

17 - Prática: Docker e Containers para Aplicações (III)

Higor Miller Grassi

Descrição da Atividade: utilização de comandos Docker para gerenciar containers, imagens, volumes e redes, incluindo a criação, execução, remoção, e manipulação de containers e imagens, além de mapeamento de volumes e portas. Também abrange a exportação de containers como imagens e a construção de novas imagens a partir de Dockerfiles.

Commandos:

Docker CLI: interface de linha de comando do Docker, que permite interagir com o Docker Engine para gerenciar contêineres, imagens, volumes, redes e outros recursos, tendo várias linhas de comando que irei pontuar a seguir:

Define o recurso e qual ação queremos gerenciar

-docker {object} {action}

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

exemplos de utilização:

- docker container rm alpine
- docker image ls
- docker container ls

Caso precise de ajuda em comandos

-docker container --help

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container --help
Usage:  docker container COMMAND
Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Execute a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Create and run a new container from an image
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

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

Para criação de container

-docker container create --name {name} {image}

-docker container create --name test alpine

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container create --name test alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
52f827f72350: Pull complete
Digest: sha256:56fa17d2a7e7f168a043a2712e63aed1f8543aeafdcee47c58dcffe38ed51099
Status: Downloaded newer image for alpine:latest
49043af1e6de9bbc815c70f767d863ed0b04be8e71683f4c0ddcc6314b71eb69
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container start {test}
```

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
49043af1e6de	alpine	"/bin/sh"	56 seconds ago	Created		test

Para deletar container

-docker container rm {name} docker container rm test

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container rm test
test
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
heloamillergrassi@MacBook-Air-de-Heloa ~ %
```

Para startar um container

-docker container start {name}

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container start test
test
```

Permite conectar-se aos fluxos padrão de entrada, saída e erro de um contêiner em execução

-docker container attach {name}

```
heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container attach test
/ # ls
bin      etc      lib      mnt      proc     run      srv      tmp      var
dev      home    media    opt      root     sbin     sys      usr
/ #
```

Criação de arquivo e pegar o mesmo

-touch {nome}

-echo "nome_container" > {nome}

```
[(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % touch Higor
[(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % echo "test" > Higor
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ %
```

Realiza operações sem necessariamente entrar no container

```
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container exec test cat /etc/hosts
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters
172.17.0.2    ca3e876664df
```

Para renomear container

-docker container rename {name} {new_name}

Executar comandos diretamente dentro de um contêiner em execução

-docker container exec {container_name} {command}

Copiar arquivos ou diretórios entre os containers Docker e vice versa

-docker container cp {path_to_file} {container_name}:{destination_path} docker
container cp {container_name}:{destination_path} {path_to_file}

Permite obter informações detalhadas sobre um contêiner

-docker container inspect {container_name_or_id}

Mapeando volumes:

Utilizado para salvar os estados de contêineres, permitindo que você armazene dados fora do contêiner, garantindo que as informações sejam preservadas mesmo que o contêiner seja removido ou recriado

-docker container run -v {host_dir}:{container_dir} {image}

