

Universidade Federal do Piauí - UFPI

Campus Senador Helvídio Nunes Barros - CSHNB

Sistemas de Informação - Sistemas Distribuídos - prof. Rayner Gomes

Aluna: Vandırleya Barbosa da Costa

Tema: Processos

5º Lista de Exercícios de Fixação - 5º Semana

Atenção: Para todas as perguntas, cite o *site*, ou o material utilizado como fonte da pesquisa. Como uma atividade de formação intelectual, não copie e cole, o intuito não é saber sua capacidade de apertar CTRL-C e CTRL-V, mas sua capacidade de resumir, explicar e transmitir um novo conteúdo.

- 1. [DOCKER] Para todos os itens abaixo, faça uma pesquisa de suas funcionalidades e de como são usados. Faça um mini-manual de cada um. Adicione no final de sua resposta com exemplos em sua máquina, registre as telas de execução (print screen) de todos os passos e explique-os.**

a. Dockerfile

O Dockerfile é um arquivo de configuração utilizado para automatizar a criação de imagens Docker. Ele contém uma série de instruções em texto que o Docker usa para definir os passos de configuração de um ambiente, como instalar pacotes, definir variáveis de ambiente e especificar comandos que serão executados quando o contêiner iniciar. Cada linha do Dockerfile representa uma camada na construção da imagem, o que permite uma gestão eficiente e reutilização de recursos. O uso do Dockerfile facilita o processo de criação e

configuração de contêineres, garantindo que o ambiente seja replicável e padronizado.

Como usar:

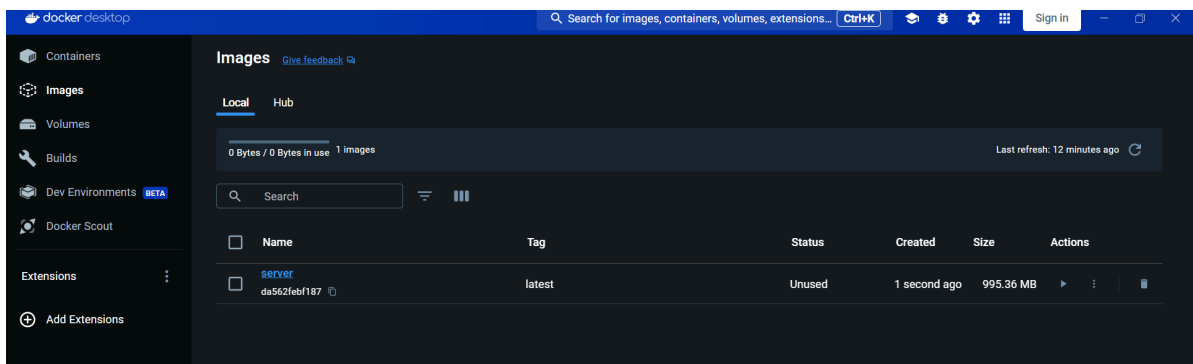
1. Crie um arquivo Dockerfile na raiz do seu projeto.

```
ServerDockerFile.dockerfile X
Lista 1 > ServerDockerFile.dockerfile > ...
1 # Use uma imagem base do Python
2 FROM python:3.8
3
4 # Defina o diretório de trabalho no container
5 WORKDIR /app
6
7 # Copie o arquivo do servidor para o container
8 COPY server.py .
9
10 # Execute o servidor quando o container for iniciado
11 CMD ["python", "server.py"]
12
```

2. Escreva as instruções necessárias para configurar o ambiente do seu container.

```
Start a build
PS C:\Users\vandi\OneDrive\Documentos\RedesIT\Lista 1> docker build -t server -f ServerDockerFile.dockerfile .
2024/10/28 20:55:20 http2: server: error reading preface from client //.pipe/docker_engine: file has already been closed
[+] Building 48.8s (8/8) FINISHED
=> [internal] load build definition from ServerDockerFile.dockerfile
=> => transferring dockerfile: 323B
=> [internal] load metadata for docker.io/library/python:3.8
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/python:3.8@sha256:d411270700143fa2683cc8264d9fa5d3279fd3b6afff62ae81ea2f9d070e390c
=> => resolve docker.io/library/python:3.8@sha256:d411270700143fa2683cc8264d9fa5d3279fd3b6afff62ae81ea2f9d070e390c
=> => sha256:7aa279fb41dad2962d3c915aa6f6615134baa412ab5aafa9d4384dcaaa0af15d 2.32kB / 2.32kB
=> => sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2511c508fa4c0a9b1975ffe20e 49.56MB / 49.56MB
=> => sha256:a47c7f7f31e941e78bf63ca19f0811b675283e2c00ddea10c57f78d93b2bc343 24.05MB / 24.05MB
=> => sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480b51f768a19332dfa83d7722a5 64.39MB / 64.39MB
=> => sha256:d411270700143fa2683cc8264d9fa5d3279fd3b6afff62ae81ea2f9d070e390c 10.35kB / 10.35kB
=> => sha256:3ea6ead4f175bd42f39dae10098b1820ee522628ff04670b3ac8b89fe00c1c8 5.97kB / 5.97kB
=> => sha256:01272fe8adbacc44afd2b92994b31c40a151f4324ca392050d9e8d580927dd32 211.27MB / 211.27MB
=> => extracting sha256:cdd62bf39133c498a16f7a7b1b6555ba43d02b2511c508fa4c0a9b1975ffe20e
=> => sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a91dab6c092bb353f82098b09 6.16MB / 6.16MB
=> => extracting sha256:a47c7f7f31e941e78bf63ca19f0811b675283e2c00ddea10c57f78d93b2bc343
=> => sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3335fe9b7811b8f764abb1a4ac 18.06MB / 18.06MB
=> => sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6a4f2a3fbf7baa902f6a 249B / 249B
=> => extracting sha256:a173f2aee8e962ea19db1e418ae84a0c9f71480b51f768a19332dfa83d7722a5
=> => extracting sha256:01272fe8adbacc44afd2b92994b31c40a151f4324ca392050d9e8d580927dd32
=> => extracting sha256:cddc73e4e6c704bfa2325e53c32ddb3553c8fc3a91dab6c092bb353f82098b09
=> => extracting sha256:cc48f13b5f0f44b2e298de83a94a99fe7abdfb3335fe9b7811b8f764abb1a4ac
=> => extracting sha256:5a98c896c047f960c5fd29d44fa778899a68e7ebfb6a6a4f2a3fbf7baa902f6a
=> [internal] load build context
=> => transferring context: 1.06kB
=> [2/3] WORKDIR /app
=> [3/3] COPY server.py .
=> => exporting to image
=> => exporting layers
=> => writing image sha256:da562febf187df8acc4f0573e5f5a772a0ac164f83dcf779257c4f74356b987
=> => naming to docker.io/library/server
```

