Relatório 17 - Docker e Containers para Aplicações

Guilherme Loan Schneider

Descrição da atividade

Nas aulas do atual card, foi demonstrado a utilização do Docker com o CLI (Command Line Interface), permitindo compreender os comandos que podem ser feitos quanto a containers, imagens, volumes, dentre outros. Além disso, o autor explica cada tipo de objeto e seu funcionamento.

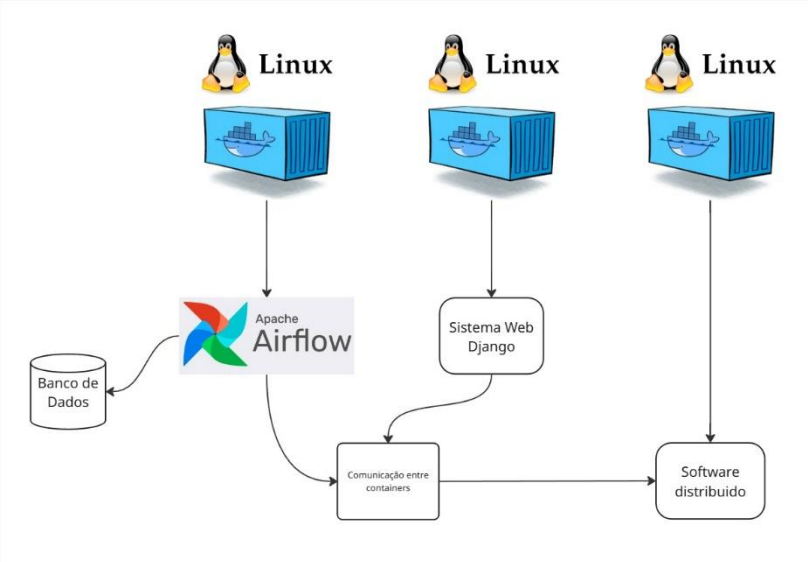
Layout de comandos a serem utilizados no CLI

```
docker {objeto} {ação}

docker {container | image | network | volume | etc } {ls | inspect | rm | create}

$ docker container rm alpine

$ docker  image ls
```

Utilização básica de Containers

De forma simples, um container no Docker funciona como se fosse um ambiente pronto para implementação, onde lá já existirão alguns arquivos básicos para o que o usuário deseja fazer, como um container com um ambiente Linux Ubuntu.

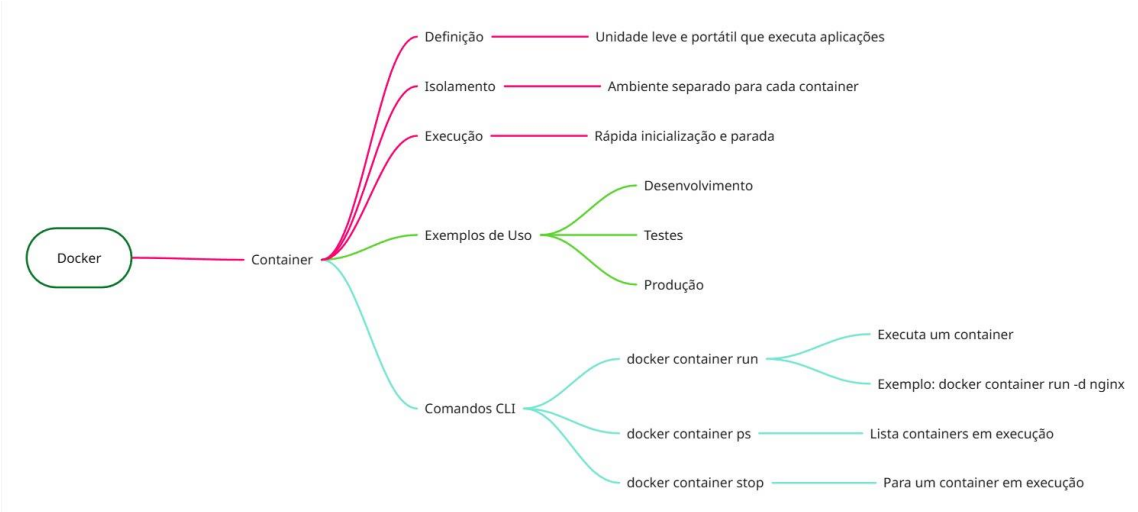
No entanto, utilizar um container apenas para rodar um ambiente Linux não é muito interessante, mas sim utilizá-lo para disponibilizar um serviço do Apache, features, arquivos, dentre outros, localmente ou até mesmo na internet.