```
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container run -v /Users/heloamillergrassi/bkp:/bkp --name teste-volume -it alpine sh
/ # ls /bkp
```

Os volumes são utilizados para salvar os status dos containers, assim, **{container_dir}** passa a ser compartilhado entre o contêiner e a máquina local, permitindo que qualquer alteração feita em arquivos nesse diretório seja imediatamente visível em ambos os lados.

Mapeando Portas:

As portas expostas durante a construção de uma imagem Docker podem ser associadas a uma porta específica da máquina local juntamente com a imagem nginx

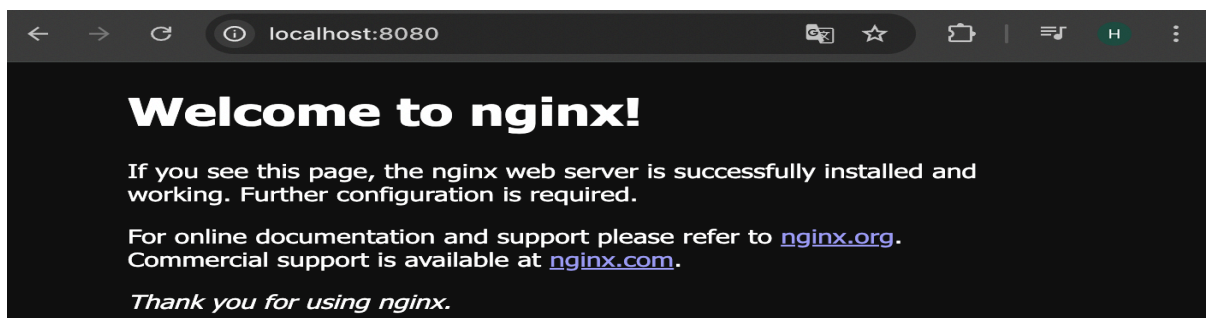
-docker run -p {host_port}:{container_port} {image}

ex: docker run --name my-nginx -d -p 8080:80 nginx

```
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % docker run --name my-nginx -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
f5c6876bb3d7: Pull complete
bdb964b66a74: Pull complete
b5368be906ec: Pull complete
755bf136756e: Pull complete
aafc2e219c20: Pull complete
261c6a94b398: Pull complete
a8066ea4829b: Pull complete
Digest: sha256:42e917aaa1b5bb40dd0f6f7f4f857490ac7747d7ef73b391c774a41a8b994f15
Status: Downloaded newer image for nginx:latest
5116278a14db27ae7e86de2afe0a56e9ea65545ea335588a7ae7ae7a31e02277
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container port my-nginx
80/tcp -> 0.0.0.0:8080
```

Quando o Docker Engine é instalado em uma máquina, ele cria algumas redes automaticamente e sempre que um contêiner é iniciado, ele se conecta por padrão a uma rede bridge, e com esta rede é permitido que o host tenha acesso ao contêiner. No entanto, se uma porta não for mapeada, o contêiner usará apenas o IP da rede bridge, e o acesso direto à porta pelo host não será possível.

Depois de executar o código acima, conseguimos acessar o https através do IMocalhost:8080 como definido



Mostra o log do localhost
-docker container logs my-nginx

```
(base) heloamillergrassi@MacBook-Air-de-Heloa ~ % docker container logs my-nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/01/09 17:24:04 [notice] 1#1: using the "epoll" event method
2025/01/09 17:24:04 [notice] 1#1: nginx/1.27.3
2025/01/09 17:24:04 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/09 17:24:04 [notice] 1#1: OS: Linux 6.10.14-linuxkit
2025/01/09 17:24:04 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/09 17:24:04 [notice] 1#1: start worker processes
2025/01/09 17:24:04 [notice] 1#1: start worker process 29
2025/01/09 17:24:04 [notice] 1#1: start worker process 30
2025/01/09 17:24:04 [notice] 1#1: start worker process 31
```


Ao rodar o container nginx conseguimos executar comandos dentro do container, neste exemplo rodei juntamente ao shell

```

(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container start my-nginx
my-nginx
(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container exec -it my-nginx sh
# ls
bin          dev          docker-entrypoint.sh  home  media  opt   root  sbin  sys  usr
boot         docker-entrypoint.d  etc      lib   mnt   proc  run   srv   tmp  var
# █

```

Imagens e Containers:

Basicamente é a exportação um container em forma de imagem assim qualquer pessoa com essa imagem pode executar os processos exatamente da mesma forma que o criador configurou

Criação do container que será utilizado como exemplo, juntamente com o diretório

