Certified Tech Developer
The Ultimate Degree

Objetivos

Construir um Dockerfile para compilar e executar uma aplicação em Java, aplicação que simula o funcionamento de uma máquina de café.

O que vamos construir?

Vamos fazer um Dockerfile para criar uma imagem personalizada para depois executarmos um container baseado nesta imagem.
Baixe o código fonte da Coffee Machine: [digitalhouse/CoffeeMachine.java at main · nidiodolfini/digitalhouse (github.com)]

Instruções

Abrir sua IDE preferida, bloco de notas ou algum editor de texto no Linux como VI, VIM ou nano.
Então devemos criar um arquivo chamado Dockerfile
O arquivo deve ter esta estrutura:

```
FROM openjdk:11
WORKDIR /diretorioprinc/
COPY CoffeeMachine.java /diretorioprinc/
RUN ["javac", "CoffeeMachine.java"]
ENTRYPOINT ["java", "CoffeeMachine"]
```

Dica: Se analisarmos o que acabamos de escrever, vamos ver que é composto por cinco termos:

- FROM: Este termo serve para solicitar a imagem base ao regristry do Docker, no caso o DockerHub, vamos utilizar uma imagem que já vem com o OpenJDK 11 instalado e utiliza o Linux como S.O.
- WORKDIR: Este cria um diretório e faz ele ser nosso diretório default ao acessar via bash.
- COPY: Este comando copia arquivos ou pastas que especificados na primeira parte do comando no nosso caso o `./src/CoffeeMachine.java` para o local especificado na segunda parte do comando no caso o `/diretorioprinc/` .
- RUN: vai executar os comandos especificados, no caso o compilador java para compilar nosso arquivo e transformar em um .class
- ENTRYPOINT: E finalmente, este termo é um parâmetro que diz a nossa imagem que os comandos especificados devem ser executados quando o container for iniciado.

Utilizaremos o comando build para construir a imagem baseada no nosso Dockerfile.

```
docker image build . --tag nomedaimagem
```

Dica: Se analisarmos o comando que acabamos de executar, vamos ver que é composto por:

- docker: para invocar o já famoso cliente Docker ou Docker CLI (Command Line Interface)
- image: este termo engloba todas as operações relacionadas às imagens no docker.
- build: ao usar este termo, no contexto de “image”, estamos indicando que queremos criar uma imagem baseada no Dockerfile que está no mesmo diretório, por isso a utilização do “.” (ponto) após o build.
- --tag ou -t: no contexto de “image” adiciona um nome para a imagem.

Para saber mais sobre os comandos que estão no grupo “image” podemos executar:

```
docker image --help
```

Podemos verificar se nossa imagem foi criada, executando o comando:

```
docker images --all
```

Dica: podemos usar o -a também.

```
→docker images -a
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
nomedaimagem    latest    3c5476bf4374   3 hours ago     47.8MB
```

```
docker container run --interactive --name nomedocontainer nomedaimagem
```

Dica: Este comando tem muitos termos. Vamos analisá-los:

- docker: novamente, para invocar o cliente Docker ou Docker CLI (Command Line Interface)
- container: Para dizer que vamos trabalhar no contexto de container, mas ele pode ser suprimido e utilizarmos apenas o run:
- run: este termo indica que vamos executar um novo container.
- --interactive ou -i: para indicar que vamos usar um terminal interativo, no caso o bash.
- --name: com este parâmetro, atribuímos um nome ao nosso container, neste caso 'nomedocontainer'.
- Na última instrução colocamos o nome da imagem que vai ser utilizada em nosso container, poderia ser imagens que contidas no registry do Docker, mas iremos utilizar a imagem que nós criamos

Para saber mais sobre os parâmetros que podemos usar para o comando 'run', podemos executar:

```
docker run --help
```

Se tudo deu certo teremos nossa simulação de uma máquina de café rodando no nosso terminal, o container se manterá ligado enquanto estivermos jogando:

```
~\Documents\GitHub\coffeemachine
→ docker container run --interactive --name nomedocontainer nomedaimagem

Write action (buy, fill, take, remaining, exit):
```

Caso queira jogar novamente sem criar um container novamente, podemos inicializar nosso container com o comando:

```
docker start -i nomedocontainer
```

Dica: Este comando tem poucos termos. Mas vamos analisá-los:

- docker: novamente, para invocar o cliente Docker ou Docker CLI (Command Line Interface)
- start: Para inicializar um container que já foi criado:
- -i: para indicar que vamos usar um terminal interativo, no caso o bash.
- Na última instrução colocamos o nome do container que vamos inicializar.

Para saber mais sobre os parâmetros que podemos usar para o comando 'run', podemos executar:

```
docker start --help
```

```
exportar imagem baseada em um container:

#criar container e abrir um terminal interativo

docker container run -it --name containercriado ubuntu:latest bash

#comando linux para install e update

apt-get update
apt-get install nginx -y

#sair do terminal interativo

exit

#pausar container

docker stop containercriado
```

```
#fazer commit (criar a imagem baseada no container) verificar se o repositório é o mesmo do seu repo no Docker Hub

docker container commit containercriado nomerepo/nomeimagem:version

#verificar se a imagem foi criada

docker images -a

#fazer o push para o docker hub

docker push nomerepo/nomeimagem:version

exportar imagem baseada em um container:

#criar container e abrir um terminal interativo

docker container run -it --name containercriado ubuntu:latest bash

#comando linux para install e update

apt-get update
apt-get install nginx -y

#sair do terminal interativo

exit

#pausar container

docker stop containercriado

#fazer commit (criar a imagem baseada no container) verificar se o repositório é o mesmo do seu repo no Docker Hub

docker container commit containercriado nomerepo/nomeimagem:version

#verificar se a imagem foi criada
```

```
docker images -a

#fazer o push para o docker hub

docker push nomerepo/nomeimagem:version
```