

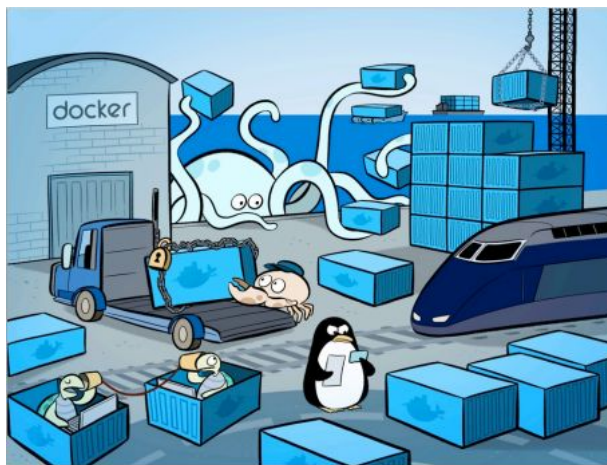


Desenvolvimento de Software para Nuvem

Prof. Dr. Fernando Antonio Mota Trinta
Prof. Dr. Paulo Antonio Leal Rego

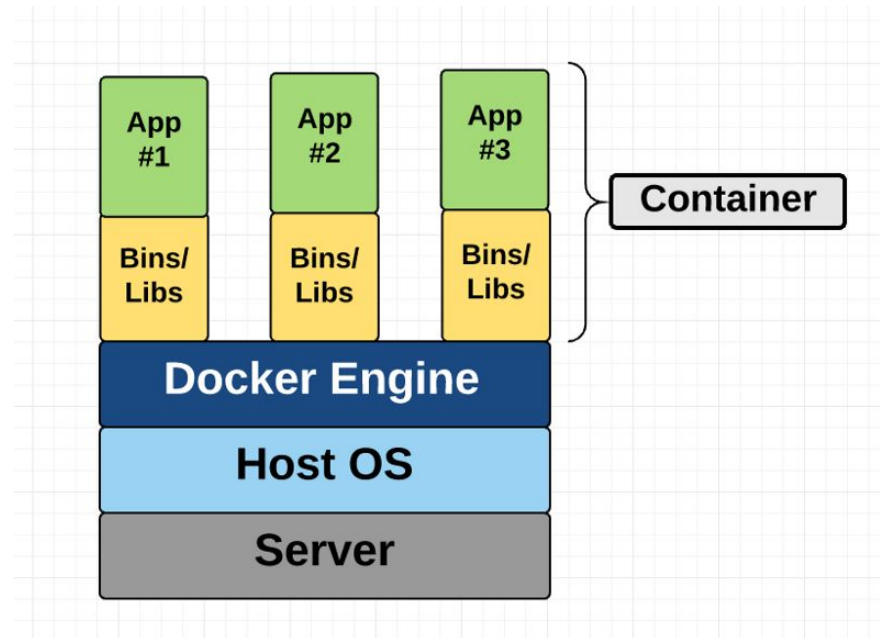
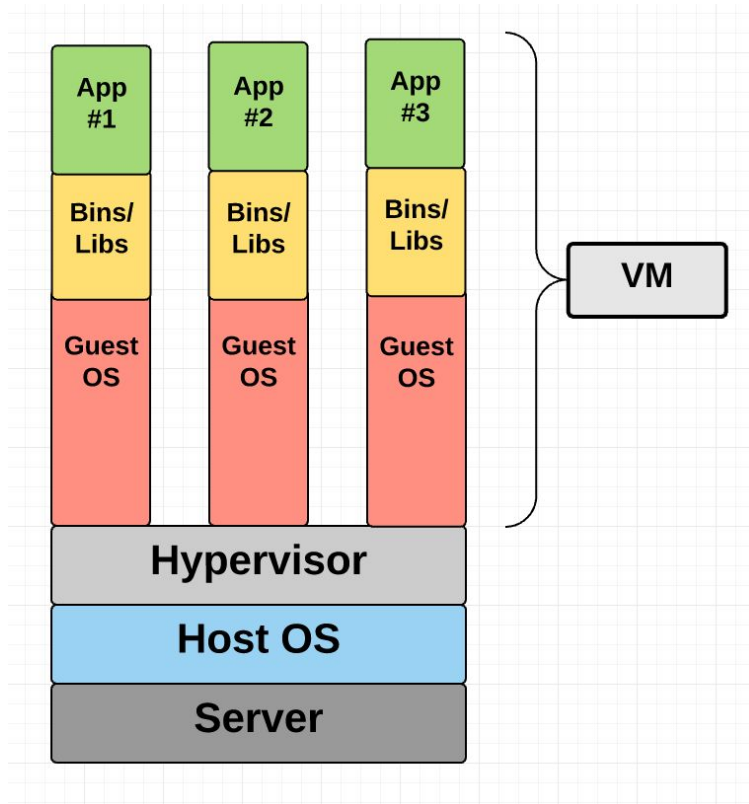
Containers Docker

Definição





- ❑ É uma plataforma aberta para desenvolvedores e administradores de sistemas criarem, fazerem deploy e executarem aplicações distribuídas.
- ❑ É uma tecnologia de software que fornece uma camada adicional de abstração e automação de virtualização de nível de sistema operacional, e empacota aplicações, códigos e dependências.

