Sistemas de Arquivos em Docker

Professor: Pedro Horchulhack

Agenda

1. O que são volumes?
2. Criação, montagem e remoção
3. Compartilhamento de dados entre contêineres
4. Configuração de contêineres e variáveis de ambiente

O que são volumes?

O que são volumes?

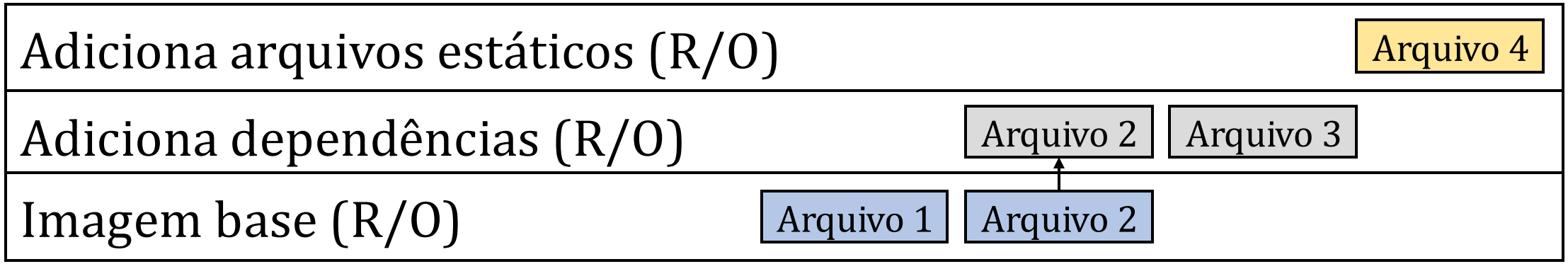
- São, literalmente, uma forma de **persistir** os dados de contêineres

- Além disso, existem para além do ciclo de vida de um contêiner
  - Dados podem persistir mesmo após um contêiner deixar de executar

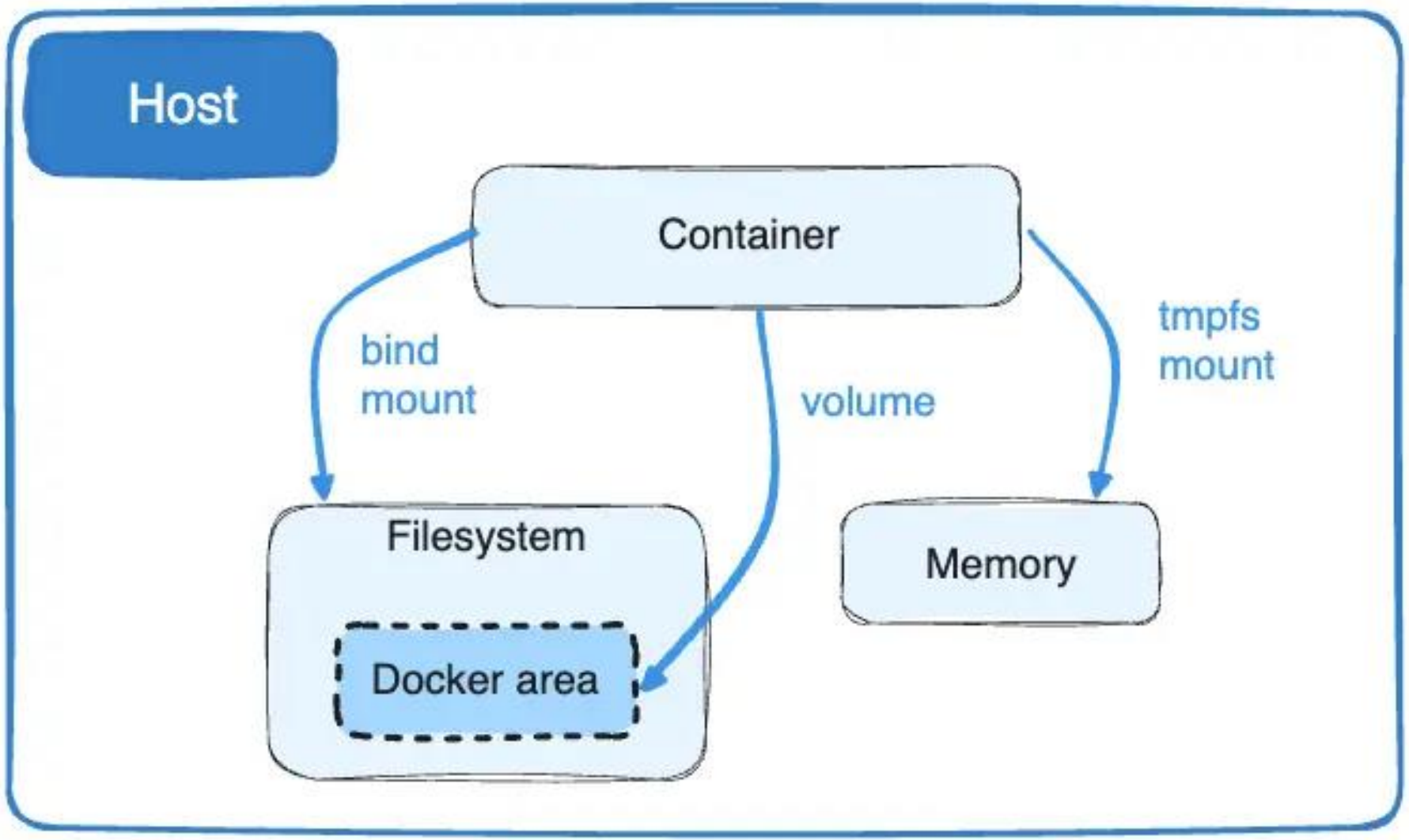
- Podem ser utilizados para:
  - Persistir dados depois de um contêiner parar ou ser removido
  - Configurar aplicações que dependem de dados persistentes (BD, por exemplo)
  - Mapear arquivos/diretórios do host para o contêiner e vice-versa
  - Compartilhar dados entre contêineres no mesmo host

