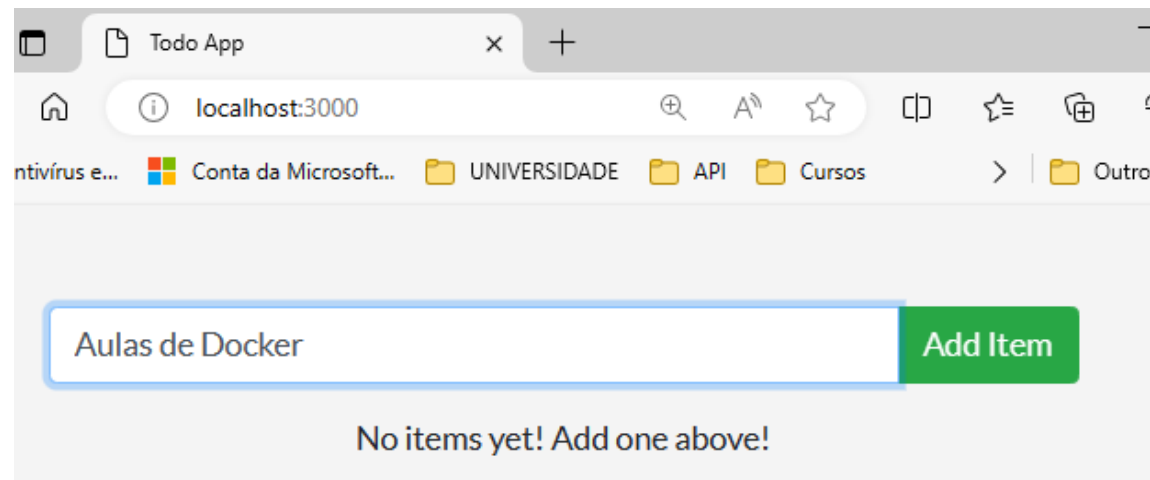
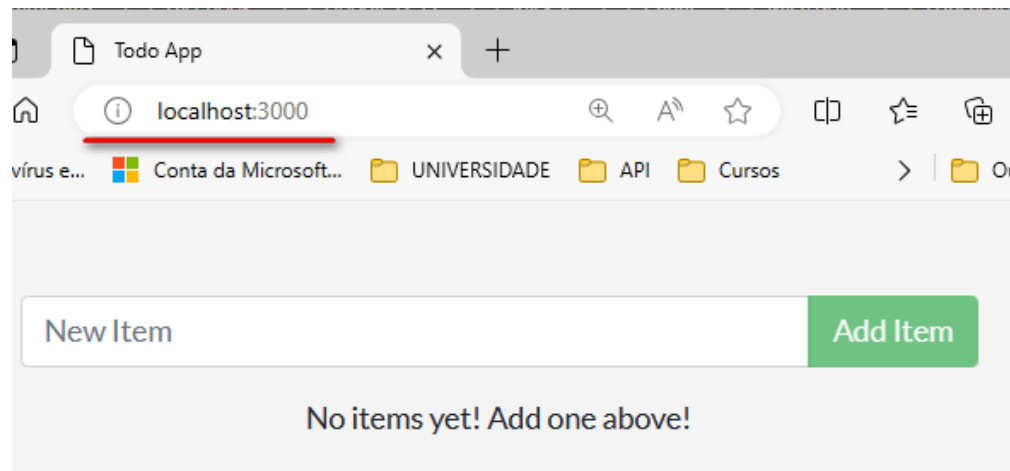
```
$ docker run -dp 3000:3000 getting-started:latest
```

```
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 26 – Docker – parte 4 – Dockerfile produto-mvc

Testar no navegador



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> |
```

