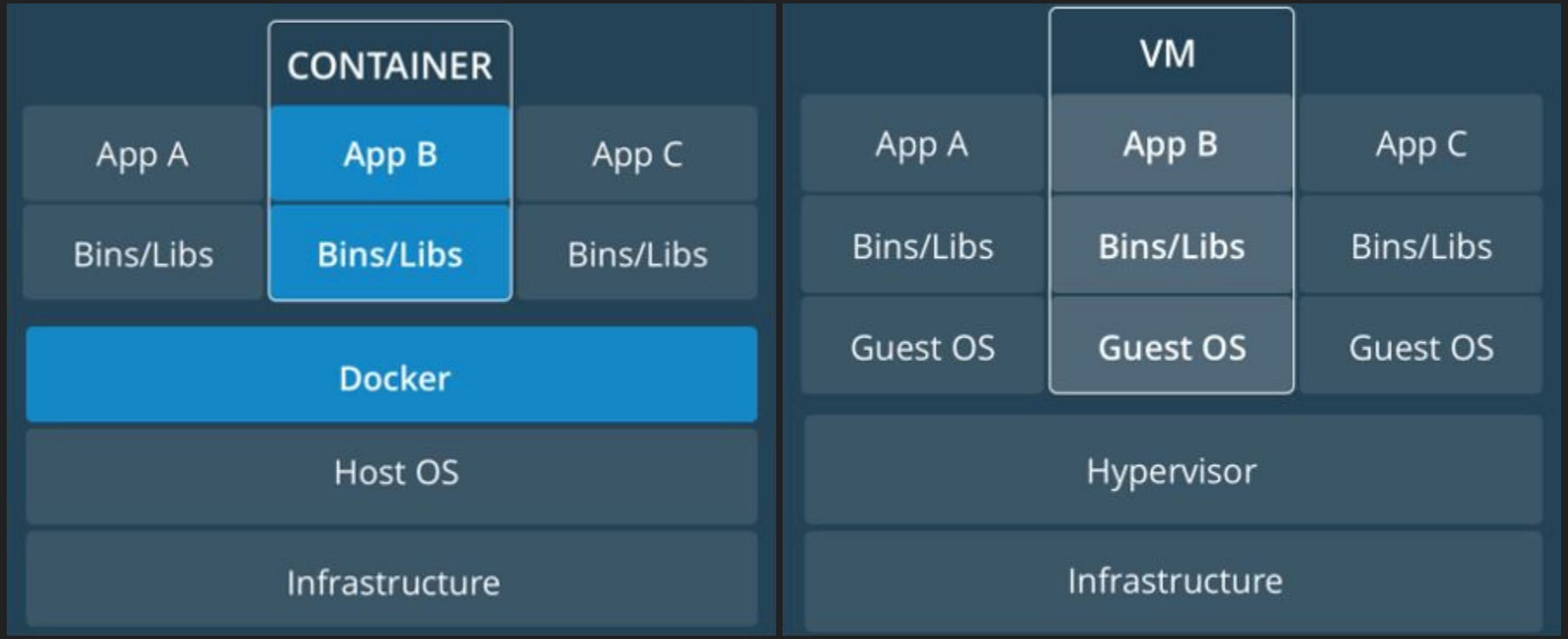
Desenvolvendo aplicações web com Flask e Docker