```

(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container run --name nginx-allumy -it alpine sh
sh: command not found: docker
(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container run --name nginx-allumy -it alpine sh
[/ # ls
bin      etc      lib      mnt      proc     run      srv      tmp      var
[dev     home     media    opt      root    /sbin    sys      usr
[/ # mkdir allumy
[/ # cd allumy/
[/allumy # touch docker
[/allumy # ls
[docker
[/allumy # echo "teste container para imagem" > docker
[/allumy # cat docker
[teste container para imagem
[/allumy #

```

Comando para a criação de uma imagem através de um container
-docker container commit [OPTIONS] CONTAINER IMAGE

```

(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container commit nginx-allumy nginx-allumy-img
sha256:cff38347a8fdfaf5ee5652e38d241f33f3bea3bd1aee48f63f67eb2db1ae632c
(base) helloamillergrassi@MacBook-Air-de-Heloa ~ %

```

```

(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker image ls
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
nginx-allumy-img    latest          cff38847a8fd   About a minute ago  8.17MB
alpine               latest          7ad00e65ee25   30 hours ago      8.17MB
nginx                latest          5e0fa356e6f4   6 weeks ago       197MB
(base) helloamillergrassi@MacBook-Air-de-Heloa ~ %

```

Acessa docker através da imagem

```

(base) helloamillergrassi@MacBook-Air-de-Heloa ~ % docker container run -it --rm nginx-allumy-img sh
/ # ls
allumy  dev      home     media    opt       root      sbin      sys      usr
bin     etc      lib      mnt      proc      run       srv       tmp      var
/ # cd allumy/
/allumy # ls
docker
/allumy # cat docker
teste container para imagem
/allumy # █

```

Criação do arquivo .tar a partir da imagem, usado para exportar e importar containers e salvar e carregar imagens docker

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image save -o nginx-allumy.tar nginx-allumy-img
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % ls
nginx-allumy.tar
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % █

```

Carregando a imagem.tar

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image load -i nginx-allumy.tar
60351aef5cc7: Loading layer 4.096kB/4.096kB
Loaded image: nginx-allumy-img:latest
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % █

```

Rodando o docker a partir do arquivo tar

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker container run -it --rm --name nginx-allumy nginx-allumy-img sh
/ # ls
allumy  dev      home     media    opt       root      sbin      sys      usr
bin     etc      lib      mnt      proc      run       srv       tmp      var
/ # █

```

Ve o histórico da imagem, tudo o q tem sido feito

-docker image {image_name}

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image history nginx-allumy-img

```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
cff38847a8fd	15 minutes ago	sh	131B	
<missing>	30 hours ago	CMD ["/bin/sh"]	0B	buildkit.dockerfile.v0
<missing>	30 hours ago	ADD alpine-minirootfs-3.21.2-aarch64.tar.gz ...	8.17MB	buildkit.dockerfile.v0

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % █

```

Criação de imagem a partir do sistema de arquivos importado do container

-docker image import {nome_imagem} {novo_nome}

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image import teste-export.tar ubuntu-import
sha256:6a93a9cd03b6fd93c94d074e2098e4a0c5ddc620023bf98ef25892c14b980645
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image ls ubuntu-import

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu-import	latest	6a93a9cd03b6	7 seconds ago	8.17MB

