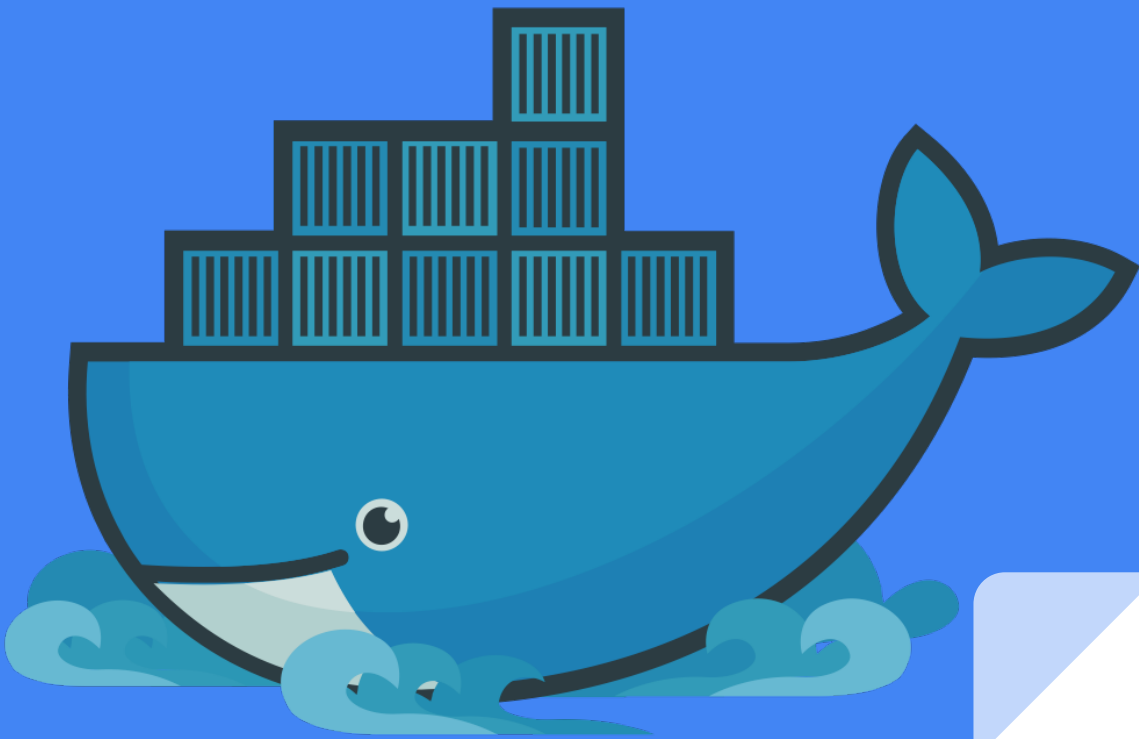
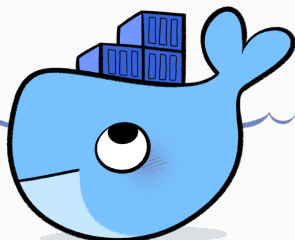
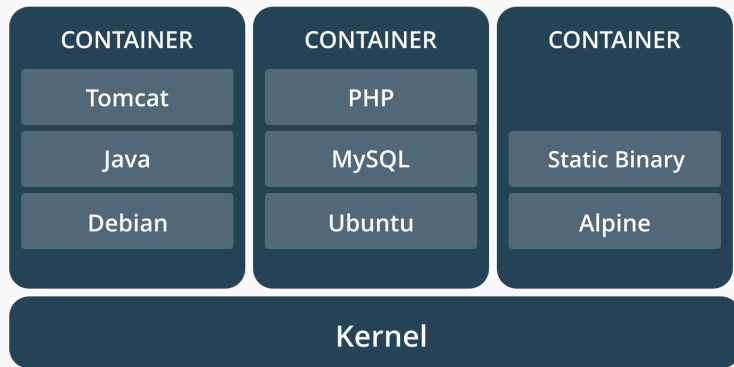


