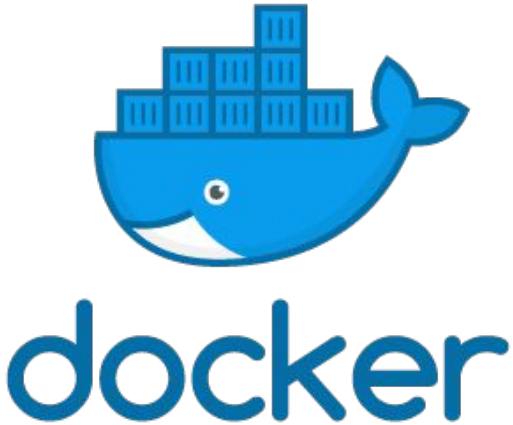
Introdução ao Docker



Stack

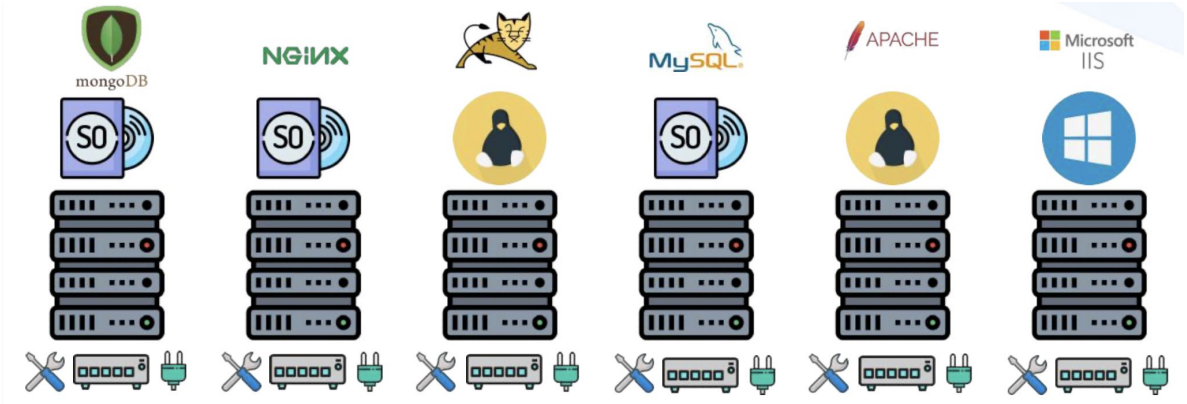
Primeiros passos com Docker

- O que são containers ?
- Como surgiram ?

Stack

Definição do problema

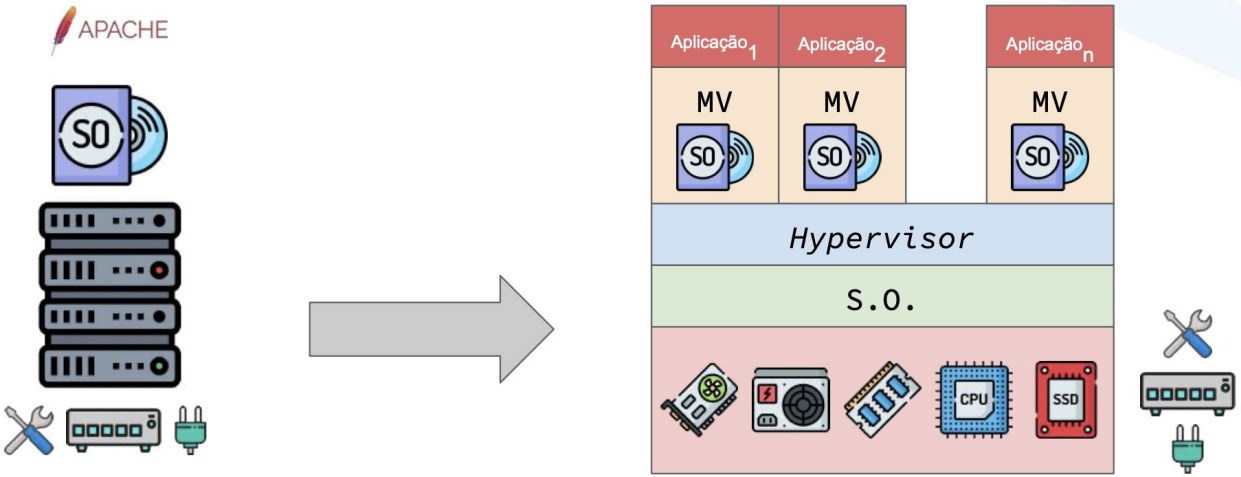
Como hospedar aplicações utilizando servidores físicos?



Stack

Máquinas virtuais

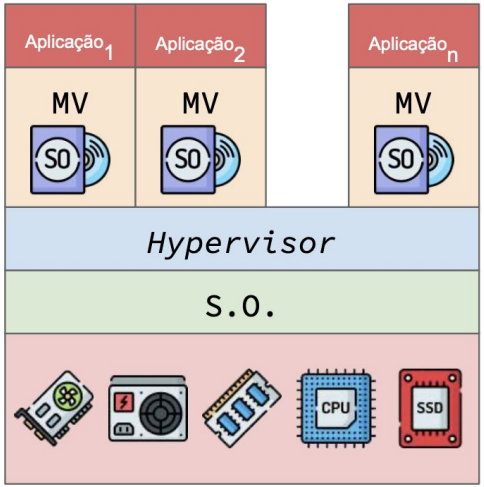
Como hospedar aplicações utilizando máquinas virtuais?



Stack

Definição do problema

Desvantagens das máquinas virtuais

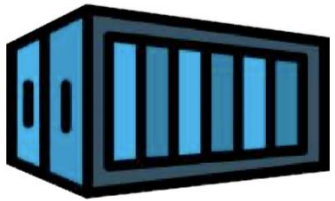


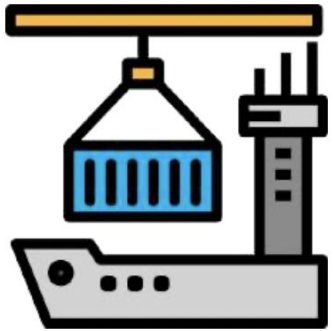
- Configuração
- Atualização
- Segurança
- Para cada máquina virtual
  - Armazenamento.
  - Processamento.
  - Memória.