Para criar um container com um ambiente linux, inicia-lo e entrar no shell desse container, utilizamos o seguinte comando no CLI:

docker container run -it –name container_teste ubuntu sh

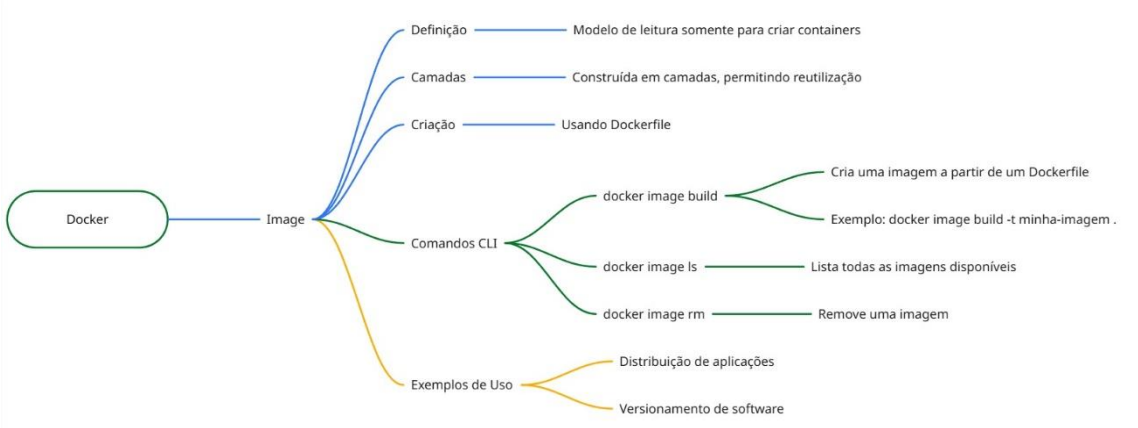
- O parâmetro -it é um conjunto de comandos, onde o -it inicia em modo interativo e já faz o “attach” no container, permitindo utilizar o shell.
- O parâmetro –name é apenas para rotular o nome do container. Caso não seja passado esse parâmetro o próprio Docker gera um nome para ele.
- Em seguida, passamos o ambiente ou serviço a ser instalado, como o Ubuntu ou NGINX.
- Por fim, o “sh” define como iremos utilizar o container, nesse caso, o próprio shell no terminal.



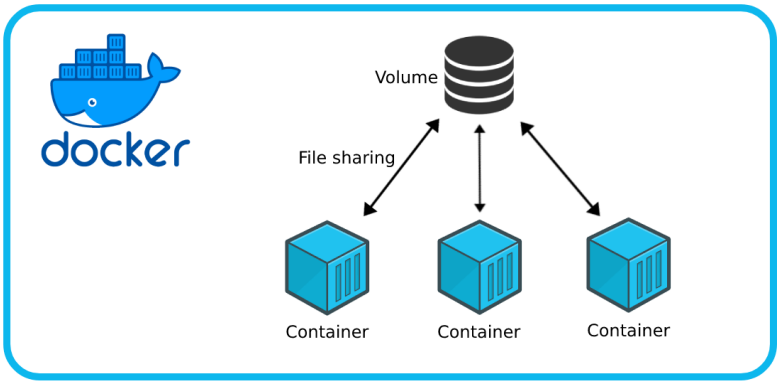
Imagens no Docker

As imagens são estados de um container em um determinado período, onde são salvos todas as configurações, arquivos, logs, dentre outros. Elas são utilizadas quando deseja-se salvar uma determinada configuração a fim de ser replicada em outros computadores, não apenas isso, pode ser utilizada para versionamento de aplicativos.

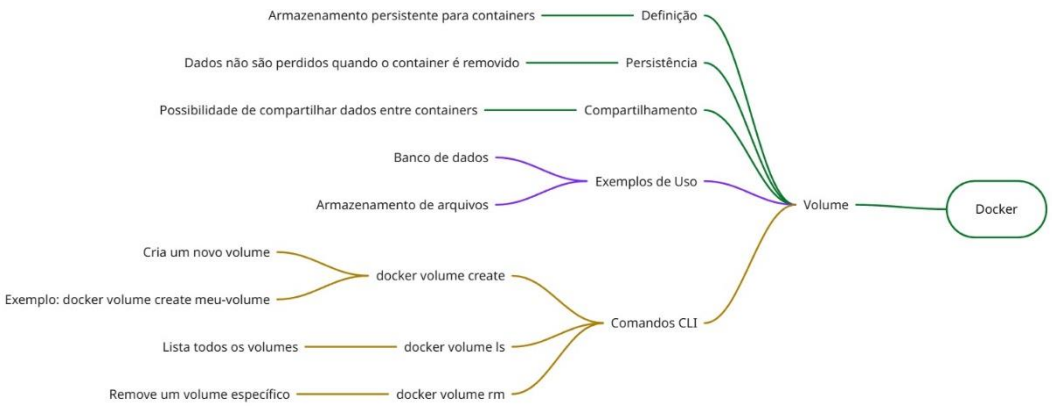
Uma forma de fazer a criação dessas imagens é utilizando o um arquivo Dockerfile montado a partir do editor de texto “vim”.