O que são volumes?

- Utilizam do mesmo sistema de arquivos em camadas que vimos nas imagens
  - Union Filesystem (UFS)



O que são volumes?

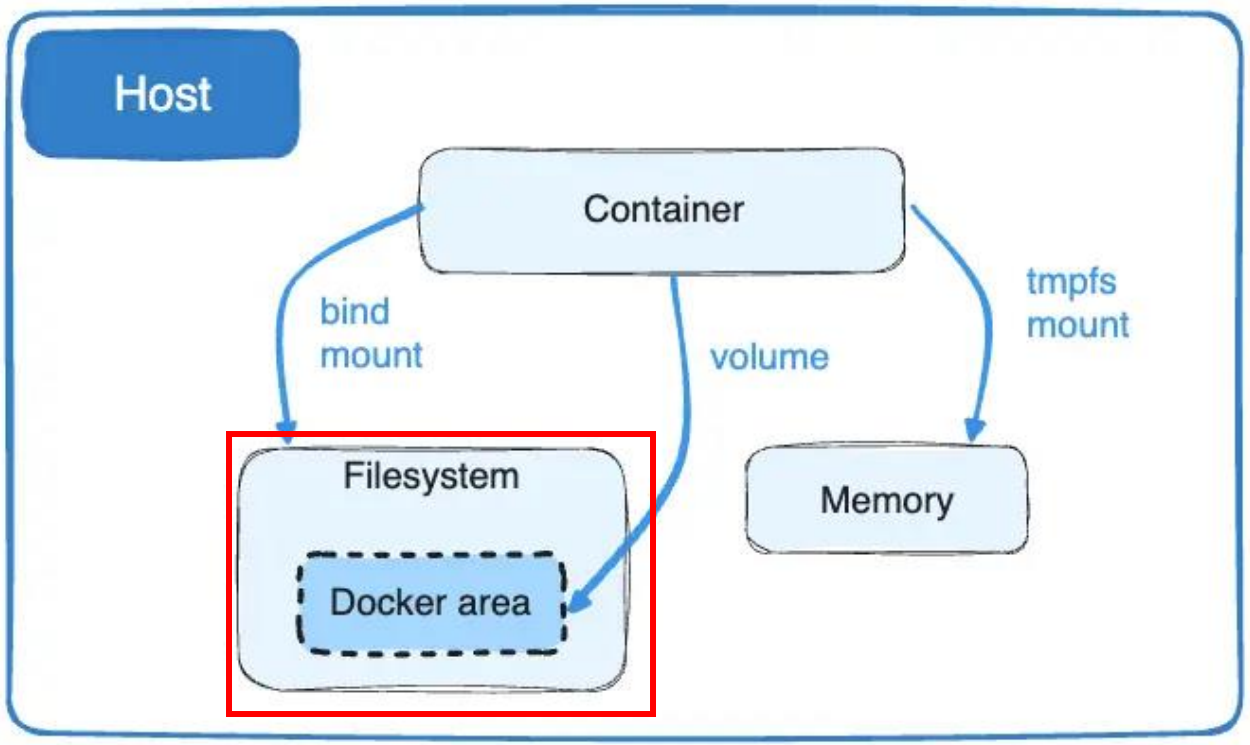


Volumes Docker (fonte: [1])

O que são volumes?

