LUIZ ZENHA

PÓS TECH

SOFTWARE ARCHITECTURE

DOCKERIZAÇÃO

AULA 03

SUMÁRIO

O QUE VEM POR AÍ? 3
HANDS ON 4
SAIBA MAIS 7
O QUE VOCÊ VIU NESTA AULA? 12
REFERÊNCIAS 13
PALAVRAS-CHAVE 14

O QUE VEM POR AÍ?

Nessa aula, abordaremos o Dockerfile e o Docker Compose, duas ferramentas fundamentais para quem trabalha com Docker.

Construiremos ainda um arquivo Dockerfile e detalharemos seus principais comandos, que serão essenciais para a criação de imagem customizadas. Também falaremos sobre o Docker Compose e veremos como organizar vários containers de forma simples, com o intuito de facilitar nosso dia a dia.

HANDS ON

Em nosso hands on, vamos criar um arquivo Dockerfile, explorando e experimentando os principais comandos e concebendo uma imagem própria. Para isso, vamos utilizar a imagem base do ubuntu e, depois, configuraremos um servidor nginx com essa imagem em que poderíamos, por exemplo, hospedar um site simples em html.

Em seguida, vamos criar o nosso docker compose, no qual subiremos o conteiner com a imagem do Wordpres e um banco de dados MariaDB para rodarmos nosssa aplicação Wordpress básica.

A seguir está o nosso Dockerfile final:

```
FROM ubuntu:22.10
RUN apt-get update
RUN apt-get install nginx -y

VOLUME ["/var/www/html"]

WORKDIR "/var/www/html/"

COPY "index.html" "index.html"

ADD     "https://images.pexels.com/photos/1521304/pexels-photo-
1521304.jpeg" "foto.jpeg"

# Como baixamos o arquivo precisamos alterar a
# permissão para que seja possível acessá-lo
RUN chmod 644 foto.jpeg

ENTRYPOINT ["/usr/sbin/nginx", "-g", "daemon off;"]

EXPOSE 80
```

Código fonte 1 – Dockerfile de exemplo
Fonte: Elaborado pelo autor (2023)

Vejamos a seguir o arquivo docker-compose.yml para executar um projeto wordpress que também utilizamos no nosso Hands On.

```
services:
  db:
    # We use a mariadb image which supports both amd64 & arm64
architecture
    image: mariadb:10.6.4-focal
    # If you really want to use MySQL, uncomment the following
line
    #image: mysql:8.0.27
    command: '--default-authentication-
plugin=mysql_native_password'
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
      - MYSQL_PASSWORD=wordpress
    expose:
      - 3306
      - 33060
  wordpress:
    image: wordpress:latest
    volumes:
      - wp_data:/var/www/html
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=wordpress
      - WORDPRESS_DB_PASSWORD=wordpress
      - WORDPRESS_DB_NAME=wordpress
volumes:
  db_data:
  wp_data:
```

Código Fonte 2 – Exemplo docker-compose,yml
Fonte: Elaborado pelo autor (2023)

SAIBA MAIS

DOCKERFILE

Imagine que, agora, precisamos criar uma imagem para rodar nossa aplicação, seja ela Python, Java ou PHP. Subir um ambiente com todas essas configurações leva algum tempo e, no caso de alterações de dependências que precisem de bibliotecas atreladas ao sistema operacional, seria bem trabalhoso gerenciar todo o ciclo de CI/CD dessas aplicações.

Como as necessidades de cada container pode ser distinta, o melhor caminho seria deixar essas configurações de fácil manutenção na mão da pessoa especialista em desenvolvimento. Para isso temos o arquivo de Dockerfile, no qual podemos criar nossas imagens com todo o ambiente necessário para a execução de nossa aplicação.

Usualmente, precisamos apenas criar um arquivo com o nome Dockerfile sem nenhuma extensão para trabalharmos as configurações que desejarmos; basicamente, seria a receita para a criação daquela imagem, então temos diversos comandos que devem ser executados e seguidos para que aquela receita gere a imagem esperada e, assim, possamos executar o container com ela.

COMANDOS DO DOCKERFILE

FROM

Esse comando é obrigatório e é com ele que definimos a imagem base da nossa aplicação, por exemplo:

```
FROM openjdk
```

A imagem base openjdk é utilizada para executar aplicações JAVA, posto que ela retorna, além do sistema operacional que irá executar, a instalação da versão do Java que necessitamos.

Uma atenção que sempre devemos ter aqui é verificar se já não há uma imagem mais próxima dos seus requisitos, como por exemplo a openjdk, e buscar sempre optar por imagens oficiais ou que passaram por testes de vulnerabilidade e segurança para evitar qualquer comprometimento da sua aplicação.

RUN

Essa instrução também é executada no tempo de criação da imagem, que irá executar comandos da mesma forma que escrevemos comandos via terminal. Dessa forma, podemos instalar ou remover dependências no sistema, alterar configurações, permissões etc.

CMD e ENTRYPOINT

Esses dois comandos têm a mesma função do comando RUN, que é executar um comando, mas elas são executadas no momento da criação do container e não da imagem.

A diferença entre eles é que o comando CMD pode ser sobrescrito na hora de executar o container, ao passo que o comando ENTRYPOINT não permite que seja sobrescrito.

EXPOSE

Esse comando tem como objetivo documentar no seu arquivo Dockerfile com qual porta aquele container terá comunicação. Vale destacar que, mesmo que não coloquemos esse comando no Dockerfile, é possível que seu container possa utilizar determinada porta quando usamos o parâmetro -p no comando Docker run.

