docker --version

Informa versão do docker

-------------------------------------------------------------------------------------------------------------------------------

docker ps

Lista de containers em execução

docker ps -a

Lista/histórico de containers já executados

-------------------------------------------------------------------------------------------------------------------------------

docker run [image] [comando]

Executa um docker em background

docker run -ti [image] [comando]

Executa um docker em um terminal interface

docker run -ti --name [usr-container-name] [image] [comando]

Executa um docker em um terminal interface e informa um nome ao container

docker run -d [image] [comando]

Executa um docker como um daemon (um processo em background)

docker run -t [image] [comando]

Executa um docker como um daemon (um processo em background)

keyboard: ctrl+d

Mata a execução do shell e to container

keyboard: ctrl+p+q

Deixa o container executando em *background*

docker attach [container id]

Retorna a interagir com o shell do container

-------------------------------------------------------------------------------------------------------------------------------

docker create [image]

Cria o container sem executar

docker stop [container id]

Interrompe a execução do container

docker start [container id]

Inicia a execução do container

docker pause [container id]

Pausa a execução do container

docker unpause [container id]

Despausa a execução do container

docker stats [container id]

Informa consumo de hardware

docker top [container id]

Informa os processos do container

docker logs [container id]

Informa os logs do container

docker rm [container id]

Remove o container parado (stop)

docker rm -f [container id]

Remove o container em execução

-------------------------------------------------------------------------------------------------------------------------------

docker inspect [container id]

Apresenta na tela todas as configurações do container em execução

docker inspect [container id] | grep -i mem / grep -i cpu

Apresenta as configurações de memória do container em execução (no Linux)

docker run -ti --memory 512m --name [name] [image]

Roda e atribui um valor limite de memória a ser utilizado pelo container

docker update -m 256m [container id]

Atualiza o valor limite de memória a ser utilizado pelo container

docker run -ti --cpu-shares 1024 --name [name] [images]

Roda e atribui um valor limite de cpu a ser utilizado pelo container

docker update ---cpu-shares 512 [container id]

Atualiza o valor limite de cpu a ser utilizado pelo container

-------------------------------------------------------------------------------------------------------------------------------

docker run -ti -v /volume [image] [comando]

Roda um container e cria um volume (diretório) no caminho (comando)

docker run -ti -v [host-path]:/volume [image] [comando]

Roda um container e cria um volume existente no host no caminho (comando)

docker inspect -f {{.mounts}} [container id]

Apresenta o volume montados

-------------------------------------------------------------------------------------------------------------------------------

container data-only

docker create -v /data --name [name] [image]

Remove o container em execução

docker run -d -p 5432:5432 --name [name] --volumes-from [container-data-only-name] -e POSTGRESQL\_USER=docker -e POSTGRESQL\_PASS=docker -e POSTGRES\_DB=docker kamui/postgresql

Roda um container e utiliza um volume de outro container

-------------------------------------------------------------------------------------------------------------------------------

container Dockerfile

FROM [image]

Determina a imagem a ser utilizada para montar o container

MAINTAINER [user-name]

Descreve quem escreve o dockerfile

RUN apt-get update && apt-get install -y apache2 && apt-get clean

(Run) executa comandos no container

Porém evitar múltiplos run ~~apt-get~~, pois criará novas camadas (custo de hardware)

ADD file.txt /diretório/

Adiciona um arquivo ao container (inclusive arquivos .TAR, .ZIP, .RAR...)

CMD [“SH”, “-C”, “echo”, “$HOME”]

É um comando por parâmetro para o (entrypoint) principal processo de um container

LABEL Description=”bla bla bla metadados”

Adiciona metadados, fabricante, versão, e outros...