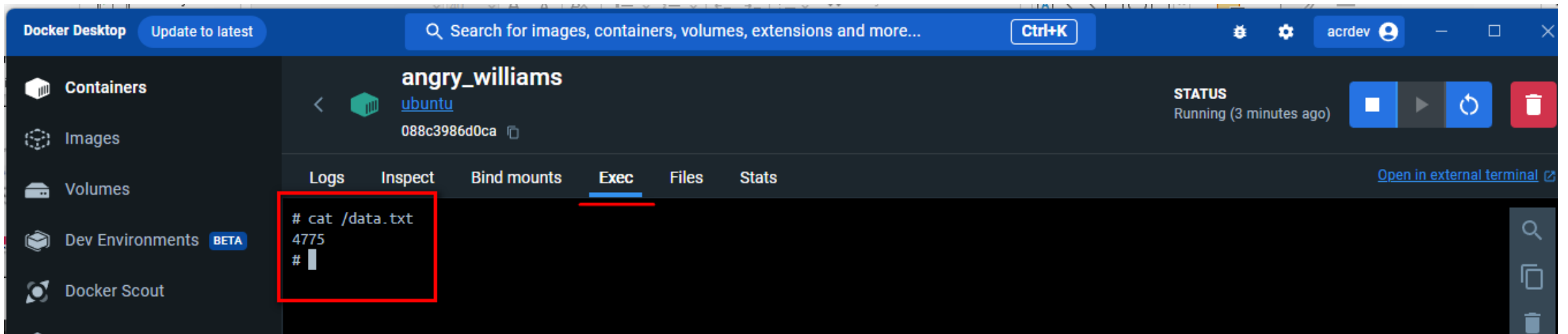
Executando o *container*

```
$ 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. One container, 'angry_williams' (ubuntu), is in a 'Running' state. A context menu is open for this container, with 'Open in terminal' highlighted. Red arrows point to the menu and the selected option.

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
suspicious_mendel	acrdev/getting-started:late	Exited	0%	3000:3000	1 hour ago	[Actions]
angry_williams	ubuntu	Running	0%		2 minutes ago	[Actions]



I 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>

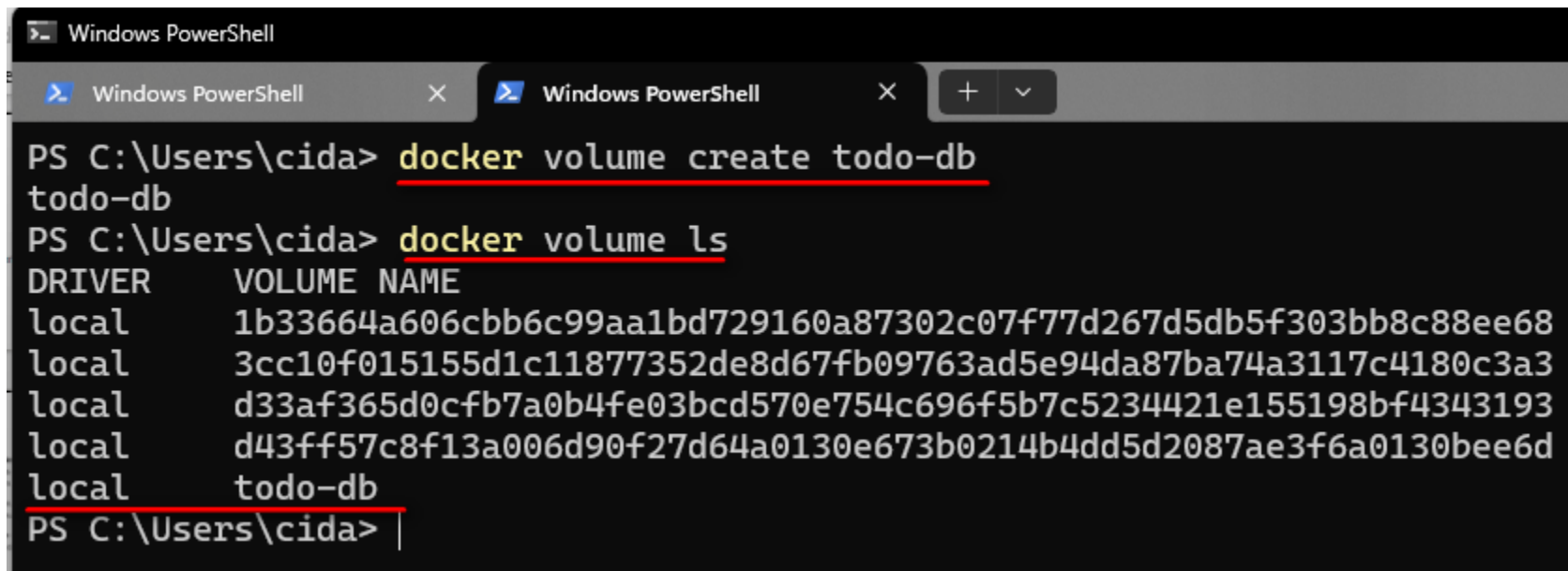
# Detalhes do volume
$ docker volume inspect <name>

# 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.

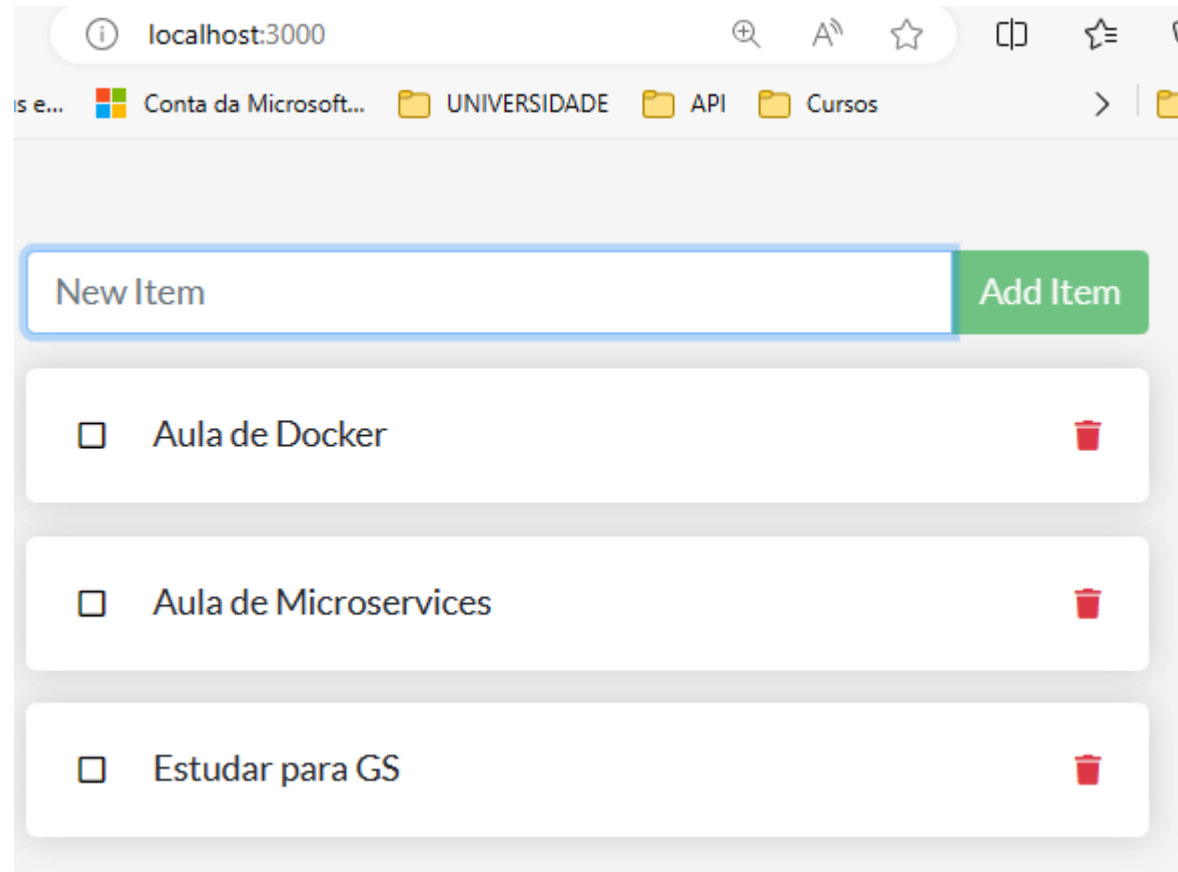
`target` - onde, no **container**, será montado esse **volume**.