VM x Container



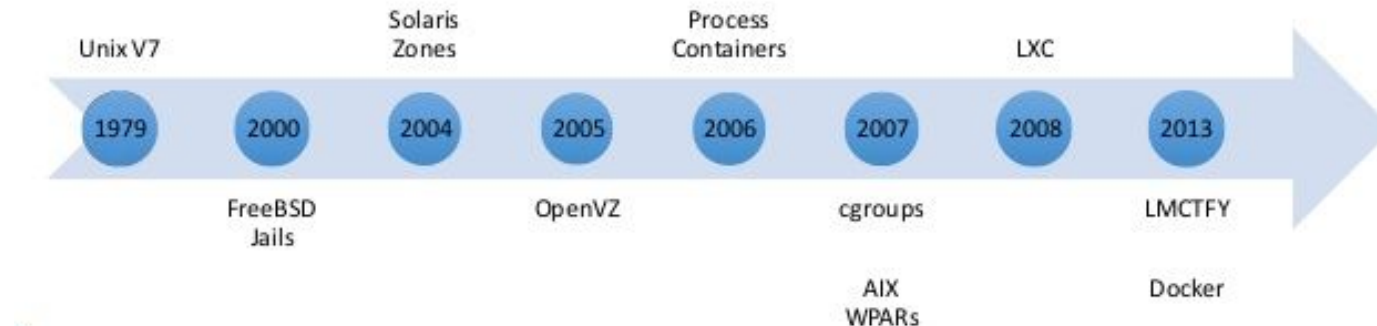
VM x Container

Criteria	 Virtual Machine	Docker 
OS support	Occupies a lot of memory space	Docker Containers occupy less space
Boot-up time	Long boot-up time	Short boot-up time
Performance	Running multiple virtual machines leads to unstable performance	Containers have a better performance as they are hosted in a single Docker engine
Scaling	Difficult to scale up	Easy to scale up
Efficiency	Low efficiency	High efficiency
Portability	Compatibility issues while porting across different platforms	Easily portable across different platforms
Space allocation	Data volumes cannot be shared	Data volumes can be shared and reused among multiple containers

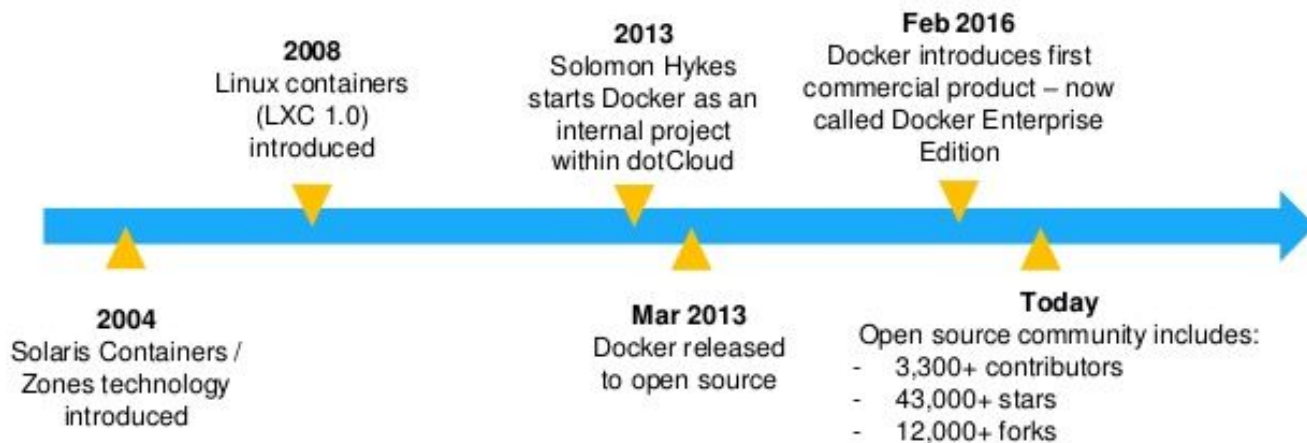
Containers

- ❑ Uso de Containers não é novidade
 - ❑ Google tem usado tecnologia própria de containers há anos.
 - ❑ Existem outras tecnologias além do Docker: Solaris Zones, BSD jails, and LXC.
- ❑ Docker is é um projeto open-source baseado na tecnologia Linux containers (LXC), que usa várias features do Kernel Linux.
 - ❑ É a plataforma de containers mais utilizada

Containers Timeline



A história do Docker

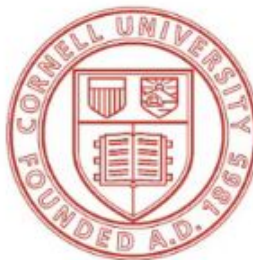


Por que usar Docker?

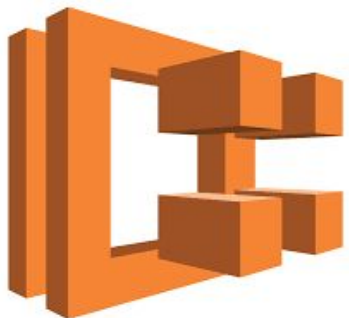
- ❑ Economia significativa de recursos
 - ❑ Containers consomem menos memória e storage
- ❑ Facilidade e velocidade para implantar
- ❑ "Pack Once, Run Everywhere"
- ❑ Gerenciamento facilitado
 - ❑ Fácil e barato de escalar
- ❑ Disponibilidade maior do sistema
 - ❑ Inicialização rápida
- ❑ Possibilidade de compartilhamento (DockerHub)
- ❑ Open source e forte comunidade
- ❑ Funciona no Linux, Windows e Mac



Quem está usando o Docker?