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % █

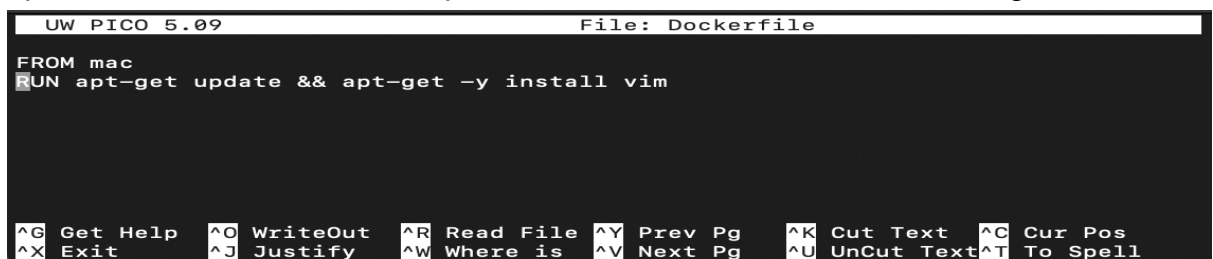
```

Comandos utilizados para a imagem:

- **Inspecionar os elementos de uma imagem:** Use o comando para obter informações detalhadas sobre a imagem, como sua configuração, camadas e outros metadados.
-docker image inspect {image_name}
- **Excluir imagens não associadas a contêineres:** Esse comando remove imagens órfãs, ou seja, imagens que não estão sendo usadas por nenhum contêiner.
-docker image prune
- **Baixar uma imagem do Docker Hub:** Para fazer o download de uma imagem para a sua máquina local a partir do Docker Hub.
-docker image pull {image_name}
- **Marcar uma imagem com um nome personalizado:** Crie um alias para a imagem, permitindo que ela seja referenciada por um nome diferente nos comandos futuros.
-docker image tag {image_name} {custom_name}
- **Enviar uma imagem para o Docker Hub:** Envie a imagem para o Docker Hub ou outro repositório remoto configurado, para que ela possa ser compartilhada ou armazenada.
-docker image push {image_name}

Construindo a sua própria imagem:

Após utilizar o nano Dockerfile para editarmos, damos o comando a seguir



```
UW PICO 5.09 File: Dockerfile
FROM mac
RUN apt-get update && apt-get -y install vim
```

The screenshot shows the nano text editor interface. The top status bar displays 'UW PICO 5.09' on the left and 'File: Dockerfile' on the right. The main editing area contains two lines of text: 'FROM mac' and 'RUN apt-get update && apt-get -y install vim'. The bottom status bar shows various keyboard shortcuts for nano, such as ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Pg, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where is, ^V Next Pg, ^U UnCut Text, and ^T To Spell.

O comando de construção

-docker image build -t {dockerfile_name} {path_to_directory}

```
(base) helloamillergrassi@MacBook-Air-de-Helios imagens % vim Dockerfile
(base) helloamillergrassi@MacBook-Air-de-Helios imagens % docker image build -t meu-ubuntu.
ERROR: "docker buildx build" requires exactly 1 argument.
See "docker buildx build --help".

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
(base) helloamillergrassi@MacBook-Air-de-Helios imagens % docker image build -t meu-ubuntu .

[+] Building 153.7s (4/6)                                     docker:desktop-linux
=> [internal] load build definition from Dockerfile           0.0s
=> => transferring dockerfile: 102B                          0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 47.3s
=> [auth] library/ubuntu:pull token for registry-1.docker.io   0.0s
=> [internal] load dockerignore                                0.0s
=> transferring context: 2B                                     0.0s
=> [1/2] FROM docker.io/library/ubuntu:latest@sha256:88dd3c3b9c6cceb9f1667e9290b3bc61b78c26780c2cbdae5ff0fa92cc6734ab 86.4s
```

Volumes são pontos de imagem localizados fora do sistema de arquivos de um contêiner, possibilitando mapear diretórios de host no contêiner.

```
-docker run -it --name {container_name} -v {path_to_shared_dir_in_the_container}
{image}
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker run -it --name volume -v /data alpine sh
/ # ls
bin      dev      home     media    opt       root      sbin     sys      usr
data     etc      lib      mnt      proc      run       srv      tmp      var
/ # cd data
[/data # ls
```

Para acessar a pasta compartilhada

```
-docker inspect {volume_id}
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker volume ls
DRIVER      VOLUME NAME
local       586c6912782309927ecadcf1b899190d1949638f13daa1cc12082936d8044f66
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens %
```

Mostrar o disco doker

```
-docker system df
```

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker system df