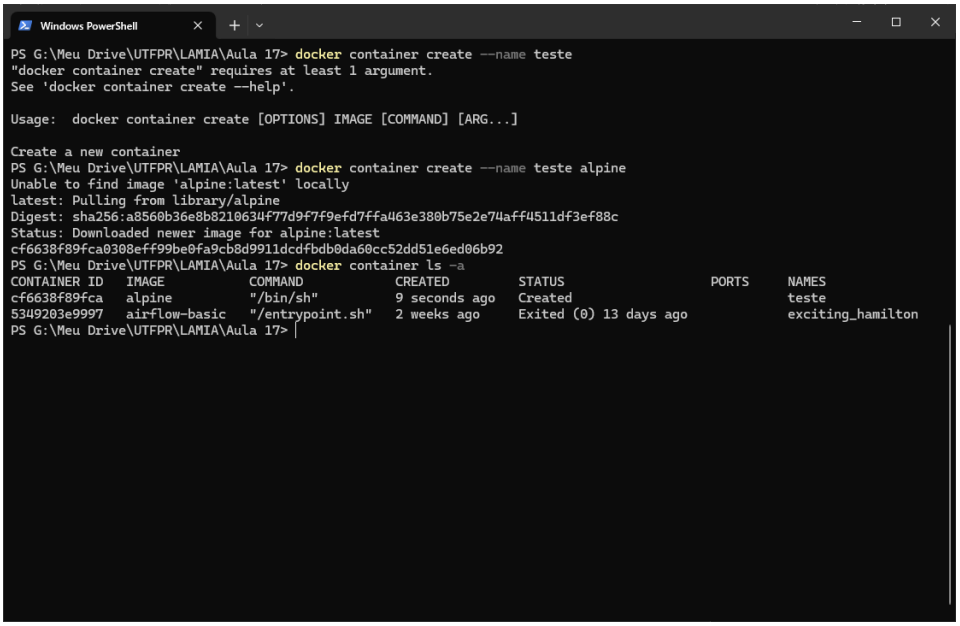
Volumes no Docker

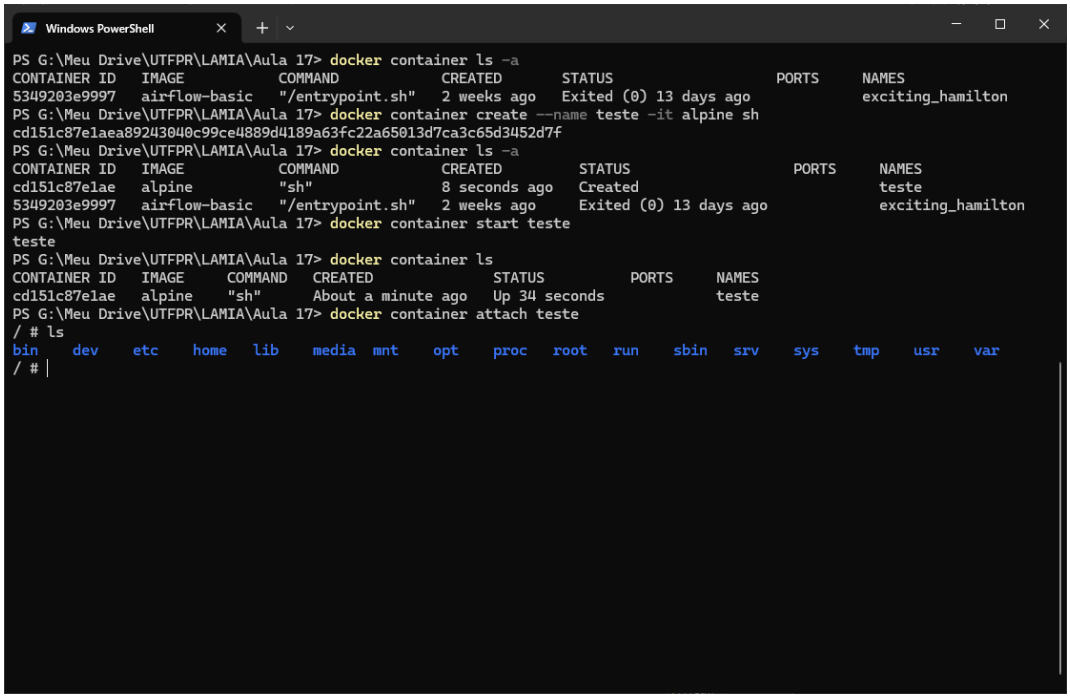


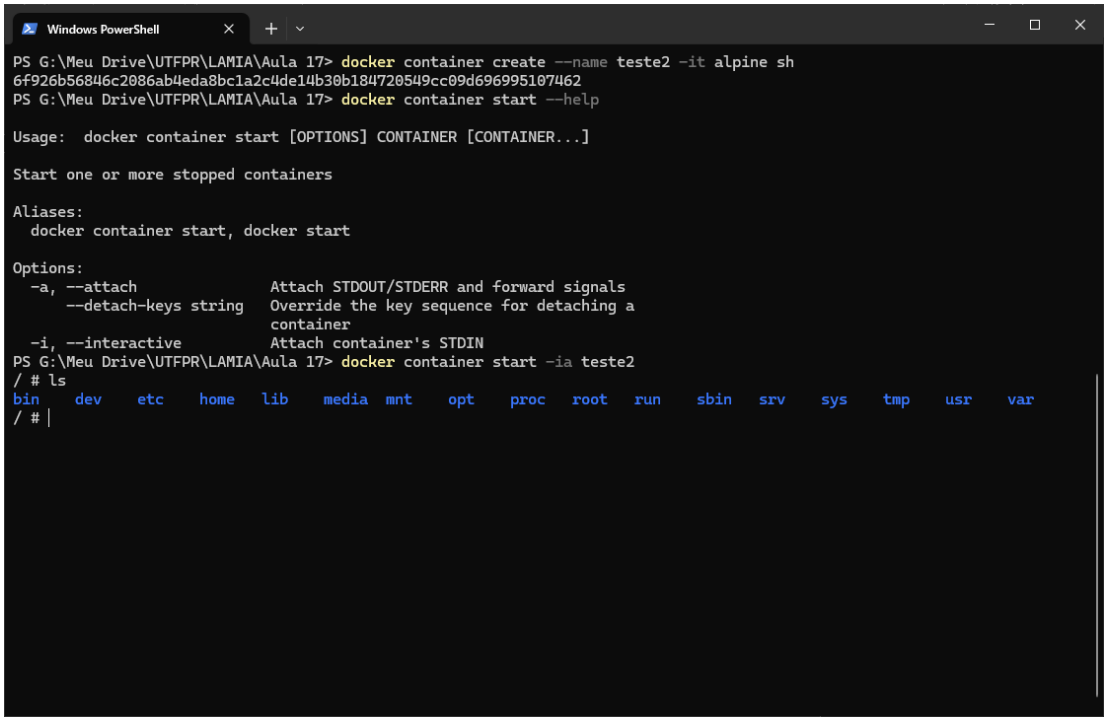
Os volumes no Docker funcionam em um sistema de compartilhamento, de forma sincronizada, de arquivos, onde é mapeado um diretório de uma Máquina A, e esse diretório é adicionado ao container na etapa de criação. É importante salientar a liquidez desses dados, onde caso um arquivo seja removido via CLI ou por algum outro meio, o arquivo é removido tanto no container, quanto na Máquina A. Para que isso não ocorra, o mapeamento normalmente é definido apenas em modo leitura.