```
Windows PowerShell
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

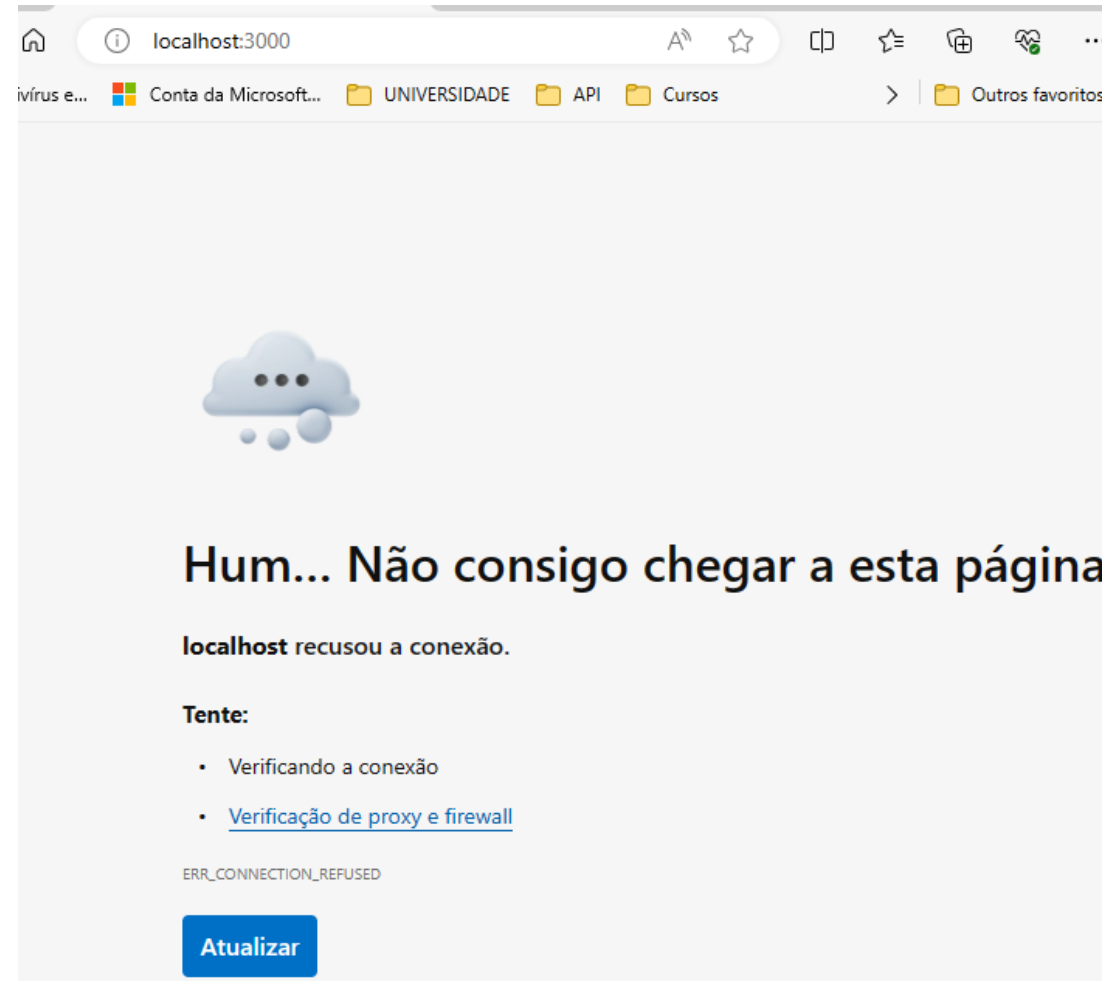
1 - Assim que o **container** for iniciado, vamos abrir o aplicativo (no navegador) e adicionar alguns itens à lista de tarefas.



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.

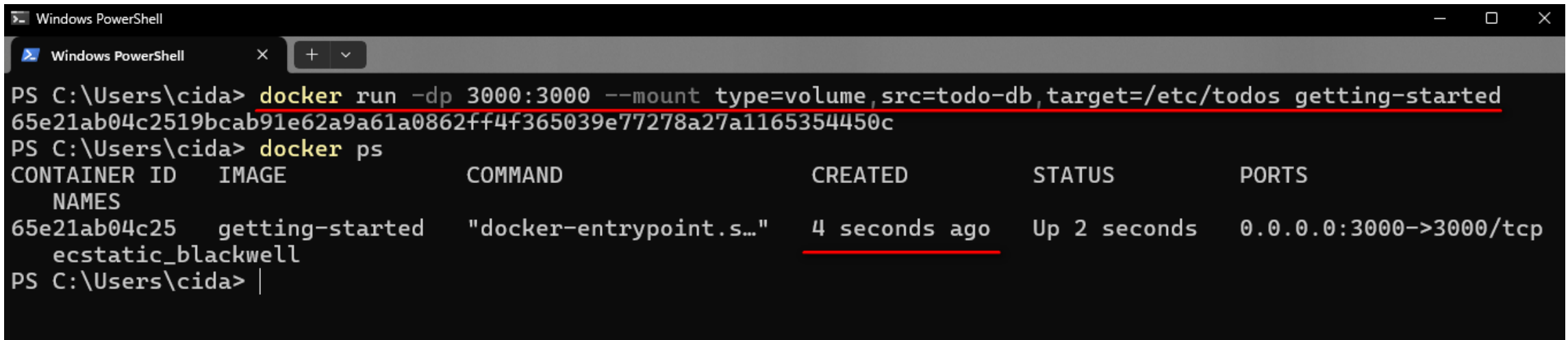
```
Windows PowerShell
PS C:\Users\cida> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
27e56aba53a1   getting-started "docker-entrypoint.s..." 6 minutes ago Up 6 minutes  0.0.0.0:3000->3000/tcp
great_mclaren
PS C:\Users\cida> docker rm -f 27e56aba53a1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

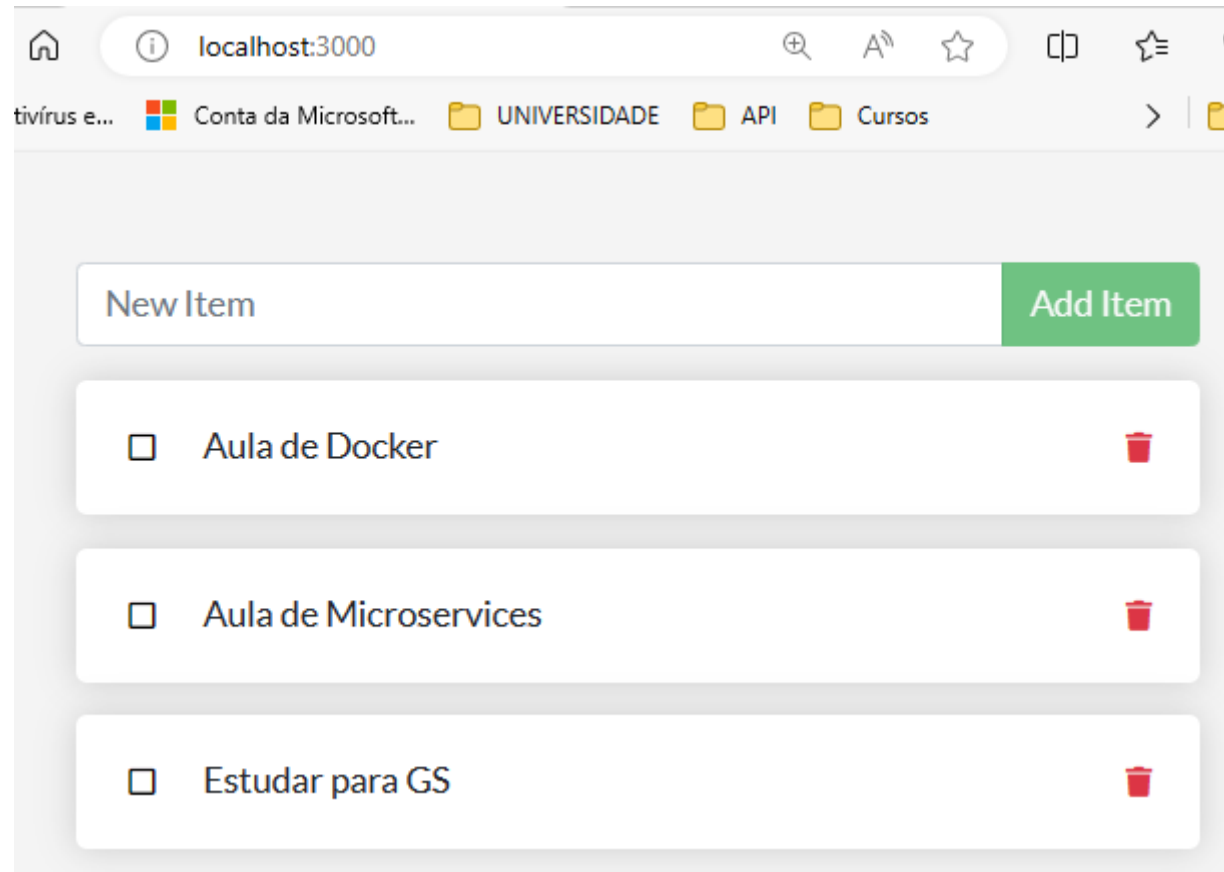
```
$ docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
```



```
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
NAMES
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**.