Plataformas de nuvem já suportam o Docker



Amazon ECS

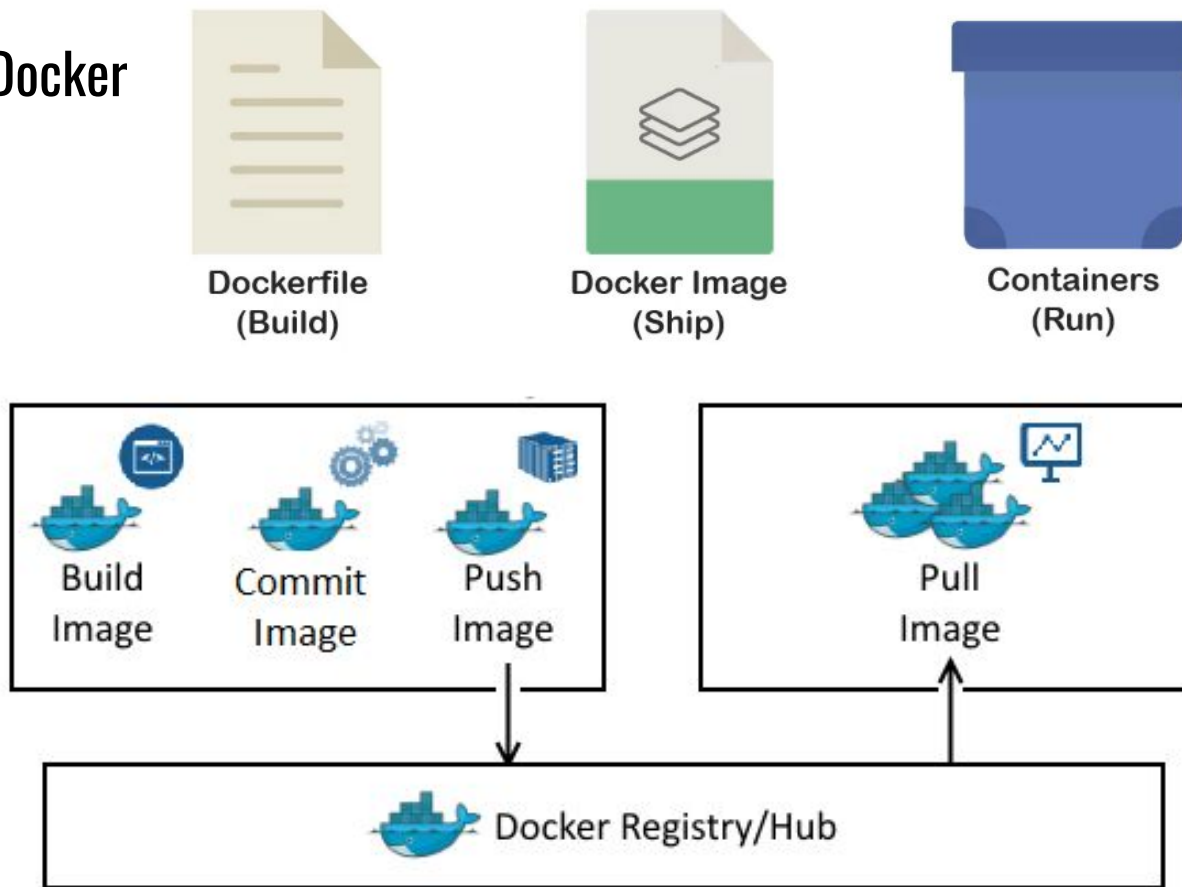


Google Compute Engine



**Heroku deploy
com container**

Ecosystem Docker



Orquestração



- ❑ O Docker mostrou-se uma boa opção para criar ambientes de sistemas que atendam a diferentes estágios de desenvolvimento (testes, homologação e produção). Mas será que o Docker conseguiria administrar o ambiente de **Produção**?
- ❑ Gerenciar diversos containers distribuídos não é uma tarefa fácil.
- ❑ Containers são passíveis de travas, lentidão, e podem sofrer instabilidades nos servidores reais e/ou serem insuficientes para uma demanda maior de serviços.

Orquestração

- ❑ As ferramentas de orquestração de containers são aplicações que permitem fazer o gerenciamento de múltiplos contêineres e seus principais objetivos são:
 - ❑ Cuidar do ciclo de vida dos containers de forma autônoma, replicando e distribuindo de acordo com as especificações ou demandas;
 - ❑ Gerenciar volumes e rede.

Ferramentas de Orquestração

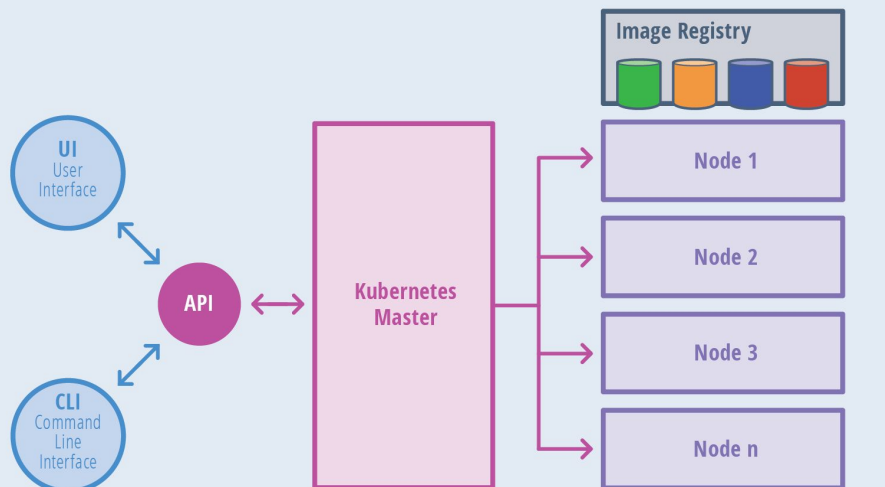


Kubernetes

- ❑ Gerencia aplicativos em containers através de múltiplos hosts de um cluster e, assim, amplia a estabilidade e controle do serviço.
- ❑ Ferramenta mais utilizada pelas empresas, por ter sido um dos primeiros orquestradores e por ser open source.
- ❑ Desenvolvida pela Google para uso interno. Em 2015, foi entregue ao Cloud Native Computing Foundation.
- ❑ Os serviços do Kubernetes suportam descoberta, escalabilidade e balanceamento.