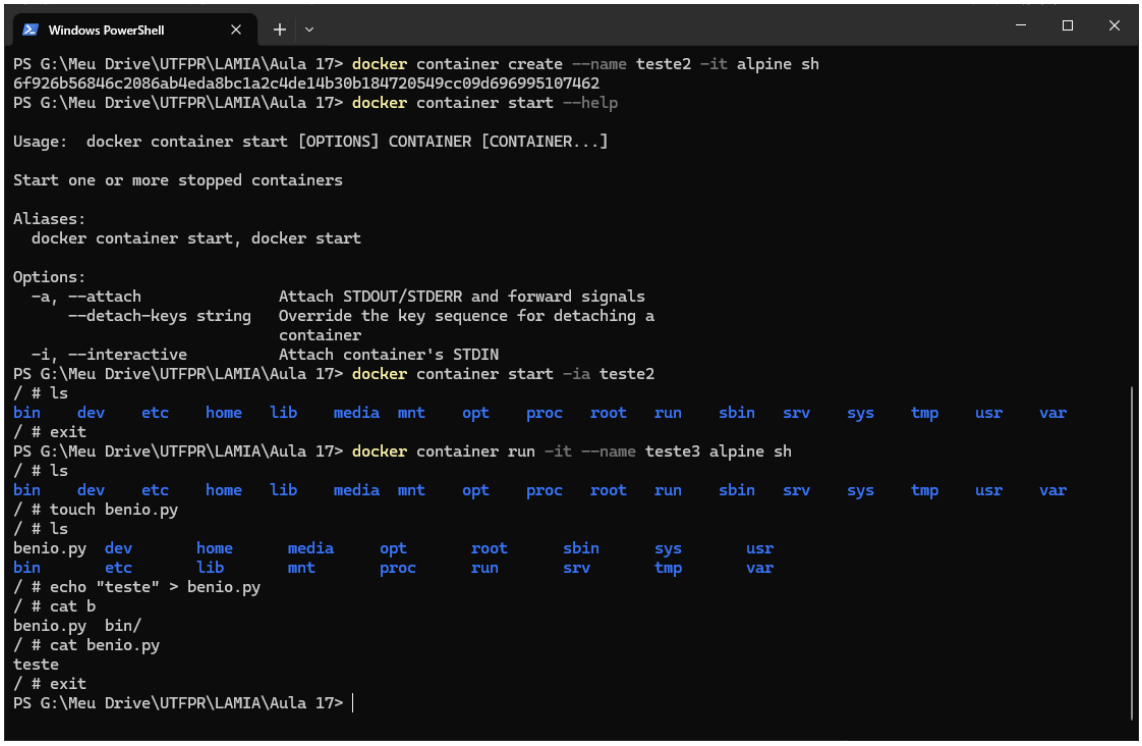


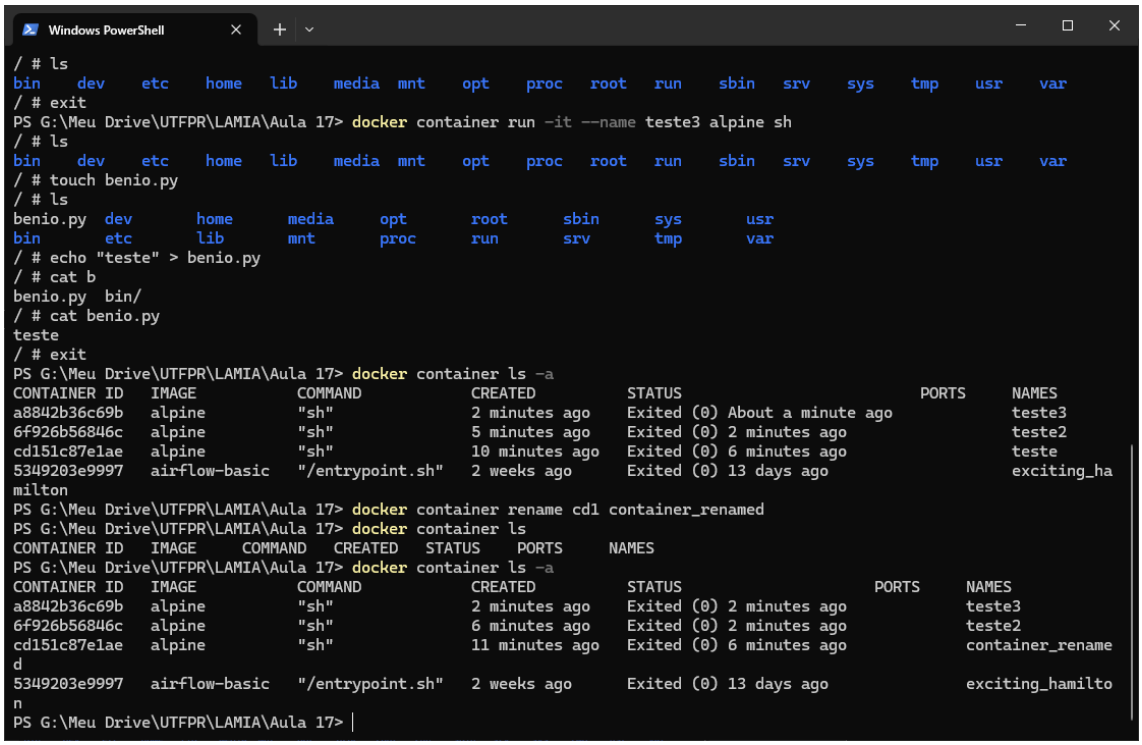
Capturas de Tela dos comandos realizados em aula

