

Docker: o que é e para que serve

Luiz Sol e Marcos Vinicius

2019-03-11

O problema

A pessoa *A* desenvolveu um *pipeline* de *Machile Learning* em *Python* no seu computador pessoal.

O *setup* da pessoa A é:

- *Majaro* 17.1 (Arch Linux)
- *Python* 3.5
- *PostgreSQL* 10.3 (Banco de Dados)
- *PyTorch* 0.9
- *Pandas* 0.14

Agora a pessoa B precisa revisar o código (ver se foi bem escrito e se funciona corretamente).

O *setup* da pessoa *B* é:

- *Windows* 10
- *Python* 3.7
- *MySQL* 5.7 (Banco de Dados)
- *TensorFlow* 0.9
- *Pandas* 0.21

E uma vez que o código for aprovado ele deverá ser colocado no servidor de produção *C* (máquina que irá rodar de fato a aplicação).

O *setup* do servidor *C* é:

- *RHEL* 7.6 (Red Hat Linux)
- *Python* 3.6
- *Cassandra* 3.11 (Banco de Dados)
- *TensorFlow* 0.7
- *Pandas* 0.26

O Pesquisador *D* está tentando comparar a diferença de performance entre duas versões diferentes da *JVM* (*Java*), mas não quer ter que desinstalar e instalar cada versão para cada teste que precisar realizar.

E agora José?

Temos um problema de reproducibilidade e isolamento de ambientes de execução.

Possíveis soluções:

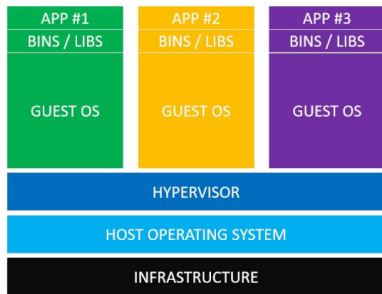
- **Solução 1:** Obrigar todos a utilizarem os mesmos softwares que estão no servidor de produção
- **Problemas da solução 1:**
 - Servidores tendem a utilizar versões antigas e estáveis de software, o que pode atrapalhar os pesquisadores
 - Nem sempre é possível usar para desenvolvimento o que se usa em produção (licenças, interface com o usuário, demanda computacional etc)
 - Restringir pesquisadores e desenvolvedores pode resultar na fuga de capital humano qualificado

- **Solução 2:** Utilizar *máquinas* virtuais que espelhem o setup de produção
- **Problemas da solução 2:**
 - Máquinas virtuais são grandes (~10GB) e consomem bastante memória (~6GB) por si só
 - Versionamento (controle de versões) de máquinas virtuais não é uma tarefa simples (arquivos binários)
 - A interação entre a máquina hópedeira e a máquina hóspede (máquina virtualizada) nem sempre é simples (arquivos, rede, *clipboard* etc)

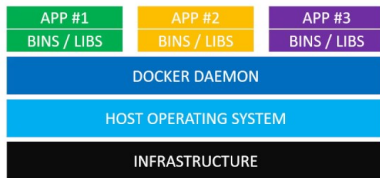
Docker ao resgate

O *Docker* tenta solucionar este problema criando “máquinas virtuais” compactas, flexíveis e reutilizáveis.

O *Docker* utiliza o próprio *kernel* (a “base”) do sistema operacional para executar as aplicações das máquinas virtualizadas.



Virtual Machines



Docker Containers

Os principais conceitos do *Docker* são:

- **Imagens:** instruções de como construir os containers
- **Containers:** as “máquinas virtuais” já construídas a partir dos containers
- **Volumes:** pastas no computador hospedeiro que o container irá utilizar para armazenar arquivos permanentemente
- **Redes:** as sub-redes das quais os containers farão parte

Imagens

São as “plantas” que serão utilizadas para construir os *containers*

São contruídas a partir de outras imagens que estão disponíveis no repositório público do Docker (entitulado Docker Hub)

Imagens de sistemas operacionais “puros”

☐ Docker Certified

Images

- ☐ Verified Publisher
- ☒ Official Images

Categories

- ☐ Analytics
- ☐ Application Frameworks
- ☐ Application Infrastructure
- ☐ Application Services
- ☐ Base Images
- ☐ Databases
- ☐ DevOps Tools
- ☐ Featured Images
- ☐ Messaging Services
- ☐ Monitoring
- ☒ Operating Systems
- ☐ Programming Languages
- ☐ Security
- ☐ Storage

