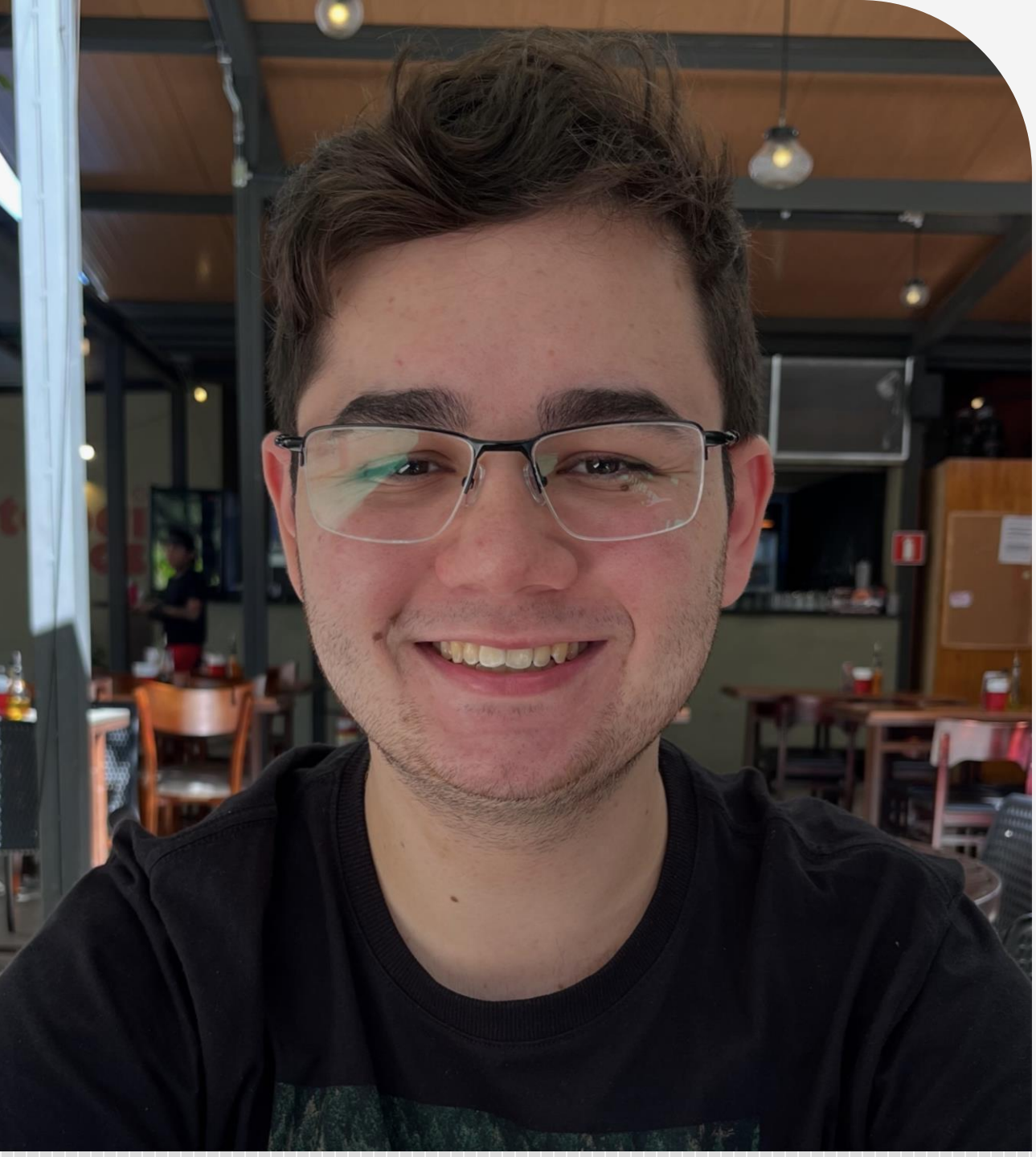
Unlocking
digital value.
Together.