COPY file.txt /diretório/

Apenas copia arquivos para o container

ENTRYPOPINT [“/usr/bin/[image-container]”, “-D”, “FOREGROUND”]

Determina um processo para ser o principal, monitorado e “dar vida” ao container

Se o processo morrer o container é finalizado e deixa de consumir recursos

ENV [variável-name]=”[value]”

Configurar variáveis de ambiente para o container

EXPOSE 80

Expõe portas do container para dar acesso

USER [name]

Determina o usuário padrão para o container

WORKDIR /catota

Determina o diretório de trabalho do container

VOLUME /diretório

Cria um volume no container

-------------------------------------------------------------------------------------------------------------------------------

docker build .

Inicia a construção do container, verifica ou faz download da imagem

Roda o container com as configurações indicadas

Roda os comandos instruídos

docker build -t [name]:[versão] .

Linux CLI to search text

-------------------------------------------------------------------------------------------------------------------------------

No Linux

mkdir /root/dockerfiles

cd /root/dockerfiles

mkdir apache

cd apache/

vim Dockerfile (can use vscode)

Dockerfile example:

1 FROM debian

2

3 RUN apt-get update && apt-get install -y apache2 && apt-get clean

4

5 ENV APACHE\_LOCK\_DIR="/var/lock"

6 ENV APACHE\_PID\_FILE="/var/run/apache2.pid"

7 ENV APACHE\_RUN\_USER="www-data"

8 ENV APACHE\_RUN\_GROUP="www-data"

9 ENV APACHE\_LOG\_DIR="/var/log/apache2"

10

11 LABEL Description=”Webserver”

12

13 VOLUME /var/www/html

14

15 EXPOSE 80

No Linux

docker build -t webserver:1.0 .

docker images

docker run -ti webserver:1.0

ps -ef

etc/init.d/apache2 start

ps -ef

ss -s

ss -a

ip addr

telnet 172.17.0.2 80

curl 172.17.0.2:80

-------------------------------------------------------------------------------------------------------------------------------

REGISTRY : DOCKER HUB : hub.docker

docker login [url]

Faz login no registry

docker inspect [image]

Inspeciona as configurações da imagem (auditoria das imagens)

Docker history [image]

Apresenta as camadas do container, apresentando a evolução da imagem

docker tag [image id] [user-name]/[name-image]

Altera o nome da imagem para poder upar ao registry (docker hub)

docker push [image id]

Envia a imagem para o Registry docker hub

docker push [user-name]/[name-image]

Envia a imagem para o Registry docker hub

docker search [user-name]

Busca imagens do usuário no registry (docker hub)

docker rmi -f [image id]

Remove a imagem

docker rmi -f [user-name]/[name-image]

Remove a imagem

docker pull [user-name]/[name-image]

Efetua o download da imagem do docker hub (Registry)

-------------------------------------------------------------------------------------------------------------------------------

REGISTRY : DOCKER HUB : create your own

docker run -d -p 5000:5000 --restart=always --name [name] registry:2

Baixa e executa um container do registry

docker tag [image id] [image-name]

Envia a imagem para o Registry docker hub

**É importante mudar o nome para redirecionar o registry**

**Do hub.docker para o localhost**

docker push [image-name]

Envia a imagem para o registry local (**não para o docker hub**)

Linux CLI: curl localhost:5000/v2/\_catalog

Apresenta o conjunto de imagens do Registry local

-------------------------------------------------------------------------------------------------------------------------------

Redes

docker0 é a ponte de comunicação de rede dos containers

docker run -ti --dns 8.8.8.8 [image]

Passa um servidor de DNS para o container

Linux CLI: cat /etc/resolv.conf

Apresenta as configurações, podendo verificar o DNS

docker run -ti --hostname [host-name] [image]

Efetua o download da imagem do docker hub (Registry)

docker run -ti --link [container-name] --name [container-name] [image]

Quando container estiver rodando já conhece o container ligado

Linux CLI: ping [container-name]

Demonstra que há conexão

docker run -ti --expose 80 [image]

Expõe a porta do container

docker run -ti -p 8080:80 --name [container-name] [image]

Redireciona da porta do host para o container

Linux CLI: iptables -t nat -L -n

Apresenta a configuração de DNAT (destination nat), o redirect

docker run -ti --net=host [image]

