

Relatório 17 - Docker e Containers para Aplicações (III)

Lucas Scheffer Hundsdorfer

Descrição da atividade

A primeira aula é ensinado a instalar o Docker no seu SO e algumas configurações básicas dentro dele. A segunda aula é para mostrar alguns comandos utilizando o Docker Commandline ou CLI, segue sempre o mesmo padrão, docker na frente, (objeto) e a (ação):

Interface de linha de comando para controlar a docker engine.

docker {objeto} {ação}

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

\$ docker container rm alpine

\$ docker image ls

É mostrado que existe o comando ‘--help’ para que dentro do cmd seja listado tudo que é possível fazer:

```
C:\Users\Lucas>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
```

Mostrando as opções de ação com o meu objeto, o container.

É falado dentro do curso que os containers são descartáveis, pois eles são criados a partir de imagens imutáveis. É deixado bem claro durante as

vídeo-aula que dentro do CLI é possível mais de um comando ter o mesmo efeito, como ele mostra por exemplo que para criar o docker, iniciar e entrar nele dá para ser feito com 3 comandos diferentes:

```
C:\Users\Lucas>docker container create --name teste2 -it alpine
be55ba53ea4aa565475b798953ec9e58731628b4693cafa5cb892e6d5bd57168

C:\Users\Lucas>docker container start teste
teste

C:\Users\Lucas>docker container attach teste
/ # |
```

Também pode ser feito em uma linha de comando só:

```
C:\Users\Lucas>docker container run -it --name teste3 alpine sh
/ # |
```

Também nos é mostrado que é possível sair do container e ainda deixar ele em execução, como criar pastas dentro do container sem estar dentro dele, e como copiar essas pastas tanto do meu sistema para o container como do container para o sistema.

Manipulando containers

Listando containers

```
$ docker container ls -a -s
```

Parando containers

```
$ docker container stop <container>
```

Inicializando containers

```
$ docker container start <container>
```

Copiando containers

```
$ docker container cp <src> <dst>
```

Removendo containers

```
$ docker container rm <nome_id>
```

Anexando-se a containers em execução

```
$ docker container attach <nome_id>
```

Inspeccionando containers

```
$ docker container inspect <nome_id>
```

Renomeando containers

```
$ docker container rename <nome_antigo> <novo_nome>
```

Aqui estão alguns dos comandos mais básicos para manipulação de containers.

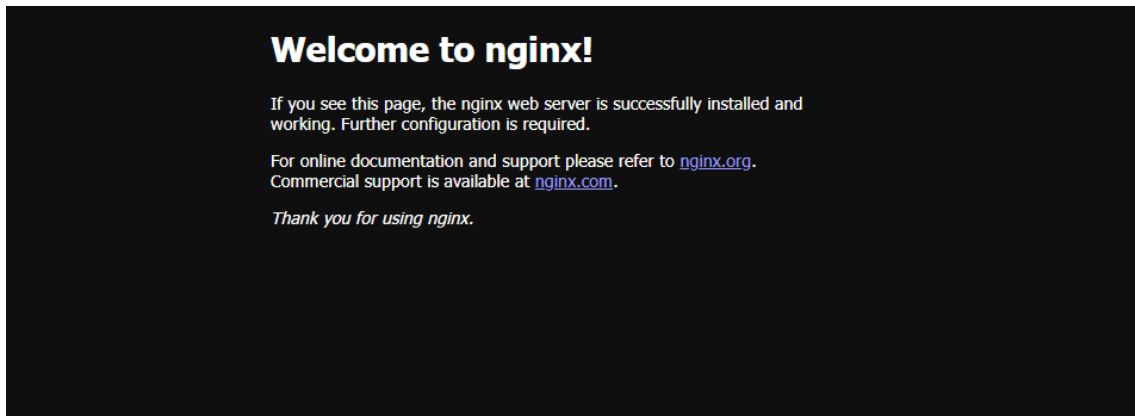
O que é possível ser feito também é o mapeamento de volumes, volumes são utilizados para salvar o estado dos containers. Na aula ele dá um exemplo que ele cria através do host pastas dentro do container, porém ele deixa claro que é

um exemplo simples mas pode ser algo bem mais complexo e útil, como o banco de dados ou outras aplicações.

