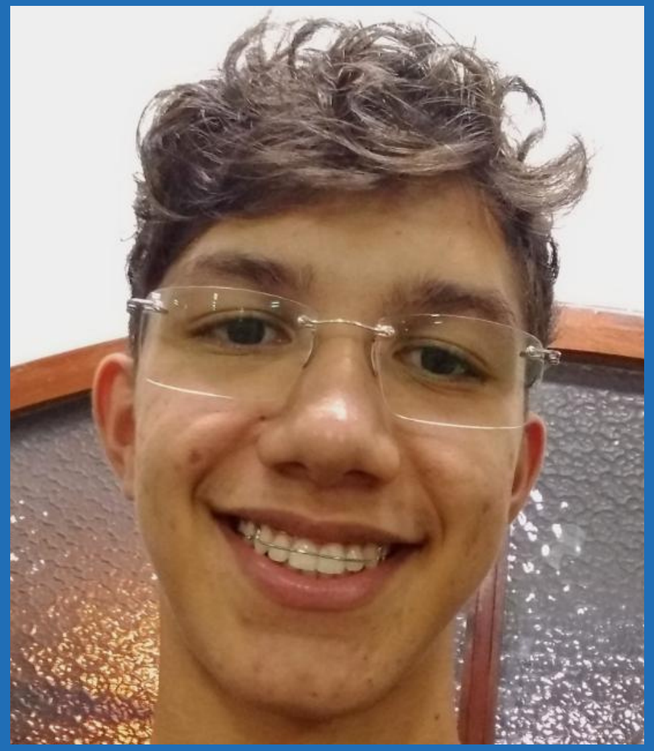
Infraestrutura Como um Código: Docker

DevOps | Container | Git | Automatização | Cloud

Um pouco sobre nós

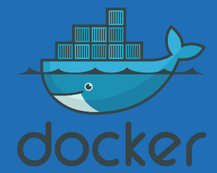


Graduando em Redes de Computadores pelo IFRN-CNAT, ex-Bolsista TAL, na disciplina de Programação para Redes e atualmente trabalha no GCTI, na função de Analista de Redes e Infraestrutura. Migrando para funções de Analista de Sistemas/DevOps. Possui alguns certificados na área de Redes e Linux.

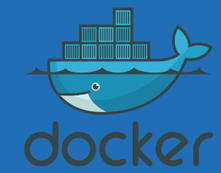
Graduando em Redes de Computadores pelo IFRN - CNAT, técnico em informática pelo IFRN - Campus Parnamirim. Profissional de infraestrutura de TI com mais de 2 anos de experiência na administração de ambientes Linux. Especialista em monitoração e observabilidade.

Um pouco sobre vocês

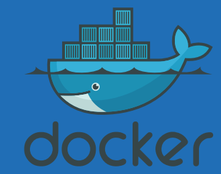
SUMÁRIO



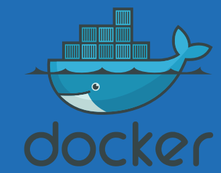
1 O que é Container Docker?



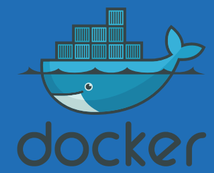
Software de Código aberto (Open-Source), que ajuda na implementação rápida de aplicações/serviços, baseados em Linux.



Para quê serve Container Docker?

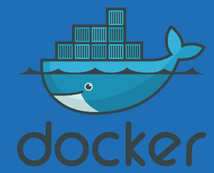


Serve para virtualizar as aplicações e serviços, de forma que também facilita a criação, implementação e administração dos mesmos, isoladamente.