Stack

Definição do problema

Como resolver o problema das máquinas virtuais?

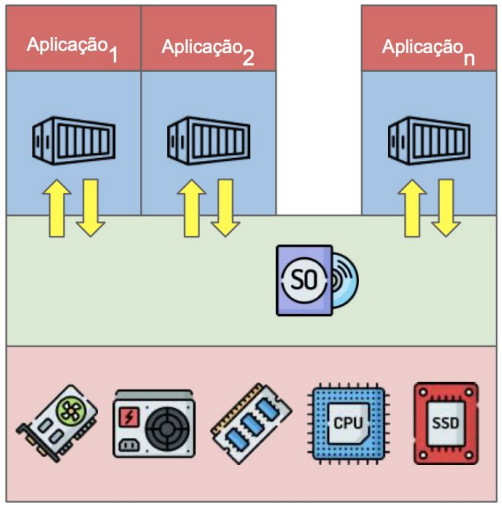




Stack

Containers

O que é um container?

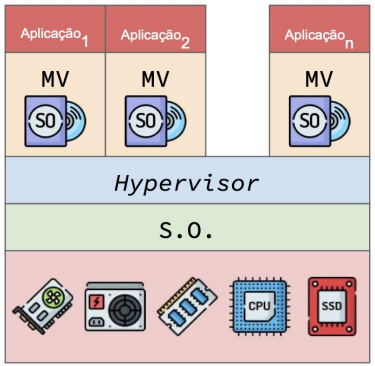


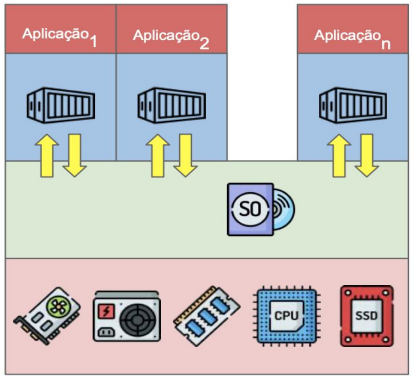
- Um container será um ambiente de virtualização
  - Com menos consumo de processamento,
  - Com menor custo de manutenção/configuração,
  - Com menor tempo de inicialização/finalização.

Stack

Máquinas virtuais vs Containers

- Máquinas virtuais fornece hardware virtual.
- Containers não utilizam hardware virtualizado.
- A tecnologia de containers simplesmente utiliza do conceito de containers já disponível no sistema operacional da máquina host.

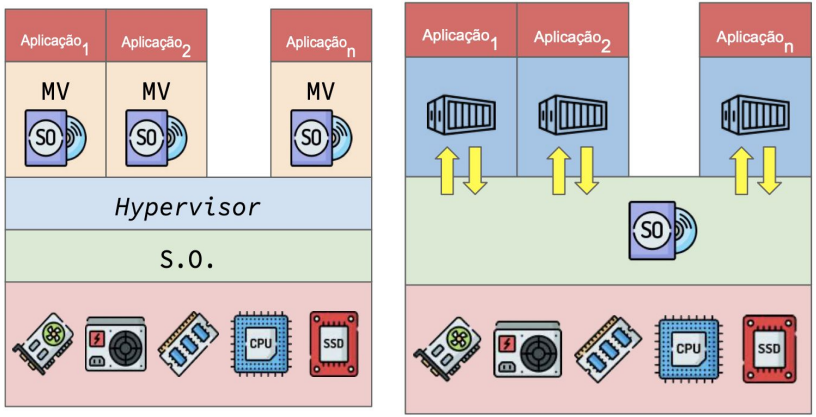




Stack

Máquinas virtuais vs Containers

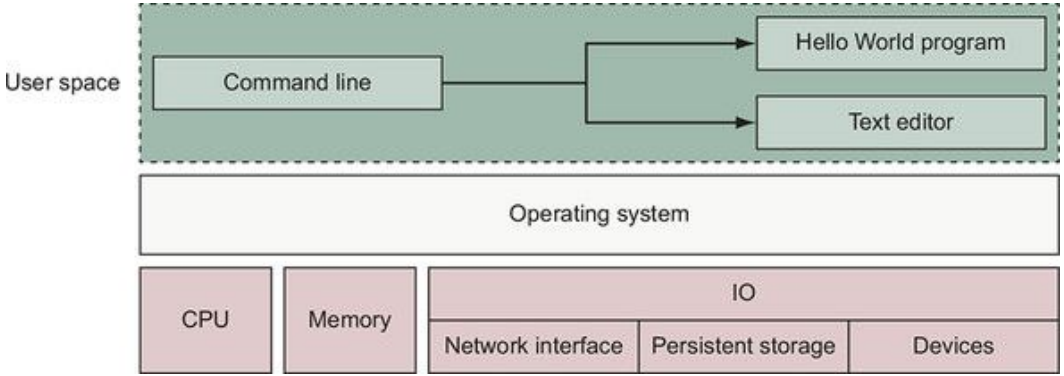
Máquinas virtuais fornecem abstrações de hardware e assim podemos executar sistemas operacionais.
Já os containers são uma feature do sistema operacional.



Stack

Executando softwares em containers para isolamento

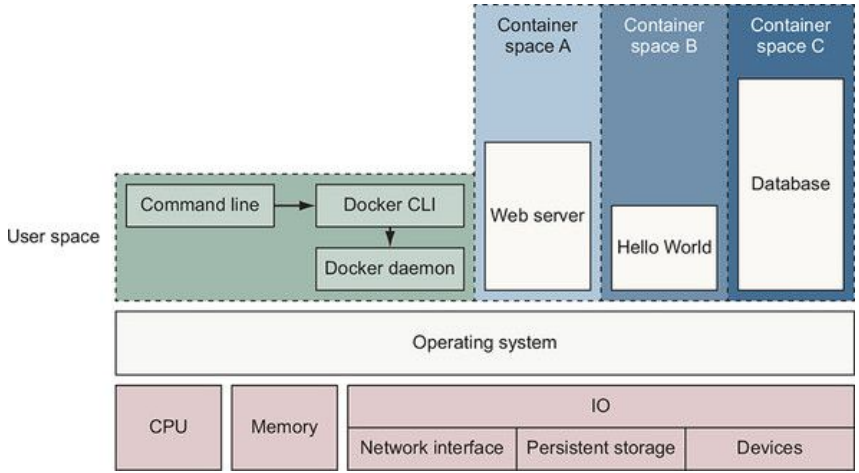
Espaço do usuário, sistema operacional e componentes de hardware.