# Introdução ao Docker

Por Guilherme S. Salustiano



# A unidade mágica básica:



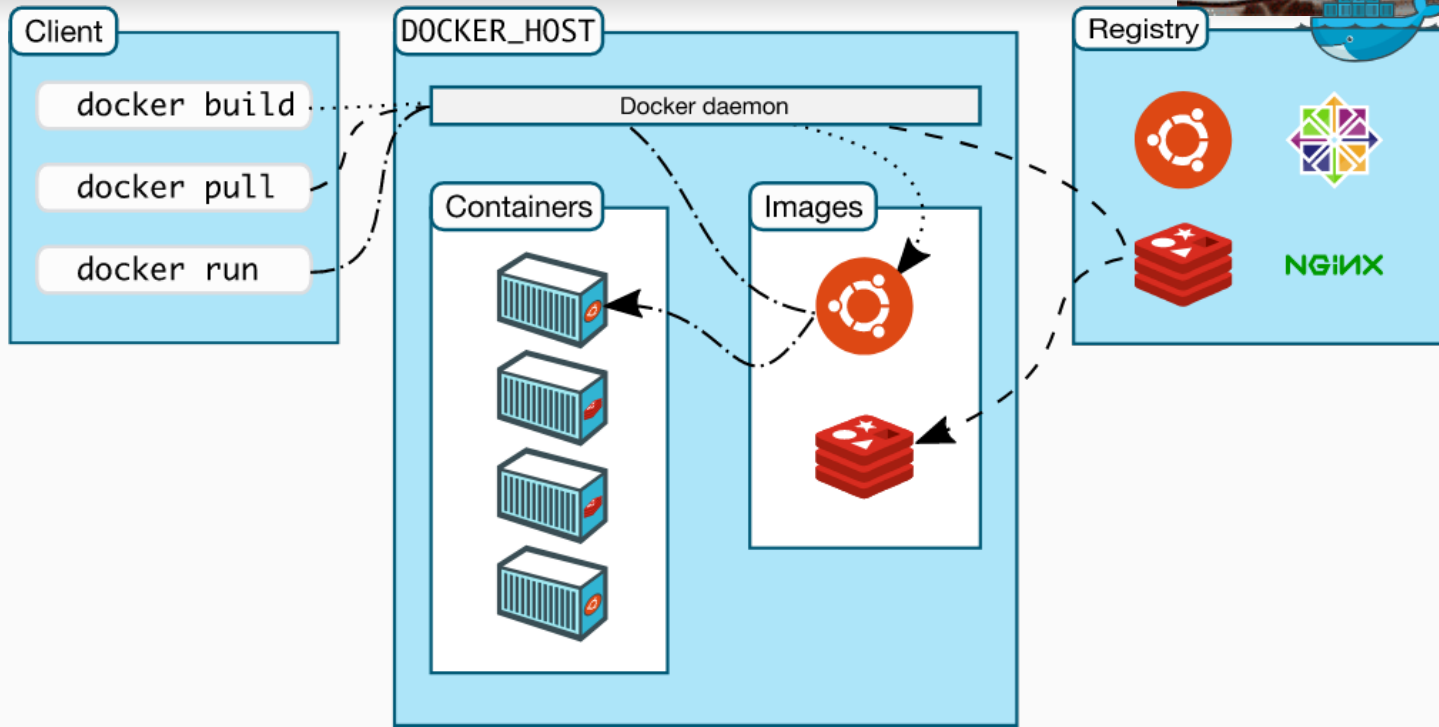
- Container empacota e executa um aplicativo em um ambiente isolado, seguro e independente
- Diversos container rodam juntos sem problemas
- Facilmente compartilhável por meio de imagens

# Por que docker?

- Gerenciar dependências
- Isolar ambientes
  - Escala
  - Segurança
- Leve
- Reprodutibilidade



# Workflow



Demo:

Rodando e acessando  
nosso primeiro container

# Descrição da demo

[https://hub.docker.com/\\_/julia/](https://hub.docker.com/_/julia/)


```
$ docker run -it --rm julia
```

```
$ docker run -it --rm ubuntu
```

```
$ docker run -it --rm --entrypoint bash julia:1.0
```

# Abrindo o docker para o mundo:

```
docker run -d \  
  --name turingdb \  
  -p 5432:5432 \  
  -e POSTGRES_PASSWORD=ateu \  
  -v $PWD/data:/var/lib/postgresql/data \  
  postgres
```



OPEN UP! OPEN UP!

