PHP – Docker

Aula 0.2 - Criar uma Dockerfile no seu projeto PHP

O Dockerfile nada mais é do que um meio que utilizamos para criar nossas próprias imagens. Em outras palavras, ele serve como a receita para construir um container, permitindo definir um ambiente personalizado e próprio para meu projeto pessoal ou empresarial.

```
FROM php:8.2-cli
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
CMD [ "php", "./your-script.php" ]
```

Vou explicar cada parte do Dockerfile:

1. ***FROM php:8.2-cli:*** Esta linha especifica a imagem base que será usada como ponto de partida para a construção da imagem Docker. Neste caso, a imagem base é "php:8.2-cli", que é uma imagem oficial do PHP 8.2 para uso em ambientes de linha de comando (CLI).
2. ***COPY . /usr/src/myapp:*** Essa instrução copia o conteúdo do diretório local (o diretório onde o Dockerfile está localizado) para o diretório "/usr/src/myapp" dentro do contêiner que está sendo construído. Isso permite que os arquivos e diretórios do host sejam copiados para o contêiner, para que possam ser usados durante a execução do contêiner.
3. ***WORKDIR /usr/src/myapp:*** Esta linha define o diretório de trabalho padrão dentro do contêiner como "/usr/src/myapp". Isso significa que, quando o contêiner for executado, o Docker mudará automaticamente para este diretório como o diretório de trabalho atual. Isso facilita a execução de comandos e scripts a partir desse diretório.
4. ***CMD ["php", "./your-script.php"]***: Esta instrução define o comando padrão a ser executado quando o contêiner for iniciado. Ele especifica que o comando a ser executado é "php ./your-script.php", o que significa que o PHP será executado com o script "your-script.php" como argumento. Isso inicia a execução do script PHP assim que o contêiner é iniciado.

Em resumo, este Dockerfile cria uma imagem Docker baseada na imagem "php:8.2-cli", copia os arquivos do diretório local para o contêiner, define o diretório de trabalho e especifica o comando padrão a ser executado, que é a execução do script PHP "your-script.php". Quando você construir um contêiner a partir deste Dockerfile e iniciá-lo, o PHP será executado com o script fornecido no diretório "/usr/src/myapp" dentro do contêiner.

Em seguida, execute os comandos para construir e executar a imagem Docker:

```
docker build -t my-php-app .
docker run -it --rm --name my-running-app my-php-app
```

Vou explicar cada comando:

1. ***docker build -t my-php-app .:***
   - ***docker build:*** Este comando é usado para construir uma imagem Docker.

◦ ***-t my-php-app:*** A opção -t permite nomear a imagem que está sendo construída. Neste caso, a imagem será nomeada como "my-php-app".

◦ ***.:*** O ponto representa o contexto de construção. Ele indica que o Dockerfile está no diretório atual. O Docker usará o Dockerfile no diretório atual como base para construir a imagem.

Portanto, este comando construirá uma imagem Docker com o nome "my-php-app" usando o Dockerfile no diretório atual.

2. docker run -it --rm --name my-running-app my-php-app:

◦ ***docker run***: Este comando é usado para iniciar a execução de um contêiner a partir de uma imagem Docker.

◦ ***-it:*** Essas opções combinadas alojam um terminal interativo no contêiner. O "i" representa interatividade, permitindo que você interaja com o contêiner, e o "t" especifica a alocação de um terminal.

◦ ***--rm:*** Esta opção instrui o Docker a remover automaticamente o contêiner após a sua finalização. Isso é útil para evitar que contêineres não utilizados ocupem espaço em disco.

◦ ***--name my-running-app:*** Define um nome personalizado para o contêiner. Neste caso, o contêiner é nomeado como "my-running-app" para facilitar a sua identificação.

◦ ***my-php-app:*** Este é o nome da imagem que será usada como base para criar o contêiner. No caso, ele faz referência à imagem Docker criada no passo anterior.

Portanto, este comando executará um contêiner interativo baseado na imagem Docker "my-php-app" que foi criada anteriormente. Esse contêiner será nomeado como "my-running-app". Você poderá interagir com o contêiner por meio de um terminal interativo e, quando a execução do contêiner for encerrada, o Docker o removerá automaticamente, devido à opção "--rm".