Stack

Executando softwares em containers para isolamento

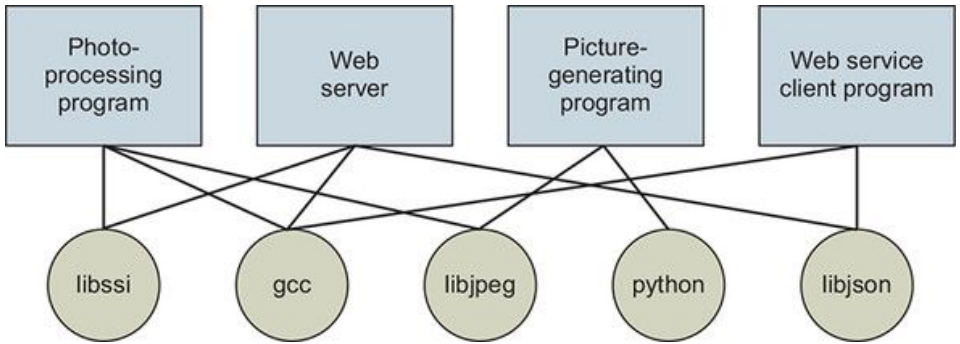
Docker CLI, Daemon e containers.



Stack

Qual problema o Docker resolve ?

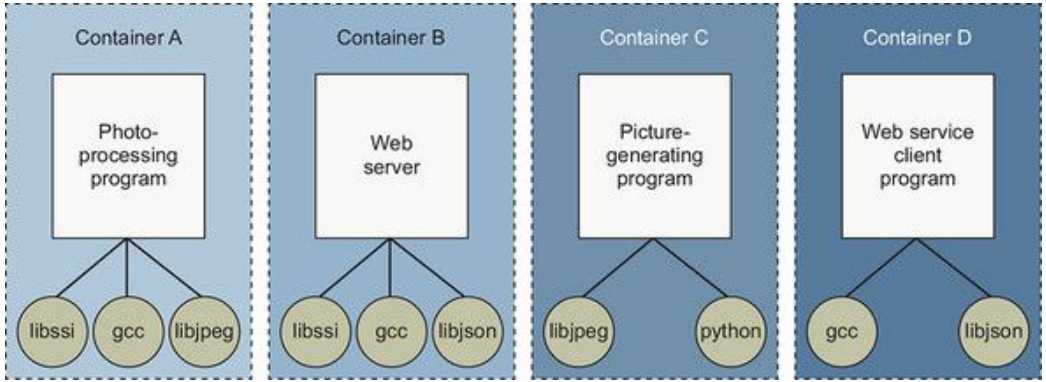
- Redução da complexidade para deploy de softwares.
- Dependência de outras instalações.



Stack

Qual problema o Docker resolve ?

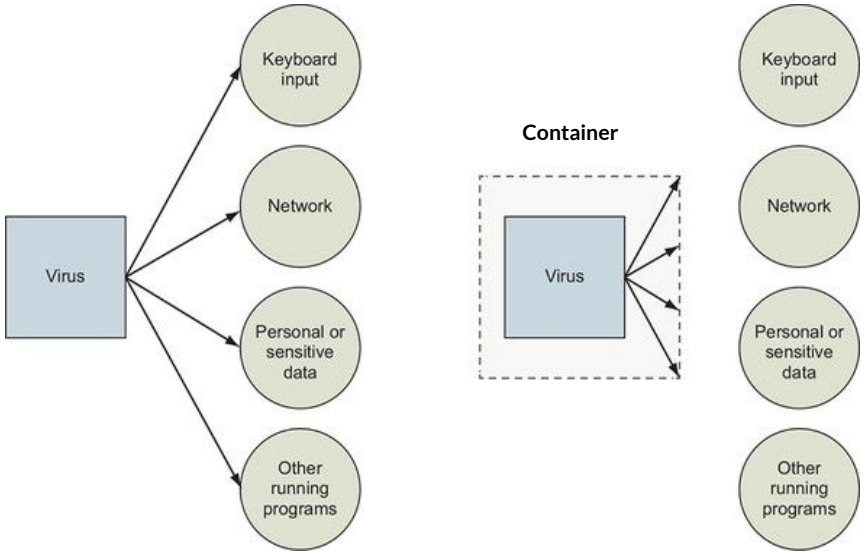
- Redução da complexidade para deploy de softwares.
- Dependência de outras instalações.
- Organização



Stack

Qual problema o Docker resolve ?

- **Portabilidade.**
- **Segurança.**



Stack

Porque o Docker é importante?

- Abstração.
  - Simplificação
  - Padronização (Processo de deploy é sempre o mesmo)
  - Economia de tempo, dinheiro e energia.
- Adotado pela comunidade.
- Docker pode ser utilizado em grandes ambientes (Kubernetes)

Stack

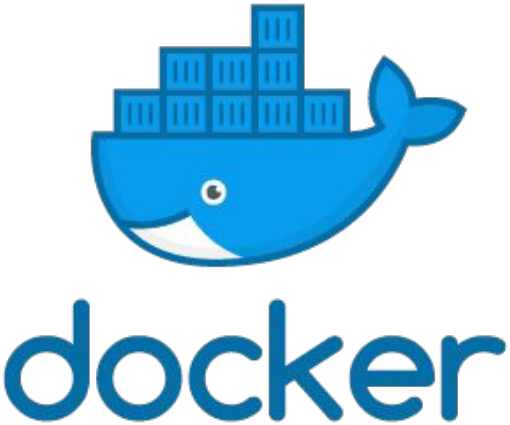
O que sabemos até aqui?

- Docker utiliza uma abordagem logística para resolver problemas comuns de software.
- É um programa de linha de comando com processos em background e um conjunto de serviços remotos.
- A abstração do container é o core da abordagem.
- Trabalhar com containers proporciona uma interface mais consistente e permite o desenvolvimento de ferramentas mais sofisticadas.
- Containers oferece mais segurança.
- Docker é disponível em Linux, MacOS e Windows.
- Docker utiliza da tecnologia de containers já disponível nos sistemas operacionais.
- Docker pode ser utilizado em grandes ambientes.