```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	6	2	540.5MB	343.4MB (63%)
Containers	6	4	1.243kB	19B (1%)
Local Volumes	1	1	0B	0B
Build Cache	5	0	65B	65B

```

(base) helloamillergrassi@MacBook-Air-de-Heloa imagens %

```


Para mostrar as informações do docker
-docker system info

```
((base) heloamillergrassi@MacBook-Air-de-Heloa imagens % docker system info
Client:
 Version:      27.4.0
 Context:      desktop-linux
 Debug Mode:   false
 Plugins:
  ai: Ask Gordon - Docker Agent (Docker Inc.)
     Version:  v0.5.1
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-ai
  buildx: Docker Buildx (Docker Inc.)
     Version:  v0.19.2-desktop.1
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
     Version:  v2.31.0-desktop.2
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-compose
  debug: Get a shell into any image or container (Docker Inc.)
     Version:  0.0.37
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-debug
  desktop: Docker Desktop commands (Beta) (Docker Inc.)
     Version:  v0.1.0
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-desktop
  dev: Docker Dev Environments (Docker Inc.)
     Version:  v0.1.2
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-dev
  extension: Manages Docker extensions (Docker Inc.)
     Version:  v0.2.27
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-extension
  feedback: Provide feedback, right in your terminal! (Docker Inc.)
     Version:  v1.0.5
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-feedback
  init: Creates Docker-related starter files for your project (Docker Inc.)
     Version:  v1.4.0
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-init
  sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc.)
     Version:  0.6.0
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-sbom
  scout: Docker Scout (Docker Inc.)
     Version:  v1.15.1
     Path:      /Users/heloamillergrassi/.docker/cli-plugins/docker-scout

Server:
 Containers: 6
  Running: 4
  Paused: 0
  Stopped: 2
 Images: 6
 Server Version: 27.4.0
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 2
 Plugins:
  Volume: local
```

Exercício

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker container run --name exercicio -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
8bb55f067777: Already exists
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
root@e57245e832fc:/#
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker container run --name exercicio -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
8bb55f067777: Already exists
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
root@e57245e832fc:/# apt-get install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package nginx
```

```
[root@e57245e832fc:/# /etc/init.d/nginx start
* Starting nginx nginx
root@e57245e832fc:/#
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' ca23cec95c08172.17.0.3
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens %
```

Welcome to nginx!

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

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

Thank you for using nginx.

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker commit exercicio exercicio
sha256:4e5b7ca03f40a1f5f528ddde22614255e9d79c09b8a753c45353d43cd23b757a
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image ls exercicio
REPOSITORY TAG IMAGE ID CREATED SIZE
exercicio latest 4e5b7ca03f40 9 seconds ago 154MB
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image save exercicio -o exercicio.tar
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % ls
Dockerfile nginx-allumy.tar
exercicio.tar teste-export.tar
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens %
```

```
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens % docker image rm exercicio
Untagged: exercicio:latest
Deleted: sha256:4e5b7ca03f40a1f5f528ddde22614255e9d79c09b8a753c45353d43cd23b757a
Deleted: sha256:b28ae72e7a239eec0ee04635560533898a9be086cdf969c0c816de56adfa60ab
(base) helloamillergrassi@MacBook-Air-de-Heloa imagens %
```

```
((base) heloamillergrassi@MacBook-Air-de-Heloia imagens % docker image load -i exercicio.tar
720a22f1b0c5: Loading layer 54.07MB/54.07MB
Loaded image: exercicio:latest
(base) heloamillergrassi@MacBook-Air-de-Heloia imagens % docker container run -it exercicio sh
[# █
```

Conclusão: A prática com Docker oferece uma compreensão sobre a criação, gerenciamento e compartilhamento de containers e imagens, otimizando o desenvolvimento e garantindo a portabilidade e persistência de dados, essenciais para soluções escaláveis e eficientes.