Inspeccionando o Volume

Inspeccionando 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

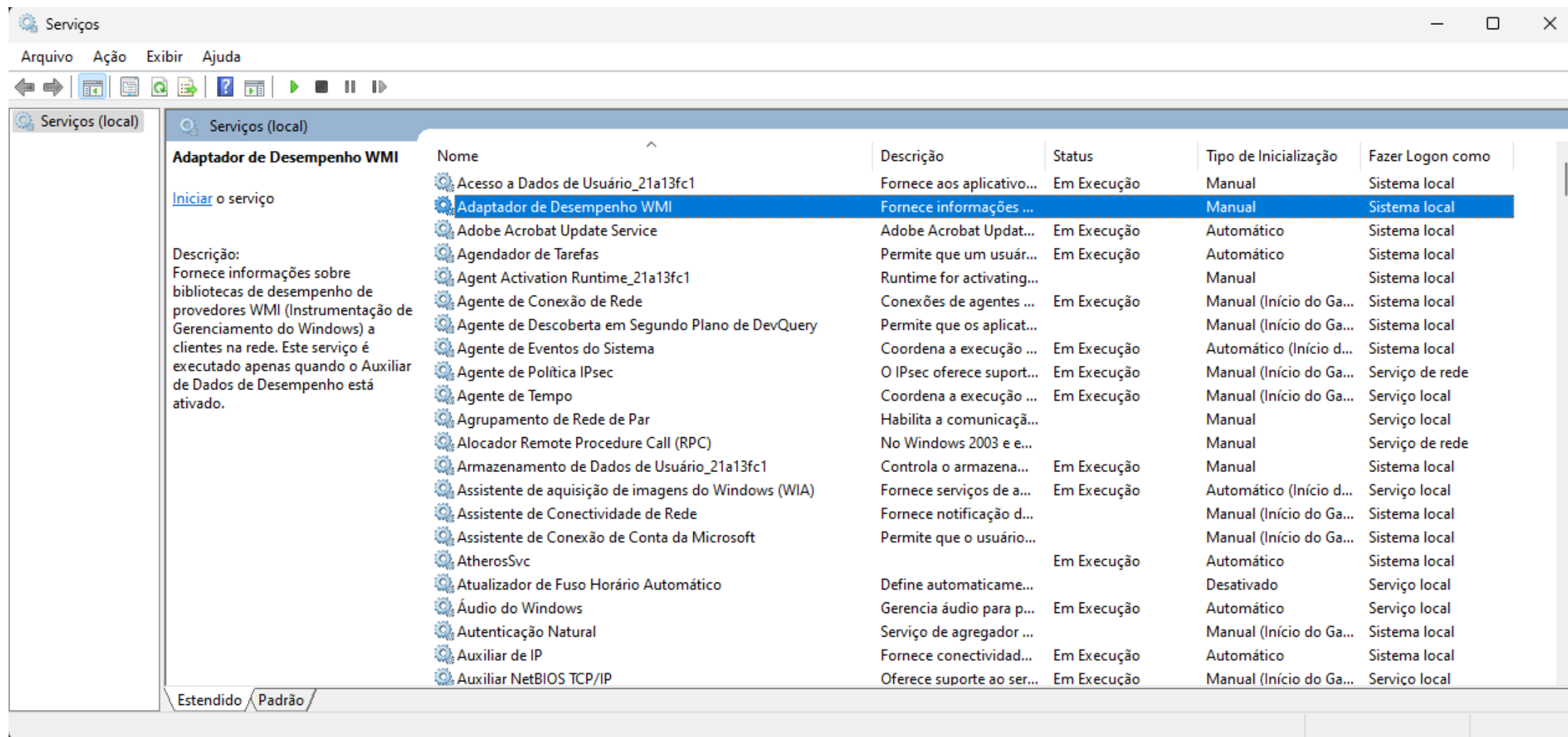
Manage volumes

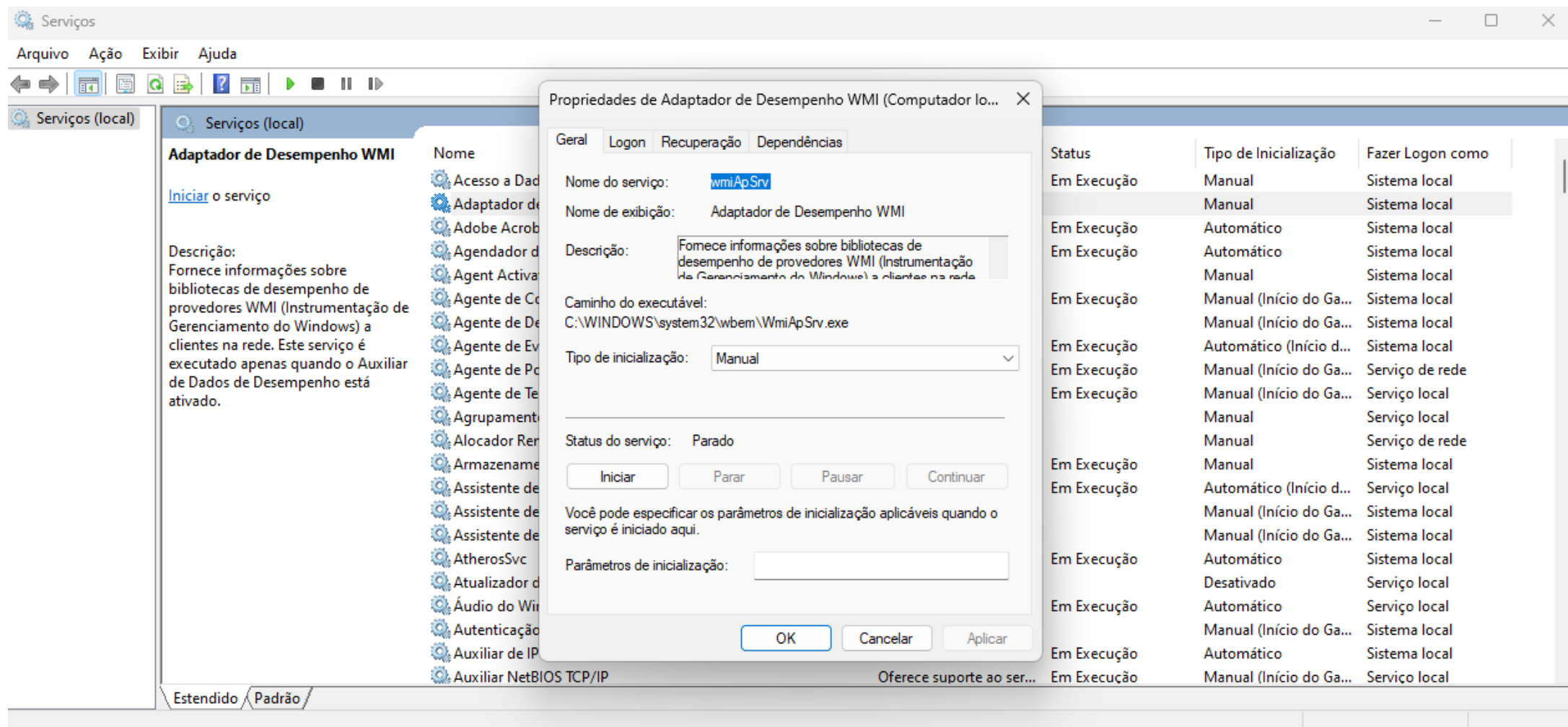
Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove unused local volumes
  rm          Remove one or more volumes