Kubernetes

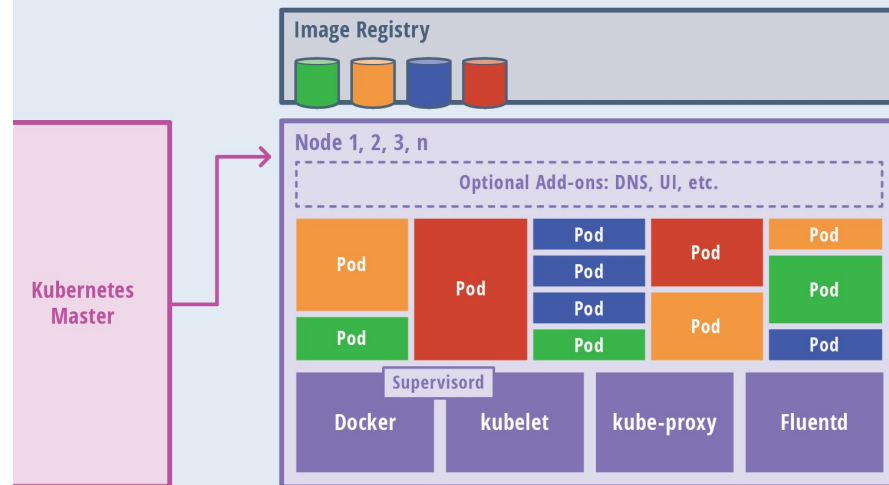
Kubernetes Architecture



Source: Janakiram MSV

THE NEW STACK

Kubernetes Node



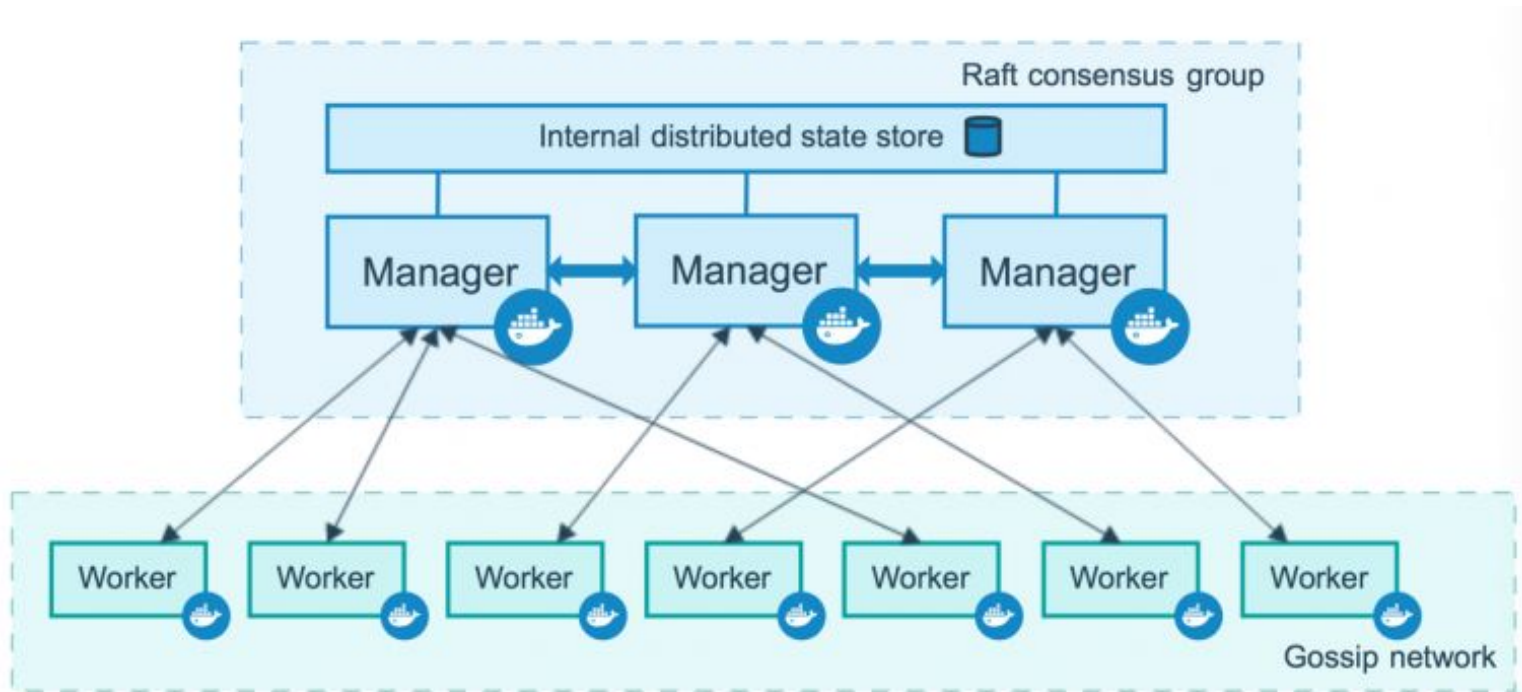
Source: Janakiram MSV

THE NEW STACK

Docker Swarm

- ❑ Ferramenta para cluster da Docker.
 - ❑ Era uma ferramenta a parte e passou a ser nativa.
- ❑ O “modo Swarm” é um recurso que fornece funcionalidades de orquestração de containers, incluindo clustering nativo de hosts do Docker e agendamento de cargas de trabalho.
- ❑ Funcionalidades:
 - ❑ auto-scaling
 - ❑ balanceamento de carga
 - ❑ descoberta de serviços