VOLUME

Essa instrução é muito importante para que, na criação da imagem, sejam gerados os metadados de que existe esse volume quando o container for executado. Mesmo assim precisamos realizar o mapeamento do volume no momento da execução, na pasta de destino máquina host, para que ela possa ser acessível.

Apesar de podermos mapear qualquer caminho no momento de execução do container, é interessante também como documentação mapearmos as pastas/volumes que podem ser úteis na nossa imagem.

WORKDIR

O WORKDIR é onde iremos definir nosso local de execução dos comandos, principalmente dos **CMD, RUN, ENTRYPOINT, ADD e COPY**. Esse caminho que definirmos também será o ponto de abertura do container quando executarmos ele no modo interativo.

ADD e COPY

Ambos os comandos têm nomes bastante intuitivos e fazem quase a mesma coisa, porém o **COPY** só faz a cópia de arquivos e pastas da máquina que estamos realizando o build da imagem para dentro da imagem. Já o comando **ADD** possui mais algumas funcionalidades, como baixar o arquivo de uma URL ou extrair de forma automática arquivos compactados em formato tar.

Com isso, caso não precise extrair automaticamente um arquivo tar ou baixar o arquivo de uma fonte externa, deve-se sempre preferir usar o **COPY** para um controle maior do que está acontecendo no processo de build da imagem.

Esses são os principais comandos e os mais utilizados na criação de uma imagem. Em nosso vídeo, veremos como usar cada um deles, como construir nosso Dockerfile e como utilizar todos esses comandos.

DOCKER COMPOSE

Docker Compose é uma ferramenta de orquestração de contêineres que permite definir, configurar e executar vários containers Docker como um aplicativo. Com ele, é possível criar e gerenciar aplicativos Docker complexos, compostos por vários serviços e contêineres, com facilidade e eficiência.

A principal vantagem do Docker Compose é que ele permite que se gerencie todos os aspectos de seus aplicativos Docker de uma só vez. Ao invés de gerenciar

manualmente cada container e serviço, o Docker Compose permite que os indivíduos que o utilizam definam as configurações e parâmetros de cada serviço em um único arquivo YAML.

Uma boa prática recomenda que esse arquivo seja versionado e compartilhado com outros membros da equipe, a fim de que todos(as) possam trabalhar com o mesmo conjunto de definições e configurações, facilitando assim principalmente a integração e configuração do ambiente de desenvolvimento de novos membros desse time.

Com o Docker Compose, é possível definir rede, volumes, portas e dependências de cada serviço, além de especificar imagens personalizadas e variáveis de ambiente. Ele também facilita a escalabilidade e a implementação de aplicativos em vários ambientes, permitindo criar e testar os aplicativos em suas máquinas locais antes de implantá-los em produção.

Podemos, por exemplo, preparar um ambiente de desenvolvimento para funcionar localmente que necessite de um servidor web, com um banco de dados e uma ferramenta de cache em que ficarão alguns dados de consulta. Com o Docker Compose é possível escrever essa configuração em um arquivo e subir toda essa infraestrutura com apenas uma linha de comando.

Para isso, no Docker Compose, usamos arquivos YAML, sendo que iremos descrever e definir todos os serviços que vamos precisar.

O nome padrão do arquivo é docker-compose.yml e ele deve ser colocado na pasta raiz do nosso container. Para executá-lo, basta rodar o comando:

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

Comando 1 – Executar docker compose
Fonte: Elaborado pelo autor (2023)

Caso você precise especificar um arquivo de nome diferente, podemos fazer isso utilizando a flag -f no comando docker compose:

```
$ docker compose -f FILENAME.yml up -d
```

Comando 2 – Iniciando docker composse com comando -f
Fonte: Elaborado pelo autor (2023)

O parâmetro -d serve para não ficarmos com o terminal preso na execução dos containers,

Para parar os containers, utilizamos o comando down e podemos executar sem passar o nome do arquivo, caso estejamos na mesma pasta do arquivo docker-compose.yml ou usando o a flag -f para especificar qual arquivo de composse queremos desligar:

```
$ docker compose down
```

Comando 3 – Parando docker compose
Fonte: Elaborado pelo autor (2023)

O QUE VOCÊ VIU NESTA AULA?

Em resumo, o Dockerfile e o Docker Compose são duas ferramentas essenciais para quem trabalha com Docker e aplicações em contêineres.

Com o Dockerfile e o Docker Compose, é possível criar, testar e executar aplicativos Docker com rapidez e facilidade. Com o Dockerfile podemos conceber as imagens específicas para nossas aplicações e o Docker Compose simplifica o processo de gerenciamento de múltiplos containers, permitindo que indivíduos especialistas em desenvolvimento se concentrem em escrever o código sem se preocupar com a infraestrutura.

Estamos no Discord para te ajudar com dúvidas, conversar sobre os conteúdos e muito mais! Acesse a comunidade e se conecte conosco.

REFERÊNCIAS

ARTINE, D. **Docker**: Desvendando o Dockerfile. 2022. Disponível em: <https://www.alura.com.br/artigos/desvendando-o-dockerfile>. Acesso em: 14 mar. 2023.

MICROSOFT. **Usar o Docker Compose para implantar vários contêineres**. 2022. Disponível em: <https://learn.microsoft.com/pt-br/azure/cognitive-services/containers/docker-compose-recipe>. Acesso em: 14 mar. 2023.

PALAVRAS-CHAVE

Palavras-chave: Docker. Dockerfile. Docker composse. Comandos Docker. Orquestração container.