Docker

Sumário

1 - Algumas observações

O Docker é uma ferramenta poderosa usada para desenvolver, empacotar e implantar aplicativos de forma eficiente. O Docker é um serviço de gerenciamento de contêineres. O Docker foi lançado em 2013. Ele é de código aberto e está disponível para diferentes plataformas, como Windows, macOS e Linux. O Docker está enviando, testando e implantando código rapidamente. Para reduzir seu atraso entre escrever o código e executá-lo na produção. Você pode criar ambientes autocontidos conhecidos como contêineres. Que podem ser executados consistentemente em diferentes plataformas. (https://www.geeksforgeeks.org/docker-tutorial/?ref=outind)

Não pretendo criar aqui um tutorial sobre docker, mas apenas anotar algumas observações e compartilhar algumas experiências sobre;

A tecnologia do momento para os programadores e devops. Em seu site eles dizem "Desenvolva mais rápido e execute em qualquer lugar".

Com docker geralmente se critica aqueles programadores, em geral que usam windows, que cria um software que roda em seu desktop mas não roda no servidor.

Com docker, podemos ter em nosso desktop, um ambiente similar ao ambiente do servidor.

Antes do docker tivemos máquinas virtuais, tivemos e ainda temos Vagrant. O docker é muito popular atualmente e valoriza quem sabe.

O docker basicamente trabalha com imagens de sistemas operacionais e com container. Uma imagem docker geralmente contém somente o sistema operacional, mas também já pode vir com o que desejar seu criador.

Já o container, é algo que criamos em nosso desktop ou em nosso servidor ao instalar uma imagem.

Geralmente o conteúdo criado em um container é perdido ao fecharmos o mesmo. Para que ele guarde as alterações precisamos executar certos procedimentos.

1.1 - Imagem interativa

Podemos criar uma base, contendo algumas imagens e pacotes e gerar uma imagem resultante interativamente selecionando imagens e pacotes, geralmente através de um .env.

1.2 - Versões de distros e pacotes

Como um forte objetivo do docker é criar container e imagens que sejam similares a servidores e assim evitar cconflitos ao enviar para o servidor e também como gosto de usar o Debian para minhas imagens/containers, então preciso saber que versão do Debian usar para ter certa versão do PHP, no meu caso.

Debian	Versão do PHP
Squeeze 6	5.3.3
Wheezy 7	5.4.45
Jessie 8	5.6.40
Stretch 9	7.0
Buster 10	7.3
Bullseye 11	7.4
Bookworm 12	8.2
Unstable	8.2

https://wiki.debian.org/PHP

Tem muito, muito mais no docker e docker-compose do que abordei aqui. A intenção é mostrar algo do que aprendi para tentar suavisar a jornada de quem está iniciando seu aprendizado.

2 - Instalação do Docker

2.1 - No Linux Mint 20

Primeiramente iremos instalar as dependências necessárias, para isso execute os comandos abaixo:

sudo apt update
sudo apt -y install apt-transport-https ca-certificates curl software-properties-common

Agora importaremos a chave do pacote docker:
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Agora iremos adicionar o repositório do docker ao nosso sistema:
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(. /etc/os-release; echo "$UBUNTU_CODENAME") stable"

Em seguida iremos atualizar a lista de pacotes:
sudo apt update

Por fim, instalaremos o docker e o docker-compose:
sudo apt -y install docker-ce docker-compose

Será necessário adicionar nosso usuário ao grupo do docker para não haver necessidade de utilização do "sudo".
sudo usermod -aG docker $USER
su - $USER
senha
exit

reiniciar o computador

Verificando a versão do docker instalado:

docker -v
Docker version 20.10.18

Reiniciar o computador para poder usar o docker

FONTE: https://computingforgeeks.com/

2.2 - No Debian 11 e 12

sudo apt update

sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

And now if you do: docker-compose --version

You'll see that docker-compose is now on the PATH

sudo apt -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker-archive-keyring.gpg

```
sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/debian \
  $(lsb_release -cs) \
  stable"
```

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io -y

sudo systemctl enable --now docker

sudo usermod -aG docker $USER

docker version

docker-compose ps

https://computingforgeeks.com/install-docker-and-docker-compose-on-debian

https://download.docker.com/linux/debian/dists/bullseye/pool/stable/amd64/

Testar

sudo service docker status

sudo docker run hello-world

reiniciar o desktop

3 - Caso queira remover imagens e containers

```
docker-compose stop
docker-compose rm -f

docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker rmi -f $(docker images -a -q)
```

4 - Ambiente de Exemplo

Vou mostrar como aprendi a criar um ambiente de desenvolvimento usando Debian

Estarei usando Debian, Apache, MySQL e PHP.

Nosso ambiente será composto de:

- mysql/
  my.sql
- php/
  Dockerfile5.6
  Dockerfile7.4
  Dockerfile8.1
- web/
  crud-ma/
  crud-sq/
  micro/
  index.php
- docker-compose.yml
- perms
- README.md
- sample.env
- start.sh
- start2.sh

4.1 - README

Tradicionalmente projetos no Github vem com um arquivo README.md que apresenta o projeto. O md é de markdown, linguagem usada no arquivo. Acho bem importante oferecer uma boa quantidade de informações sobre o projeto neste arquivo, além de oferecer um .sql para que o colega possa testar o projeto com facilidade, além de outras boas informações e facilidades.