Comparação entre servidores

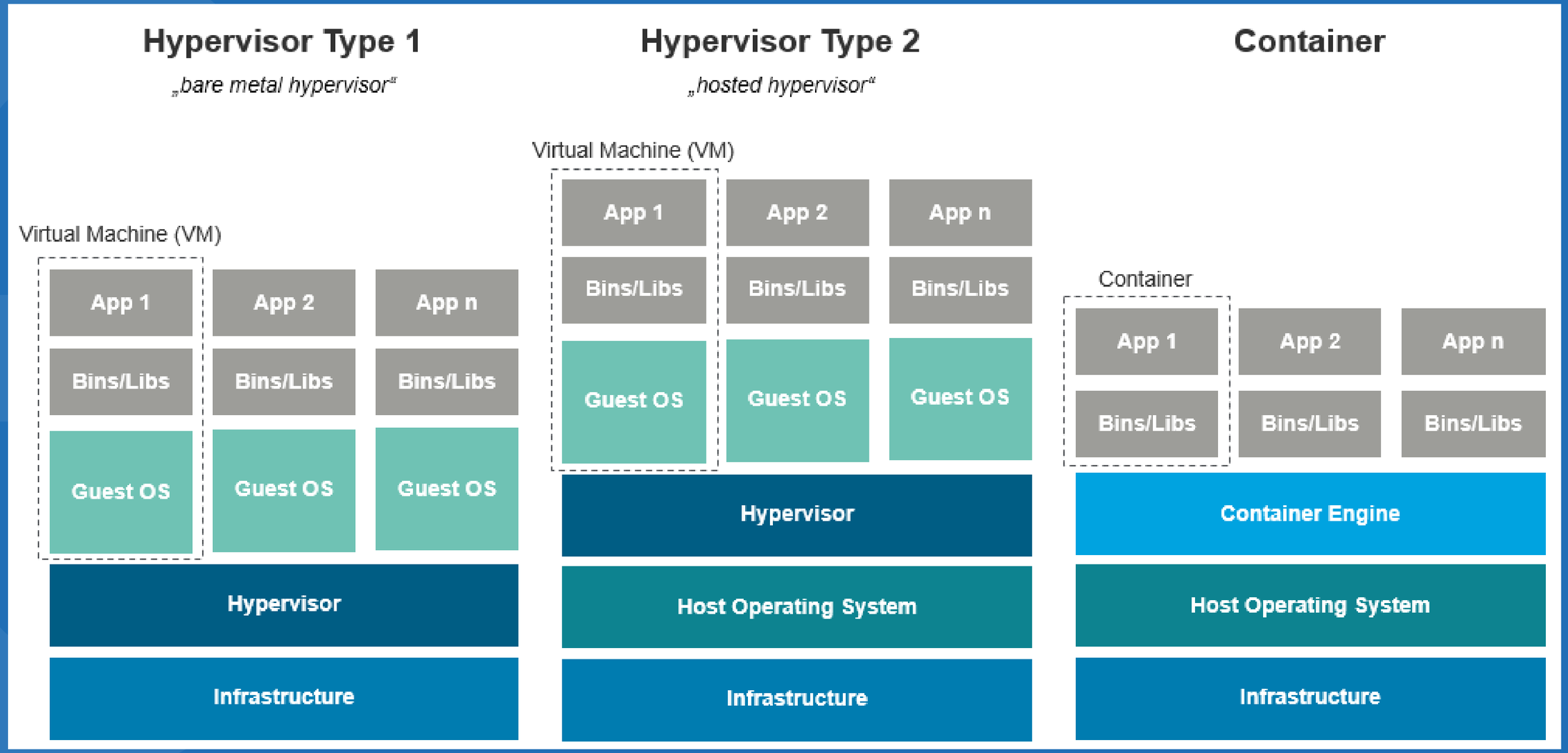
- Físicos
- Virtualizados
- Containers



Físicos

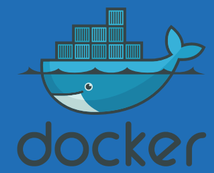
Virtualizados

Virtualizações



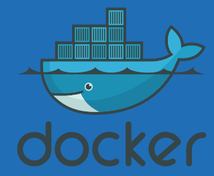
4

Dockerfile: Imagens & Comandos



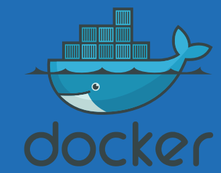
Exemplo de Dockerfile

```
1   FROM debian:buster
2
3   RUN apt-get update \
4       && apt-get install -y \
5       apache2 apache2-utils libapache2-mod-php \
6       php php-cli php-curl php-xml php-imap \
7       nano
8
9   RUN apt-get upgrade -y
10
11  COPY process_02.sh .
12  COPY process_03.sh .
13  RUN ./process_03.sh
```