dti

Introdução ao Docker

Da teoria à prática com Docker e Docker Compose



Artur Bomtempo

Desenvolvedor de Software

- Crafter desde setembro/2025
- Enhenharia de Software | PUC
- Criador de conteúdo digital

Sumário

Problema que o Docker resolve

- Diferenças de ambiente (SO, versões, dependências)
- Setup demorado e frágil
- “Funciona na minha máquina”

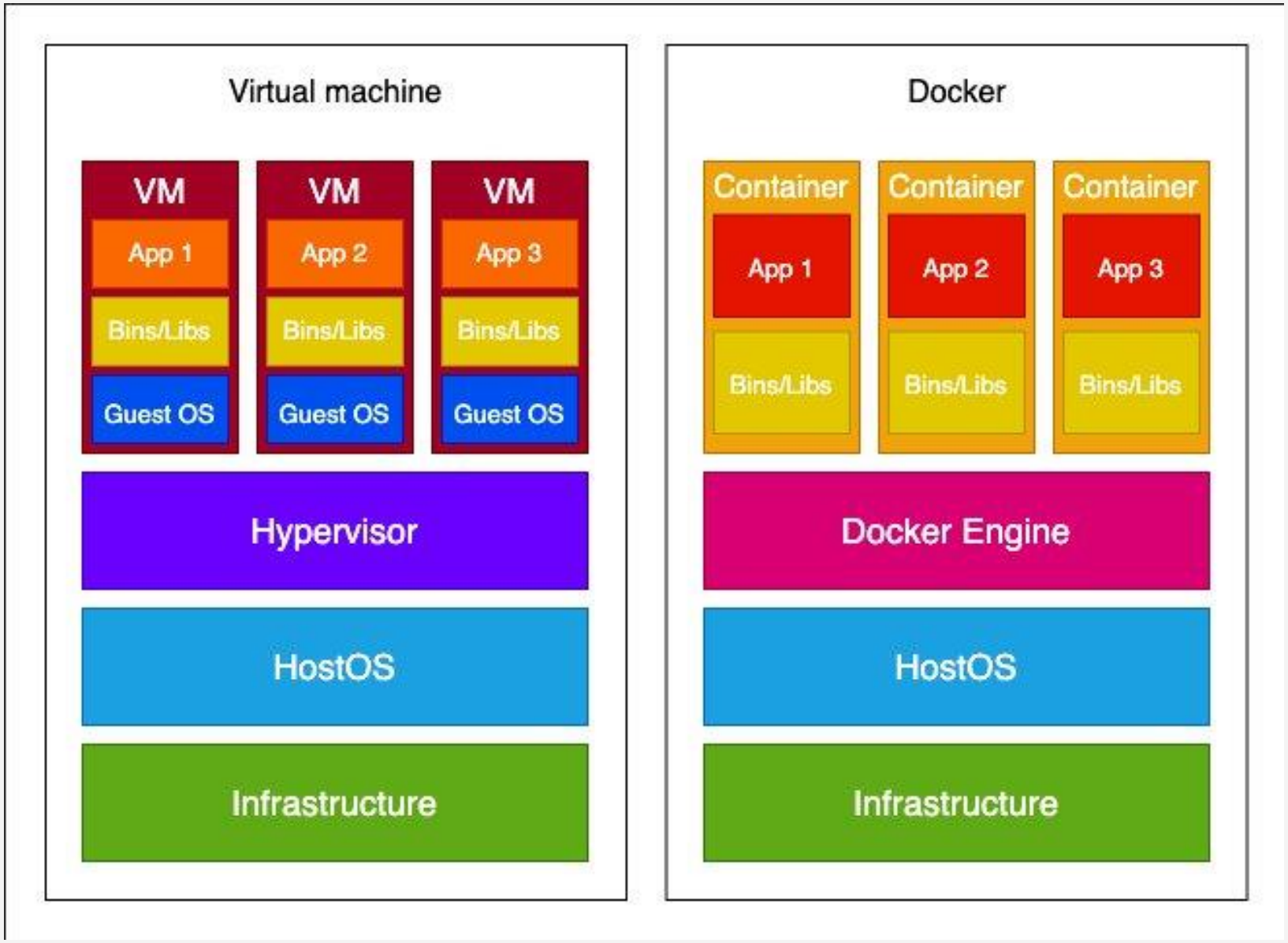
O que são containers?

