



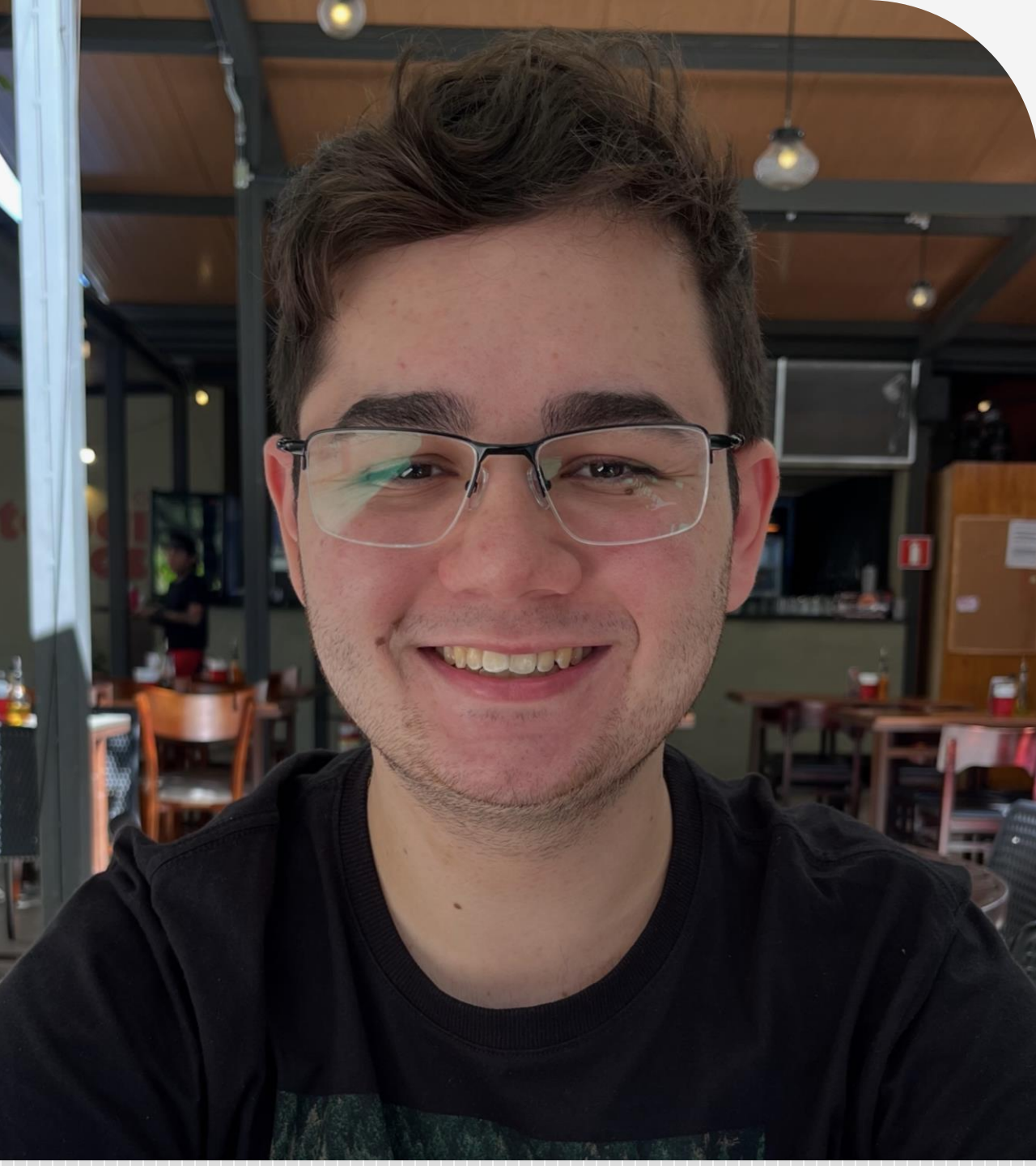
Unlocking  
digital value.  
**Together.**



# Introdução ao Docker

Da teoria à prática com Docker e  
Docker Compose





# Artur Bomtempo

**Desenvolvedor de Software**

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

# sumário

1. Container
2. Arquitetura e Workflow
3. Volumes
4. Docker Compose

# 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



Hypervisor

HostOS

Infrastructure

### Docker



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
- Portáteis
- Ideais para arquiteturas modernas
- Menor tempo para release