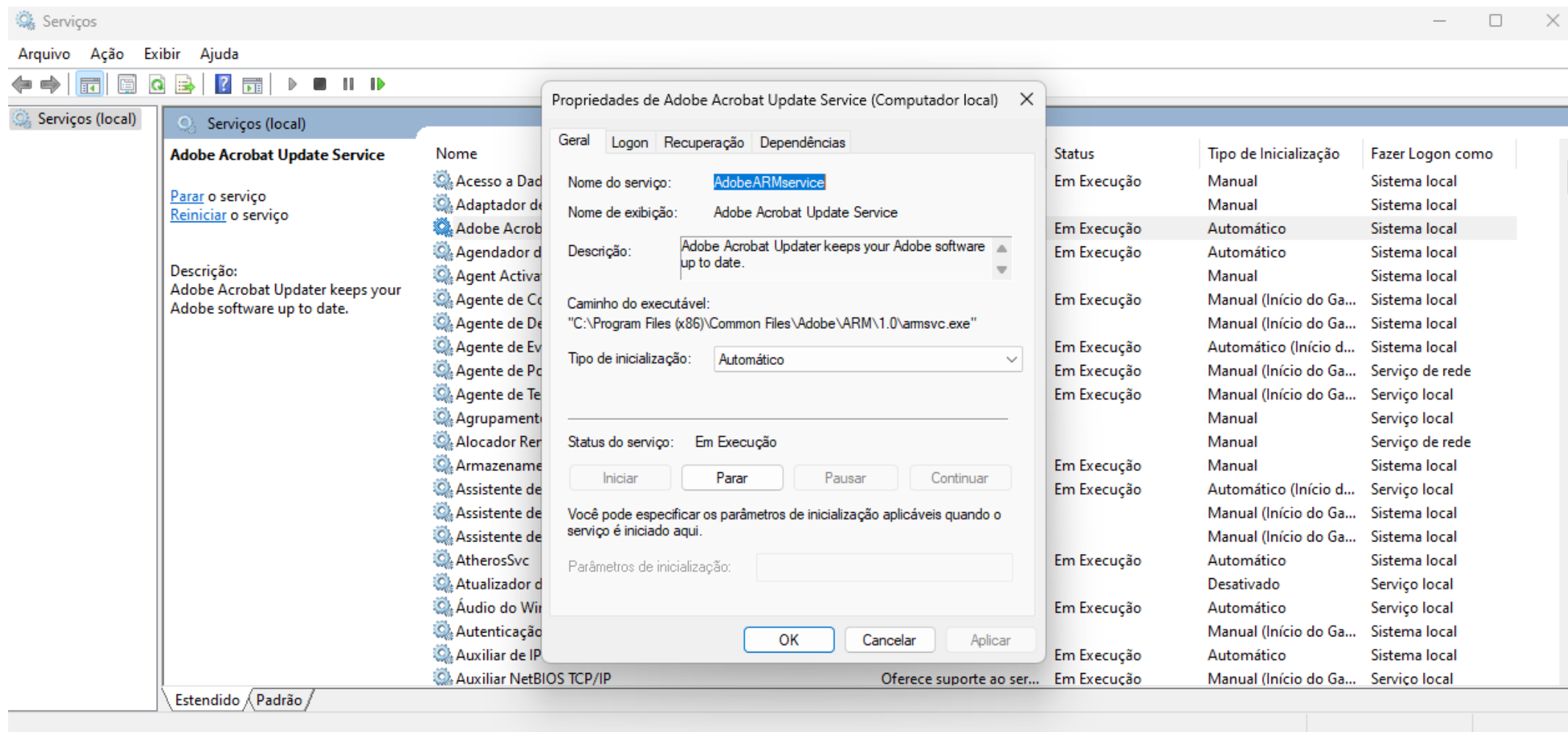
Run 'docker volume COMMAND --help' for more information on a command.
```

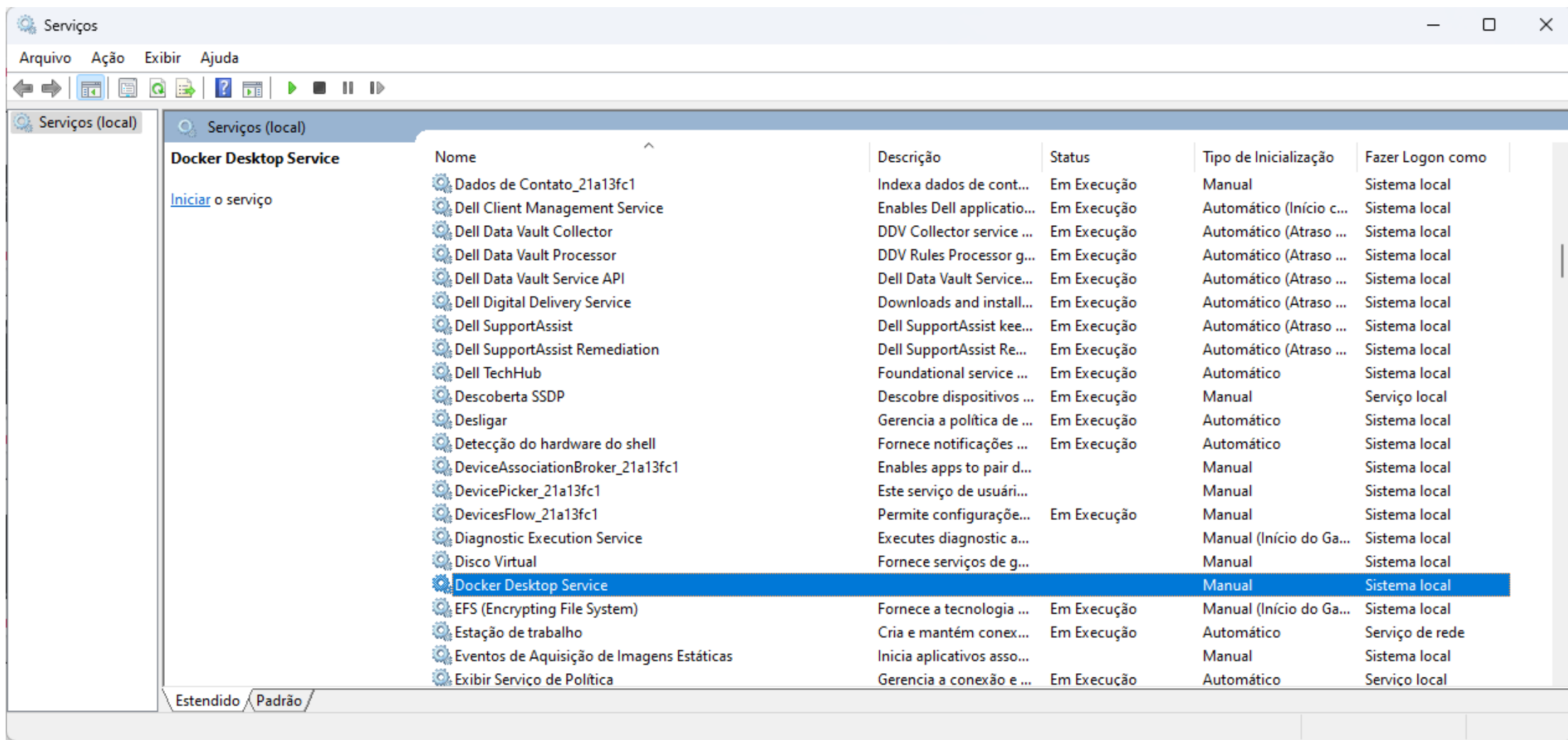
Serviços

