DOCKER

O Docker é um plataforma (aberta) que permite que você crie, rode e faça deploy de containers. De maneira simples, um container é o empacotamento da sua aplicação mais as dependências dela.

Sobre

- ***Docker Client:*** isto é o que está sendo executado em nossa máquina. É o binário do Docker que estaremos interagindo sempre que abrirmos um terminal e executarmos $ docker pull ou $ docker run. Ele se conecta ao Docker daemon que faz todo o trabalho pesado, seja no mesmo host (no caso do Linux) ou remotamente (no nosso caso, interagindo com a nossa VirtualBox VM).

- ***Docker Daemon:*** isso é o que faz o trabalho pesado de construção, execução e distribuição de seus containers Docker.

- ***Docker Images:*** imagens do Docker são as "blueprints" para nossas aplicações. Mantendo a analogia dos containers/peças de Lego, eles são nossas "plantas" (como em arquitetura, não biologia) para realmente construir um exemplo real e funcional. Uma imagem pode ser um sistema operacional como o Ubuntu, mas também pode ser um Ubuntu com o seu aplicativo web e todos os seus pacotes necessários instalados.

- ***Docker Container:*** containers são criados a partir das imagens do Docker, e eles são as instâncias reais de nossos containers / peças de Lego. Eles podem ser iniciados, executados, parados, deletados, e movidos.

- ***Docker Hub (Registry):*** o Docker Registry é um registro hospedado em um servidor que pode conter imagens do Docker. Docker (a empresa) oferece um Docker Registry público chamado de Docker Hub que vamos utilizar neste tutorial para baixar algumas imagens, mas eles oferecem todo o sistema open-source para que pessoas possam ter seus próprios servidores e armazenar imagens privadas.

Comandos

SEARCH

Procura uma imagem.
comando: `$ docker search IMAGE_NAME` **exemplo:** `$ docker search ubuntu`

PULL

Baixa uma imagem.
comando: `$ docker pull IMAGE_NAME`
exemplo: `$ docker pull ubuntu`

RUN

Executa a imagem, acessa um container e etc... Olhar o `docker help run`.
comando: `$ docker run IMAGE_NAME`

exemplo:

```
$ docker run ubuntu /bin/echo "Hello World" | Executa um novo container e passa um comando par
$ docker run -i -t ubuntu | Executa um novo container e conecta nele.
$ docker run -i -t -p host_port:server_port node-express
```

START / STOP

Inicia a execução de um container
comando: `$ docker start/stop CONTAINER_NAME`

COMMIT

Cria uma imagem a partir de alterações feita em outra imagem.
comando: `$ docker commit -a "You Name <you@email.com>" -m "node and express" CONTAINER_ID image-name:versionnumber`

exemplo:

```
$ docker commit -a "Vinícius Galvão <viniciusj16@gmail.com>" -m "node and express" ee13a6de9a0

- a | author
- m | message
```

TAG

É uma boa prática taggear imagens com uma versão específica para que outras pessoas possam saber exatamente qual a imagem que eles estão executando. Adicionar a tag latest ajuda com que outras pessoas possam simplesmente se referir a sua imagem pelo nome dela, sem a versão, para baixá-la ou executá-la (node-express no nosso caso).

comando: `$ docker tag from-image:version new-image:version` **exemplo:**

```
$ docker tag node-express:0.1 node-express:latest
```

PUSH

Envia o container para o docker hub

comando / exemplo:

```
$ docker login
$ docker tag image-name your_docker_hub_username/image-name
$ docker rmi image-name (remove image)
$ docker push your_docker_hub_username/image-name
```

PS

Lista os containers em execução, para listar todos, deve ser adicionada a flag `-a`.
comando: `$ docker ps | $ docker ps -a`

exemplo:

```
$ docker ps
$ docker rm -a
```

RM

Remove um container.
comando: `$ docker rm YOUR_CONTAINER_ID`

exemplo:

```
$ docker rm node-express
$ docker rm -f node-express | -f = force
```

RMI

Remove uma imagem.
comando: `$ docker rmi YOUR_CONTAINER_ID`

exemplo:

```
$ docker rmi ubuntu
$ docker rmi -f ubuntu | -f = force
```