Conteúdo

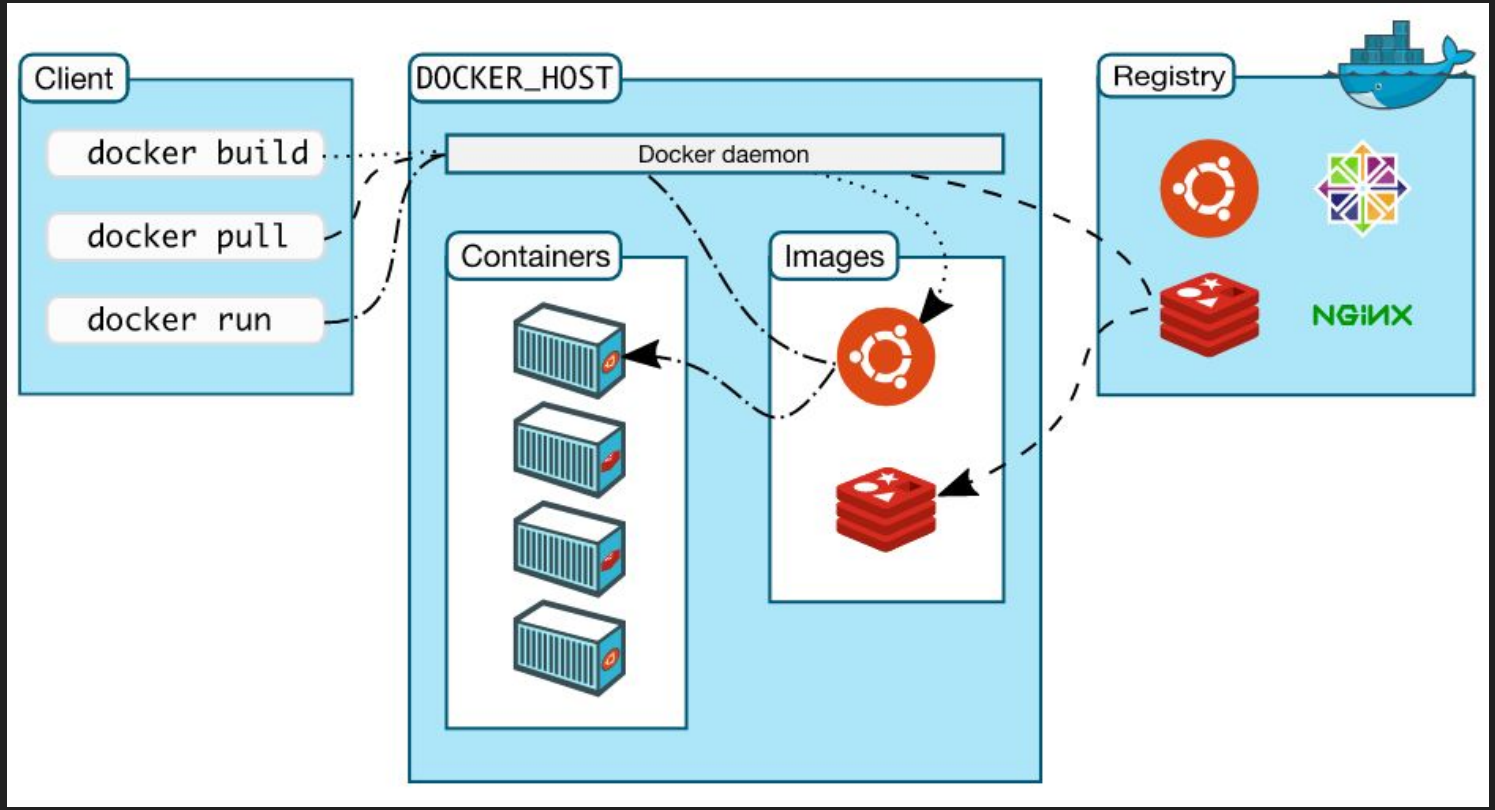
- O que é Docker e porque isso importa
- Estruturando a aplicação
- Build e Deploy