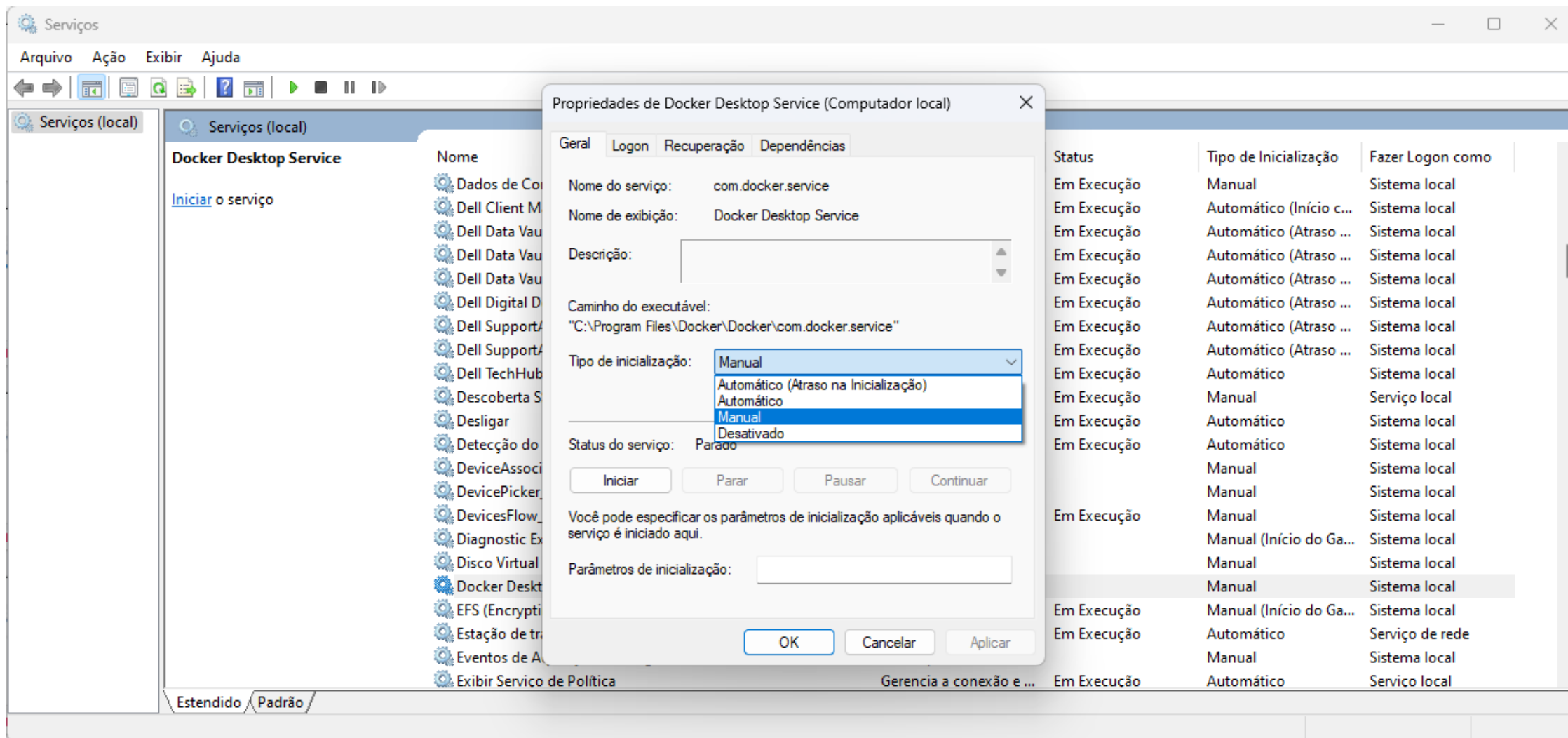
Windows







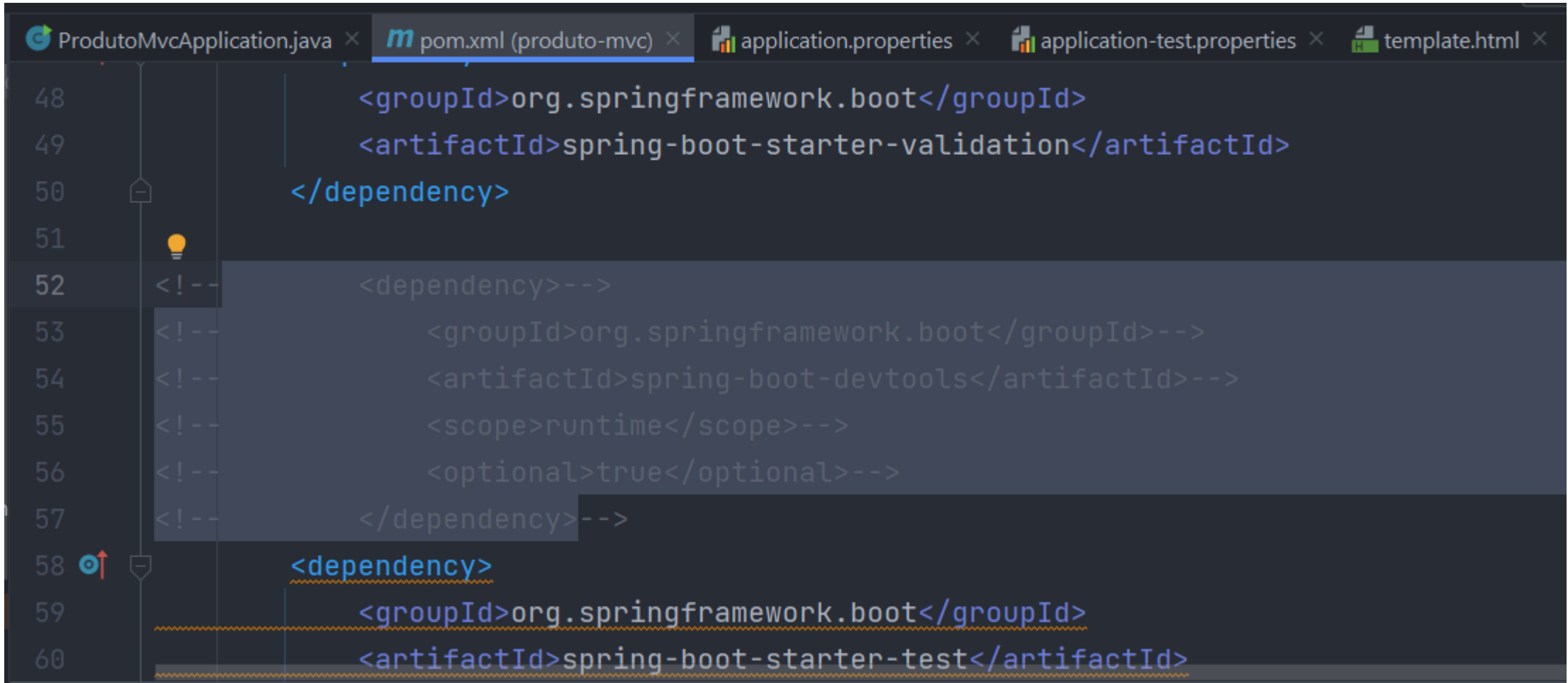




Dockerizando projeto produto-mvc

- Baixar projeto produto-mvc – aula 23

Alterar pom.xml