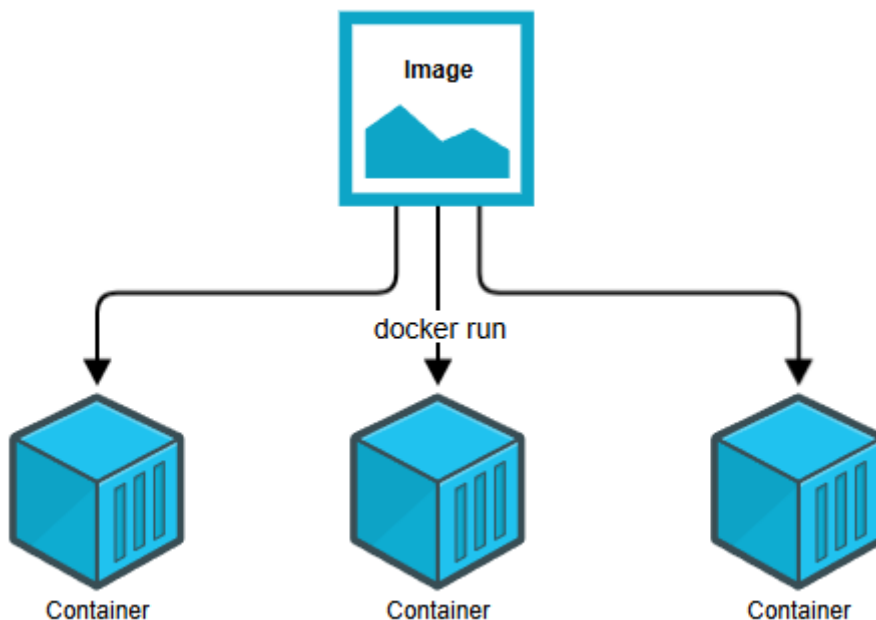
O que é uma porta?

Uma porta é um canal de comunicação que permite que os serviços dentro de um contêiner troquem dados com o mundo externo.

Após criar um container através da imagem do nginx, e é possível acessar ao localhost dele, imagem abaixo:



Para um melhor entendimento de imagens e containers ele usa o exemplo de programação orientada a objetos, que a imagem é uma classe e o container é um objeto.



Em uma das aulas é ensinado a criar uma imagem a partir de um container modificado, ele dá um exemplo básico da criação de uma pasta e um arquivo dentro dela e cria uma imagem, e mostra que já se pode criar um container a

partir dessa imagem criada.

```
C:\Users\Lucas>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 # cat docker
TESTE CONTIANER PARA IMAGEM
/allumy # |
```

Aqui foi criado um container a partir da imagem criada e a pasta já estava lá com o arquivo dentro.

```
C:\Users\Lucas\imagens>docker image save -o nginx-allumy.tar nginx-allumy-img
```

Esse comando salva uma imagem onde ela estiver no diretório com o nome desejado, nesse caso foi um arquivo .tar com o nome nginx-allumy.

```
C:\Users\Lucas\imagens>docker image load -i nginx-allumy.tar
Loaded image: nginx-allumy-img:latest

C:\Users\Lucas\imagens>docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
nginx-allumy-img    latest       191cale3011a  25 minutes ago 12.2MB
nginx                latest       fb39280b7b9e  6 weeks ago   279MB
alpine               latest       a8560b36e8b8  3 months ago  12.1MB
hello-world          latest       dd01f97f2521  4 months ago  20.4kB

C:\Users\Lucas\imagens>
```

Aqui o comando carrega a imagem a partir do arquivo .tar.

Também é possível ver o histórico da imagem, com o comando abaixo:

```
C:\Users\Lucas>docker image history nginx-allumy-img
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
191cale3011a   28 minutes ago  sh                  20.5kB
<missing>      3 months ago    CMD ["/bin/sh"]     0B        buildkit.dockerfile.v0
<missing>      3 months ago    ADD alpine-minirootfs-3.21.3-x86_64.tar.gz /... 8.5MB     buildkit.dockerfile.v0

C:\Users\Lucas>
```

Resumo dos comandos mais úteis dos containers e das imagens:

Docker Container:

1. `docker container run`: Cria e inicia um novo contêiner.
2. `docker container ls`: Lista os contêineres.
3. `docker container stop`: Para um ou mais contêineres em execução.
4. `docker container start`: Inicia um ou mais contêineres parados.
5. `docker container rm`: Remove um ou mais contêineres parados.

