Docker: Isolando aplicações

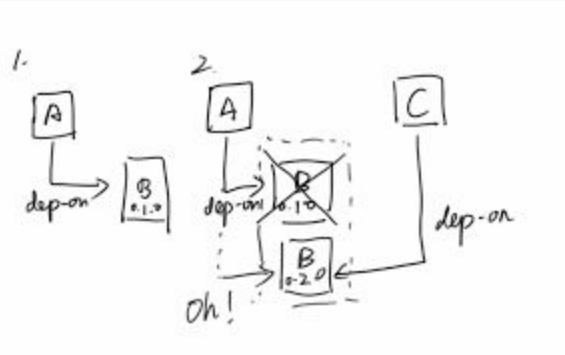
Pedro Mello

Problema

Colocando a aplicação no ambiente de produção:

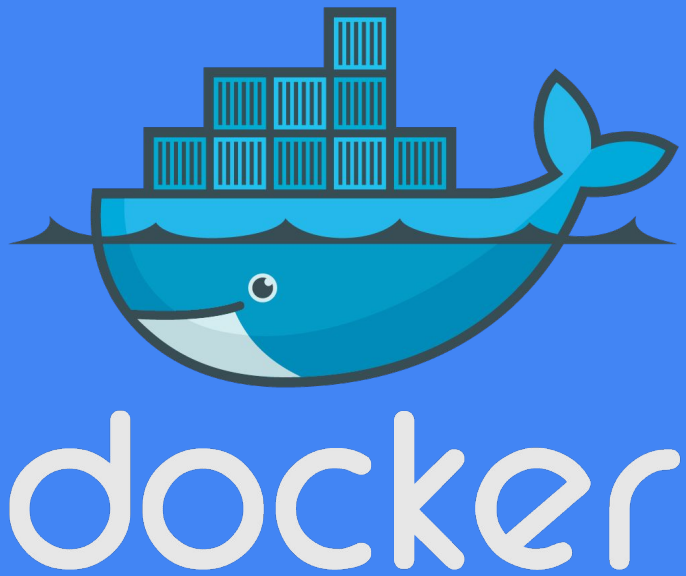
- SO específico
- Compilador / Interpretador da linguagem
- Bibliotecas e dependências
- A própria aplicação

Dependency hell



E se tivesse uma forma...

- De tornar as aplicações realmente independentes?
- Que permita a escalabilidade, ao mesmo tempo?
- Que evite o “funciona na minha máquina?”?
- Que seja facil de “deployar”?


docker

Docker - História

- Dotcloud, uma empresa PaaS, fundada em 2010
- Desenvolvido em GO
- Março de 2013 libera a primeira versão open-source
- Parcerias com empresas como google, IBM.

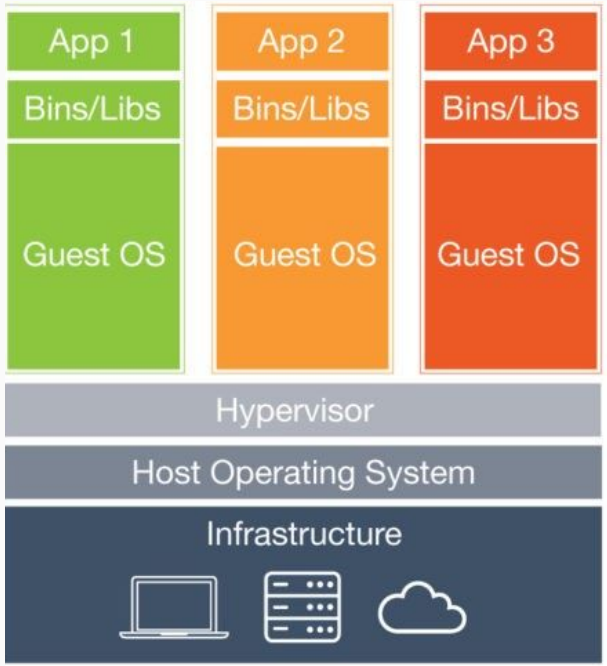
Docker – O que é?