Bind-mount

- Mapeamento host -> contêiner
- Necessita criação se não existir

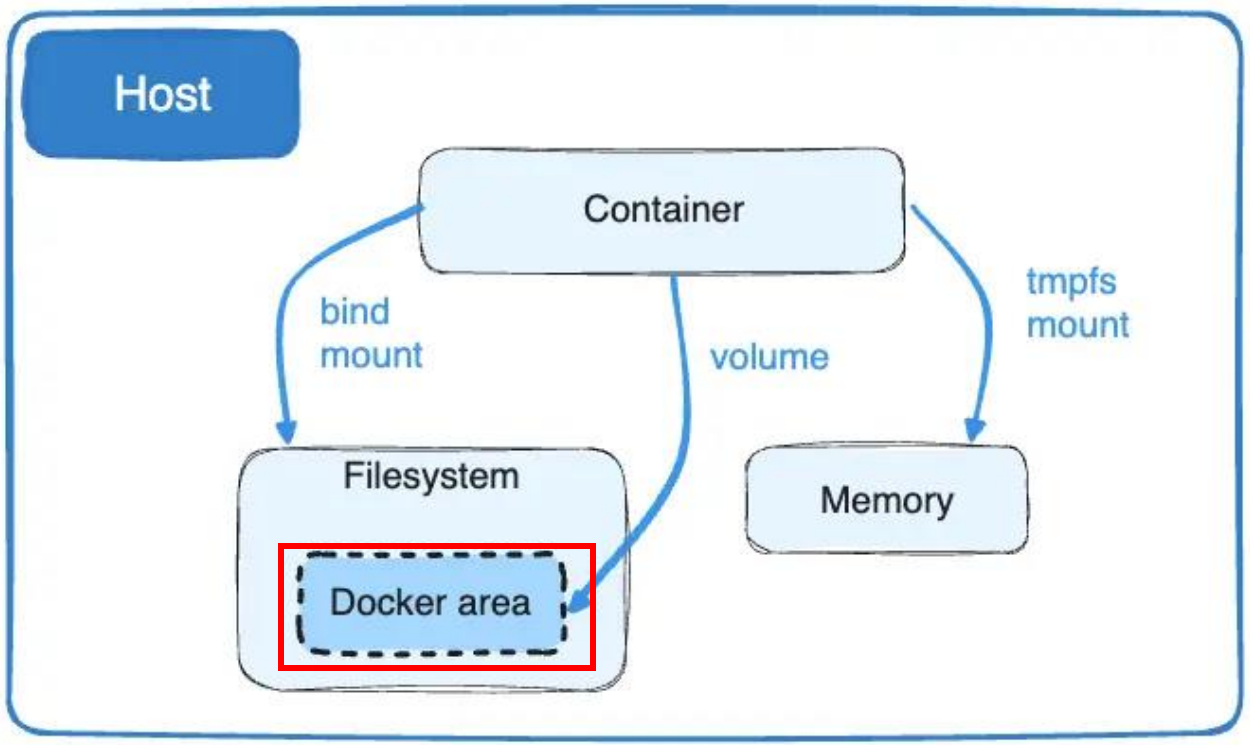


Volumes Docker (fonte: [1])

O que são volumes?

Volume

- Criação e gestão pelo Docker
- São “isolados” do host
- Representação lógica
- Anônimos ou nomeados
  - Anônimos removidos com `--rm`
  - Nomeados persistem