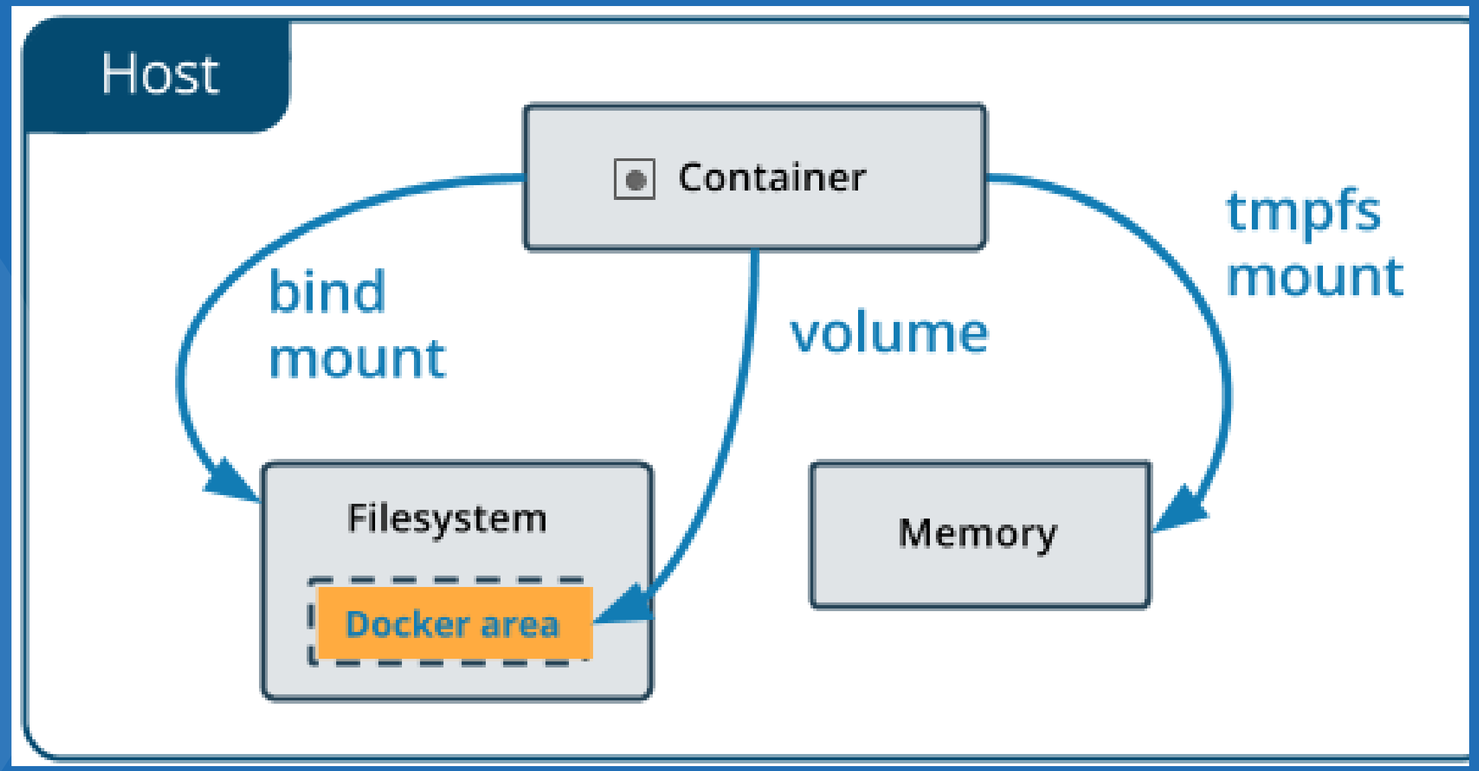


5

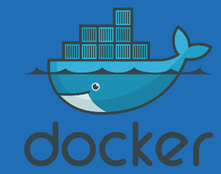
Volumes



Persistência de dados das aplicações, para backups e redeploy.



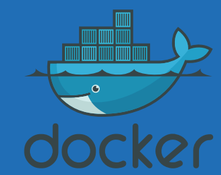
Comandos Docker



Principais comandos argumentos

- docker **pull**
- docker **build**
- docker **run**
- docker **ps**
- docker **images**
- docker **volume**
- docker **rm**
- docker **rmi**
- docker **compose**

- **-p:** Mapeamento de Portas
- **-v:** Mapeamento de Volumes
- **-d:** Execução em Background
- **-t:** Nome para Imagem
- **-f:** Forçar Execução
- **-a:** Mostrar Todos
- **--help:** Ajuda para Todos os Comandos



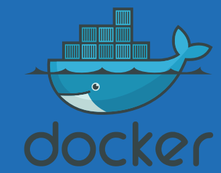
Práticas iniciais

docker

Subir WebServer 1.0

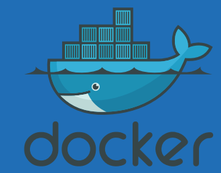
- docker run -p 8080:80 --name web nginx
- Acesse localhost:8080