As configurações de rede são do host

-------------------------------------------------------------------------------------------------------------------------------

Docker machine : installation

wget [link] `uname -s`-`uname -m`

Efetua o download da imagem do docker hub (Registry)

mv docker-machine-Linux-x86\_64 /usr/local/bin

move o arquivo para a pasta /bin do Linux

chmod +x /usr/local/bin/docker-machine-Linux-x86\_64

Habilita o comando docker-machine-x86\_64 para o Linux CLI

-------------------------------------------------------------------------------------------------------------------------------

Docker machine : using | criar docker host para executar em outros ambientes

docker-machine-Linux-x86\_64 create --driver virtualbox [name]

Roda a docker-machine para montar um docker host com nome escolhidio

docker-machine-x86\_64 env [name]

Apresenta as variáveis de ambientes a serem configuradas

eval “$(docker-machine-Linux-x86\_64 env [name])”

Conecta no host da docker machine

docker-machine-Linux-x86+64 ls

Apresenta conjunto de docker-machine disponíveis

docker-machine-Linux-x86+64 ip [name]

Apresenta o IP do docker-machine

docker-machine-Linux-x86+64 ssh [name]

Apresenta o IP do docker-machine

docker-machine-Linux-x86+64 inspect [name]

Apresenta informações sobre o docker-machine

docker-machine-Linux-x86+64 stop [name]

Interrompe a execução do docker

docker-machine-Linux-x86+64 start [name]

Iniciar execução do docker

docker-machine-Linux-x86+64 rm [name]

Interrompe a execução do docker

-------------------------------------------------------------------------------------------------------------------------------

Docker compose

db:

image: postgres

web:

build: .

command: python manage.py runserv 0.0.0.0:8000

volumes:

- .:/code

ports:

- 8000:8000

links:

- db

docker-compose scale [parameter]=[value]

Função para ampliar número de containers, escalar a aplicação

Docker compose installing:

Segue o padrão docker-machine: **wget + mv + chmod**, porém docker tools já instala

docker-compose --version

Apresenta a versão do docker-compose

build

build: .

Indica o caminho para o dockerfile

command

command: bulde exec thin -p 3000

Executa um comando dentro do container

container\_name

container\_name: my-web-container

Fornece um nome ao container

dns

dns: 8.8.8.8

Fornece um serviço DNS ao container

dns\_search

dns\_search: example.com

Fornece um serviço de dns search

dockerfile

dockerfile: Dockerfile-alternate

Especifica um Dockerfile alternativo

env\_file

env\_file: .env

Especifica um arquivo com todas as variáveis de ambiente

environment

environment:

RACK\_ENV: development

Adiciona variáveis de ambiente

expose

expose:

- “3000”

- “8000”

expõe a porta do container

external\_links

external\_links:

- “redis\_1”

- “project\_db\_1:mysql”

Linka container com outro container não listado no Dockerfile

extra\_hosts

extra\_hosts:

- “somehost:162.242.195.82”

- “otherhost:50.31.209.229”

Adiciona uma entrada no /etc/hsots do container

image

image: ubuntu:14.04

Indica uma imagem

labels

labels:

com.example.description: “Accounting webapp”

com.example.department: “Finance”

Adiciona metadata ao container

links

links:

- db

Linka containers dentro de um mesmo docker-compose

log\_drive

1. log\_drive: syslog
2. logging: syslog

Indica o formato de log a ser gerado

log\_opt || logging

1. log\_opt:

syslog-addres: “tcp://192.168.0.42:123”

1. logging:

driver: syslog

options:

syslog-addres: “tcp://192.168.0.42:123”

Indica onde mandar os logs, pode ser local ou em um syslog remoto

net

net: “bridge”

net: “host”

Modo de uso da rede

ports

ports:

- “3000”

- “8000:8000”

Expõe as portas do container e do host

volumes, volumes\_drive

volumes:

# Just specify a path and let the Engine create a volume

- /var/lib/mysql

# Specify an absolute path mapping

- /opt/data:/var/lib/mysql

# Path on the host, relative to the Compose file

- ./cache:/tmp/cache

Monta volumes no container

volumes\_from

volumes\_from:

- service\_name

- service\_name:ro

Monta volumes através de outro container

-------------------------------------------------------------------------------------------------------------------------------

**docker-compose.yml**

db:

image: postgres

web:

build: .

command: python manage.py runserv 0.0.0.0:8000

volumes:

- .:/code

ports:

- 8000:8000

links:

- db

**Dockerfile**

FROM python:2.7

ENV PYTHONBUFFERED 1

RUN mkdir /code

WORKDIR /code

ADD requirements.txt /code/

RUN pip install -r requirements.txt

ADD . /code/

**requirements.txt**

Django

psycopg2

docker-compose run web django-admin.py startproject composexample .

cd composeexample/

vim settings.py

**settings.py**

DATABASES = {

...

host: “db”

…

}

docker-compose up -d

Levanta o container dock

-------------------------------------------------------------------------------------------------------------------------------

**Docker 1.12 | Atualização** |Docker cluster

docker swarm init --listen-addr [addres]

Inicia o cluster, manager (swarm-manager)

docker swarm join \

Acessa o cluster

docker node ls

Verifica nódulos no cluster

docker service create --networks ingress --name mysql env MYSQL\_ROOT\_PASSWORD=wordpress --env MYSQL\_DATABASE=wordpress --env MYSQL\_USER=wordpress -env MYSQL\_PASSWORD=wordpress mysql:5.7

Criar um service mysql (que faz a distribuição entre os containers)

docker service create --network ingress --name wordpress --env WORDPRESS\_DB\_HOST=[service-name]:[port] --env WORDPRESS\_DB\_PASSWORD=wordpress --publish 8000:80

Criar um service wordpress (que faz a distribuição entre os containers)

docker service ls

Apresenta os services

ipconfig eth0

Apresenta os services

docker service create --name nginx nginx

Criar um service nginx (que faz a distribuição entre os containers)

docker service scale nginx=4

Cria 4 containers desse service

docker service inspect [service-name]

Detalhes do serviço

docker node inspect

Detalhes do node onde estão rodando o containers

docker service tasks [service-name]

Onde estão rodando os containers

docker node inspect [service-id]

Apresenta configurações do nodulo

docker network ls

Apresenta os services

docker volume ls

Apresenta os services

-------------------------------------------------------------------------------------------------------------------------------

Linux LPI

Processo é um software em execução

Processo tem um PID (**process id**)

Processos com PPID signfica que foram iniciados por outro processo

Processos com UID significa que foram iniciados por um usuário

ps

Visualizar processos

ps aux

Visualizar processos

a - todos os processos

x - todos os processos que não estão no terminal

u - horário e usuário

USER - usuário

PID - process id

%CPU - % uso cpu

%MEM - % uso memória

VSZ - virtual memory size (toda memória que esse processo pode acessar)

RSS - resident set size (uso de memória física - memória ram)

TTY - atrelado a terminal ou não.

STAT - status do processo (r-running; s - sleep; z-zombie; t-stoppepd;...)

START -

TIME -

COMMAND - processo (comando que executou)

top

Apresenta os processos em tempo real

espaço - atualiza

shift+< ou > - navega

k - kill, escolhe PID + enter e mata o processo

man top

Help do TOP para aprender a utilizar

htop (atp-get install htop)

Uma melhoria do top

h - help

k - kill, digita PID + enter e mata o processo

kill [pid]

Visualizar processos com valor -15

kill -9 [pid]

Visualizar processos com valor -9, força finalização

kill -1 [pid]

Visualizar processos com valor -1, reload

pgrep [processo]

Visualizar pid de processo

chroot

Chroot programas que rodam com um ambiente chrootado não conseguem acessar arquivos e comandos fora do ambiente. “*chroot jail*”. Básicamente subistitui o diretório ‘/’ (root) de um processo e seus filhos.

wget -qO- <https://get.docker.com/> | sh

sudo usermod -aG docker <usuário>