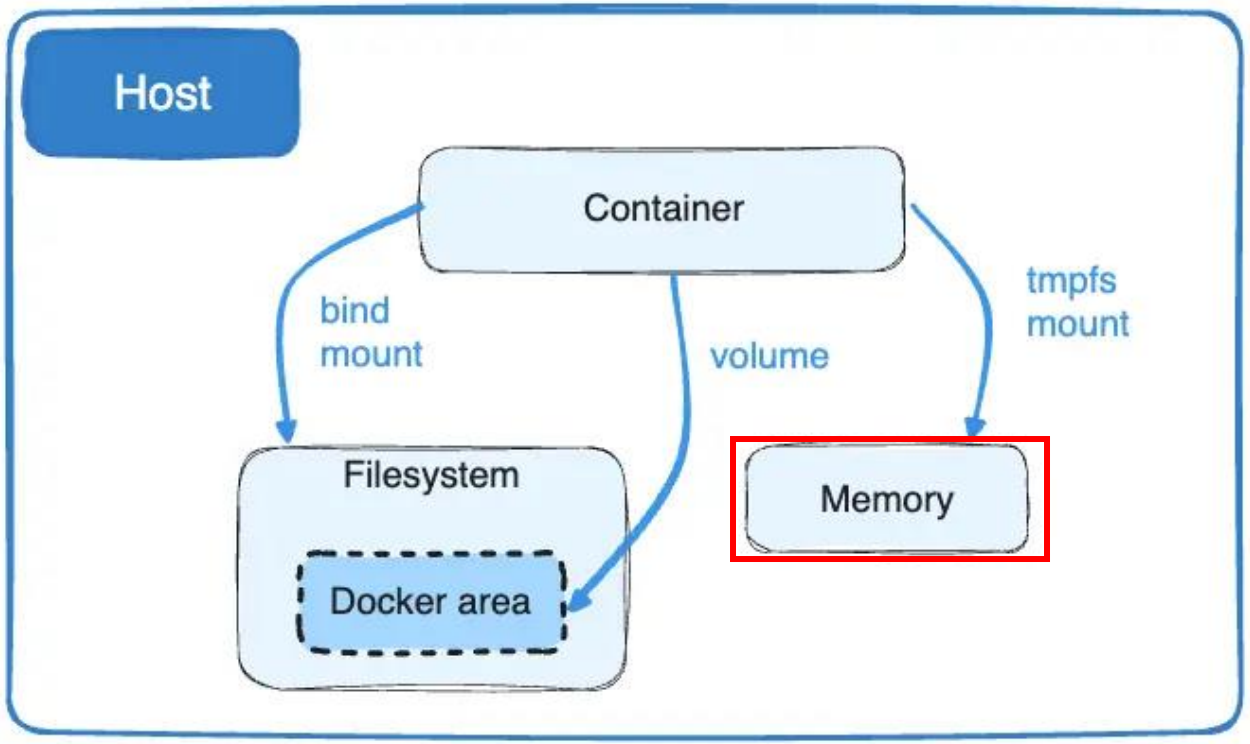


Volumes Docker (fonte: [1])

O que são volumes?

tmpfs

- Não persistentes
- Utilizados durante o ciclo de vida do contêiner
- Informações sensíveis ou estado não-persistente



Volumes Docker (fonte: [1])

Criação, montagem e remoção de volumes

Criação, montagem e remoção de volumes

- Temos uma interface para a gestão de volumes
  - `docker volume`

```
~ » docker volume --help                                    hrchlhck@hrchlhck

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
```

Criação, montagem e remoção de volumes

Criação de volumes

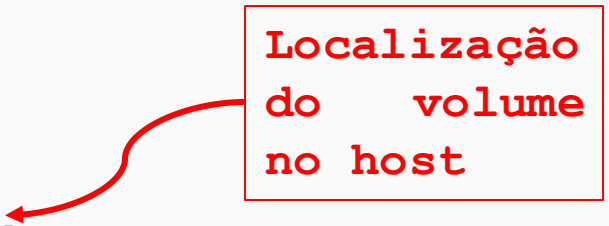
- Vamos criar um volume e inspecioná-lo

```
1. docker volume create teste
2. docker volume inspect teste
```

Criação, montagem e remoção de volumes

Criação de volumes

```
~ » docker volume create teste                                    hrchlhck@hrchlhck
teste
------------------------------------------------------------------------------------
~ » docker volume inspect teste                                   hrchlhck@hrchlhck
[
    {
        "CreatedAt": "2024-05-27T14:04:25Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/teste/_data",
        "Name": "teste",
        "Options": null,
        "Scope": "local"
    }
]
```



Criação, montagem e remoção de volumes

Montagem de volumes

- A montagem é feita através da flag `-v` / `--volume` no `docker run`

- Vamos montar o volume teste em um contêiner qualquer
  - `docker run --rm -it -v teste:/app --name aula07 ubuntu bash`

