Tópico 1: Conceitos Básicos do Docker

1. **Comando: docker --version**
    - **Descrição:** Verificar a versão do Docker instalada.
    - **Execução:** Execute o comando acima no terminal.
2. **Comando: docker info**
    - **Descrição:** Exibir informações detalhadas sobre o ambiente Docker.
    - **Execução:** Execute o comando acima no terminal.

Tópico 2: Manipulação de Contêineres

1. **Comando: docker pull nginx:latest**
    - **Descrição:** Baixar a imagem mais recente do servidor web Nginx do Docker Hub.
    - **Execução:** Execute o comando acima no terminal e aguarde o download.
2. **Comando: docker images**
    - **Descrição:** Listar todas as imagens Docker baixadas.
    - **Execução:** Execute o comando acima para ver a imagem do Nginx que você baixou.
3. **Comando: docker run -d --name meu_nginx -p 8080:80 nginx:latest**
    - **Descrição:** Executar um contêiner a partir da imagem Nginx, mapeando a porta 8080 do host para a porta 80 do contêiner.
    - **Execução:** Execute o comando acima para iniciar o contêiner.
4. **Comando: docker ps**
    - **Descrição:** Listar os contêineres em execução.
    - **Execução:** Execute o comando acima para ver o contêiner Nginx em execução.
5. **Comando: docker exec -it meu_nginx bash**
    - **Descrição:** Acessar o terminal interativo dentro do contêiner Nginx.
    - **Execução:** Execute o comando acima para entrar no terminal do contêiner.

Tópico 3: Gerenciamento de Imagens e Contêineres

1. **Comando: docker stop meu_nginx**
    - **Descrição:** Parar o contêiner Nginx em execução.

- **Execução:** Execute o comando acima para parar o contêiner.

2. **Comando: docker rm meu_nginx**
    - **Descrição:** Remover o contêiner Nginx.
    - **Execução:** Execute o comando acima para remover o contêiner.
3. **Comando: docker images**
    - **Descrição:** Verificar novamente a lista de imagens disponíveis.
    - **Execução:** Execute o comando acima para ver que o contêiner foi removido, mas a imagem permanece.
4. **Comando: docker rmi nginx:latest**
    - **Descrição:** Remover a imagem Nginx que não é mais necessária.
    - **Execução:** Execute o comando acima para remover a imagem.

Tópico 4: Criação de Imagens Personalizadas

1. **Criação do Dockerfile:** Crie um arquivo chamado **Dockerfile** em um diretório de sua escolha. Adicione o seguinte conteúdo ao Dockerfile:

```
# Use a base image

FROM ubuntu:latest

# Set the working directory

WORKDIR /app

# Copy a sample file into the container

COPY hello.txt .

# Define the command to run

CMD ["cat", "hello.txt"]
```

1. **Comando: docker build -t my-custom-image:latest /path/to/your/dockerfile/directory**
    - **Descrição:** Construir uma nova imagem personalizada usando o Dockerfile no diretório especificado.
    - **Execução:** Substitua **/path/to/your/dockerfile/directory** pelo caminho real e execute o comando acima para criar a imagem personalizada.
2. **Comando: docker images**
    - **Descrição:** Verificar a nova imagem personalizada criada.

- **Execução:** Execute o comando acima para ver a imagem personalizada.

Tópico 5: Compartilhamento de Imagens

1. **Comando: docker login**
    - **Descrição:** Fazer login na sua conta do Docker Hub.
    - **Execução:** Execute o comando acima para fazer o login.
2. **Comando: docker tag my-custom-image:latest seu-usuario/my-custom-image:latest**
    - **Descrição:** Marcar sua imagem personalizada para que possa ser enviada para o Docker Hub.
    - **Execução:** Substitua **seu-usuario** pelo seu nome de usuário no Docker Hub e execute o comando.
3. **Comando: docker push seu-usuario/my-custom-image:latest**
    - **Descrição:** Enviar a imagem personalizada para o Docker Hub.
    - **Execução:** Execute o comando acima para enviar a imagem.