- docker rm -f web
- docker rmi -f nginx



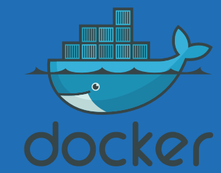
Subir WebServer 2.0

1. git clone https://github.com/denilsonbonatti/linux-site-dio.git
2. cd linux-site-dio
3. Dockerfile:
    a. FROM nginx:latest
    b. COPY . /usr/share/nginx/html
4. docker build -t web-img .
5. docker run -p 8080:80 --name web nginx



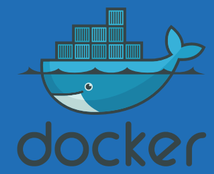
8

Docker Compose

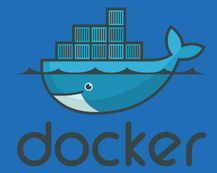


Exemplo de Docker Compose

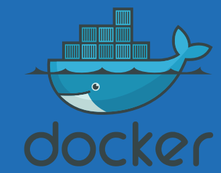
```yaml
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_DEBUG: True
  redis:
    image: "redis:alpine"
```

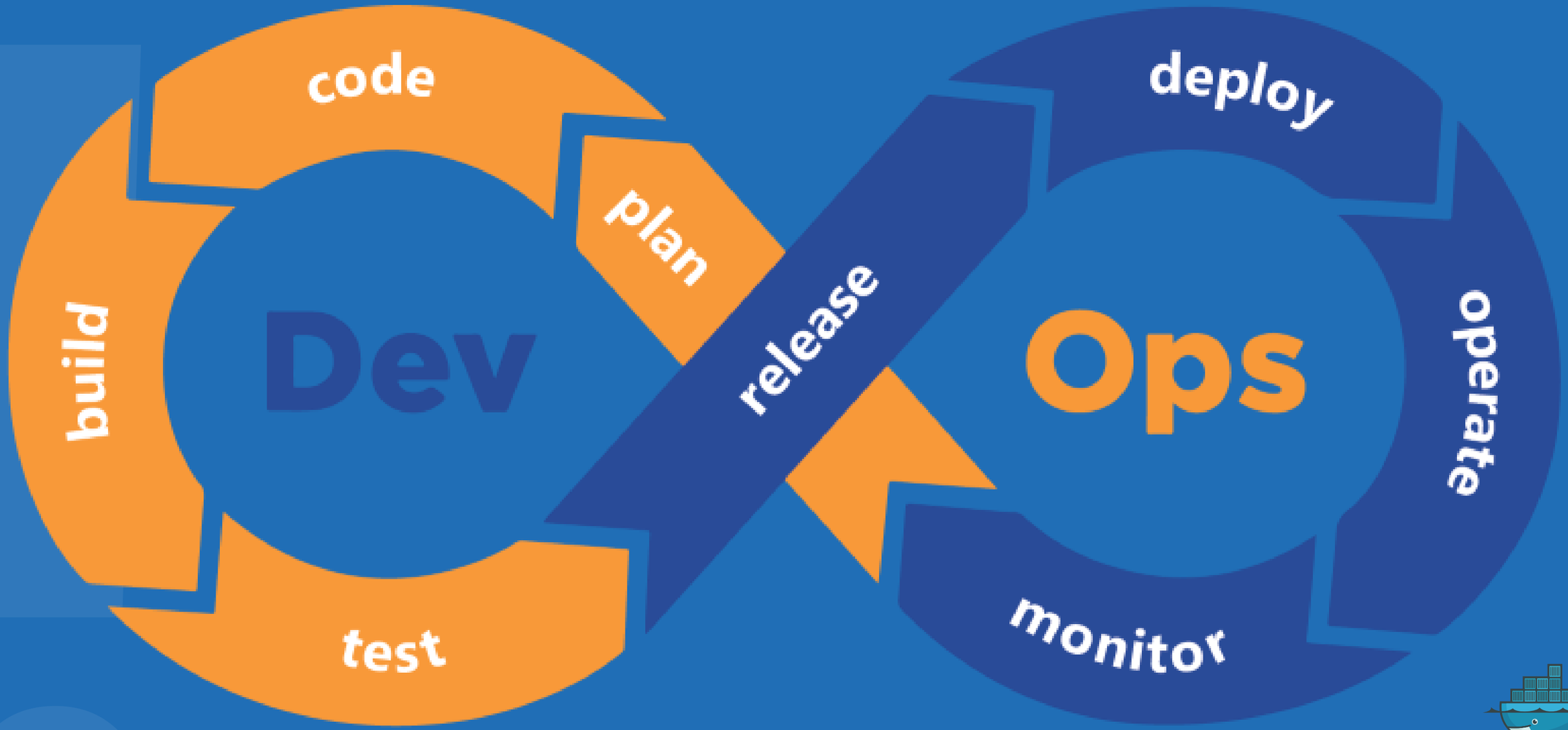


Prática final



Bônus

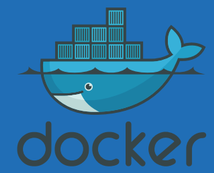



code
plan
build
Dev
test
release
deploy
operate
Ops
monitor

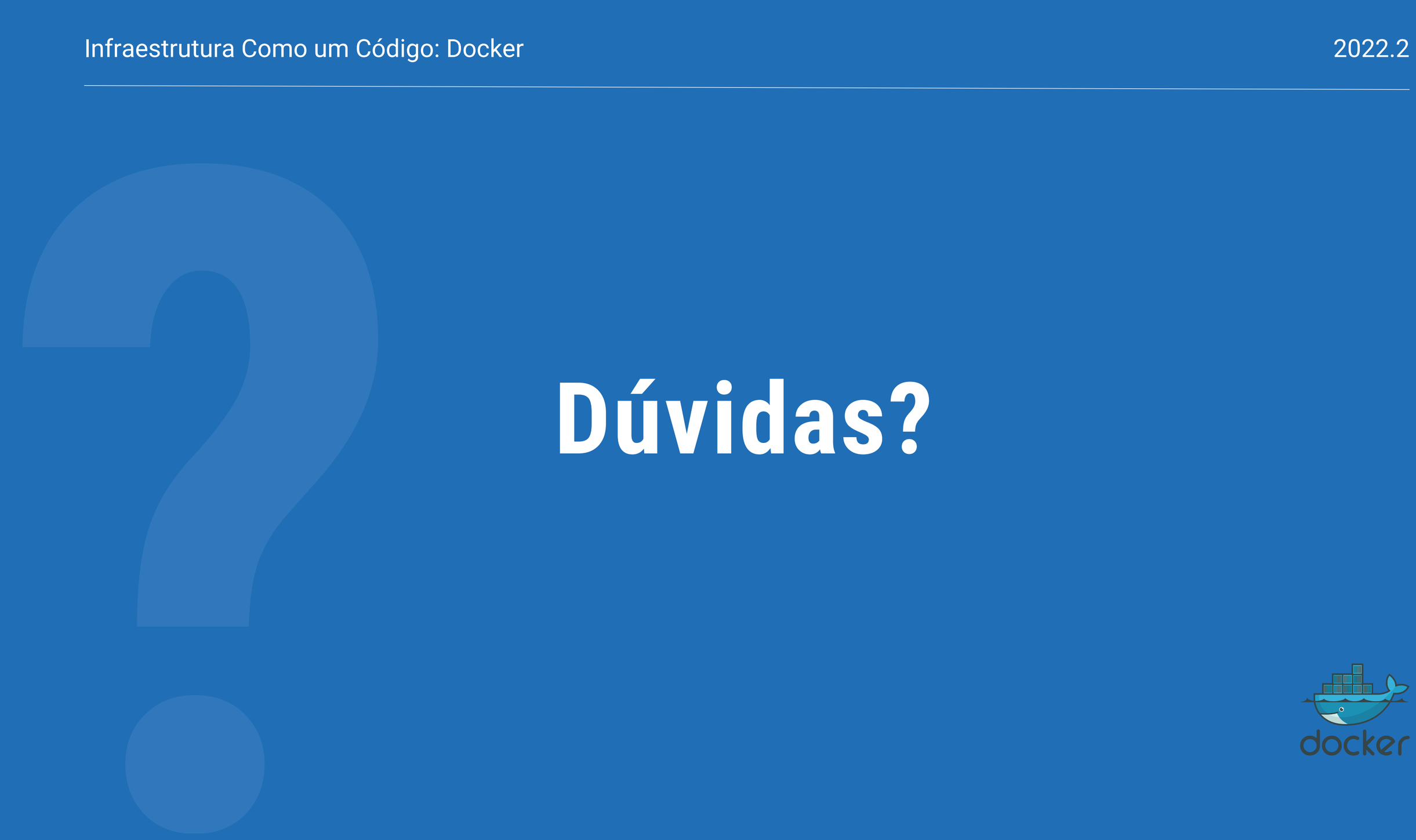
Documentação

Pesquisa

GitHub & GitLab



Dúvidas?



Obrigado pela atenção!