Demo:  
Usando docker  
externamente



# Descrição da demo

[https://hub.docker.com/\\_/postgres/](https://hub.docker.com/_/postgres/)

```
$ docker run -d --name turingdb -p 5432:5432 -e POSTGRES_PASSWORD=ateu -v  
$PWD/data:/var/lib/postgresql/data postgres
```

```
$ docker ps (ou docker container ls)
```

```
$ docker logs turingdb
```

# Descrição da demo

\$ docker inspect turingdb # Mostra as informações do container

\$ docker exec -it turingdb bash # Entra no container

\$ docker exec -it turingdb psql -U postgres -W # ateu # \l

\$ docker container stop turingdb

\$ docker container rm turingdb

# Criando suas próprias imagens



[Referência oficial aqui!](#)



```
FROM python:3.8
```

```
WORKDIR /usr/src/app
```

```
RUN apt-get -y update
```

```
COPY requirements.txt requirements.txt
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 5000
```

```
COPY . .
```

```
CMD ["python", "./app.py"]
```

# Demo:

# Criando Imagem

# Descrição da demo

[código aqui](#)

```
$ bat Dockerfile
```

```
$ docker build -t hello-flask:1 .
```

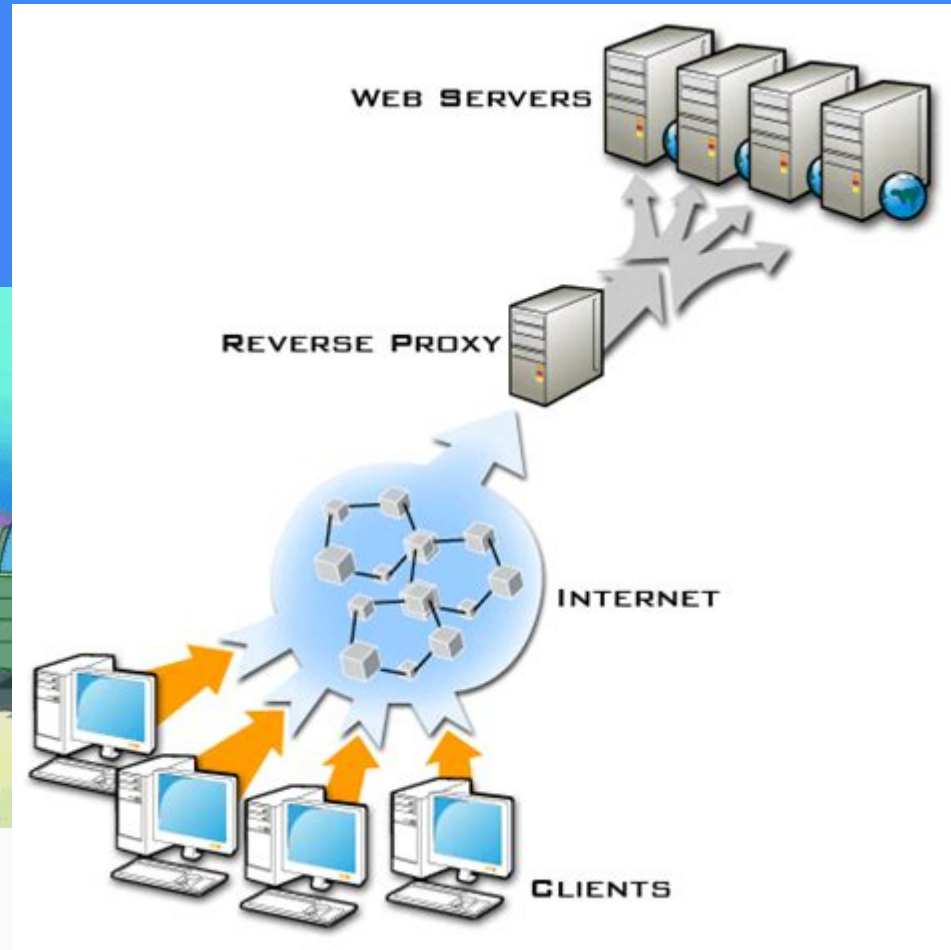
```
$ docker run --rm -p 80:5000 hello-flask:1
```

```
$ docker push 847479151133.dkr.ecr.us-east-1.amazonaws.com/hello-flask:1
```