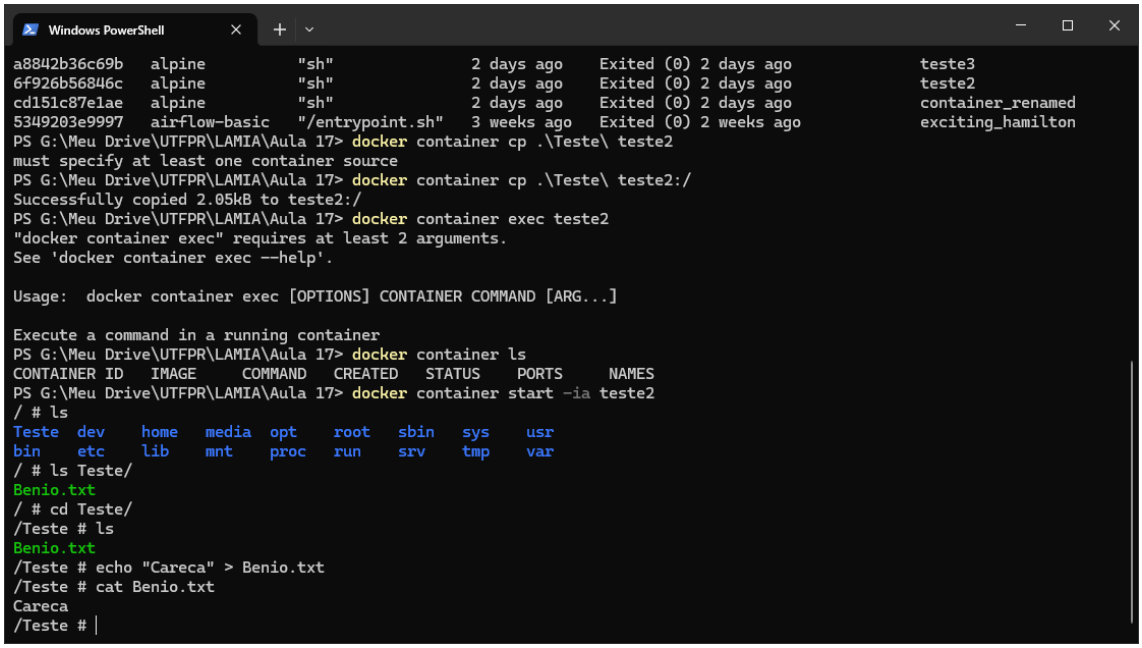


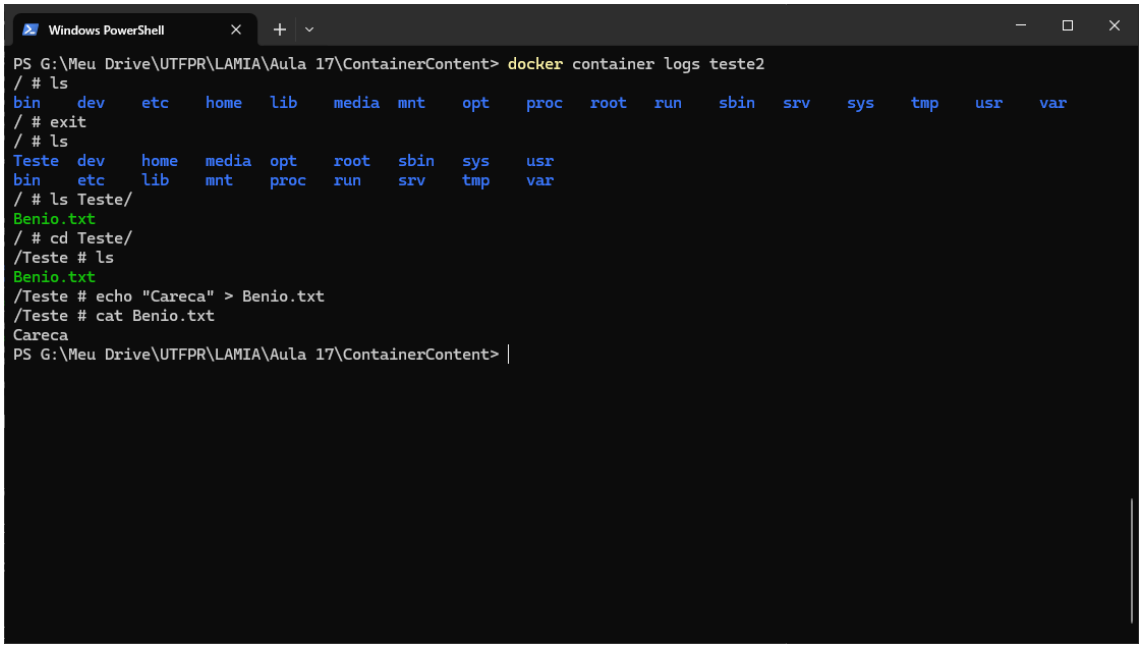

Windows PowerShell
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5349203e9997 airflow-basic "/entrypoint.sh" 2 weeks ago Exited (0) 13 days ago exciting_hamilton
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container create --name teste -it alpine sh
cd151c87e1aea89243040c99ce4889d4189a63fc22a65013d7ca3c65d3452d7f
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cd151c87e1ae alpine "sh" 8 seconds ago Created teste
5349203e9997 airflow-basic "/entrypoint.sh" 2 weeks ago Exited (0) 13 days ago exciting_hamilton
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start teste
teste
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cd151c87e1ae alpine "sh" About a minute ago Up 34 seconds teste
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container attach teste
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ #


Windows PowerShell
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container create --name teste2 -it alpine sh
6f926b56846c2086ab4eda8bc1a2c4de14b30b184720549cc09d696995107462
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start --help
Usage: docker container start [OPTIONS] CONTAINER [CONTAINER...]
Start one or more stopped containers
Aliases:
docker container start, docker start
Options:
-a, --attach Attach STDOUT/STDERR and forward signals
--detach-keys string Override the key sequence for detaching a container
-i, --interactive Attach container's STDIN
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start -ia teste2
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ #


Windows PowerShell
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container create --name teste2 -it alpine sh
6f926b56846c2086ab4eda8bc1a2c4de14b30b184720549cc09d696995107462
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start --help
Usage: docker container start [OPTIONS] CONTAINER [CONTAINER...]
Start one or more stopped containers
Aliases:
docker container start, docker start
Options:
-a, --attach Attach STDOUT/STDERR and forward signals
--detach-keys string Override the key sequence for detaching a container
-i, --interactive Attach container's STDIN
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start -ia teste2
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ # exit
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container run -it --name teste3 alpine sh
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ # touch benio.py
/ # ls
benio.py dev home media opt root sbin sys usr
bin etc lib mnt proc run srv tmp var
/ # echo "teste" > benio.py
/ # cat b
benio.py bin/
/ # cat benio.py
teste
/ # exit
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17>