Docker

É um programa que executa na **linha de comando**, um **processo em background** e um conjunto de serviços remotos que resolvem problemas comuns de softwares e simplifica a experiência de instalação, execução, publicação e remoção de sistemas.

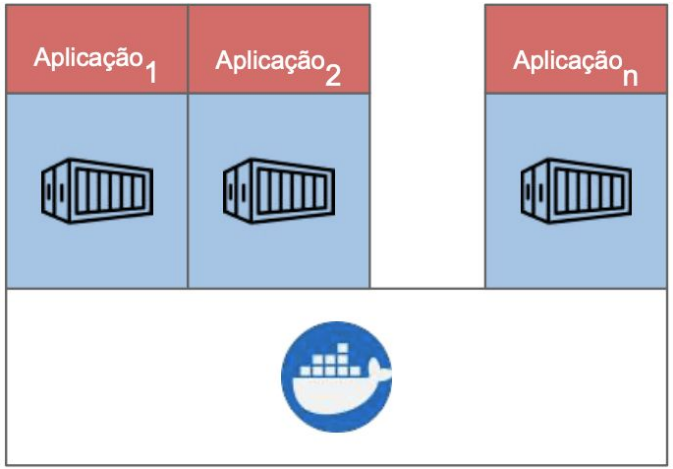
Para isso ele utiliza uma tecnologia de sistemas operacionais chamada containers.



Stack

Docker

Alternativa de virtualização em que o kernel da máquina hospedeira é compartilhado com a máquina virtualizada ou o software em operação.



Stack

Docker

Hypervisor x OS-level virtualization

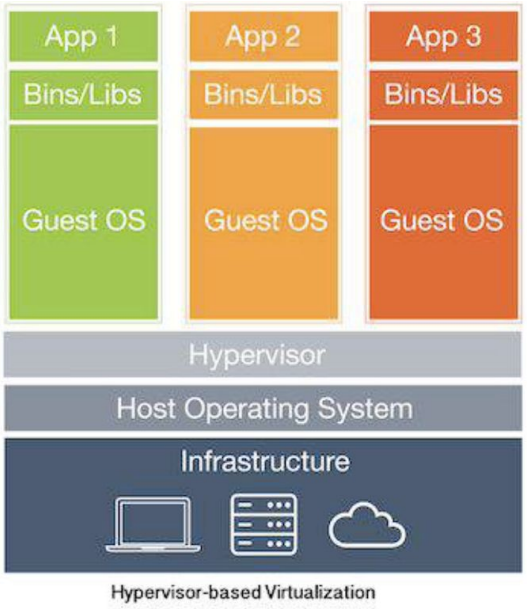
O hypervisor apresenta aos sistemas operacionais convidados uma plataforma operacional virtual e gerencia a execução dos sistemas operacionais convidados.

Várias instâncias de vários sistemas operacionais podem compartilhar os recursos de hardware virtualizado: por exemplo, as instâncias Linux, Windows e macOS podem ser executadas em uma única máquina x86 física.

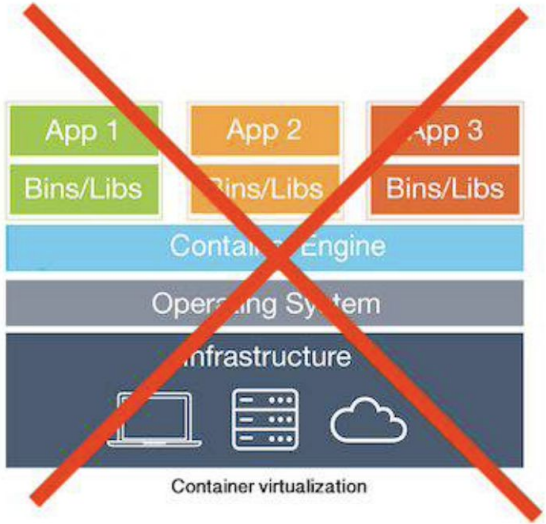
Isso contrasta com a virtualização no nível do sistema operacional, onde todas as instâncias (contêineres) devem compartilhar um único kernel, embora os sistemas operacionais convidados possam diferir no espaço do usuário, como distribuições Linux diferentes com o mesmo kernel.