3. Execute Docker build para criar a imagem baseada no Dockerfile.



b. Volumes

Volumes no Docker são um mecanismo de armazenamento que permite que dados sejam mantidos de forma persistente, mesmo que o contêiner em execução seja removido ou recriado. Ao contrário do sistema de arquivos temporário dos contêineres, os volumes são armazenados fora do ciclo de vida do contêiner, tornando-os ideais para dados que precisam ser preservados entre execuções, como bancos de dados e arquivos de configuração. Essa abordagem de armazenamento permite uma integração eficiente entre o sistema de arquivos local e o contêiner, oferecendo flexibilidade e segurança na gestão de dados.

Como usar:

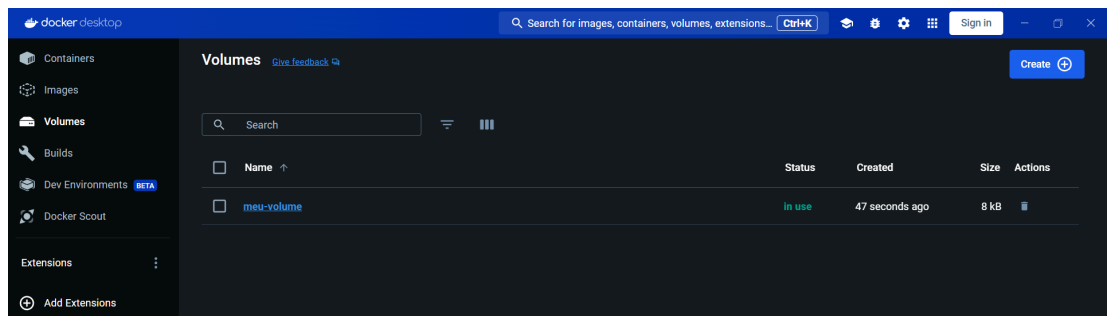
1. Crie um volume Docker com `docker volume create`

```
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> docker volume create meu-volume
meu-volume
```

2. Monte o volume usando a flag `-v` ao rodar um container.

```
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> docker run -d -v meu-volume:/app/data ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
ff65ddf9395b: Pull complete
Digest: sha256:99c35190e22d294cdace2783ac55effc69d32896daaa265f0bbbedbcde4fbc3e5
Status: Downloaded newer image for ubuntu:latest
ea8639184420d3739317dafc373f56af59cc6a12e6de2ad41e79f177aecc4fab
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> |
```

3. Acesse e manipule dados no diretório especificado dentro do container.



c. Gerenciamento Imagens

O gerenciamento de imagens no Docker abrange uma série de operações, como listar, baixar, remover e limpar imagens para otimizar o uso de armazenamento e manter o ambiente organizado. As imagens são "templates" essenciais para a criação dos contêineres, contendo todo o necessário para executar uma aplicação, incluindo o sistema operacional, dependências e configurações. O Docker facilita esse processo por meio de comandos como `docker images`, para visualizar imagens, e `docker rmi`, para remover aquelas que não são mais necessárias. Com isso, o gerenciamento de imagens se torna uma prática importante para assegurar que o ambiente de contêineres permaneça eficiente e que os recursos de armazenamento estejam sempre sob controle.

Como usar:

1. Utilize `docker build` para criar novas imagens (crie imagens e liste elas).

```
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
server        latest    da562febf187   20 minutes ago 995MB
ubuntu        latest    59ab366372d5   2 weeks ago   78.1MB
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1>
```

2. Gerencie imagens existentes usando `docker images` e `docker rmi` (remover uma).

```
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> docker rmi server
Untagged: server:latest
Deleted: sha256:da562febf187df87acc4f0573e5f5a772a0ac164f83dcf779257c4f74356b987
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1>
```

3. Distribua imagens via docker pull para repositórios remotos.

```
PS C:\Users\vandi\OneDrive\Documentos\RedesII\Lista 1> docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:99c35190e22d294cdace2783ac55effc69d32896daaa265f0bbbedbcde4fbe3e5
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Bom Trabalho!