Docker Image:

1. `docker image pull`: Baixa uma imagem de um registro.
2. `docker image ls`: Lista as imagens locais.
3. `docker image build`: Constroi uma imagem a partir de um Dockerfile.
4. `docker image tag`: Cria um alias para uma imagem.
5. `docker image rm`: Remove uma ou mais imagens locais.

Para criar imagens a partir de um dockerfile que está no mesmo diretório é possível usar esse comando

`docker image build --no-cache -t meu-ubuntu .`

É possível ver como o Docker está e quanto ele está consumindo do seu disco, com esse comando:

```
C:\Users\Lucas>docker system df
TYPE                TOTAL    ACTIVE    SIZE      RECLAIMABLE
Images              7        1         649.9MB   646.3MB (99%)
Containers          1        0         16.38kB   16.38kB (100%)
Local Volumes       1        1          0B        0B
Build Cache         4        0         12.29kB   12.29kB

C:\Users\Lucas>
```

Para liberação de memória, existe o docker prune, para evitar ter que ficar deletando dependências entre si, para sim desinstalar o que quer, para não dar dor de cabeça tem o docker system prune:

\$ docker system prune (pede confirmação)

Irá deletar:

- Todos os containers parados
- Todos os volumes não utilizados
- Todas as redes não utilizadas
- Todas as imagens não utilizadas

Também é possível fazer isso para cada tipo de objeto:

Limpar todas as imagens que não estão associadas a containers

```
$ docker image prune
```

Limpar todos os containers que não estão em execução

```
$ docker container prune
```

Limpar todos os volumes que não estão associados a containers

```
$ docker volume prune
```

Limpar todas as redes que não estão associadas a containers

```
$ docker network prune
```

Exercício proposto:

Rodar em modo interativo um container a partir da imagem do ubuntu, usando apt-get instalar o NGINX, inicializar o servidor NGINX. •

Com o comando commit, gerar uma imagem chamada ubuntu-allumy;

Exportar a image para o arquivo ubuntu-allumy.tar através do comando save;

Remover a imagem ubuntu-allumy;

Importar a imagem com ubuntu-allumy.tar através do comando load.

Rodar um container a partir da imagem importada.

Rodar em modo interativo um container a partir da imagem do ubuntu:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker container run --name exercicio -it ubuntu sh
# |
```

apt-get update para podermos instalar o nginx:

```
# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [22.1 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1087 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1092 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1442 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
69% [12 Packages 13.4 MB/19.3 MB 69%]
```

apt-get install nginx para instalar o nginx:

```
# apt-get install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  iproute2 krb5-locales libatm1t64 libbpf1 libcap2-bin libelf1t64 libgssapi-krb5-2 libk5crypto3 libkeyutils1 libkrb5-3
  libkrb5support0 libmn10 libpam-cap libtirpc-common libtirpc3t64 libxtables12 nginx-common
Suggested packages:
  iproute2-doc python3:any krb5-doc krb5-user fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  iproute2 krb5-locales libatm1t64 libbpf1 libcap2-bin libelf1t64 libgssapi-krb5-2 libk5crypto3 libkeyutils1 libkrb5-3
  libkrb5support0 libmn10 libpam-cap libtirpc-common libtirpc3t64 libxtables12 nginx nginx-common
0 upgraded, 18 newly installed, 0 to remove and 2 not upgraded.
Need to get 2734 kB of archives.
After this operation, 8075 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libelf1t64 amd64 0.190-1.1ubuntu0.1 [57.8 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 libbpf1 amd64 1:1.3.0-2build2 [166 kB]
```

docker commit para salvar a imagem:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker commit exercicio ubuntu-allumy
sha256:b1a28a11f4758e5ccf11ec99ca58e9874cf9c8a825a34394ec1584a260652f6a
```

docker image save para exportar a imagem para um arquivo .tar:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker save ubuntu-allumy -o ubuntu-allumy.tar
C:\Users\Lucas\Documents\Lamia\Card 17>|
```

docker image rm para remover a imagem:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker image rm ubuntu-allumy
Untagged: ubuntu-allumy:latest
Deleted: sha256:b1a28a11f4758e5ccf11ec99ca58e9874cf9c8a825a34394ec1584a260652f6a
```