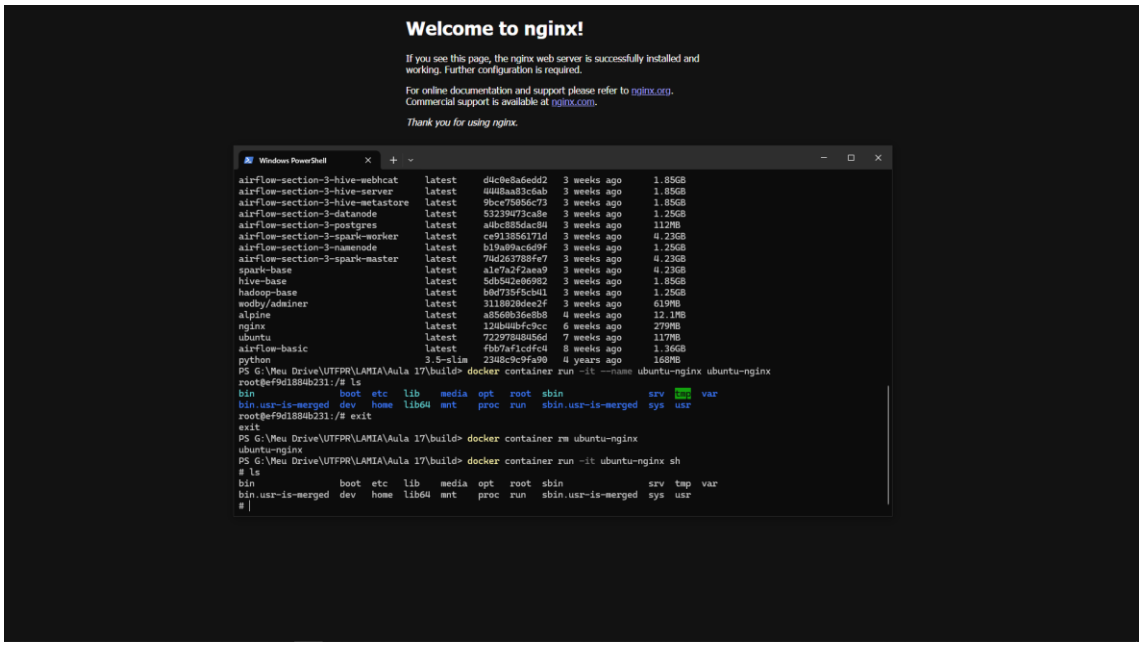

Windows PowerShell
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ # exit
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container run -it --name teste3 alpine sh
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ # touch benio.py
/ # ls
benio.py dev home media opt root sbin sys usr
bin etc lib mnt proc run srv tmp var
/ # echo "teste" > benio.py
/ # cat b
benio.py bin/
/ # cat benio.py
teste
/ # exit
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a8842b36c69b alpine "sh" 2 minutes ago Exited (0) About a minute ago teste3
6f926b56846c alpine "sh" 5 minutes ago Exited (0) 2 minutes ago teste2
cd151c87e1ae alpine "sh" 10 minutes ago Exited (0) 6 minutes ago teste
5349203e9997 airflow-basic "/entrypoint.sh" 2 weeks ago Exited (0) 13 days ago exciting_hamilton
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container rename cd1 container_renamed
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a8842b36c69b alpine "sh" 2 minutes ago Exited (0) 2 minutes ago teste3
6f926b56846c alpine "sh" 6 minutes ago Exited (0) 2 minutes ago teste2
cd151c87e1ae alpine "sh" 11 minutes ago Exited (0) 6 minutes ago container_renamed
5349203e9997 airflow-basic "/entrypoint.sh" 2 weeks ago Exited (0) 13 days ago exciting_hamilton
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17>


Windows PowerShell
a8842b36c69b alpine "sh" 2 days ago Exited (0) 2 days ago teste3
6f926b56846c alpine "sh" 2 days ago Exited (0) 2 days ago teste2
cd151c87e1ae alpine "sh" 2 days ago Exited (0) 2 days ago container_renamed
5349203e9997 airflow-basic "/entrypoint.sh" 3 weeks ago Exited (0) 2 weeks ago exciting_hamilton
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container cp .\Teste\ teste2
must specify at least one container source
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container cp .\Teste\ teste2:/
Successfully copied 2.05kB to teste2:/
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container exec teste2
"docker container exec" requires at least 2 arguments.
See 'docker container exec --help'.
Usage: docker container exec [OPTIONS] CONTAINER COMMAND [ARG...]
Execute a command in a running container
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17> docker container start -ia teste2
/ # ls
Teste dev home media opt root sbin sys usr
bin etc lib mnt proc run srv tmp var
/ # ls Teste/
Benio.txt
/ # cd Teste/
/Teste # ls
Benio.txt
/Teste # echo "Careca" > Benio.txt
/Teste # cat Benio.txt
Careca
/Teste #