Unidades executáveis de software que empacotam o código da aplicação juntamente com suas bibliotecas e dependências.

https://www.ibm.com/br-pt/think/topics/containers

Container vs Máquina Virtual

- Container é um processo isolado
- Compartilha o kernel do host
- Não possui um SO próprio


Virtual machine
VM
App 1
Bins/Libs
Guest OS
VM
App 2
Bins/Libs
Guest OS
VM
App 3
Bins/Libs
Guest OS
Hypervisor
HostOS
Infrastructure
Docker
Container
App 1
Bins/Libs
Container
App 2
Bins/Libs
Container
App 3
Bins/Libs
Docker Engine
HostOS
Infrastructure

Por que containers são mais leves?

- Compartilham o kernel
- Não precisam de Sistema Operacional completo
- Inicializam em segundos

Vantagens dos containers

- Leves
- Portáveis
- Ideais para arquiteturas modernas
- Menor tempo para release

Arquitetura e Workflow

Imagem Docker

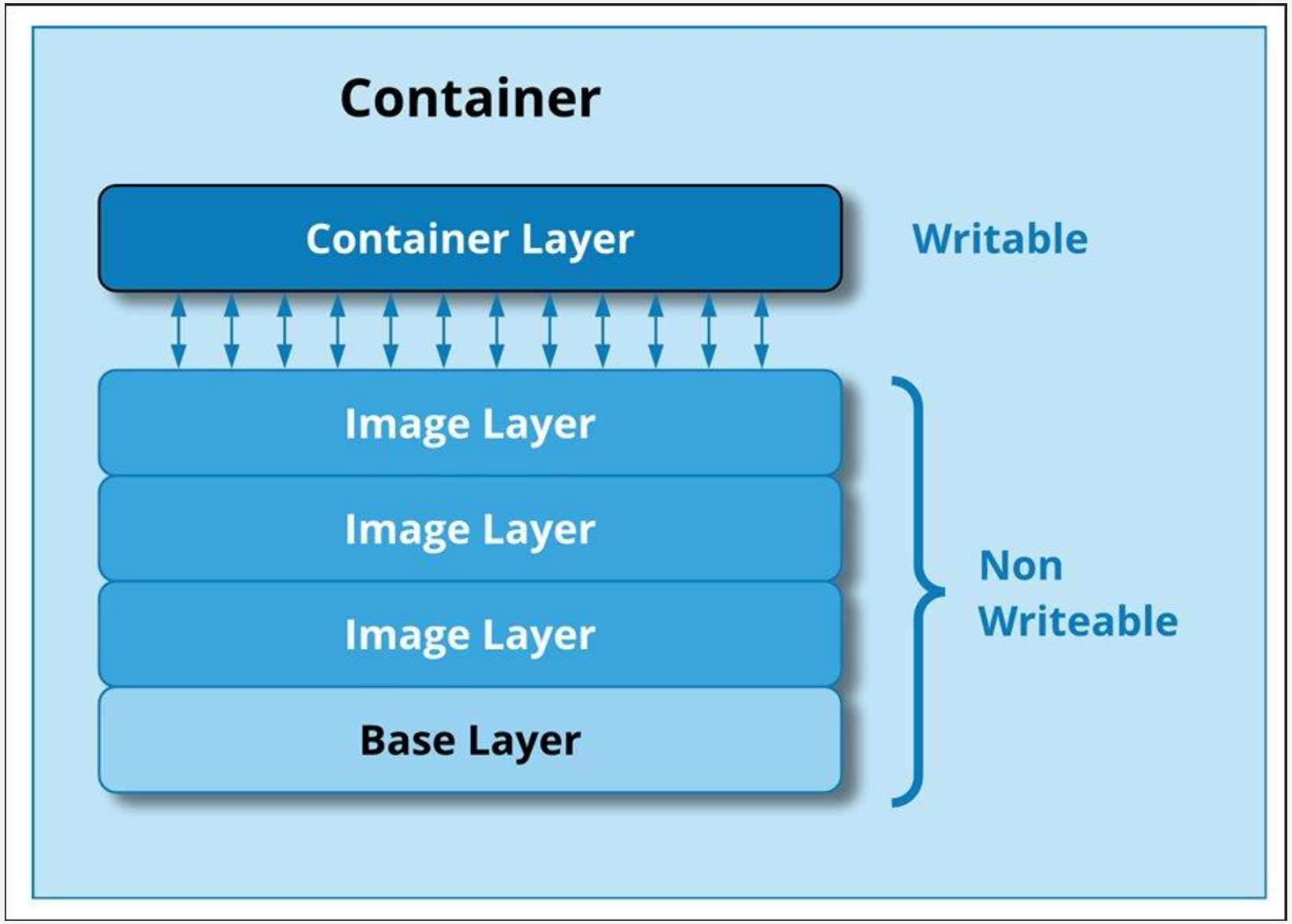
- Template *read-only*
- Snapshot da aplicação
- Reutilizável
- Base para criação de containers