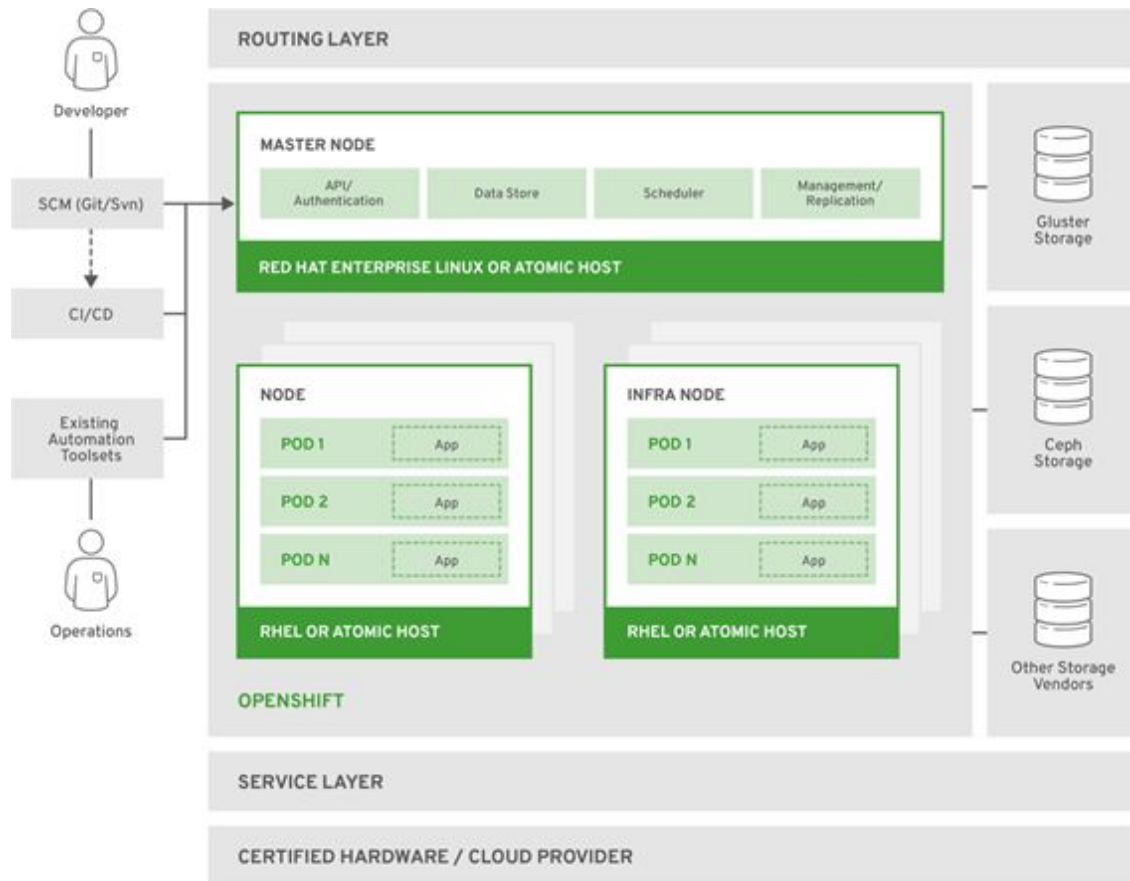
Docker Swarm



Openshift

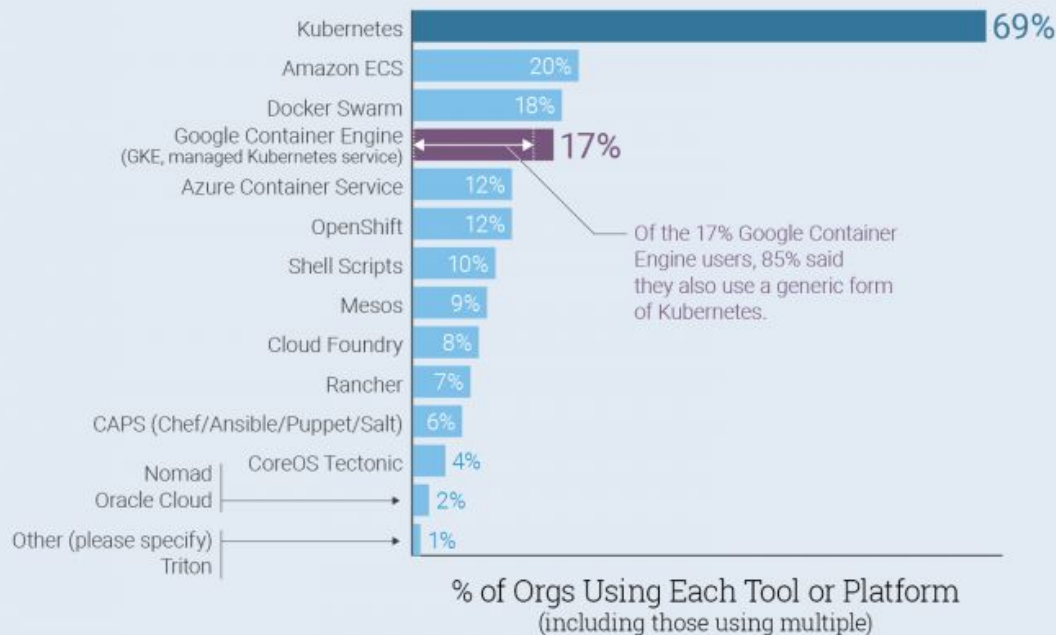
- ❑ Plataforma de código aberto desenvolvida pela Red Hat que auxilia no processo de orquestração de containers baseada em Kubernetes e containers Linux, independente da plataforma em que os containers estão sendo executados.
- ❑ Permite controlar todo o ciclo de vida de uma aplicação baseada em containers, desde o deploy até a execução, graças à integração com diferentes ferramentas e SDKs.
- ❑ Funcionalidades: gerenciamento dos processos de integração/entrega contínua (CI/CD), gerenciamento de configurações e logs, monitoramento da saúde das aplicações e containers.

Openshift



Ferramentas de Orquestração

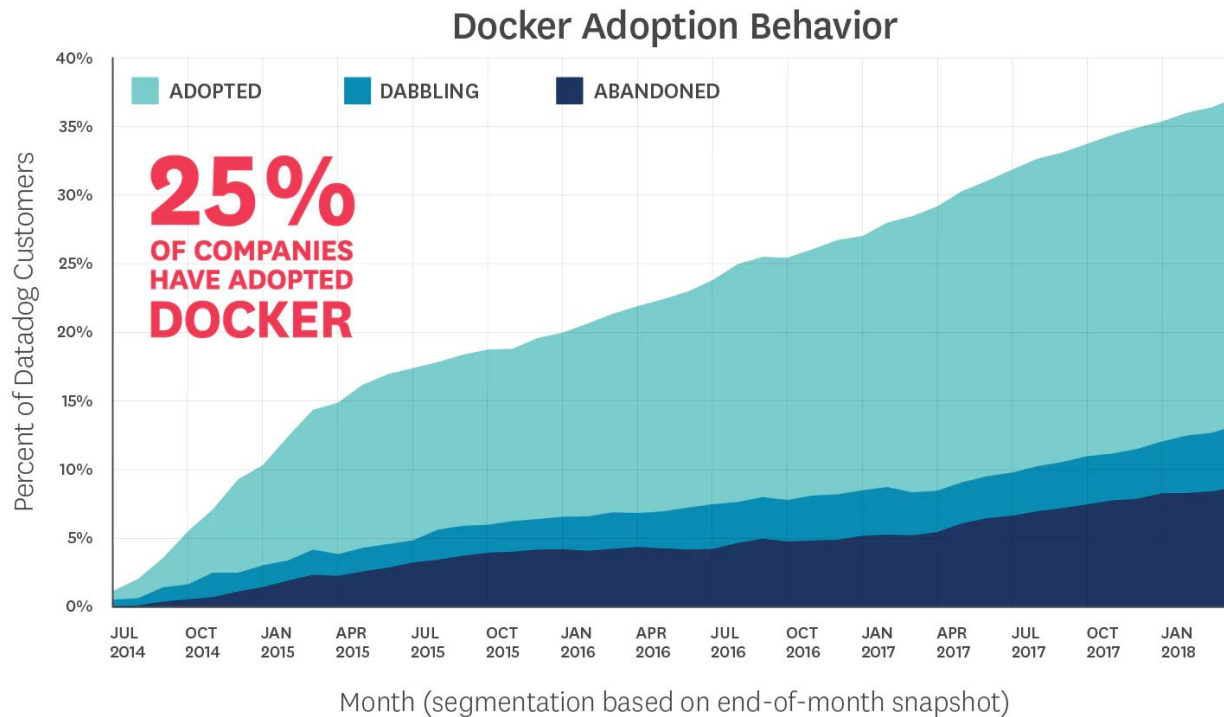
Kubernetes Manages Containers at 69% of Organizations Surveyed



Source: The New Stack Analysis of Cloud Native Computing Foundation survey conducted in Fall 2017.
Q. Your organization manages containers with... (check all that apply)? n=763.