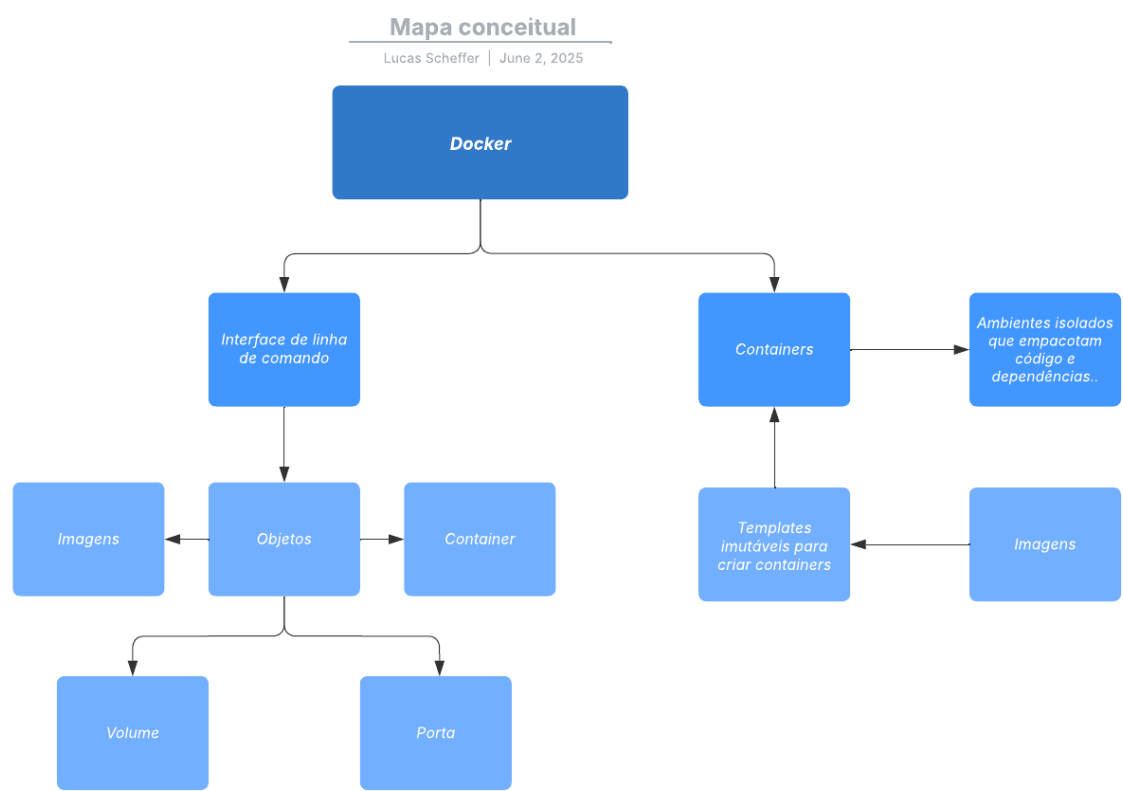
exportar a imagem através do docker image load:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker load -i ubuntu-allumy.tar
Loaded image: ubuntu-allumy:latest
```

e por fim rodar um container a partir da imagem exportada:

```
C:\Users\Lucas\Documents\Lamia\Card 17>docker container run --name teste -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
#
```

Insight visual original feito usando a ferramenta LucidChart:



Conclusões

O curso demonstra claramente que o uso do Docker e o conhecimento da sua interface de linha de comando são extremamente importantes para diversas aplicações e tarefas do dia a dia. A habilidade de manipular containers e imagens de maneira eficiente, juntamente com a compreensão dos conceitos de volumes e redes, facilita o desenvolvimento e a implantação de software de forma mais ágil e portátil. Além disso, o domínio do Docker contribui para a otimização dos recursos do sistema e para a manutenção de ambientes de trabalho consistentes, o que resulta em maior produtividade e na simplificação de processos complexos.

Referências