USANDO DOCKER PLATAFORM

O Desafio 49 requer a criação de uma infraestrutura que utilize o servidor-web NGNIX, um aplicativo PHP e um banco de dados MySQL.
Seguimos os seguintes passos para realizar esse desafio:

1- Estudamos os conceitos teóricos sobre o que é o docker plataform
2 - Abrimos o terminal no sistema operacional
3 - Fizemos a verificação de um container de teste simples para ver o funcionamento do docker na máquina. Executamos o ‘docker run hello-world’ que foi bem sucedido.

4- Em consulta ao ChatGPT, solicitamos o seguinte prompt “criar um arquivo docker-compose.yml que configure um ambiente composto por um servidor NGINX, um aplicativo PHP-FPM, um banco de dados MySQL e volumes de dados.”
5- Como resposta, foi nos apresentada a seguinte configuração

```
version: '3'

services:
  nginx:
    image: nginx
    ports:
      - "80:80"
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d
      - ./nginx/html:/usr/share/nginx/html
    depends_on:
      - php-fpm

  php-fpm:
    image: php:7.4-fpm
    volumes:
      - ./php-fpm:/var/www/html
    depends_on:
      - db

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: example
      MYSQL_USER: example
      MYSQL_PASSWORD: example
    volumes:
      - ./mysql/data:/var/lib/mysql
```

6 - Na máquina local, criamos uma pasta chamada ‘docker’
5- Abrimos o programa notepad (editor de texto) e colamos o código para configuração dos serviços que foi informado no chatGPT
7- O arquivo foi salvo como docker-compose.yml

8- Retornando ao terminal do sistema operacional, navegamos até o diretório onde o arquivo foi salvo e executamos o comando **docker-compose up** para iniciar os serviços definidos no arquivo
9- O Docker Compose começou a criar e configurar os contêineres necessários com base nas definições do arquivo

10 - Durante o processo de configuração deu erro na imagem NGNIX, em razão de não haver portas disponíveis *(ERROR: for docker_nginx_1 Cannot start service nginx: Ports are not available: exposing port TCP 0.0.0.0:80 -> 0.0.0.0:0: listen tcp 0.0.0.0:80: bind: address already in use ERROR: for nginx Cannot start service nginx: Ports are not available: exposing port TCP 0.0.0.0:80 -> 0.0.0.0:0: listen tcp 0.0.0.0:80: bind: address already in use ERROR: Encountered errors while bringing up the project)*.

11 - Para nos certificarmos de que a configuração dos serviços no arquivo docker-compose.yml estava funcionando adequadamente, fizemos um teste no https://labs.play-with-docker.com/ (docker play ground) e verificamos que tudo funcionou corretamente.

OBS: Não criamos dockerfiles individualmente para cada uma das imagens (servidor web, php e mysql). Partimos diretamente para a criação do docker-compose que é um único arquivo que engloba todas as imagens conjuntamente e que definiu todos os serviços e suas configurações para serem executados juntos em um ambiente Docker.
Na verdade, o docker-compose foi usado para definir um ambiente de vários containers (servidor, php e mysql) em que foram definidas as dependências entre os contêineres e suas configurações.

Nesse contexto, é importante lembrar que o docker-compose é uma ferramenta que usa o arquivo docker-compose.yml para automatizar o processo de criação, inicialização e interconexão de vários contêineres Docker, ou seja, é possível usar o docker-compose para executar vários contêineres ao mesmo tempo em vez de executá-los individualmente usando comandos docker.

—----------
EXPLICAÇÃO da configuração do NGNIX, PHP, MYSQL, suas configurações e dependências

A configuração do docker-compose.yml que você compartilhou define três serviços: nginx, php-fpm e db. Vou explicar cada um deles e suas configurações:

1. Serviço nginx:

- Imagem: Utiliza a imagem padrão do NGINX disponível no Docker Hub.
- Portas: Mapeia a porta 80 do contêiner NGINX para a porta 80 do host.
- Volumes: Monta dois volumes para o contêiner NGINX. O primeiro volume ./nginx/conf.d mapeia o diretório local nginx/conf.d para o diretório /etc/nginx/conf.d no contêiner. O segundo volume ./nginx/html mapeia o diretório local nginx/html para o diretório /usr/share/nginx/html no contêiner.
- Depends_on: Define a dependência do serviço nginx em relação ao serviço php-fpm. Isso garante que o serviço php-fpm seja iniciado antes do serviço nginx.

2. Serviço php-fpm:
    - Imagem: Utiliza a imagem PHP 7.4-FPM disponível no Docker Hub.
    - Volumes: Monta um volume que mapeia o diretório local php-fpm para o diretório /var/www/html no contêiner PHP-FPM.
    - Depends_on: Define a dependência do serviço php-fpm em relação ao serviço db. Isso garante que o serviço db seja iniciado antes do serviço php-fpm.
3. Serviço db:
    - Imagem: Utiliza a imagem MySQL 5.7 disponível no Docker Hub.
    - Restart: Configura o serviço para reiniciar sempre que ele for parado ou ocorrer um erro.
    - Environment: Define variáveis de ambiente para configurar o MySQL. Isso inclui a senha do root (MYSQL_ROOT_PASSWORD), o nome do banco de dados (MYSQL_DATABASE), o nome de usuário (MYSQL_USER) e a senha do usuário (MYSQL_PASSWORD).
    - Volumes: Monta um volume que mapeia o diretório local mysql/data para o diretório /var/lib/mysql no contêiner MySQL.

Essa configuração define uma pilha básica para um ambiente de desenvolvimento local com NGINX, PHP-FPM e MySQL. O NGINX é usado como servidor web, o PHP-FPM é usado para processamento de scripts PHP e o MySQL é usado como banco de dados. Cada serviço é configurado com suas dependências e volumes adequados para compartilhamento de dados.

Lembre-se de que essa é apenas uma configuração inicial e você pode ajustá-la conforme necessário, adicionando mais serviços, configurando portas, volumes e variáveis de ambiente de acordo com os requisitos da sua aplicação.