Operating Systems

- ☐ Linux
- ☐ Windows

Architectures

- ☐ ARM
- ☐ ARM 64
- ☐ IBM POWER
- ☐ IBM Z
- ☐ PowerPC 64 LE
- ☐ x86
- ☐ x86-64

alpine
Updated an hour ago
A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!
Container Linux x86-64 IBM Z PowerPC 64 LE 386 ARM 64 ARM Featured Images Base Images
10M+ 5.0K Downloads Stars

ubuntu
Updated an hour ago
Ubuntu is a Debian-based Linux operating system based on free software.
Container Linux IBM Z PowerPC 64 LE 386 ARM 64 ARM x86-64 Base Images Operating Systems
10M+ 9.2K Downloads Stars

centos
Updated 44 minutes ago
The official build of CentOS.
Container Linux ARM 64 ARM x86-64 PowerPC 64 LE 386 Base Images Operating Systems
10M+ 5.2K Downloads Stars

debian
Updated 44 minutes ago
Debian is a Linux distribution that's composed entirely of free and open-source software.
Container Linux PowerPC 64 LE ARM 64 386 ARM x86-64 IBM Z Base Images Operating Systems
10M+ 3.0K Downloads Stars

fedora
Updated 44 minutes ago
Official Docker builds of Fedora
Container Linux IBM Z PowerPC 64 LE ARM 64 ARM x86-64 Base Images Operating Systems
10M+ 767 Downloads Stars

amazonlinux
10M+ 587 Downloads Stars

Imagens de sistemas operacionais com linguagens pré-instaladas

The screenshot displays the Docker Hub interface with a sidebar on the left and a main content area on the right. The sidebar includes filters for 'Images' (Verified Publisher, Official Images), 'Categories' (Analytics, Application Frameworks, etc.), 'Operating Systems' (Linux, Windows), and 'Architectures' (ARM, ARM 64, etc.). The main content area lists several programming language images: golang, python, openjdk, php, ruby, and java. Each entry shows the language logo, name, update time, description, and a list of supported architectures and operating systems. The 'Official Images' badge is visible for each entry.

☐ Docker Certified

Images

- ☐ Verified Publisher
Docker Certified And Verified Publisher Content
- ☒ Official Images
Official Images Published By Docker

Categories

- ☐ Analytics
- ☐ Application Frameworks
- ☐ Application Infrastructure
- ☐ Application Services
- ☐ Base Images
- ☐ Databases
- ☐ DevOps Tools
- ☐ Featured Images
- ☐ Messaging Services
- ☐ Monitoring
- ☐ Operating Systems
- ☒ Programming Languages
- ☐ Security
- ☐ Storage

Operating Systems

- ☐ Linux
- ☐ Windows

Architectures

- ☐ ARM
- ☐ ARM 64
- ☐ IBM POWER
- ☐ IBM Z
- ☐ PowerPC 64 LE
- ☐ x86
- ☐ x86-64

golang
Updated 41 minutes ago
Go (golang) is a general purpose, higher-level, imperative programming language.
Container Linux Windows PowerPC 64 LE x86 ARM 64 ARM x86-64 IBM Z Programming Languages
10M+ 2.3K Downloads Stars

python
Updated 41 minutes ago
Python is an interpreted, interactive, object-oriented, open-source programming language.
Container Windows Linux PowerPC 64 LE x86 ARM 64 ARM x86-64 IBM Z Programming Languages
10M+ 3.9K Downloads Stars

openjdk
Updated 41 minutes ago
OpenJDK is an open-source implementation of the Java Platform, Standard Edition
Container Linux Windows ARM 64 x86-64 x86 ARM IBM Z PowerPC 64 LE Programming Languages
10M+ 1.3K Downloads Stars

php
Updated 41 minutes ago
While designed for web development, the PHP scripting language also provides general-purpose use.
Container Linux ARM 64 ARM x86-64 IBM Z PowerPC 64 LE x86 Programming Languages
10M+ 4.3K Downloads Stars

ruby
Updated 41 minutes ago
Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.
Container Linux IBM Z PowerPC 64 LE x86 ARM 64 ARM x86-64 Programming Languages
10M+ 1.6K Downloads Stars

java
Updated 40 minutes ago
10M+ 2.0K Downloads Stars

Imagens de sistemas operacionais com bancos de dados pré-instalados

☐ Docker Certified

Images

☐ Verified Publisher
Docker Certified And Verified Publisher Content

☒ Official Images
Official Images Published By Docker

Categories

☐ Analytics

☐ Application Frameworks

☐ Application Infrastructure

☐ Application Services

☐ Base Images

☒ Databases

☐ DevOps Tools

☐ Featured Images

☐ Messaging Services

☐ Monitoring

☐ Operating Systems

☐ Programming Languages

☐ Security

☐ Storage

Operating Systems

☐ Linux

☐ Windows

Architectures

☐ ARM☐ ARM 64☐ IBM POWER☐ IBM Z☐ PowerPC 64 LE☐ x86☐ x86-64

**mongo**
Updated an hour ago

OFFICIAL IMAGE

10M+ 5.6K
Downloads Stars

MongoDB document databases provide high availability and easy scalability.