Stack

Docker não é um Hypervisor



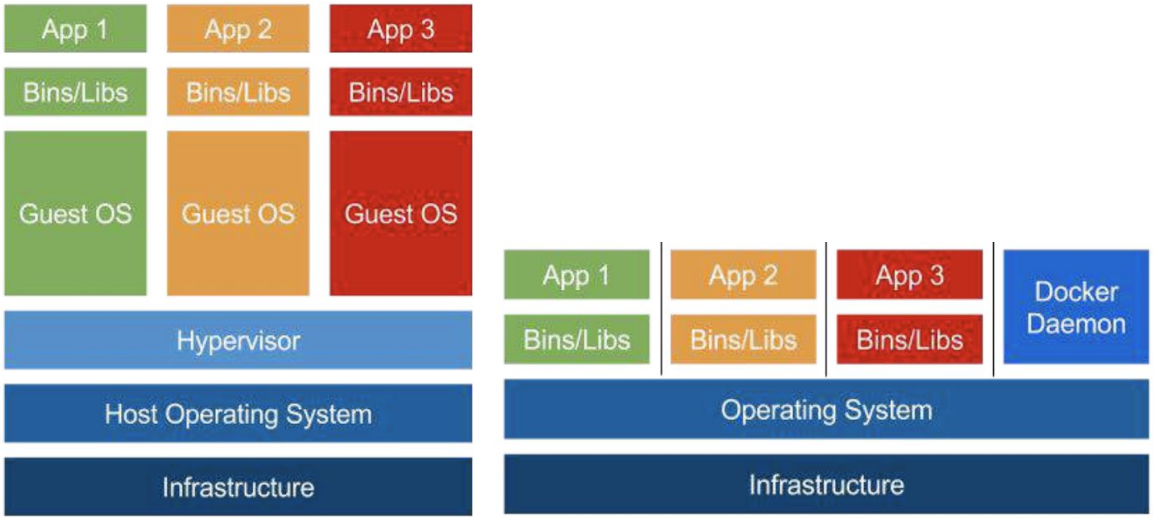
Hypervisor-based Virtualization



Container virtualization

Stack

Docker não é um Hypervisor



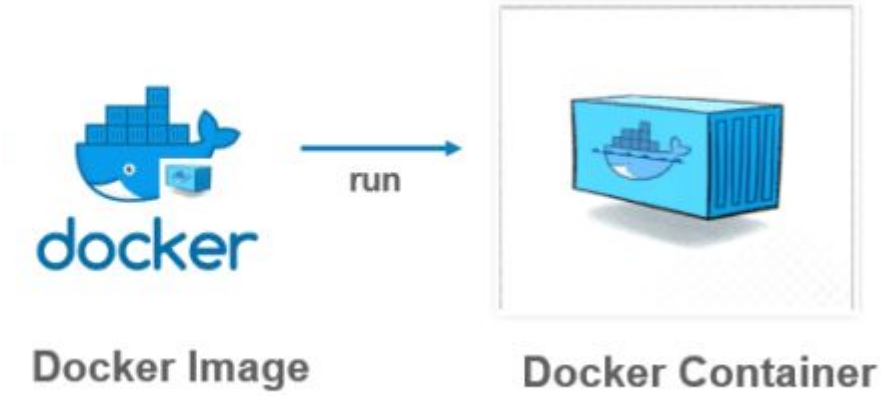
Stack

Alguns conceitos importantes....

Stack

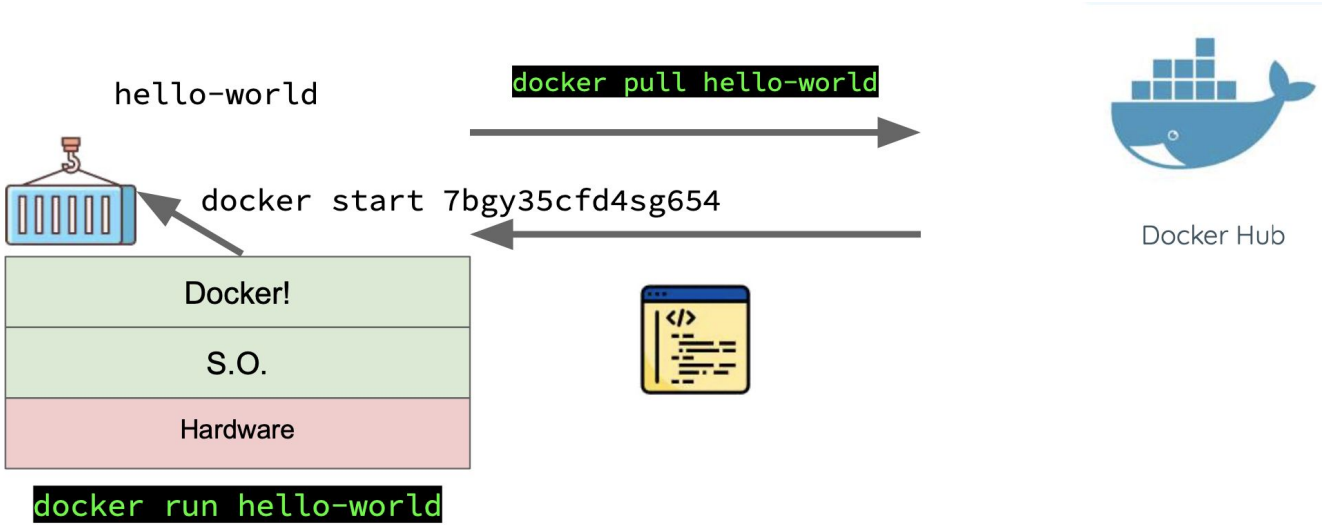
Docker Image

Imagem é uma coleção de arquivos e metadados.
Nos metadados contém específicos programas para executar e outros detalhes de configuração.



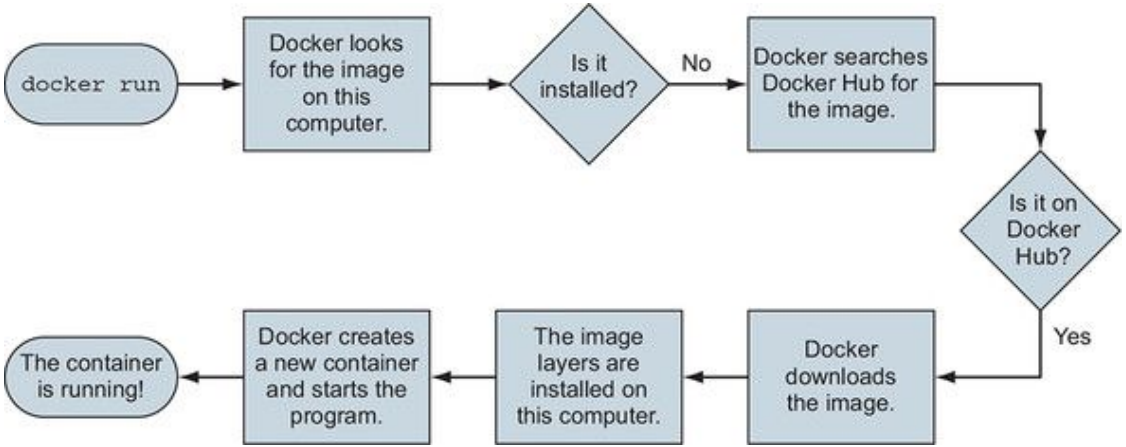
Stack

O fluxo de execução...



Stack

Docker run



Stack

Docker run

Stack

Hands On!
Instalação do Docker

(Play with Docker)

https://labs.play-with-docker.com/

Stack

Docker no Windows

Windows 10 PRO (Docker Desktop)
https://www.docker.com/products/docker-desktop

Windows 7,8, 10 Home (DockerToolbox)
https://github.com/docker/toolbox/releases
https://devconnected.com/how-to-install-docker-on-windows-7-8-10-home-and-pro/

Stack

Hello World!

docker container run hello-world

Para verificar a versão atual do docker...
docker version

Stack

Criando outro container...