- Crie um arquivo no diretório `/app` e saia do contêiner

Criação, montagem e remoção de volumes

Montagem de volumes

– Onde está o arquivo que criamos?

```
ubuntu@arm:~$ docker volume inspect teste
[
    {
        "CreatedAt": "2024-05-27T14:03:55Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/snap/docker/common/var-lib-docker/volumes/teste/_data",
        "Name": "teste",
        "Options": null,
        "Scope": "local"
    }
]
```

Criação, montagem e remoção de volumes

Montagem de volumes

– Onde está o arquivo que criamos?

```
ubuntu@arm:~$ sudo su
[sudo] password for ubuntu:
root@arm:/home/ubuntu# cd /var/snap/docker/common/var-lib-docker/volumes/teste/_data/
root@arm:/var/snap/docker/common/var-lib-docker/volumes/teste/_data# ls
a
root@arm:/var/snap/docker/common/var-lib-docker/volumes/teste/_data# cat a
ola
```

Criação, montagem e remoção de volumes

Definindo volumes nas imagens

- Podemos criar volumes no momento de criação da imagem (no `Dockerfile`)
  - Volumes são criados automaticamente
  - SHA-256 associado
  - Comum em aplicações que **precisam** de persistência (bancos de dados, por exemplo)

- Instrução VOLUME no `Dockerfile`

Criação, montagem e remoção de volumes

Definindo volumes nas imagens

- Criação de um volume no caminho `/app/data` (dentro do contêiner)

- Vamos inspecionar o volume

```
FROM python:3.12
RUN mkdir -p /app
...
CMD ["python", "main.py"]
VOLUME ["/app/data"]
```

Criação, montagem e remoção de volumes

Definindo volumes nas imagens

```
ubuntu@arm:~$ docker ps
CONTAINER ID   IMAGE     COMMAND             CREATED         STATUS         PORTS                                       NAMES
d871f0078b31   teste     "python main.py"    2 minutes ago   Up 2 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   tender_johnson
```

Criação, montagem e remoção de volumes

Definindo volumes nas imagens

```
ubuntu@arm:~$ docker inspect d8 -f "{{json .Mounts }}" | jq
[
  {
    "Type": "volume",
    "Name": "8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b",
    "Source": "/var/snap/docker/common/var-lib-docker/volumes/8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b/_data",
    "Destination": "/app/data",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
]
```

Criação, montagem e remoção de volumes

Definindo volumes nas imagens

```
ubuntu@arm:~$ docker volume ls
DRIVER    VOLUME NAME
local     3bc00f2d7300b0fc128dd1892d3a662037a71a50c35d0fb3dd19ab3ee85c73cb
local     8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b
local     teste
ubuntu@arm:~$ docker volume inspect 8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b
[
    {
        "CreatedAt": "2024-05-27T17:13:14Z",
        "Driver": "local",
        "Labels": {
            "com.docker.volume.anonymous": ""
        },
        "Mountpoint": "/var/snap/docker/common/var-lib-docker/volumes/8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b/
_data",
        "Name": "8ef9e92bb2aec841b77cdf9a14a84e9abd194a5b294ea70173c9a872f077c11b",
        "Options": null,
        "Scope": "local"
    }
]
```

Compartilhamento de dados entre contêineres

Compartilhamento de dados entre contêineres

Montagem de volumes

- Vamos compartilhar esse volume com um novo contêiner
  - `docker run -v teste:/app --rm -it alpine sh`

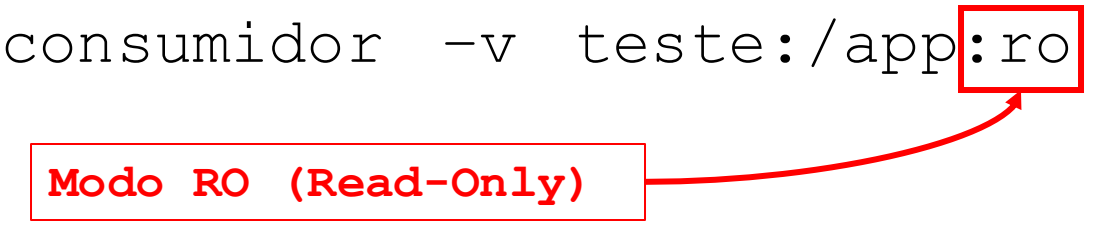
```
ubuntu@arm:~$ docker run -v teste:/app -it --rm alpine sh
/ # ls
app    dev    home   media  opt    root   sbin   sys    usr
bin    etc    lib    mnt    proc   run    srv    tmp    var
/ # cd app
/app # ls
a
/app # cat a
ola
/app #
```