Imagem vs Container

- Imagem → definição
- Container → execução
- Um container nasce de uma imagem

Camadas da imagem

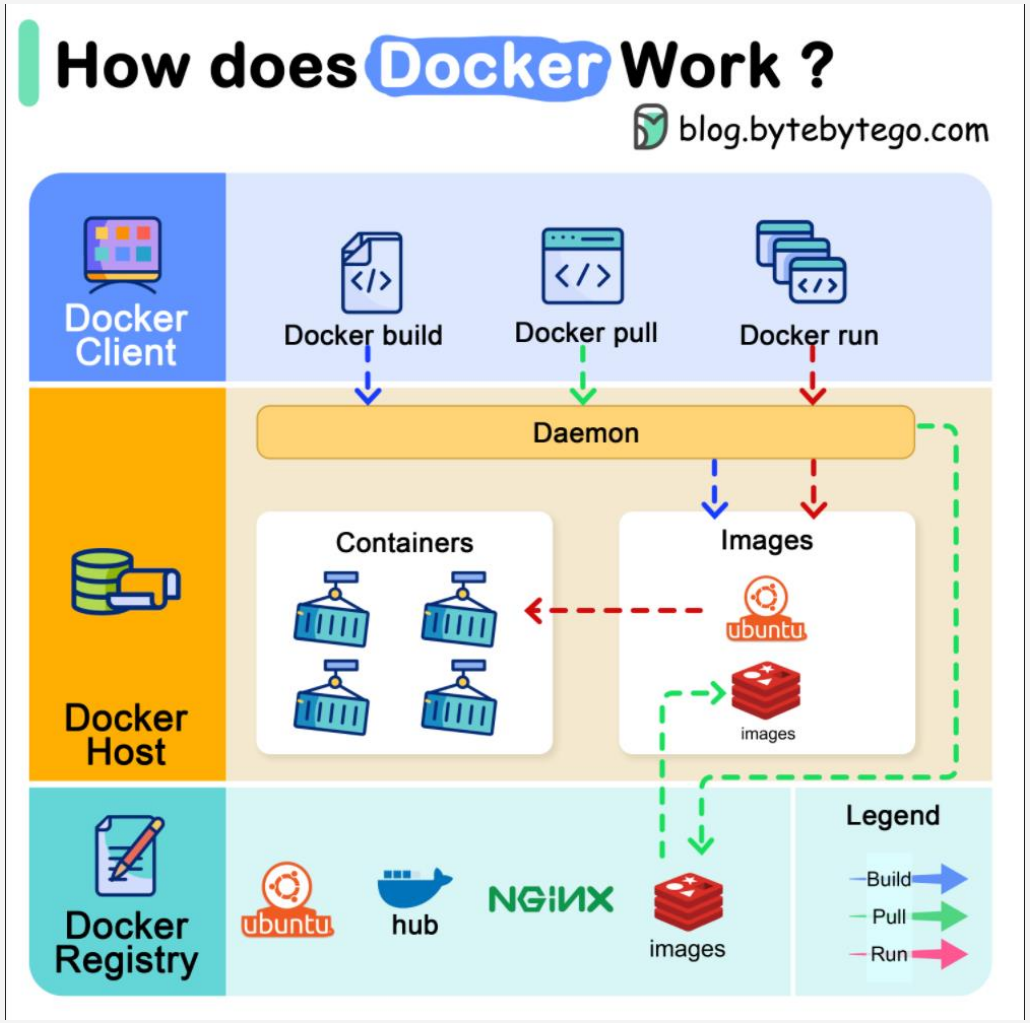
- Cada instrução do Dockerfile gera uma camada
- Camadas são reutilizadas (cache)
- Apenas a camada do container é gravável