Criando um container com o Alpine.
docker container run alpine

Listando imagens
docker image ls

Stack

Imagens populares

https://hub.docker.com/search/?type=image

Comandos básicos

Verificando os containers ativos.
docker ps

Verificando os containers inativos e parados.
docker ps -a

Criando um container e executando.
docker container run alpine echo "Ola Mundo"

Especificando uma tag para a criação do container, se não especificar sempre vai pegar a **latest**.
docker container run ubuntu:18.04 cat /etc/issue

Stack

Comandos básicos

Instanciando um container e executando comandos.
docker container run -d alpine sleep 20

Instanciando um container, definindo um nome e executando comandos.
docker container run -d --name linux alpine sleep 20

Parando um container
docker container stop <container-id>

O comando run sempre cria novos containers. Criando um novo container removendo o anterior.
docker container run --rm alpine echo "Ola Mundo"

Especificando uma tag para a criação do container, se não especificar sempre vai pegar a **latest**.
docker container run ubuntu:18.04 cat /etc/issue

Verificando os logs do container
docker container logs <container-id/container-name>

Verificando os logs do container de forma iterativa.
docker container logs -f <container-id/container-name>

Stack

Exercícios

1. Crie um container CentOS.
2. Execute o comando sleep e suspenda o SO por 2000 segundos.
3. Verifique os containers em execução.
4. Pare o container que está executando o comando sleep.

Stack

Comandos básicos

Ligando o terminal da nossa máquina com o terminal dentro do container.
docker container run -it <container-id/container-name> bash

Executando comandos em um container em execução.
docker container exec <container-id> ls

Para remover um container.
docker container rm <container-id/container-name>

Stack

Comandos básicos

Parar um container.
docker container stop <container-id/container-name>

Para inicializar um container.
docker container start <container-id/container-name>

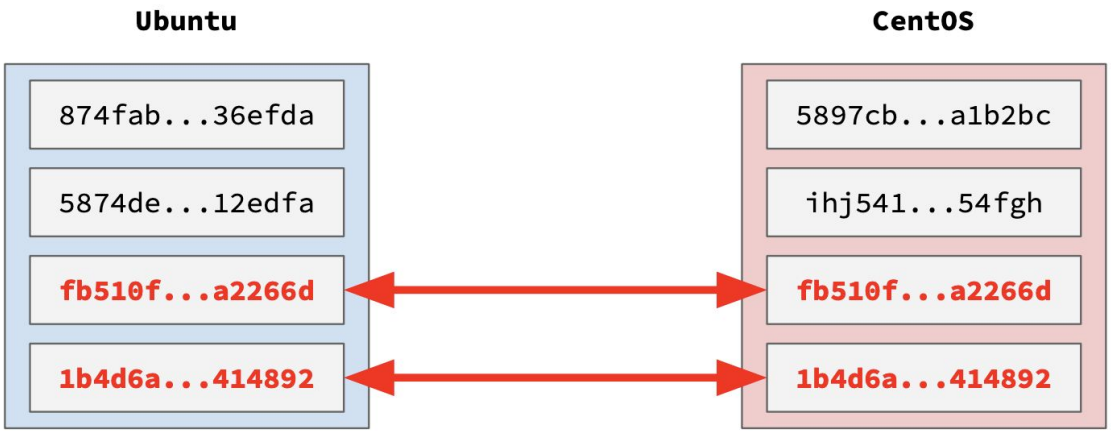
Para remover uma imagem.
docker image rmi <container-id/container-name>

Para remover todo os containers inativos.
docker container prune

Stack

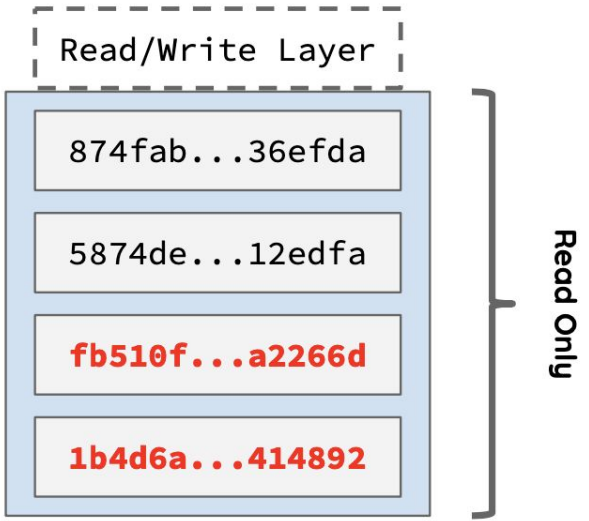
Sistema de Camadas no Docker

Layered File System

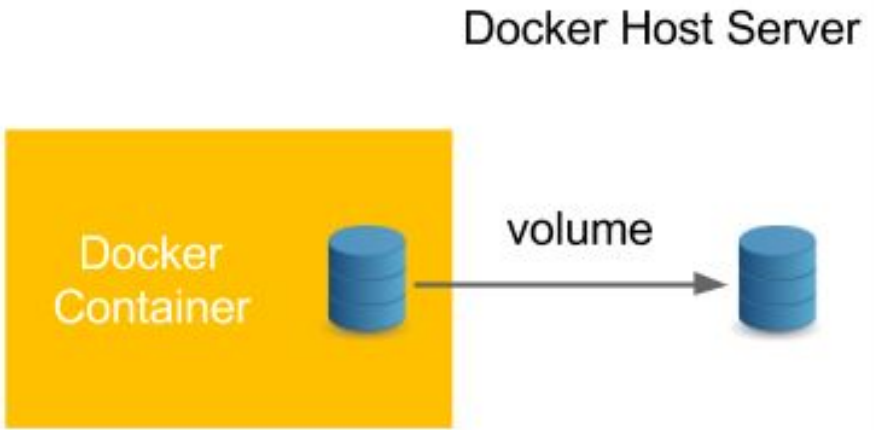


Sistema de Camadas no Docker

Layered File System



Sistema de volumes



Comandos básicos

Criando um container e especificando um volume para escrita no host
docker container run -d --name linux -v "<dir-host:dir-container>" image

Criando um container e especificando um volume para escrita no host
docker container run -d --name linux -v "/data/server/:/tmp" ubuntu:18.04