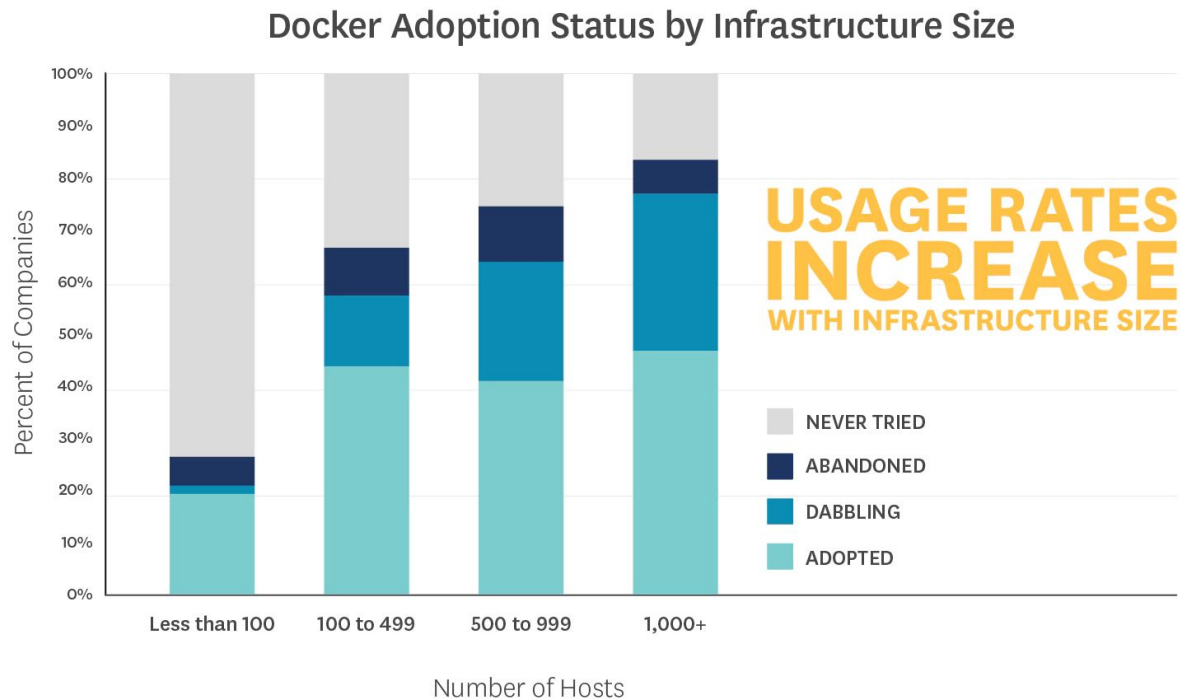
THE NEW STACK

Docker Facts



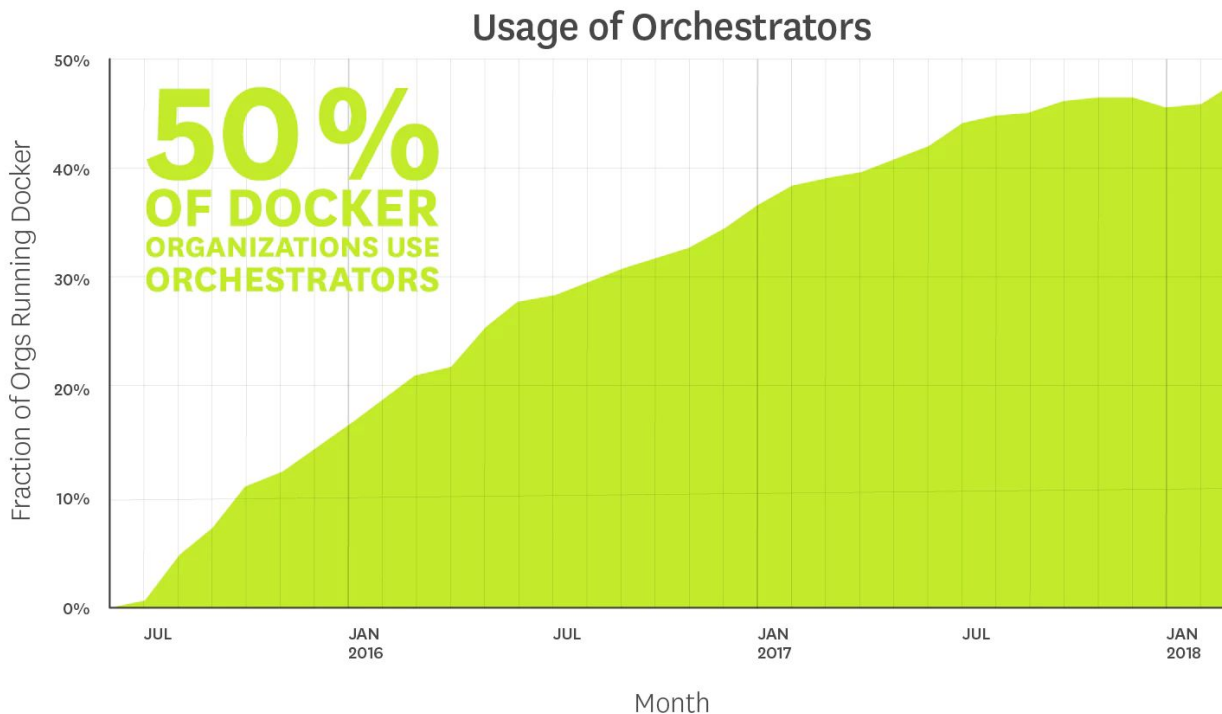
Source: Datadog

Docker Facts



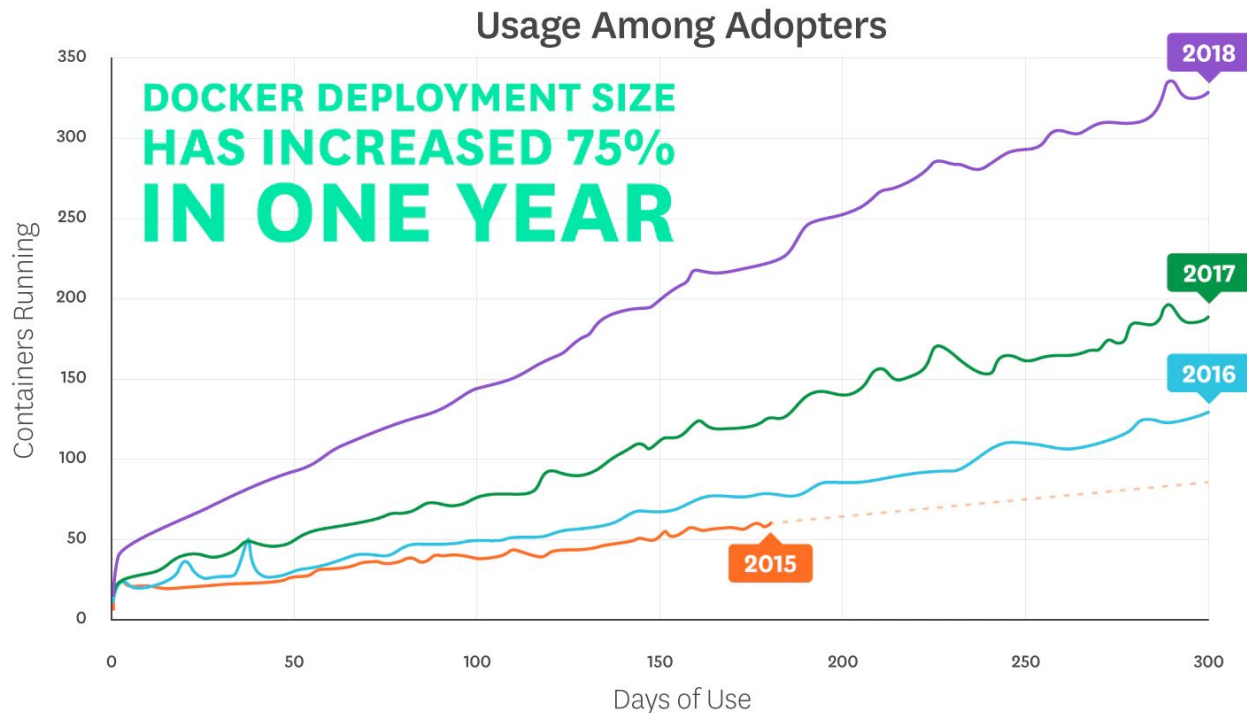
Source: Datadog

Docker Facts



Source: Datadog

Docker Facts



Source: Datadog

Mais: <https://www.datadoghq.com/docker-adoption/>

Desenvolvimento de Software para a Nuvem

Instalação e Configuração do Docker

Instalar Docker Community Edition (CE) no Ubuntu 18.04

1. Atualize sua lista de pacotes:
 - a. `sudo apt update`
2. Instale pacotes que permitem que o APT utilize pacotes via HTTPS:
 - a. `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
3. Então adicione a chave GPG para o repositório oficial do Docker em seu sistema:
 - a. `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
4. Adicione o repositório do Docker às fontes do APT:
 - a. `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"`
5. Atualize sua lista de pacotes (vai atualizar os pacotes do repositório recém adicionado):
 - a. `sudo apt update`
6. Instale o Docker
 - a. `sudo apt install docker-ce`
7. Verifique se o Docker está sendo executado
 - a. `sudo systemctl status docker`

Configurar o Docker para ser executado sem a necessidade do comando sudo

- ❏ O comando docker só pode ser executado pelo usuário root ou por um usuário do grupo docker (que é criado durante o processo de instalação).

- ❏ Configurar:
 1. Adicione o usuário ao grupo docker
 - a. `sudo usermod -aG docker <usuário>`
 2. Refaça o login para que a modificação surta efeito
 - a. `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
 3. Confirme se o usuário foi de fato adicionado ao grupo docker
 - a. `id -nG`

Prática Docker

Comandos

<code>attach</code>	Attach local standard input, output, and error streams to a running container
<code>build</code>	Build an image from a Dockerfile
<code>commit</code>	Create a new image from a container's changes
<code>cp</code>	Copy files/folders between a container and the local filesystem
<code>create</code>	Create a new container
<code>diff</code>	Inspect changes to files or directories on a container's filesystem