Veja o conteúdo do README.md deste projeto:

Docker debian devel

Esta imagem contém o LAMP (Debian, Apache, MariaDb e PHP 5.6, 7.4 ou 8.1)

Com dois CRUDs, um com mariadb e outro com sqlite

Os CRUDs usam PHP, PDO, paginação e bootstrap.

Como usar

Execute

cp sample.env .env

Abra o .env e altere a gosto a versão do PHP

Remover

Para garantir, caso queira remover todas as imagens e containers existentes:

Parar todos os containers rodando:

```bash
*docker stop $(docker ps -a -q)*
*docker rm $(docker ps -a -q)*
*Remover todas as imagens*
*docker rmi $(docker images -a -q)*
```
Criar o container através da imagem
```bash
*docker-compose up -d*
```
Ao final
```bash
*docker images*
*docker run -it nomeimagem*
```
Executar:
```bash
*start.sh - Se setou o .env para 56*

*start2.sh - Se setou o .env para 74 ou 81*

*source /root/.bashrc*

ip a
```
Abra no desktop

http://172.17.0.2

Veja que na pasta php existem 3 arquivos Dockerfile que serão selecionados no .env e capturados pelo docker-compose.yml. Então vejamos os dois arquivos principais, que são o docker-compose.yml e o Dockerfile.

4.2 - docker-compose.yml

```
version: "3.6"

services:
   lamp:
     container_name: docker-lamp
     build:
       context: .
       dockerfile: php/Dockerfile${PHPVERSION}
     environment:
       APACHE_RUN_USER: www-data
       APACHE_RUN_GROUP: www-data
       WEB_DOCUMENT_ROOT: /var/www/html
       MYSQL_DATABASE: ${DB_DATABASE}
       MYSQL_PASSWORD: ${DB_PASSWORD}
       MYSQL_USER: ${DB_USERNAME}
       restart: unless-stopped
     volumes:
       - ./web:/var/www/html
       - ./mysql:/var/lib/mysql
     ports:
       - "8888:80"
       - "3333:3306"

volumes:
  web:
  mysql:
```

Observe que este arquivo pega a versão do PHP do .env e chama o respectivo Dockerfile. Vejamos então um Dockerfile, o da vers/ao 7.4, como exemplo

4.3 - Dockerfile7.4

```
FROM debian:bullseye as lampp-img

LABEL maintainer="Ribamar FS <ribafs@gmail.com>"

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get upgrade -y
RUN apt-get install -y apache2 nano unzip wget

RUN apt-get update
RUN apt-get install -y software-properties-common dirmngr
RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get -yq install mariadb-server

RUN apt-get install -y php7.4 libapache2-mod-php7.4 \
    php7.4 \
    php7.4-common \
    php7.4-fpm \
    php7.4-cli \
    php7.4-curl \
    php7.4-json \
    php7.4-xsl \
    php7.4-xdebug \
    php7.4-gd \
    php-pear \
    php7.4-mysql \
    php7.4-sqlite

RUN apt-get install -y curl composer

RUN curl -sL https://deb.nodesource.com/setup_16.x -o nodesource_setup.sh
RUN bash nodesource_setup.sh
RUN apt-get install -y nodejs

# No docker-compose.yml montar um volume mysql:/var/lib/mysql
ADD mysql/ /var/www/html/
ADD web/ /var/www/html/
RUN rm /var/www/html/index.html

COPY start2.sh /usr/local/bin/start2.sh
RUN chmod +x /usr/local/bin/start2.sh

COPY perms /usr/local/bin/perms
RUN chmod +x /usr/local/bin/perms

RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

# install apache to run and configure
RUN sed -i "s/AllowOverride\ None/AllowOverride\ All/" /etc/apache2/apache2.conf
```

```
RUN sed -i "s/display_errors = Off/display_errors = On/" /etc/php/7.4/apache2/php.ini

WORKDIR /var/www/html/

EXPOSE 80
EXPOSE 3306
```

Aqui precisamos ficar atentos, pois pode acontecer de a versão do Debian debian:bullseye já não estar trazendo a versão 7.4 do PHP.

O que acabei descobrindo é que é bom ter uma relação da distro que usamos com suas versões e as versões dos pacotes desejados no Dockerfile.

Algo interessante e que me ajudou foi criar localmente um container com a versão que desejo testar, atualizar ela e instalar o pacote, no caso o PHP, para verificar se a versão é realmente a que preciso.

```
docker run -it debian:bullseye
apt update
apt install php
Já deve ter visto a versão durante a instalação, caso queira
php -v
```

Assim podemos nos certificar se a versão da distribuição realmente traz a versão dos pacotes desejados.

5 – Remover

Para remover imagens e containers da memória.

Caso queira remover imagens e containers

docker-compose stop
docker-compose rm -f

docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker rmi -f $(docker images -a -q)

```
docker image prune
```

Remover iimagens não usadas por containers

```
docker image prune -a

docker container prune

docker volume prune

docker network prune
```

6 – Boas Referências

https://github.com/sprintcube/docker-compose-lamp

A basic LAMP stack environment built using Docker Compose. It consists of the following:

- PHP
- Apache
- MySQL
- phpMyAdmin
- Redis

As of now, we have several different PHP versions. Use appropriate php version as needed:

- 5.4.x
- 5.6.x
- 7.1.x
- 7.2.x
- 7.3.x
- 7.4.x
- 8.0.x
- 8.1.x
- 8.2.x
- 8.3.x