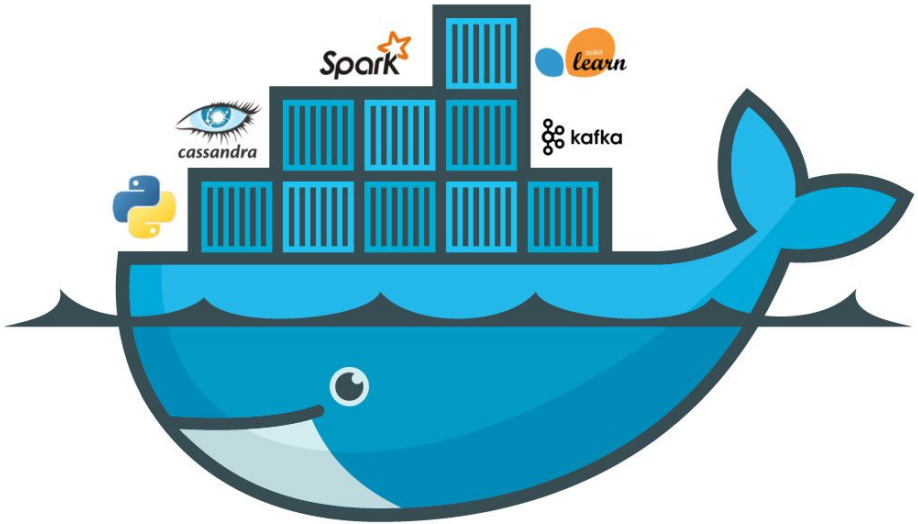
Criando um container e especificando como volume para escrita no host
docker container run -d --name linux -v "$PWD/data:/tmp" ubuntu:18.04

Enviando uma variável de ambiente para o container.
docker container run --name linux2 -e vartest="variavel de teste" ubuntu env

Criando um container de um servidor web e roteando a porta 8080 do host com a 80 do container..
docker container run -d --name webserver -p 8080:80 nginx

Stack

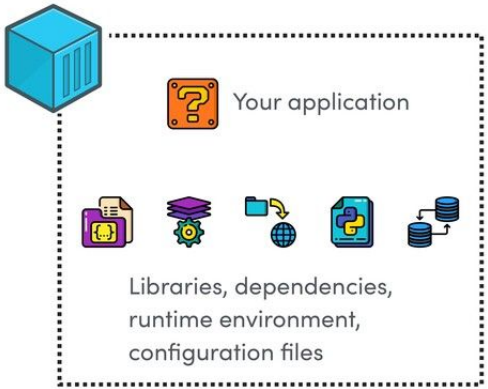
Trabalhando com Docker em projetos de Data Science

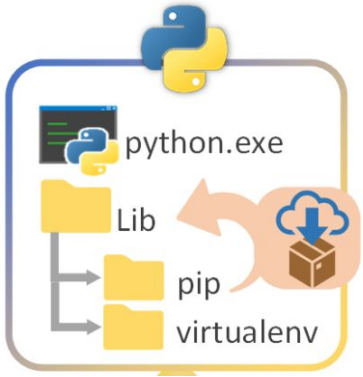


Stack

Docker vs pip

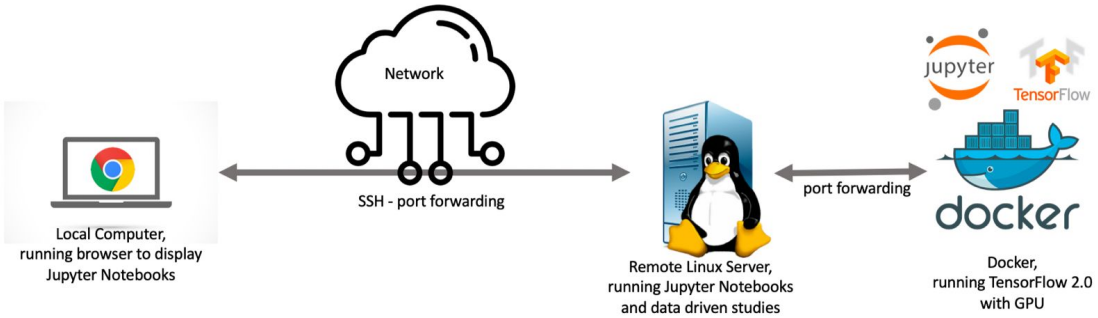
1. Armazenar todas as dependências com o **pip freeze > requirements.txt** não é suficiente para um deploy eficiente.
2. Em containers temos além de todas as bibliotecas e recursos do ambiente Python **configurações do sistema operacional, compiladores, drivers, configuration files.**





Stack

Ambiente de Desenvolvimento para Data Science



Stack

Ambiente de Data Science com Docker

Criando o container e fazendo as configurações necessárias.

```
docker container run -it --name jupyter-server \
                -v  "$PWD/data:/tmp"  \
                -p "8888:8888" ubuntu:16.04 bash
```

Instalação de bibliotecas e softwares dentro do container.

- Atualizando o repositório de pacotes do sistema.
  - `apt-get update --fix-missing`
- Instalando o script wget para fazer download e softwares básicos.
  - `apt-get install -y wget bzip2 ca-certificates build-essential byobu curl git-core pkg-config python3-dev python-pip python-setuptools python-virtualenv unzip`

Stack

Ambiente de Data Science com Docker

Instalação de bibliotecas e softwares dentro do container.

- Limpando os arquivos temporários e listas.
  - apt-get clean
  - rm -rf /var/lib/apt/lists/
- Definindo a variável de ambiente com a localização para instalação do anaconda.
  - echo 'export PATH=/opt/conda/bin:$PATH' > /etc/profile.d/conda.sh
- Fazendo o Download do Anaconda (Verifique se o link está correto)
  - wget --quiet https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86_64.sh -O ~/anaconda.sh

Stack

Ambiente de Data Science com Docker

Instalação de bibliotecas e softwares dentro do container.