Tópico 6: Limpando o Ambiente

1. **Comando: docker system prune -a**
    - **Descrição:** Limpar todos os contêineres, redes e imagens não utilizados.
    - **Execução:** Execute o comando acima para limpar o ambiente.

Criando um Ambiente Docker com MySQL e phpMyAdmin usando Docker Compose

Passo 1: Criar o arquivo docker-compose.yml

1. Crie um diretório chamado **docker-mysql-phpmyadmin** em um local de sua escolha.
2. Dentro deste diretório, crie um arquivo chamado **docker-compose.yml** com o seguinte conteúdo:

```
version: '3'

services:

  mysql:
```

```
    image: mysql:latest
    container_name: my-mysql-container
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: my_database
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin:latest
    container_name: my-phpmyadmin-container
    environment:
      PMA_HOST: mysql
      PMA_PORT: 3306
    ports:
      - "8080:80"
volumes:
  mysql-data:
```

Passo 2: Executar o Docker Compose

1. No terminal, navegue até o diretório **docker-mysql-phpmyadmin** que você criou.
2. Execute o seguinte comando para iniciar os contêineres:

```
docker-compose up -d
```

3. O Docker Compose começará a baixar as imagens necessárias e criar os contêineres. Aguarde até que a operação seja concluída.

Passo 3: Acessar o phpMyAdmin

1. Abra um navegador da web e acesse **http://localhost:8080**.
2. Faça login com o usuário **root** e a senha que você definiu no arquivo **docker-compose.yml** (no exemplo, **root_password**).
3. Você terá acesso ao phpMyAdmin e poderá administrar o banco de dados MySQL criado.

Passo 4: Interagir com o MySQL

1. Você pode se conectar ao MySQL usando ferramentas de linha de comando ou clientes MySQL, usando o host **localhost**, a porta **3306**, o nome de usuário **root** e a senha definida.

Passo 5: Encerrar e Limpar o Ambiente

1. No diretório **docker-mysql-phpmyadmin**, execute o seguinte comando para encerrar os contêineres:

```
docker-compose down
```

2. Isso encerrará e removerá os contêineres e as redes criadas pelo Docker Compose.

Volumes no Docker: Em um ambiente Docker, os volumes são uma maneira de persistir dados gerados e usados pelos contêineres. Eles permitem que os dados sejam compartilhados e mantidos mesmo quando um contêiner é destruído ou recriado. Os volumes são essenciais para garantir que os dados sejam armazenados além do ciclo de vida dos contêineres.

No Docker, você pode montar um volume em um contêiner usando a opção **-v** ou **--volume** no momento da execução ou definindo volumes em um arquivo de composição, como o **docker-compose.yml**. Volumes podem ser montados a partir de diretórios do host ou de um contêiner existente.

Volumes no MySQL: No contexto do MySQL, o uso de volumes é crucial para garantir que os dados do banco de dados sejam preservados entre diferentes execuções do contêiner MySQL. Ao criar um contêiner MySQL, você pode definir um volume para armazenar os dados do banco de dados, como as tabelas, os esquemas e outras informações. Dessa forma, quando o contêiner é encerrado ou recriado, os dados persistem no volume.

No exemplo do exercício anterior, você viu um volume sendo usado para o contêiner MySQL no arquivo **docker-compose.yml**:

```
volumes:
  - mysql-data:/var/lib/mysql
```

Aqui, **mysql-data** é o nome do volume, e ele é mapeado para o diretório **/var/lib/mysql** dentro do contêiner MySQL. Isso permite que os dados do banco de dados sejam armazenados no volume e não se percam quando o contêiner é encerrado.

O uso de volumes para o MySQL é altamente recomendado em ambientes de produção, pois garante que os dados do banco de dados sejam mantidos mesmo em caso de atualizações, recriações ou falhas do contêiner.

Em resumo, volumes no MySQL e Docker são uma maneira de garantir a persistência e a disponibilidade dos dados do banco de dados, permitindo que eles sejam mantidos independentemente do ciclo de vida dos contêineres.