O que é Docker | VMs vs Docker



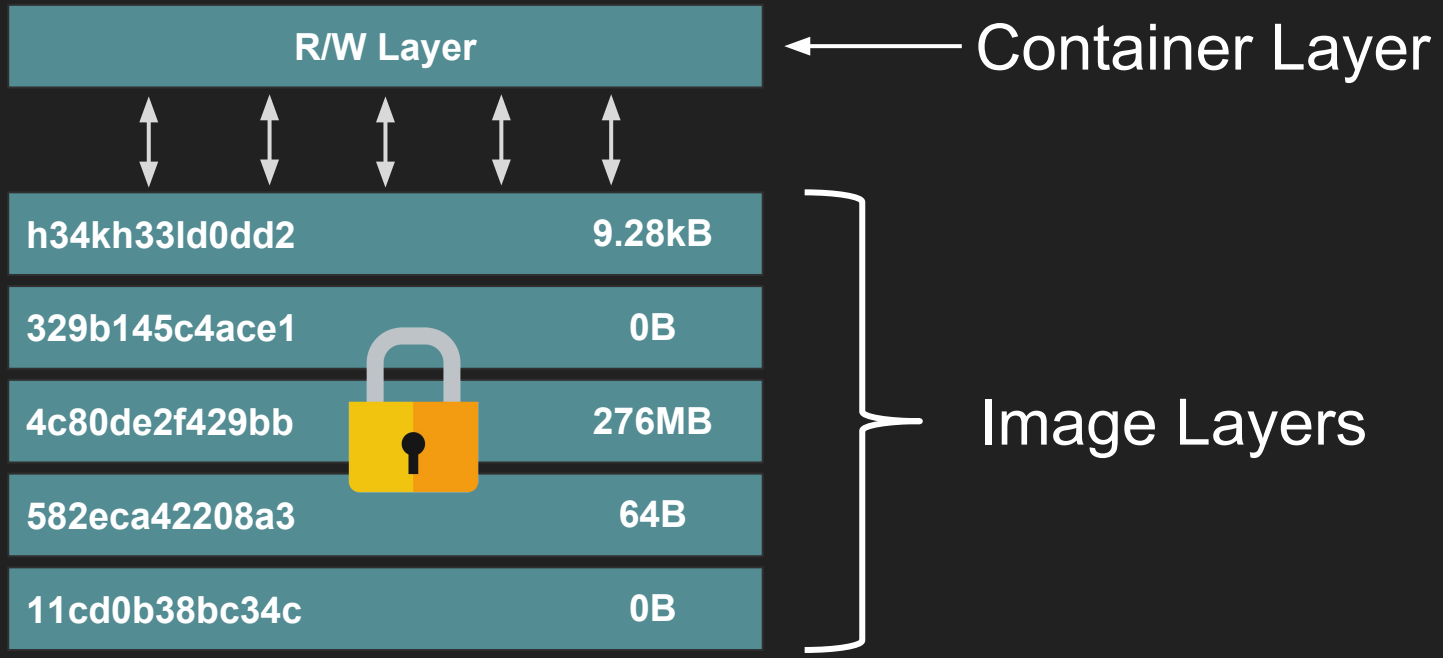
Fonte:https://docs.docker.com/get-started/#containers-and-virtual-machines