Container Windows Linux x86-64 ARM 64 Databases

**postgres**
Updated an hour ago

OFFICIAL IMAGE

10M+ 6.3K
Downloads Stars

The PostgreSQL object-relational database system provides reliability and data integrity.

Container Linux ARM 64 ARM x86-64 IBM Z PowerPC 64 LE 386 Databases

**redis**
Updated an hour ago

OFFICIAL IMAGE

10M+ 6.6K
Downloads Stars

Redis is an open source key-value store that functions as a data structure server.

Container Linux Windows IBM Z PowerPC 64 LE 386 ARM 64 ARM x86-64 Databases

**mariadb**
Updated an hour ago

OFFICIAL IMAGE

10M+ 2.6K
Downloads Stars

MariaDB is a community-developed fork of MySQL intended to remain free under the GNU GPL.

Container Linux 386 PowerPC 64 LE ARM 64 x86-64 Databases

**mysql**
Updated an hour ago

OFFICIAL IMAGE

10M+ 7.8K
Downloads Stars

MySQL is a widely used, open-source relational database management system (RDBMS).

Container Linux x86-64 Databases

**crate**
Updated an hour ago

OFFICIAL IMAGE

10M+ 115
Downloads Stars

Ex: Se eu quiser construir uma imagem para um container *Linux* que irá fazer *web scrapping* utilizando *Celery* eu posso criar a imagem:

- A partir da imagem de um *Linux* puro (*Alpine*, *Ubuntu* etc)
- A partir da imagem oficial do *Python* (que por sua vez foi criada a partir da imagem de um *Linux* puro)
- A partir de uma imagem que já tenha *Celery* instalado e configurado

Trechos do Dockerfile do *MySQL*

```
# Usando outra imagem como ponto de partida
FROM debian:stretch-slim # ...
# Executando comandos para instalar pacotes
RUN apt-get update && # ...
# Definindo variáveis de ambiente
ENV GOSU_VERSION 1.7 # ...
# Expondo uma das pastas para o hospedeiro
VOLUME /var/lib/mysql # ...
# Copiando arquivos para a imagem
COPY config/ /etc/mysql/ # ...
# Expondo a porta 3306 para o hospedeiro
EXPOSE 3306 33060 # ...
# Executando o daemon do MySQL
CMD ["mysqld"]
```

Fonte: MySQL Docker File

Esses comandos serão executados toda vez que um novo container de *MySQL* for construído.

Observações importante: **o sistema de arquivo dos *containers* é volátil!**
(Isso ficará mais claro quando falarmos sobre volumes)

Containers

Containers são instâncias de imagens.

- Um *container* é gerado a partir de uma imagem
- Um hospedeiro pode executar vários *containers* simultaneamente

Analogia: a imagem é o molde e os *containers* são os objetos criados a partir desse molde

Lembram-se que o sistemas de arquivos das imagens são efêmeros?

Vamos fazer um teste.

Vamos criar uma instância (*container*) a partir da imagem oficial do *Ubuntu*:

```
[luiz@SPWSPRD37L Downloads]$ docker create ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
6cf436f81810: Pull complete
987088a85b96: Pull complete
b4624b3efe06: Pull complete
d42beb8ded59: Pull complete
Digest: sha256:7a47ccc3bbe8a451b500d2b53104868b46d60ee8f5b35a24b41a86077c650210
Status: Downloaded newer image for ubuntu:latest
54c6c11aeac6e0226ea2aa70c2d5b73eac514c27c9773df77980eea438dbe5f7
[luiz@SPWSPRD37L Downloads]$ docker run -it -d ubuntu bash
80def1a3ad4993b6a62c0527cd9874a0499ba04357b3d3c687930f66b6ab5b93
[luiz@SPWSPRD37L Downloads]$ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	POR
TS	NAMES				
80def1a3ad49	ubuntu	"bash"	9 seconds ago	Up 7 seconds	
	friendly_galileo				

Agora que o *container* está rodando vamos acessar o container, criar um arquivo o arquivo `/root/file.txt` e depois parar o container.