- Instalando o Anaconda e suas bibliotecas.
  - **/bin/bash ~/anaconda.sh -b -p /opt/conda**
- Excluindo o script de instalação.
  - **rm ~/anaconda.sh**
- Atualizando a variável de ambiente PATH
  - **export PATH="/opt/conda/bin:$PATH"**
- Inicializando o Jupyter Notebook
  - **jupyter notebook --ip=0.0.0.0 --allow-root --no-browser**

Stack

Automatizando com o DockerFile

Stack

DockerFile

- Formato
  - # Comment
  - **INSTRUCTION arguments**
- O DockerFile sempre começa com a instrução **FROM.**
  - **FROM ubuntu:16.04**
  - **FROM alpine**
- Instrução **RUN.**
  - **RUN <command> (shell, /bin/sh -c on Linux or cmd /S /C on Windows)**
  - **RUN ["executable", "param1", "param2"] (exec)**

DockerFile

- Instrução **RUN**.
  - RUN <command> (shell, /bin/sh -c on Linux or cmd /S /C on Windows)
  - RUN ["executable", "param1", "param2"] (exec)
    - RUN ["echo hello"]
    - RUN ["c:\\windows\\system32\\tasklist.exe"] (Barras invertidas no Windows)
- Instrução **CMD**.
  - Apenas um CMD em todo o DockerFile.
  - Diferente do RUN, o CMD não executa em tempo de build.
  - CMD ["executable","param1","param2"] (exec form)
  - CMD ["param1","param2"] (parâmetros para o ENTRYPOINT)
  - CMD command param1 param2 (shell)
    - CMD echo "This is a test." | wc -
    - CMD ["/usr/bin/wc","--help"]

DockerFile

- Instrução **LABEL.**
  - **LABEL <key>=<value> <key>=<value> <key>=<value> ...**
  - **LABEL "versao"="1.0"**
  - **LABEL "autor"="Felipe"**
  - **LABEL "descricao"="Esta imagem contém todas as bibliotecas para..."**
- Instrução **EXPOSE.**
  - **EXPOSE <port> [<port>/<protocol>...]**
  - **EXPOSE 80/tcp**
    - **docker run -p 80:80 ...**

DockerFile

- Instrução **ENV.**
  - **ENV <key>=<value> ...**
  - **ENV MY_VAR my-value**
  - **ENV dir=/home/user**
  - **ENV dir /home/user**
- Instrução **EXPOSE.**
  - **EXPOSE <port> [<port>/<protocol>...]**
  - **EXPOSE 80/tcp**
    - **docker run -p 80:80 ...**

Stack

DockerFile

- Instrução **COPY**.
  - COPY [--chown=<user>:<group>] <src>... <dest>
  - ADD hom* /mydir/
  - ADD hom?.txt /mydir/
  - ADD test.txt relativeDir/
    - Caminho relativo ao WORKDIR no container.
- Instrução **ENTRYPOINT**.
  - ENTRYPOINT ["executable", "param1", "param2"]
  - ENTRYPOINT command param1 param2
    - FROM ubuntu
    - ENTRYPOINT ["top", "-b"]
    - Transforma o container em um executável. O comando será sempre executado.

Stack

DockerFile

- Instrução **ENTRYPOINT.**
  - **FROM debian:stable**
  - **RUN apt-get update && apt-get install -y --force-yes apache2**
  - **EXPOSE 80 443**
  - **ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]**
- Instrução **WORKDIR.**
  - **WORKDIR /path/to/workdir**
  - **WORKDIR /dir1**
  - **WORKDIR dir2**
  - **WORKDIR dir3**
  - **RUN pwd**
    - **diretório de trabalho para as instruções RUN, CMD, ENTRYPOINT, COPY e ADD.**

Stack

DockerFile Final

```
FROM ubuntu:16.04

RUN apt-get update --fix-missing && apt-get install -y wget bzip2 ca-certificates \
    build-essential \
    byobu \
    curl \
    git-core \
    htop \
    pkg-config \
    python3-dev \
    python-pip \
    python-setuptools \
    python-virtualenv \
    unzip \
    && \
apt-get clean && \
rm -rf /var/lib/apt/lists/*

RUN echo 'export PATH=/opt/conda/bin:$PATH' > /etc/profile.d/conda.sh && \
    wget --quiet https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86_64.sh -O ~/anaconda.sh -O ~/anaconda.sh && \
    /bin/bash ~/anaconda.sh -b -p /opt/conda && \
    rm ~/anaconda.sh

ENV PATH /opt/conda/bin:$PATH

EXPOSE 8888

RUN mkdir ds
ENV HOME=/ds
ENV SHELL=/bin/bash
WORKDIR /ds

CMD ["jupyter", "notebook","--ip=0.0.0.0","--allow-root", "--no-browser"]
```

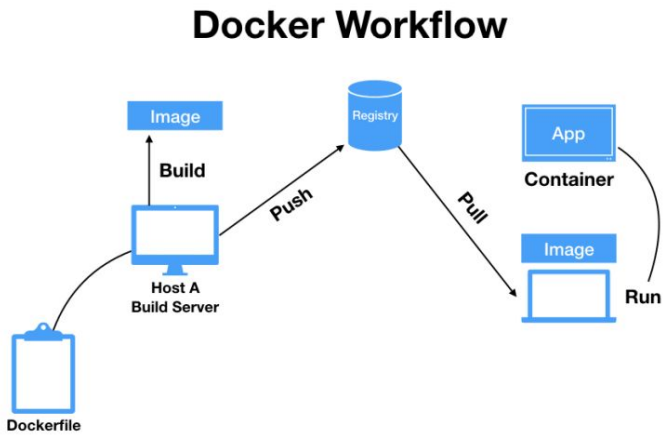
Docker Build e Tag

- Com o nosso DockerFile criado o próximo passo é construir a nossa imagem.
  - **docker build -t felipesf05/dsimage .**
- Listando a imagem criada.
  - **docker images**
- Testando a nossa nova imagem criada.
  - **docker container run -d --rm -p 8888:8888 felipesf05/dsimage**
- Adicionando uma tag para fazer o push.
  - **docker tag <imagem-fonte> <docker-id/imagem>:tag**
  - **docker tag felipesf05/dsimage felipesf05/dsimage:1.0**

Stack

Docker Push

- Realizando a autenticação.
  - **docker login -u <docker-id>**
- Submetendo o push para o Docker Hub.
  - **docker image push felipesf05/dsimage:1.0**



Stack

Documentação.

- Continue aprendendo.
  - https://docs.docker.com/engine/reference/run/

Stack