Compartilhamento de dados entre contêineres

Montagem de volumes

- Podemos compartilhar dados entre dois ou mais contêineres **simultaneamente**
  - Problema de condição de corrida se não tratado corretamente!
  - Problema do produtor/consumidor

- Exemplo (**cada comando é executado em um terminal diferente**):
  - `docker run -it --rm --name produtor -v teste:/app ubuntu bash`
  - `docker run -it --rm --name consumidor -v teste:/app:ro ubuntu bash`



Compartilhamento de dados entre contêineres

Usando volumes do host

- Podemos mapear um diretório/arquivo do host para dentro do contêiner
  - Permite a criação de ambientes de desenvolvimento
  - Os dados não serão removidos, pois estarão salvos diretamente no host

Exemplos de como fazer esse mapeamento:

- `docker run -it -v $(pwd)/src:/app/src ubuntu`
- `docker run -it -v ./src:/src ubuntu`
- `docker run -it -v /home/ubuntu/diretorio/:/app ubuntu`

Configuração de contêineres e variáveis de ambiente

Configuração de contêineres e variáveis de ambiente

- Valores de configuração de aplicações utilizam variáveis de ambiente
  - Distinção entre ambiente de teste, staging e produção
  - Personalização da aplicação **sem alterar** diretamente o contêiner/imagem

```
ubuntu@arm:~$ export
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"
declare -x HOME="/home/ubuntu"
declare -x LANG="C.UTF-8"
declare -x LC_CTYPE="C.UTF-8"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="ubuntu"
```

Configuração de contêineres e variáveis de ambiente

- Exemplo de variáveis de ambiente de um contêiner
  - Podemos adicionar/alterar através do parâmetro `--env/-e`

  `--env <nome>=<valor> / -e <nome>=<valor>`

```
ubuntu@arm:~$ docker run -it --rm alpine
/ # export
export HOME='/root'
export HOSTNAME='f4be350ab352'
export PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
export PWD='/'
export SHLVL='1'
export TERM='xterm'
```

Configuração de contêineres e variáveis de ambiente

- Exemplo de variáveis de ambiente de um contêiner
  - Podemos adicionar/alterar através do parâmetro `--env/-e`

`--env <nome>=<valor> / -e <nome>=<valor>`

```
ubuntu@arm:~$ docker run -e MINHA_VARIAVEL=aula07 -it --rm alpine
/ # export
export HOME='/root'
export HOSTNAME='ecf2b029950f'
export MINHA_VARIAVEL='aula07'
export PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
export PWD='/'
export SHLVL='1'
export TERM='xterm'
```

Configuração de contêineres e variáveis de ambiente

- Também podemos criar um arquivo de configuração e carregá-lo no contêiner (parâmetro `--env-file <arquivo de configuração>`)

```
ubuntu@arm:~$ cat dev.env
SERVER_IP=10.0.0.2
SERVER_PORT=8080

ubuntu@arm:~$ docker run -it --rm --env-file dev.env alpine
/ # export
export HOME='/root'
export HOSTNAME='e88316e5d756'
export PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
export PWD='/'
export SERVER_IP='10.0.0.2'
export SERVER_PORT='8080'
export SHLVL='1'
export TERM='xterm'
```

Referências

Referências

1. Docker. **Volumes | Docker Docs**. Docker Inc, 2024. Disponível em: <https://docs.docker.com/storage/volumes/>. Acesso em 27 de maio de 2024.
2. Docker. **docker image**. Docker Inc, 2024. Disponível em: https://docs.docker.com/reference/cli/docker/image/>. Acesso em: 12 de maio de 2024.
3. Docker. **Docker overview**. Docker Inc., 2024. Disponível em: <https://docs.docker.com/get-started/overview/#docker-architecture>. Acesso em 12 de maio de 2024.
4. Docker. **Dockerfile reference**. Docker Inc., 2024. Disponível em: <https://docs.docker.com/reference/dockerfile/>. Acesso em 12 de maio de 2024.

Referências

5. RICE, Liz. **Container security: fundamental technology concepts that protect containerized applications**. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2020.
6. SCHENKER, Gabriel Nicolas. **The ultimate Docker container book: build, test, ship, and run containers with Docker and Kubernetes**. Third edition. Birmingham, UK: Packt Publishing Ltd., 2023.

Contato: p.horchulhack@pucpr.br