```
48     <groupId>org.springframework.boot</groupId>
49     <artifactId>spring-boot-starter-validation</artifactId>
50 </dependency>
51
52 <!-- <dependency>-->
53 <!-- <groupId>org.springframework.boot</groupId>-->
54 <!-- <artifactId>spring-boot-devtools</artifactId>-->
55 <!-- <scope>runtime</scope>-->
56 <!-- <optional>true</optional>-->
57 <!-- </dependency>-->
58 <dependency>
59     <groupId>org.springframework.boot</groupId>
60     <artifactId>spring-boot-starter-test</artifactId>
```

Dockerfile

```
Dockerfile x
1  # Define your base image
2  >> FROM openjdk:17-jdk-alpine
3
4  # Define the working directory by using the WORKDIR instruction.
5  WORKDIR /app
6
7  # Copy both the Maven wrapper script and your project's pom.xml file
8  # into the current working directory /app within the Docker container.
9  COPY .mvn/ .mvn
10 COPY mvnw pom.xml ./
11
12 # Execute a command within the container.
13 # Download all dependencies for your project without building the final JAR file.
14 RUN ./mvnw dependency:go-offline
15
16 # Copy the src directory from your project on the host machine
17 # to the /app directory within the container.
18 COPY src ./src
19
20 EXPOSE 8080
21
22 # Set the default command to be executed when the container starts.
23 CMD ["/mvnw", "spring-boot:run"]
```

<https://docs.docker.com/guides/docker-concepts/building-images/multi-stage-builds/>

```
Windows PowerShell
Windows PowerShell x + v
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
acrdev/getting-started latest      2255108c0459    3 days ago     219MB
getting-started      2.0         2255108c0459    3 days ago     219MB
getting-started      latest      2255108c0459    3 days ago     219MB
getting-started      1.0        1aff6656102e    3 days ago     219MB
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26>
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> ls
```

```
Diretório: C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\
produto-mvc_aula_26
```

Mode	LastWriteTime	Length	Name
d----	14/05/2024 11:47		.idea
d----	14/05/2024 11:44		.mvn
d----	14/05/2024 11:44		src
d----	14/05/2024 11:44		target
-a----	11/02/2024 15:22	395	.gitignore
-a----	14/05/2024 14:25	819	<u>Dockerfile</u>
-a----	11/02/2024 15:22	1396	HELP.md
-a----	11/02/2024 15:22	11290	mvnw
-a----	11/02/2024 15:22	7592	mvnw.cmd
-a----	14/05/2024 11:50	2826	pom.xml

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> |
```

Gerar a *imagem* do projeto produto-mvc

```
$ docker build -t produtomvc .
```

```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker build -t produtomvc .
[+] Building 49.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 858B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine        2.8s
=> [auth] library/openjdk:pull token for registry-1.docker.io                 0.0s
=> [1/6] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd97 0.0s
=> => resolve docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd97 0.0s
=> [internal] load build context                                                0.3s
=> => transferring context: 8.90MB                                              0.3s
=> CACHED [2/6] WORKDIR /app                                                    0.0s
=> CACHED [3/6] COPY .mvn/ .mvn                                                 0.0s
=> [4/6] COPY mvnw pom.xml ./                                                  0.1s
=> [5/6] RUN ./mvnw dependency:go-offline                                       44.8s
=> [6/6] COPY src ./src                                                        0.1s
=> exporting to image                                                           0.9s
=> => exporting layers                                                         0.9s
=> => writing image sha256:3e8e5368a616fd9e68a5e1220b91b39538c3428369b7ee812b624c3635adc4ea 0.0s
=> => naming to docker.io/library/produtomvc                                  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_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> |
```



```
Windows PowerShell
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
produtomvc           latest             3e8e5368a616       9 minutes ago      487MB
acrdev/getting-started latest            2255108c0459       3 days ago         219MB
getting-started      2.0                2255108c0459       3 days ago         219MB
getting-started      latest            2255108c0459       3 days ago         219MB
getting-started      1.0               1aff6656102e       3 days ago         219MB
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> |
```