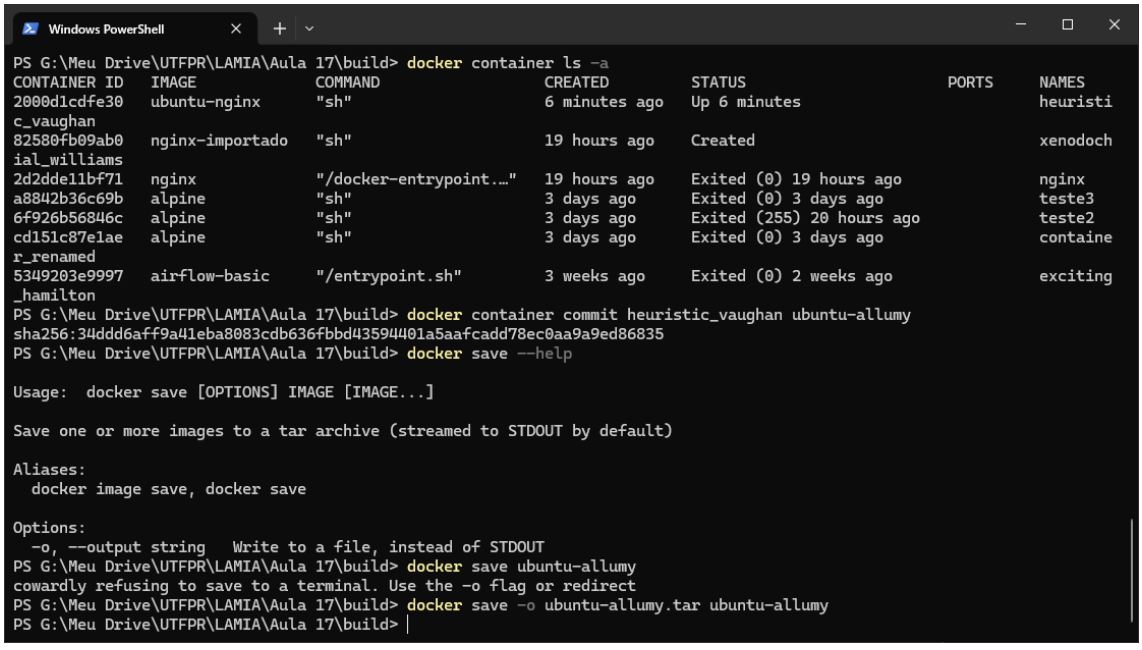

Windows PowerShell
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\ContainerContent> docker container logs teste2
/ # ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/ # exit
/ # ls
Teste dev home media opt root sbin sys usr
bin etc lib mnt proc run srv tmp var
/ # ls Teste/
Benio.txt
/ # cd Teste/
/Teste # ls
Benio.txt
/Teste # echo "Careca" > Benio.txt
/Teste # cat Benio.txt
Careca
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\ContainerContent>

Exercício

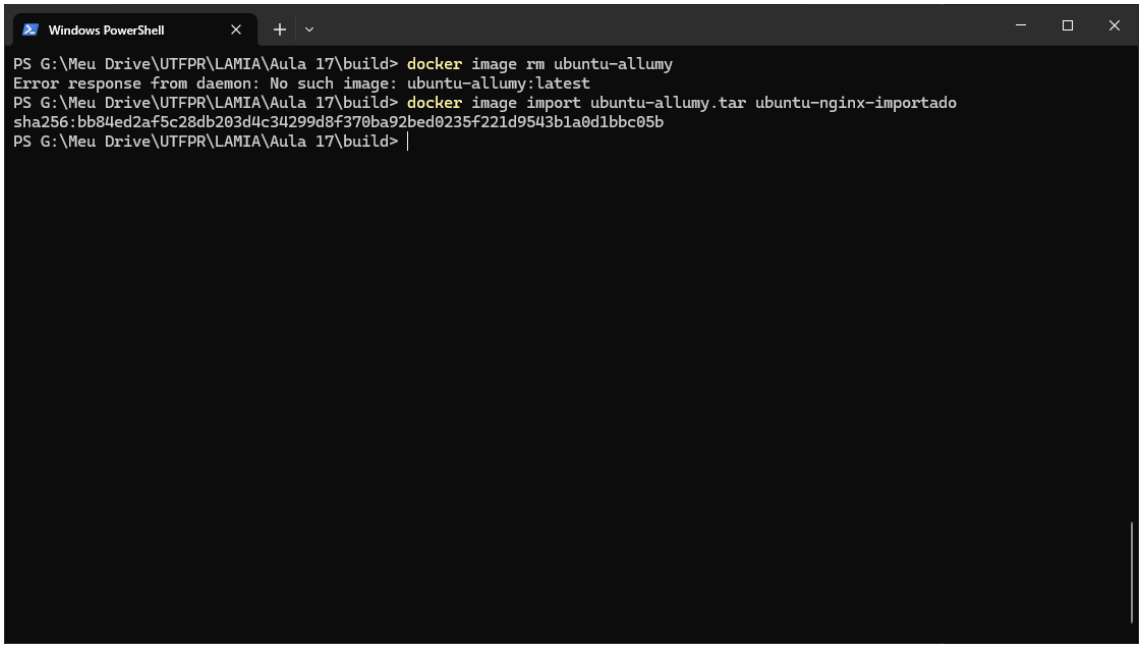
Na primeira questão do exercício, foi necessário criar um arquivo com o “Vim” para que fosse possível instalar um container com o ubuntu e o NGINX instalado.



Gerando imagem chamada “ubuntu-allumy” e exportando-a para um arquivo tar.



Removendo a imagem “ubuntu-allumy” (aqui eu já tinha removido a imagem, por isso está sendo mostrado que ela não existe) e importando a imagem que salvamos no arquivo .tar.



(Fiz os dois tipos de importações, pelo comando “import” e “load”)

```
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\build> docker image load --help

Usage:  docker image load [OPTIONS]

Load an image from a tar archive or STDIN

Aliases:
  docker image load, docker load

Options:
  -i, --input string   Read from tar archive file, instead of STDIN
  -q, --quiet          Suppress the load output
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\build> docker image load -i ubuntu-allumy.tar
Loaded image: ubuntu-allumy:latest
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\build>
```

Rodando o container a partir da imagem carregada

```
PS G:\Meu Drive\UTFPR\LAMIA\Aula 17\build> docker container run -it ubuntu-allumy sh
# ls
bin               boot  etc   lib    media  opt   root  sbin               srv  tmp  var
bin.usr-is-merged  dev   home  lib64  mnt    proc  run   sbin.usr-is-merged  sys  usr
#
```

Referencias

Introdução a Docker Containers (Hands On)