<code>events</code>	Get real time events from the server
<code>exec</code>	Run a command in a running container
<code>export</code>	Export a container's filesystem as a tar archive
<code>history</code>	Show the history of an image
<code>images</code>	List images
<code>import</code>	Import the contents from a tarball to create a filesystem image
<code>info</code>	Display system-wide information
<code>inspect</code>	Return low-level information on Docker objects
<code>kill</code>	Kill one or more running containers
<code>load</code>	Load an image from a tar archive or STDIN
<code>login</code>	Log in to a Docker registry
<code>logout</code>	Log out from a Docker registry
<code>logs</code>	Fetch the logs of a container
<code>pause</code>	Pause all processes within one or more containers
<code>port</code>	List port mappings or a specific mapping for the container

Comandos

<code>ps</code>	List containers
<code>pull</code>	Pull an image or a repository from a registry
<code>push</code>	Push an image or a repository to a registry
<code>rename</code>	Rename a container
<code>restart</code>	Restart one or more containers
<code>rm</code>	Remove one or more containers
<code>rmi</code>	Remove one or more images
<code>run</code>	Run a command in a new container
<code>save</code>	Save one or more images to a tar archive (streamed to STDOUT by default)
<code>search</code>	Search the Docker Hub for images
<code>start</code>	Start one or more stopped containers
<code>stats</code>	Display a live stream of container(s) resource usage statistics
<code>stop</code>	Stop one or more running containers
<code>tag</code>	Create a tag <code>TARGET_IMAGE</code> that refers to <code>SOURCE_IMAGE</code>
<code>top</code>	Display the running processes of a container
<code>unpause</code>	Unpause all processes within one or more containers
<code>update</code>	Update configuration of one or more containers
<code>version</code>	Show the Docker version information
<code>wait</code>	Block until one or more containers stop, then print their exit codes