# Arquitetura e Workflow

# Imagem Docker

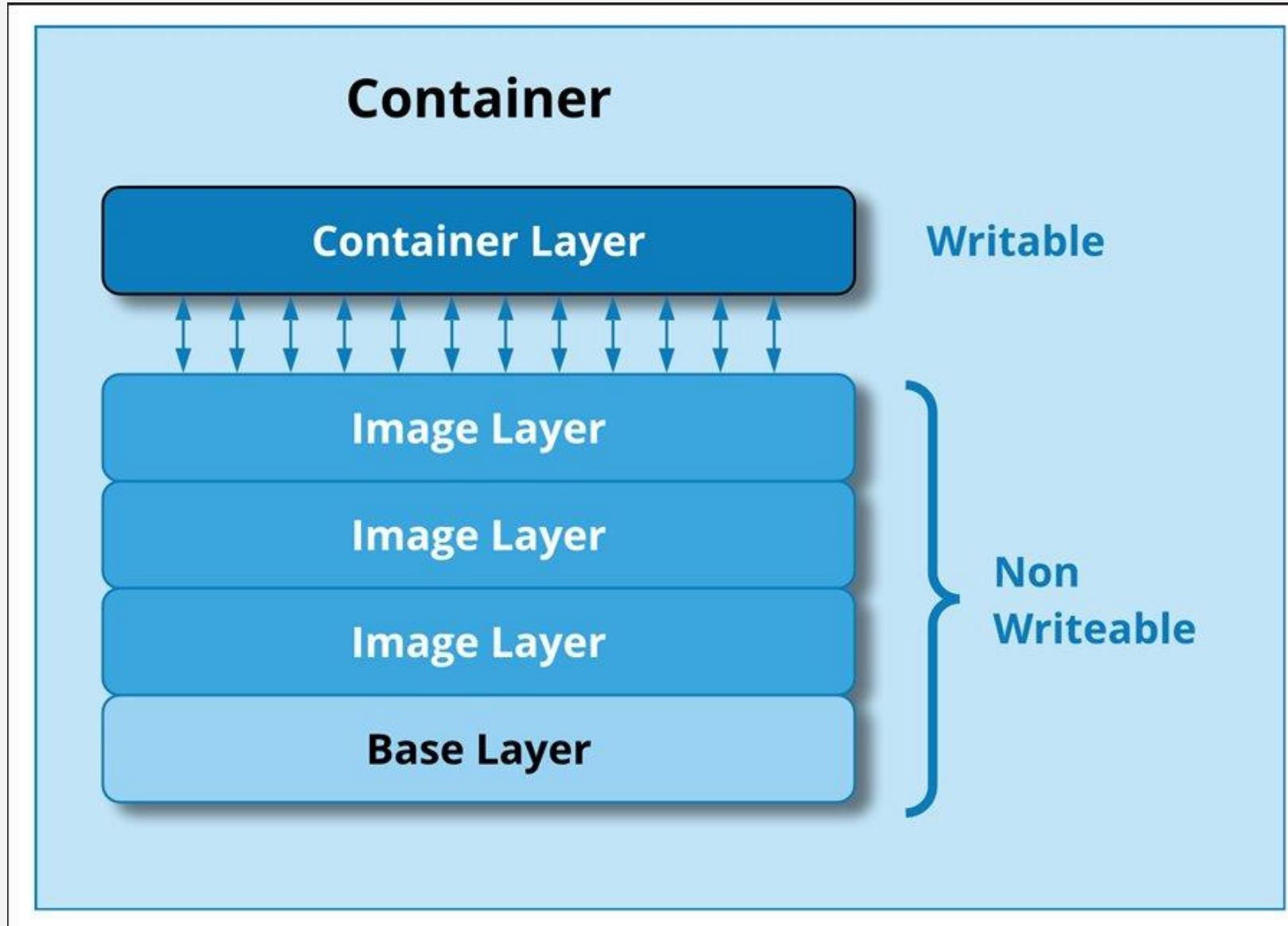
- Template *read-only*
- Snapshot da aplicação
- Reutilizável
- Base para criação de containers

# 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





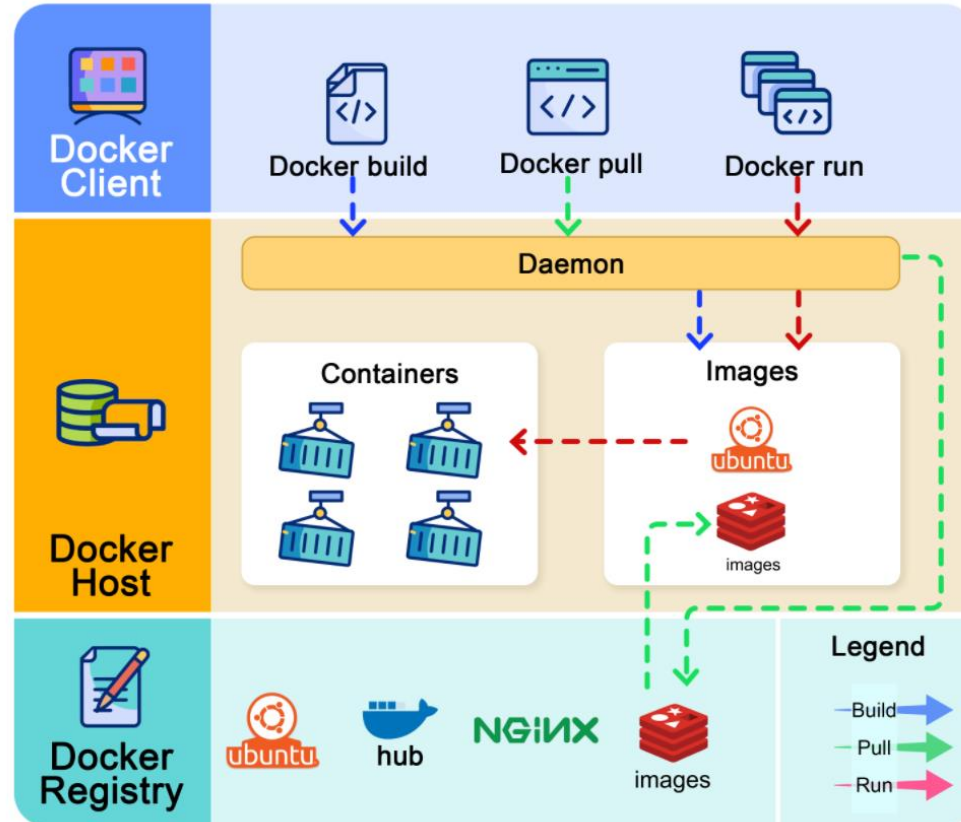
# Dockerfile

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

# How does Docker Work ?

 [blog.bytebytego.com](https://blog.bytebytego.com)



# 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](mailto:artur.bomtempo@dtidigital.com.br)