O que é Docker | Arquitetura

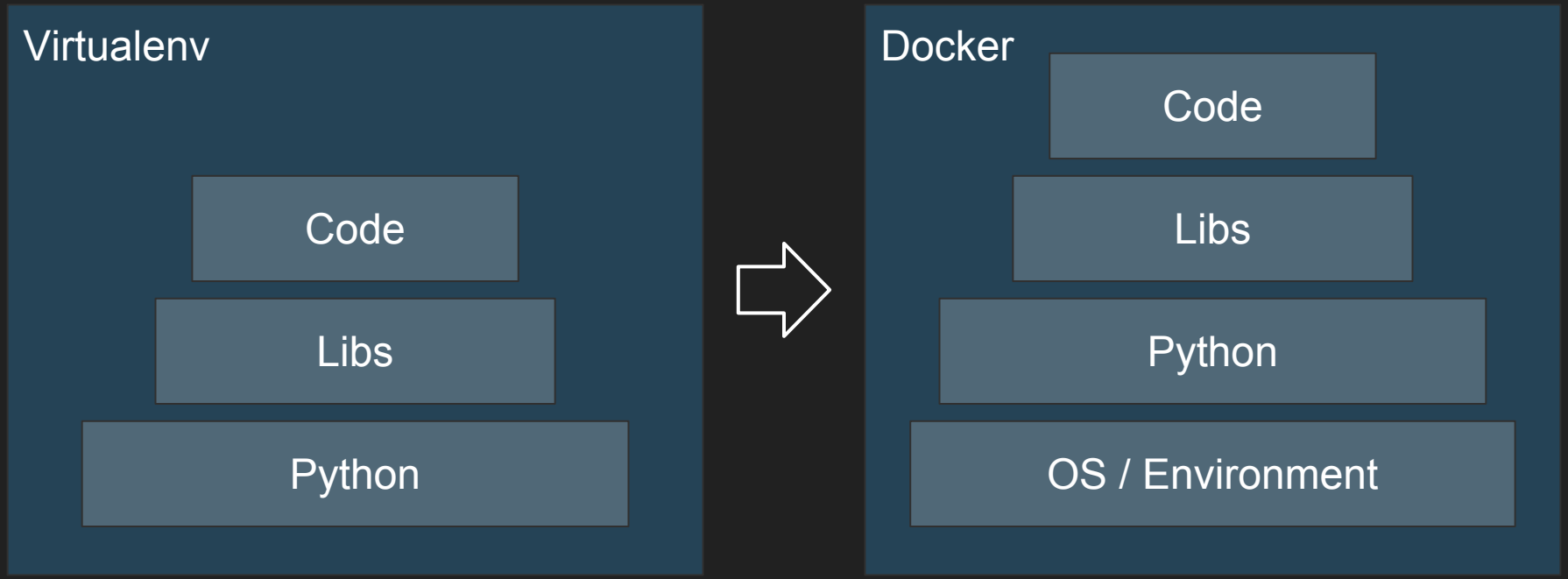


Fonte: https://docs.docker.com/engine/docker-overview/#docker-architecture

O que é Docker | Imagens e Containers



Porque isso importa



Estruturando a aplicação

12 factor app

1. Codebase
2. **Dependencies**
3. Config
4. **Backing Services**
5. Build, release, run
6. Processes
7. Port Binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. **Logs**
12. Admin Processes

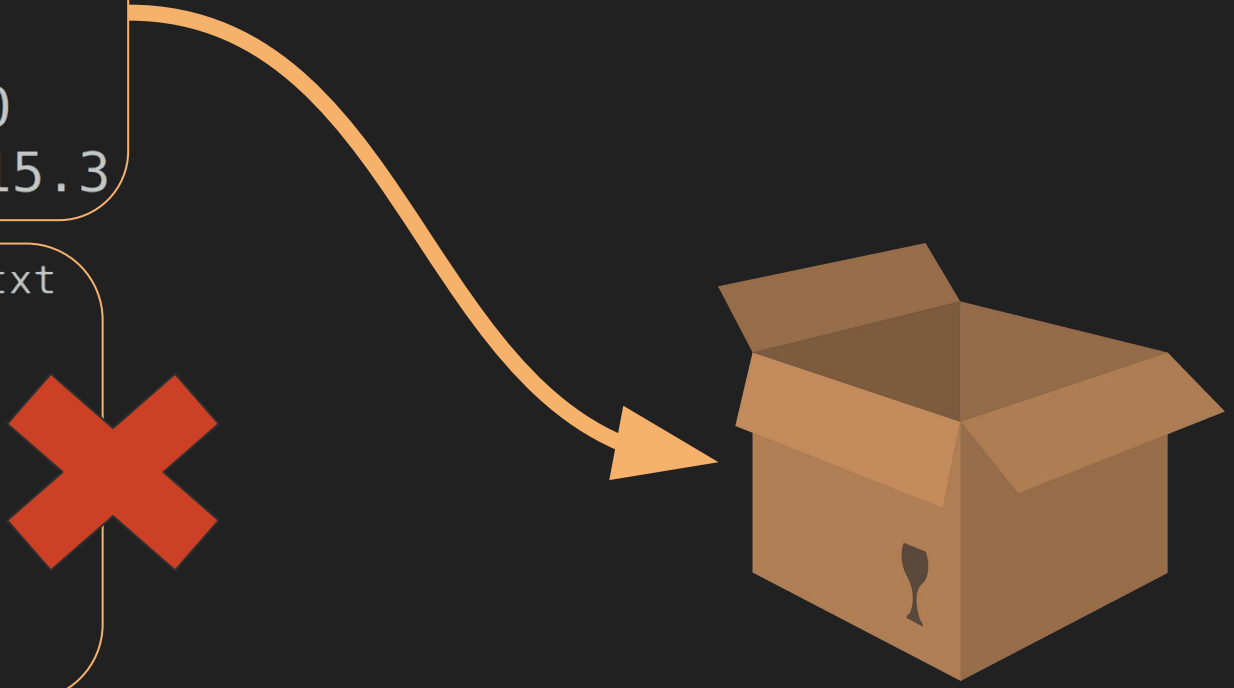
Isolamento de dependências

```
# requirements.txt

Flask==1.0.2
gevent==1.3.6
gunicorn==19.9.0
mongoengine==0.15.3
```

```
#dev_requirements.txt

coverage==4.5.1
ipython==6.5.0
mock==2.0.0
nose==1.3.7
pyflakes==1.6.0
radon==2.2.0
yapf==0.22.0
```