Comandos

1. Executar exemplo
 - a. `docker run hello-world`
2. Listar imagens no repositório local
 - a. `docker images`
3. Buscar imagens no repositório online
 - a. `docker search ubuntu`
4. Baixar imagem do Ubuntu
 - a. `docker pull ubuntu`
5. Executar uma imagem do ubuntu
 - a. `docker run -it ubuntu`
 - b. `# To detach the tty without exiting the shell, use the escape sequence Ctrl-p + Ctrl-q`
6. Listar os containers (por padrão, só os em execução/pause; -a para todos)
 - a. `docker ps -a`
7. Reconectar ao shell do container
 - a. `docker attach <código/nome do container>`

Comandos

8. Executar uma imagem do ubuntu (informando nome e mapeando portas)
 - a. `docker run --name c1 -p 5000:5000 -it ubuntu`
 - b. `#` o comando `exit` mata o shell e para o container
9. Para um container
 - a. `docker stop <nome/id do container>`
10. Executar uma imagem já detached (opção -d)
 - a. `docker run --name c2 -it -d ubuntu /bin/bash`
11. Iniciar imagem parada
 - a. `docker start <nome/id do container>`
12. Pausar todos os processos de um container
 - a. `docker pause <nome/id do container>`
13. Despausar os processos do container
 - a. `docker unpause <nome/id do container>`
14. Remover um container
 - a. `docker rm <nome/id do container>`

Comandos

15. Remover todos os containers
 - a. `docker rm $(docker ps -qa)`
16. Salvar a imagem de um container no repositório local após fazer modificações
 - a. `docker commit -m "MSG" -a "AUTOR" <ContainerID> <Repo>/<NomeImagem>`
17. Logar no DockerHub
 - a. `docker login --username <username> --password <password>`
18. Marcar sua imagem com uma TAG
 - a. `docker tag <NomeDalmagem> <UsuárioDocker>/<NomeDalmagem>`
19. Enviar imagem para repositório remoto (DockerHub)
 - a. `docker push <UsuárioDocker>/<NomeDalmagem>`
20. Mostrar informações sobre o container
 - a. `docker inspect <containerID/name>`
21. Mostrar logs do container
 - a. `docker logs <containerID/name>`
22. Criar imagem com Dockerfile (o arquivo deve estar no diretório atual .)
 - a. `docker build -t <NomeDalmagem> .`

Comandos

23. Salvar container em um arquivo tar:
 - a. `docker save -o <NomeDoArquivo>.tar <NomeDaImagem>`
24. Carregar uma imagem a partir de um arquivo .tar
 - a. `docker load -i <NomeDoArquivo>.tar`
25. Renomear um container
 - a. `docker rename <AntigoNome> <NovoNome>`
26. Mostrando processos em execução em container:
 - a. `docker top <container id/name>`
27. Executa comando quando container já está em execução
 - a. `docker exec <container id/name> <comando>`

Personalizar e salvar imagem

- ❑ Executar imagem ubuntu
- ❑ Instalar Flask
- ❑ Salvar em nova imagem
- ❑ Enviar para DockerHub

```
# Instalar pacotes necessários
apt update -y
apt install -y python-pip python-dev
pip install flask
vi app.py
python app.py
```

```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

Fazer build de uma imagem com Dockerfile

- ❑ Executar imagem ubuntu
- ❑ Instalar Flask
- ❑ Salvar em nova imagem

```
# requirements.txt
Flask==1.0.3
```

```
# Dockerfile
FROM ubuntu:latest
MAINTAINER NOME_DO_AUTOR "EMAIL"
RUN apt-get update -y
RUN apt-get install -y python-pip python-dev
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENTRYPOINT ["python"]
CMD ["app.py"]
```

```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

Dockerfile

- ❑ FROM: Informa a partir de qual imagem será gerada a nova imagem, lembrando que em poucos casos (Veremos em posts futuros), uma imagem será gerada se um imagem base;
- ❑ MAINTAINER: Campo opcional, que informa o nome do mantenedor da nova imagem;
- ❑ RUN: Especifica que o argumento seguinte será executado, ou seja, realiza a execução de um comando;
- ❑ CMD: Define um comando a ser executado quando um container baseado nessa imagem for iniciado, esse parâmetro pode ser sobrescrito caso o container seja iniciado utilizando alguma informação de comando, como: `docker run -d imagem comando`, neste caso o CMD da imagem será sobrescrito pelo comando informado;
- ❑ LABEL: Adiciona metadados a uma imagem, informações adicionais que servirão para identificar versão, tipo de licença, ou host, lembrando que a cada nova instrução LABEL é criada uma nova layer, o Docker recomenda que você não use muitas LABEL. É possível realizar filtragens posteriormente utilizando essas LABEL.
- ❑ EXPOSE: Expõem uma ou mais portas, isso quer dizer que o container quando iniciado poderá ser acessível através dessas portas;

Dockerfile

- ❑ ENV: Instrução que cria e atribui um valor para uma variável dentro da imagem, isso é útil para realizar a instalação de alguma aplicação ou configurar um ambiente inteiro.
- ❑ ADD: Adiciona arquivos locais ou que estejam em uma url, para dentro da imagem.
- ❑ COPY: Copia arquivos ou diretórios locais para dentro da imagem.
- ❑ ENTRYPOINT: Informa qual comando será executado quando um container for iniciado utilizando esta imagem, diferentemente do CMD, o ENTRYPOINT não é sobrescrito, isso quer dizer que este comando será sempre executado.
- ❑ VOLUME: Mapeia um diretório do host para ser acessível pelo container;
- ❑ USER: Define com qual usuário serão executadas as instruções durante a geração da imagem;
- ❑ WORKDIR: Define qual será o diretório de trabalho (lugar onde serão copiados os arquivos, e criadas novas pastas);
- ❑ ONBUILD: Define algumas instruções que podem ser realizadas quando alguma determinada ação for executada, é basicamente como uma trigger.