```
[luiz@SPWSPRD37L Downloads]$ docker exec -it ubuntu shell
Error: No such container: ubuntu
[luiz@SPWSPRD37L Downloads]$ docker exec -it 80def1a3ad49 shell
OCI runtime exec failed: exec failed: container_linux.go:344: starting container process caused "exec:
\"shell\": executable file not found in $PATH": unknown
[luiz@SPWSPRD37L Downloads]$ docker exec -it 80def1a3ad49 bash
root@80def1a3ad49:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@80def1a3ad49:/# cd root/
root@80def1a3ad49:~# ls
root@80def1a3ad49:~# echo "Here comes a string..." >> file.txt
root@80def1a3ad49:~# ls
file.txt
root@80def1a3ad49:~# pwd
/root
root@80def1a3ad49:~# exit
exit
[luiz@SPWSPRD37L Downloads]$ docker stop 80def1a3ad49
80def1a3ad49
[luiz@SPWSPRD37L Downloads]$ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
TS	NAMES				

Reiniciando o *container*, acessando-o e procurando pelo arquivo temos uma surpresa:

```
[luiz@SPWSPRD37L Downloads]$ docker run -it -d ubuntu bash
20b1f1799d605b85a9c02cb5a746b739f8c7d54a724e8c2c8f48a56ceec5ca8
[luiz@SPWSPRD37L Downloads]$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
TS                NAMES
20b1f1799d60       ubuntu            "bash"             9 seconds ago      Up 7 seconds
optimistic_goldwasser
[luiz@SPWSPRD37L Downloads]$ docker exec -it 20b1f1799d60 bash
root@20b1f1799d60:/# cd root/
root@20b1f1799d60:~# ls
root@20b1f1799d60:~# ????
```

Os containers são como arquivos binários de programas que não podem ser modificados permanentemente.

E agora? Vamos ter que deixar nosso container de banco de dados rodando eternamente para não perder os dados?

Não.

Vamos usar ***Volumes!***

Volumes são pastas do sistema hospedeiro que “conectamos” (montamos) ao sistema de arquivo do container, dessa forma quando ele escrever nessas pastas ele estará escrevendo no sistema de arquivos da máquina hospedeira.

Talk is cheap, show me the code!

Vamos criar outro container, dando um nome a ele (para facilitar nossa vida) e montando a pasta `~/Downloads/volume_do_ubuntu` da máquina hospedeira na pasta `/root` do container e repetir o nosso experimento


```
[luiz@SPWSPRD37L Downloads]$ docker commit 20b1f1799d60 nova_imagem
sha256:4dc217e439e98e04b761d3813f4319d69f080714bc6ccafbf6e218fae6d9aa4c
[luiz@SPWSPRD37L Downloads]$ docker run -it -v "$PWD/volume_do_ubuntu/" :/root nova_imagem bash
root@b718de8ab764:~# cd /root/
root@b718de8ab764:~# ls
root@b718de8ab764:~# echo "I hope this works..." >> file.txt
root@b718de8ab764:~# ls
file.txt
root@b718de8ab764:~# exit
[luiz@SPWSPRD37L Downloads]$ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	POR
TS	NAMES				
20b1f1799d60	ubuntu	"bash"	21 minutes ago	Up 21 minutes	

```
[luiz@SPWSPRD37L Downloads]$ docker stop 20b1f1799d60
20b1f1799d60
[luiz@SPWSPRD37L Downloads]$ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	POR
TS	NAMES				

```
[luiz@SPWSPRD37L Downloads]$ docker run -it -v "$PWD/volume_do_ubuntu/":"/root nova_imagem bash
root@d546966b9fab:/# cd /root/
root@d546966b9fab:~# ls
file.txt
root@d546966b9fab:~# cat file.txt
I hope this works...
root@d546966b9fab:~# :)
```

É isso que fazemos com containers de bancos de dados: os arquivos utilizados pelas aplicações de banco de dados são armazenados em volumes no sistema hospedeiro.