Container
Container Layer
Writable
Image Layer
Image Layer
Image Layer
Base Layer
Non
Writeable

Dockerfile

Arquivo declarativo que define, passo a passo, como uma imagem Docker é construída, especificando a imagem base, dependências, arquivos, configurações e o comando de execução do container.

Command	Purpose
FROM	Specifies the base image
WORKDIR	Sets the working directory
COPY	Copies files/directories into the container
RUN	Executes commands during image building
EXPOSE	Exposes ports
CMD	Specifies the default command to run
ENV	Sets environment variables
ARG	Defines build-time variables
LABEL	Adds metadata to the image
VOLUME	Creates a mount point as a volume in the container


How does Docker Work ?
blog.bytebytego.com
Docker Client
Docker build
Docker pull
Docker run
Daemon
Containers
Images
ubuntu
images
Docker Host
Docker Registry
ubuntu
hub
NGINX
images
Legend
Build
Pull
Run

Containers são descartáveis

- Reiniciou → perdeu
- Recriou → perdeu
- Problema para bancos e uploads

Volumes

O que são volumes?

Volumes resolvem persistência

- Dados fora do container
- Sobrevivem ao ciclo de vida
- Gerenciados pelo Docker

Volume vs Bind Mount

- **Volume:** gerenciado pelo Docker
- **Bind mount:** pasta do host

Limitações do Docker "puro"

Docker funciona bem, mas...

Quando o projeto cresce

- Vários containers
- Dependências entre serviços
- Ordem de inicialização
- Redes e volumes manuais

Trabalho manual

- Docker build
- Docker run
- Docker network
- Docker volume
- Múltiplos comandos

Docker Compose

Orquestração local de containers

- Define múltiplos serviços
- Um arquivo YAML
- Infra como código

O que o Compose faz automaticamente

- Cria rede
- Cria volumes
- Resolve DNS entre containers
- Gerencia ordem de subida

Dockerfile vs Docker Compose

Papéis diferentes

- **Dockerfile** → constrói imagens
- **Docker Compose** → orquestra containers

Antes vs Depois

Sem Compose:

- Muitos comandos
- Configuração espalhada

Com Compose:

- Um arquivo
- Um comando: *docker compose up*

Unlocking
digital value.
Together.

Obrigado.

Artur Bomtempo

artur.bomtempo@dtidigital.com.br