“Ferramenta de empacotamento de uma aplicação e suas dependências em um container virtual”

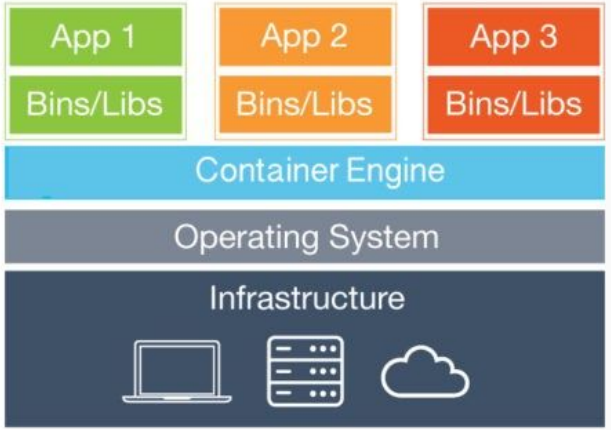
- Ambiente de execução auto-contido
- Compartilhamento do Kernel
- Isolamento dos demais containeres
- Baixo overhead e tempo de boot

Docker é uma máquina virtual?

Não, Containeres possuem uma arquitetura diferente.



Virtual Machine



Container

Prática

Instalando o docker - Linux

```
$ sudo apt-get update
$ sudo apt-get install docker-ce
$ sudo usermod -aG docker <usuario>
```

Ou de outras formas: *docs.docker.com*

Subindo um container

Para subir o container basta executar um *docker run* e qual imagem deve ser utilizada:

```
$ docker run --name <nome-container> ubuntu:18.04
```

Subindo um container

```
$ docker run --name <nome-container> ubuntu:18.04 /bin/echo/ “Hello World”
```

Por que o container não fica em pé?

“Containeres só são executados enquanto o comando especificado está ativo”

Mantendo container em pé – Alternativa 1

Container Interativo:

```
$ docker run -it --name <nome-container> ubuntu:18.04 /bin/bash
```

Mantendo container em pé – Alternativa 2

Container Detached:

```
$ docker run -d --name <nome-container> ubuntu:18.04 /bin/bash -c “while true; do
echo hello world; sleep 1; done”
```

Construindo imagem própria

Uma imagem é um Template/Modelo somente leitura, que é utilizado para subir um container.

Você precisa baixar uma imagem através do *docker hub* (ou outro repositório) para rodar os containeres ou montar a sua própria imagem utilizando o *dockerfile*.

Dockerfile

```
FROM python:2.7-slim
WORKDIR /app
COPY . /app
RUN pip install --trusted-host pypi.python.org -r flask
EXPOSE 80
ENV NAME World
CMD ["python", "app.py"]
```

Demo

Dockerfile - Instruções

```
FROM imagem[:tag] # A partir de qual imagem estamos nos baseando
RUN comando # Basicamente o que escrevemos em um script bash
WORKDIR /app # Diretorio "raiz" para os comandos seguintes
COPY . /app # Copia arquivos para dentro do container
VOLUME /app # Volumes expostos para fora do container
EXPOSE 3000 # Portas liberadas para fora do container
CMD ["comando", "parametros", …] # Que comando deve ser executado assim que um container sobe
```

Dockerfile – Construindo uma imagem

Após ter um *Dockerfile* escrito é preciso construir a imagem:

```
$ docker build -t <minha-imagem:0.1> .
```

Dockerfile – Subindo container

E agora podemos subir o container através da nossa imagem:

```
$ docker run -dit --name <meu-container> minha-imagem:0.1
```

Containeres e portas

Para acessar portas dentro do container utilizamos a flag ‘-p’:

```
$ docker run -dit --name my-apache -p 8888:80 httpd:2.4
```

Obs:

```
-p porta-da-minha-maquina:porta-dentro-do-container
```

Apagando containeres e imagens

Para apagar um container utilize o comando *docker rm*:

```
$ docker rm <nome-container>
```

Para apagar uma imagem utilize o comando *docker rmi*:

```
$ docker rmi <minha-imagem:1.0>
```

Referências:

- Documentação do Docker
- Docker - Hello World
- DockerFile
- Docker hub
- Documentação imagem httpd - Docker hub

Obrigado.