Networks

Containers participam de uma rede só deles

Para que outros sistemas consigam se comunicar com eles é necessário conectar portas dos containers a portas da máquina hospedeira (isso ficará mais claro à frente)

Você pode estar pensando: Solucionamos um problema criando outro: toda vez que eu for rodar minhas aplicações terei que digitar dezenas de linhas de comandos para “subir” e “descer” os containers

Chamamos o gerenciamento de containers de **orquestração**

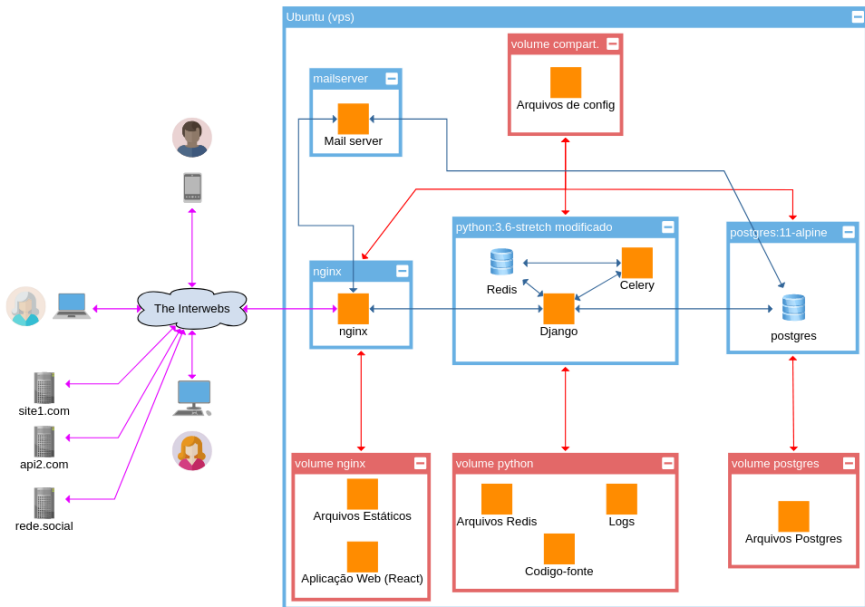
As soluções de orquestração de containers

- *Docker Compose*
- *Docker Swarm*
- *Kubernetes*

Chega de bla bla bla, vamos a um exemplo: o servidor de `luizsol.com`

Mas antes, vamos dar o nome aos bois:

- *Nginx*: servidor *proxy* (“porteiro” das requisições web, lida com *caching*, *timeout* etc)
- *Django*: *framework web* em *Python* (implementa sites, *APIs* etc)
- *Celery*: *framework* em *Python* de gestão de *tasks* (execução automática ou periódica de funções, ex: *Web Scraping*, monitoramento, geração de relatórios etc)
- *PostgreSQL*: banco de dados relacional *SQL* utilizado para persistência de várias aplicações
- *Redis*: banco de dados *NoSQL* de armazenamento em *RAM* do tipo chave-valor (super rápido, usado pelo *Celery* para despache de *tasks*)



Toda essa arquitetura de rede pode ser reproduzida em menos de 2 minutos por qualquer pessoa em qualquer sistema que tenha *Docker* instalado e acesso à internet com o comando:

```
docker-compose up
```

E todos os arquivos necessários para isso são arquivos de texto que estão armazenados e versionados em um repositório

luizsol / **luizsol.com** Private

Unwatch 1 Star 0 Fork 0

<> Code 1 Issues 1 Pull requests 0 Projects 1 Wiki Insights Settings

Repository for personal website Edit

Manage topics

47 commits 1 branch 0 releases 1 contributor

⚠ We found a potential security vulnerability in one of your dependencies.

Only the owner of this repository can see this message.

[Manage your notification settings](#) or [learn more about vulnerability alerts](#).

See security alert

Branch: master New pull request Create new file Upload files Find file Clone or download

luizsol Data Latest commit 11cd398 7 days ago

backend	Fixes comment editor bugs; Starts implementing Home page; Improves Ab...	14 days ago
web	Data	7 days ago
.gitignore	Adds Makefile to automate repetitive tasks	a month ago
Makefile	Fixes the blog post view; Adds support to friendly url	a month ago
README.md	Adds back to top button	3 months ago

Concluindo

- *Docker* é uma plataforma que oferece alta simplicidade, flexibilidade e reproducibilidade de ambientes de aplicações
- Vantagens:
 - Baixo *overhead*
 - Demanda menos espaço no disco e na memória
 - Toda o controle e configuração é feita via arquivos de texto, viabilizando versionamento
 - *Industry Standard*
 - Transforma um sistema de várias partes complexas em uma solução *plug-and-play*
- Desvantagens:
 - Não é trivial de se aprender
 - Exige algum nível de conhecimento do ambiente Linux (na maioria dos casos)
 - A fase inicial de configuração pode ser sofrida