Construir o *container* da *imagem*

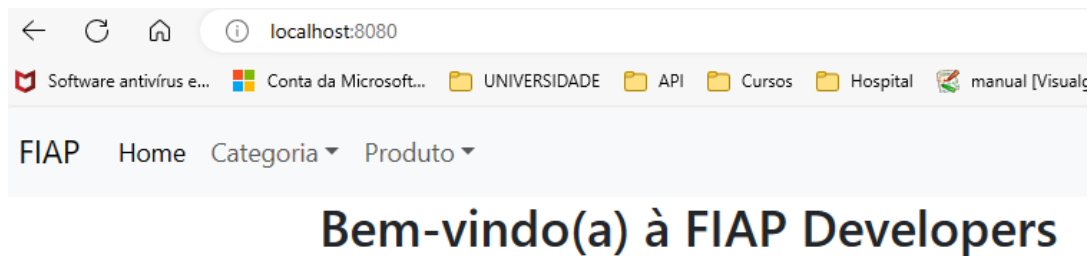
```
$ docker run -dp 8080:8080 produtomvc:latest
```

Listar *containers*

```
$ docker ps
```

```
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26>
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker run -dp 8080:8080 produtomvc:latest
d6ca339a0bd517ea6bb567d653626bb360e9540dcb01352c477373855bcadd37
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS          PORTS                               NAMES
d6ca339a0bd5   produtomvc:latest  "/.mvnw spring-boot:..."  4 seconds ago  Up 4 seconds    0.0.0.0:8080->8080/tcp             vigilant_mclean
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> |
```

Testar aplicação



Cadastro de Categoria

Nome:

Teste

Salvar

Lista de Produtos

Nome	Descrição	Categoria	Lojas	Valor	Ações	
Mouse Microsoft	Mouse sem fio	Mouse	Americanas , Magazine Luiza	250.0	Excluir	Editar
Smartphone Samsung Galaxy A54 5G	Samsung Galaxy A54 5G	Smartphone	Americanas , Submarino	1799.0	Excluir	Editar
Smart TV	Smart TV LG LED 65 polegadas	Smart TV	Fast Shop	3999.0	Excluir	Editar
Teclado Microsof	Teclado sem fio	Teclado	Fast Shop	278.5	Excluir	Editar
Apple iPhone 15	Apple iPhone 15, 128G, Preto	Smartphone	Fast Shop	4999.0	Excluir	Editar

Lista de Categorias

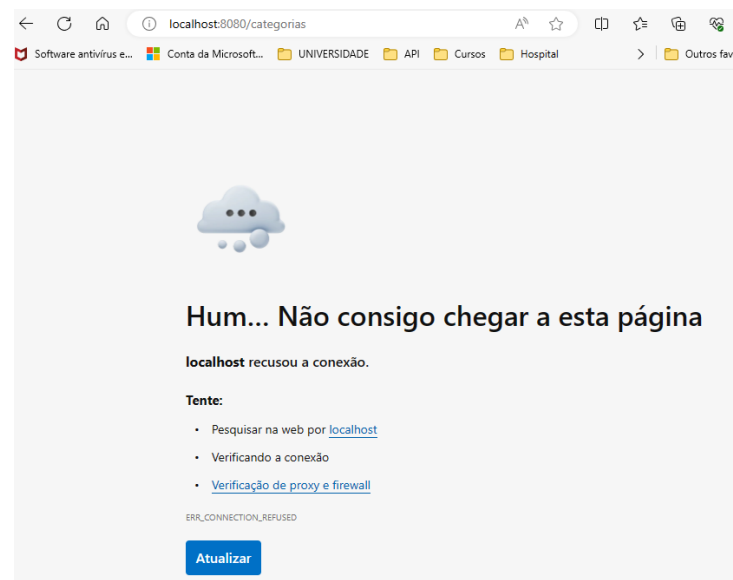
Nome	Ações	
Smartphone	Excluir	Editar
Smart TV	Excluir	Editar
Notebook	Excluir	Editar
Tablet	Excluir	Editar
Mouse	Excluir	Editar
Teclado	Excluir	Editar
Teste	Excluir	Editar

Testar aplicação

Vamos parar o *container*

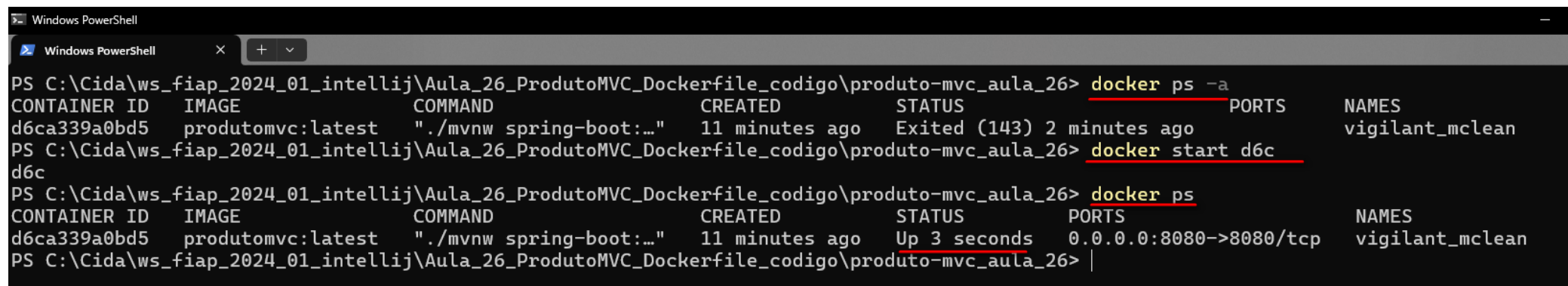
```
$ docker stop d6c
```

Testar a aplicação.



Vamos iniciar o **container** e testar a aplicação.

```
# lista todos os containers
docker ps -a
# inicia o container
docker start d6c
#lista os containers em execução
docker ps
```



```
Windows PowerShell
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
d6ca339a0bd5   produtomvc:latest  "./mvnw spring-boot:..."  11 minutes ago  Exited (143)  2 minutes ago          vigilant_mclean
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker start d6c
d6c
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
d6ca339a0bd5   produtomvc:latest  "./mvnw spring-boot:..."  11 minutes ago  Up 3 seconds  0.0.0.0:8080->8080/tcp  vigilant_mclean
PS C:\Cida\ws_fiap_2024_01_intellij\Aula_26_ProdutoMVC_Dockerfile_codigo\produto-mvc_aula_26> |
```

Testar aplicação

FIAP Home Categoria ▾ Produto ▾

Lista de Categorias

Nome	Ações
Smartphone	Excluir Editar
Smart TV	Excluir Editar
Notebook	Excluir Editar
Tablet	Excluir Editar
Mouse	Excluir Editar
Teclado	Excluir Editar



Copyright © 2024
Prof^a. 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).