Play with Docker for Developers

# Intro

Este arquivo leva em consideração que já visto o conteúdo do arquivo Play-with-Docker-for-ITs.

# Docker for Beginners - Linux

Existem diferentes maneiras de utilizar um container. Alguns são:

* Executar uma tarefa única: pode ser shell script ou um app customizado
* Interativamente: conecta-se ao container de forma similar ao SSH
* Em segundo plano: para serviços como websites e databases

## Link

<https://training.play-with-docker.com/beginner-linux/>

## Pré requisitos

* Possuir Docker ID
* Clonar o repositório no git:

git clone <https://github.com/dockersamples/linux_tweet_app>

# Executando alguns containers simples

## Tarefa única com Alpine

Comando de exemplo:

docker container run alpine hostname

O Docker mantém um container executando enquanto o processo que foi iniciado dentro deste container estiver executando.

O “hostname” encerra assim que imprime o texto. Portanto o container é parado. No entanto, o Docker não exclui os recursos por padrão. O container continua a existir no estado Encerrado (Exited).

Listando todos os containers:

docker container ls --all

Containers que executam uma única tarefa podem ser úteis para criar images que executam scripts para fazer configurações de algo. Qualquer pessoa poderá configurar o ambiente sem necessitar do próprio script ou informações de configuração, ela precisa apenas executar a image num container.

## Executar um container interativo do ubuntu

É possível rodar uma versão linux diferente da versão no host do Docker.

Os nodes de Play with Docker utilizam Alpine Linux.

Acesso ao shell do Ubuntu:

docker container run --interactive --tty --rm ubuntu bash

No comando são passados 3 parâmetros:

* Interactive: sessão interativa
* Tty: aloca um pseudo-terminal
* Rm: Docker remove o container quando a execução encerrar

No mesmo comando também é indicado ao container que bash deve ser executado como processo principal (PID 1).

O Docker consegue intermediar a entrada e a saída do shell no container. O prompt utilizado fica no formato: root@*<container id>*:/#

Para verificar qual Linux está sendo utilizado, execute:

cat /etc/issue

A saída será similar a “Ubuntu 16.04.4 LTS”.

Saia do terminal com: exit.

Verifique novamente a versão do Linux:

cat /etc/issue

A saída deve ser similar a “Welcome to Alpine Linux 3.7”.

Pontos importantes do exemplo:

* O tipo do Linux no container não precisa ser igual ao Linux no host
* No entanto, para executar qualquer Linux no container, o host deve estar executando um kernel do Linux
* Consequentemente containers no Linux não rodam (diretamente) num Docker do Windows.
* O mesmo ao contrário.

Containers interativos são ideais na hora de criar uma image. É possível executar o container para verificar todos os passos necessários para realizar o deploy do app a fim de criar um Dockerfile.

Não é recomendado utilizar o commit, ainda que tenha o mesmo resultado.

## Executando um cotainer de MySQL em segundo plano

Executar exemplo:

docker container run --detach --name mydb -e MYSQL\_ROOT\_PASSWORD=my-secret-pw mysql:latest

Parâmetros:

* Detach: executa o container em segundo plano
* Name: nomeia o banco de dados como mydb
* E: Utiliza uma variável de ambiente para especificar a senha (nunca utilizar em modo de produção)

É possível verificar que o container está sendo executado ao utilizar o comando para listar containers:

docker containers ls

É possível verificar o que está acontecendo no container por meio do comando de logs e top:

* Logs: mostra os registros do container MySQL.

docker container logs mydb

* Top: mostra os processos executando dentro do container.

docker container top mydb

Embora o MySQL esteja executando, ele está isolado dentro do container. Porque nenhuma porta foi publicada ao host (?) . Nenhum tráfego pode alcançar os containers a não ser que existam portas explicitamente públicas.

Executando comando dentro do container:

docker container exec –it mydb mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

O comando exibe, por meio do exec, a versão do MySQL.

´ É possível conectar um novo shell por meio do exec:

docker container exec –it mydb sh

Para executar o mesmo comando para verificar a versão:

mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

O comando imprime o mesmo texto, porém foi executado dentro de um shell do container.

# Empacotar e executar um app customizado com Docker

Este capítulo utiliza o repositório que foi comentado nos pré-requisitos.

Acessar a pasta do repositório:

cd ~/linux\_tweet\_app

cat Dockerfile

O comando imprime o conteúdo do Dockerfile.

Comandos do Dockerfile:

* FROM: indica qual é o image base a ser utilizado
* COPY: copia os arquivos no Docker host para a image para um local conhecido
* EXPOSE: documenta quais portas a aplicação vai utilizar
* CMD: especifica qual comando executar quando o container inicializar. Pode-se especificar o comando e os argumentos de tempo de execução

Para facilitar os comandos, será criada uma variável com o Docker ID:

export DOCKERID=*<docker-id>*

Confirmando a atribuição:

echo $DOCKERID

Criando novo image:

docker image build --tag $DOCKERID/linux\_tweet\_app:1.0 .

Anotações:

* Tag: dá um nome à image. Quando um Docker ID está incluído no nome, é possível armazenar o image no Docker Hub.
* “.” : Diz ao Docker para utilizar a pasta atual como contexto de build.

Comando para executar o container:

docker container run \

--detach \

--publish 80:80 \

--name linux\_tweet\_app \

$DOCKERID/linux\_tweet\_app:1.0

Por utilizar o NGINX web server, será utilizado o flag “publish” para publicar a porta 80 DENTRO do container na porta 80 DO HOST. Isto vai permitir que o tráfego da porta 80 do host seja direcionada para a porta 80 dentro do container.

O formato para utilizar o flag publish é:

--publish *<host-port>*:*<container-port>*

É possível mapear portas diferentes. Necessário quando dois containers utilizam a mesma porta para comunicação, sendo que a exposição de uma porta do host pode ser feita apenas uma vez.

Após executar o comando de execução, visitar o site que mostra a aplicação rodando. O link é similar ao: <http://ip172-18-0-10-bec1500gn5rg00f00l5g-80.direct.labs.play-with-docker.com/>

Comando para remover (não utilizar):

docker container rm --force linux\_tweet\_app

Este comando remove o container sem desligá-lo. Ele acaba desligando o container e remove permanentemente do Docker host.

Num ambiente de produção utilizar:

* Parar o container

docker container stop ...

* Remover permanentemente o container

docker container rm ...

# Modificar uma página executando

Task 3: Modify a running website

.....