Utilizando logs

```
def create_logger():
    logger = logging.Logger('comics', level=logging.INFO)
    handler = logging.StreamHandler(stream=sys.stdout)
    format = f'[%(asctime)s][%(levelname)s] | %(message)s]'
    handler.setFormatter(logging.Formatter(format))
    logger.addHandler(handler)
    return logger
```

Bancos de dados como serviços de apoio

```
def connect_to_database(connect_fn, db_conf, utils=utils):
    attempts = 0
    logger = utils.create_logger()
    backoff = lambda t: sleep(2 ** t)

    while attempts < 5:
        try:
            logger.info(f'Connecting to database...')
            return connect_fn(**db_conf)

        except Exception as err:
            attempts += 1
            logger.error(
                f'Unable to connect to db ({attempts}/5): {str(err)}')

            backoff(attempts)

    logger.error(
        'Failed to connect to database: Max attempts exceeded. Exiting...')
    exit(1)
```

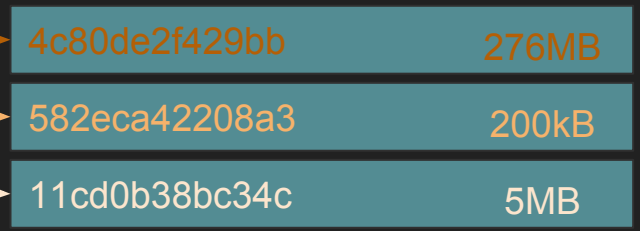
Build e Deploy

Criando um Dockerfile

```
FROM alpine:3.8 as base


COPY requirements.txt /


RUN apk add --update \
        linux-headers \
        gcc \
        g++ \
        build-base \
        python3-dev \
        shadow \
    && echo "America/Sao_Paulo" > /etc/timezone \
    && pip3 install -r requirements.txt
```



Criando uma imagem do Docker

```
$ docker build \
    -t felipemocruha/xkcd-comics:$(APP_VERSION) \
    -f deploy/Dockerfile .

---

IMAGE                          TAG      IMAGE ID       SIZE
felipemocruha/xkcd-comics      0.1.0    7e249865bad8   183MB
```

Executando um container

```
$ docker run -d -p 8080:8080 -e DEBUG=false \
    felipemocruha/xkcd-comics:0.1.0  sh start_server.sh

794cf569925527985f10e903854efed3c4a5071a930af60b27d2661dadf4cea5
```

Armazenando no Docker Hub

```
$ docker push felipemocruha/xkcd-comics:0.1.0

The push refers to repository [docker.io/felipemocruha/xkcd-comics]
d19818423a6b: Pushing [==========>                        ]  3.03MB/9.2MB
46e0b36cd38e: Pushed
18df0a81b016: Pushing [====================>              ]  11.03MB/183.1MB
c4c2dc901bd0: Pushed
73046094a9b8: Mounted from library/alpine
```

Dicas para produção

- ➔ Use load balancers e/ou proxies reversos
- ➔ Utilizar um orquestrador de containers (Kubernetes, Swarm, Mesos...)
- ➔ Seguir boas práticas na criação de imagens
- ➔ Criar volumes persistentes para os dados
- ➔ Criar imagens com responsabilidades pequenas

Onde encontrar mais

- Documentação oficial do Docker - https://docs.docker.com
- Creating Effective Docker Images - https://www.youtube.com/watch?v=vlS5EiapiII
- 12 factor app - https://12factor.net/
- Miguel Grinberg's blog - https://blog.miguelgrinberg.com/
- awesome-docker https://github.com/veggiemonk/awesome-docker
- https://github.com/humiaozuzu/awesome-flask

Obrigado

https://gitlab.com/felipemocruha/flask-conf-2018

felipemocruha@gmail.com