# Obs: Proxy reverso



[nginx reverse proxy ref](#)



# Docker Compose

medici.tv



[Compose Docs](#)

```
# docker-compose.yml

version: "3.9" # optional since v1.27.0
services:
  app1:
    build: ./app1
  app2:
    build: ./app2
    environment:
      - NAME=Turing
  proxy:
    image: nginx
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    ports:
      - "8080:80"
    links:
      - app1
      - app2
```

# Demo:

# Docker Compose



# Descrição da demo

[código aqui](#)

```
$ bat app1/Dockerfile
```

```
$ bat app1/app.py
```

```
$ bat docker-compose.yalm
```

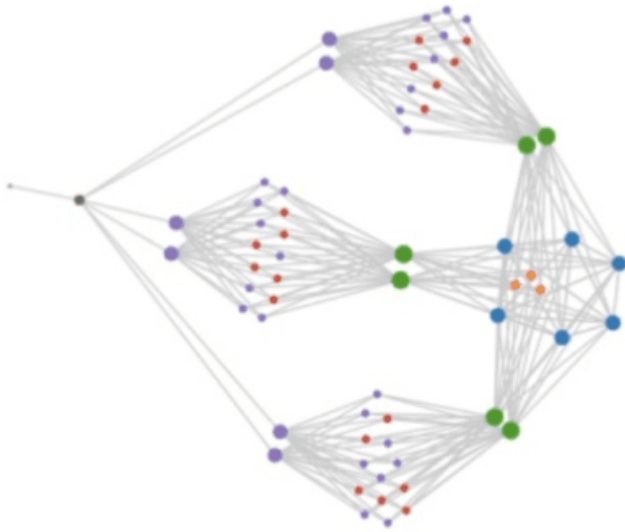
```
$ docker-compose up (-d) --build
```

```
$ docker rm $(docker ps -a -q) # Remove todos os container
```

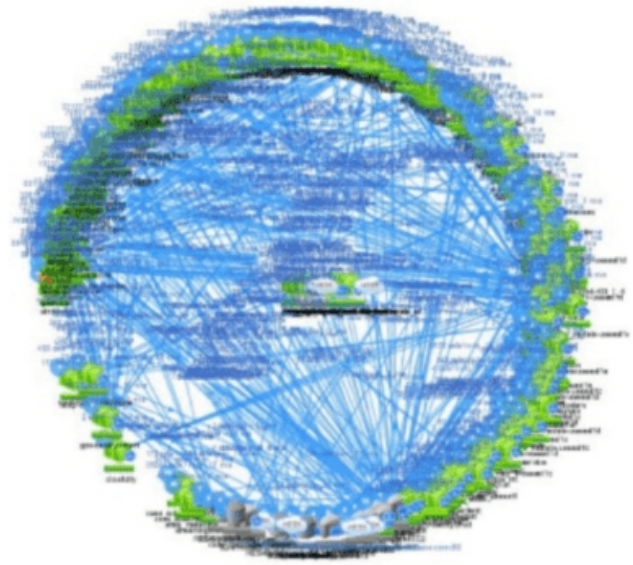
# Proximos topicos: Deploy

- Usar uma VM e gerenciar seu próprio docker ou docker-compose ( [AWS EC2](#)) (Não recomendado)
- Serverless ([AWS Lambda](#))
- Gerenciado de orquestração de contêineres ([AWS ECS](#))
- [Kubernetes](#) ([AWS EKS](#))

# Proximos topics: Arquitetura



**Simplified Architecture**



**Actual Architecture**

Fim,  
Obrigado!

