Acabando com o problema:

"Mas na minha máquina funciona!"

Tutorial de Docker

O que é o Docker?

O **Docker** é uma plataforma que permite empacotar, distribuir e executar aplicações em **ambientes isolados chamados containers**. Esses containers garantem que a aplicação execute **sempre da mesma forma**, independentemente do sistema operacional, configurações locais ou dependências.

O QUE É CONTAINER?

> Máquinas virtuais (VMs) e containers servem para isolar aplicações, mas fazem isso de forma diferente.

> Uma **VM** virtualiza todo o hardware e executa um **sistema operacional completo** para cada instância. Isso consome mais memória e tempo para iniciar, pois cada VM precisa carregar seu próprio kernel (ou sistema operacional). Existem mais tipos de virtualização, mas esse não é o foco aqui!

> Um **container** compartilha o **mesmo sistema operacional do hospedeiro**, isolando apenas o que a aplicação precisa (bibliotecas, dependências e processos). Assim, é **muito mais leve, rápido e portátil**.

Em resumo, você não precisa instalar nada em seu sistema operacional hospedeiro. Apenas no seu container.

- Além disso, você pode migrar entre quaisquer sistemas operacionais e hardware sem instalar quase nada! - Você precisa apenas de alguma plataforma gerenciadora de containers, como Docker ou LXC (nativo em OS Linux)

Conceitos Fundamentais

Conceito	Descrição
Imagem (Image)	Um modelo de ambiente pronto (ex: `jdk`, `mysql:8`, `nginx`, `apache2`) — é como uma classe.
Container	Uma instância em execução de uma imagem — é como um objeto.
Dockerfile	Um script que descreve como construir uma imagem personalizada.
Volume	Um diretório persistente para salvar dados mesmo após o container ser removido.

Conceito	Descrição
Network	Uma rede interna que conecta vários containers (ex: web + banco).
Docker Compose	Uma ferramenta para orquestrar vários containers ao mesmo tempo.

Instalação

Windows

1. Baixe o **Docker Desktop**: https://www.docker.com/products/docker-desktop
2. Durante a instalação:
    - Ative o **WSL 2 Backend** (necessário).
    - Reinicie o computador se solicitado.

> O WSL (Windows Subsystem for Linux) é um recurso do Windows que permite executar o Linux dentro do próprio Windows, sem precisar de máquina virtual.

Porque precisamos de WSL no Windows?

- O Docker foi criado originalmente para Linux, pois os containers dependem de recursos do kernel Linux para isolar processos.
- O problema é que o Windows não tem kernel Linux nativo. Então, o que o Docker faz?
- Ele usa o WSL 2 (Windows Subsystem for Linux 2) como "mini máquina Linux integrada" dentro do Windows.

1. Teste no **PowerShell ou CMD**:

```
docker -v
docker compose version
```

Linux

- No Linux é bem simples!

```
sudo apt update
sudo apt install docker.io docker-compose -y
sudo systemctl enable docker --now
```

Teste:

```
docker -v
docker compose version
```

---

macOS

Baixe o **Docker Desktop for Mac** no mesmo link oficial: https://www.docker.com/products/docker-desktop

Teste:

No terminal:

```
docker -v
docker compose version
```

> Não precisa de camada de compatibilidade com Linux no macOS, porque o sistema já é baseado em Unix (assim como o Linux).

Usando o Docker

Executar um container simples

```
docker run hello-world
```

Esse comando baixa uma imagem de teste e verifica se o Docker está funcionando corretamente.

Executar um container interativo

```
docker run -it ubuntu bash
```

- `-it` → modo interativo
- `python3` → imagem base
- `bash` → comando inicial dentro do container

Listar containers e imagens

```
docker ps            # containers em execução
docker ps -a         # todos containers (ativos e parados)
docker images        # imagens baixadas
```

Parar e remover containers

```
docker stop <id ou nome>
docker rm <id ou nome>
docker rmi <id ou nome da imagem>
```

Dockerfile — exemplo básico

```
# Usa uma imagem base node (Node.js é um ambiente de execução JavaScript que p
FROM node:20

# Define o diretório de trabalho
WORKDIR /app

# Copia os arquivos do projeto
COPY . .

# Instala dependências
RUN npm install

# Define o comando de execução
CMD ["npm", "start"]
```

Criando e executando a imagem:

```
docker build -t minha_app .
docker run -p 3000:3000 minha_app
```

Docker Compose — exemplo básico

Crie um arquivo `docker-compose.yml`:

```yaml
version: '3.8'

services:
  web:
    image: nginx # Servidor WEB (Node, Python ...)
    ports:
      - "8080:80"

  db:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: root
```

E execute tudo com:

```
docker compose up
```

Para rodexecutar em segundo plano:

```
docker compose up -d
```

Comandos mais usados

Comando	O que faz
`docker ps`	Lista containers ativos
`docker ps -a`	Lista todos os containers
`docker images`	Lista imagens
`docker exec -it <container> bash`	Acessa o terminal do container
`docker logs <container>`	Mostra logs
`docker stop <container>`	Para um container
`docker rm <container>`	Remove um container
`docker rmi <imagem>`	Remove uma imagem
`docker system prune`	Limpa tudo o que não está em uso
`docker compose up`	Sobe todos os serviços definidos

Comando	O que faz
`docker compose down`	Para e remove containers criados pelo Compose

Dicas rápidas

- Containers são **descartáveis** — sempre salve dados em **volumes**.
- Use `latest` com cuidado — fixe versões (`mysql:8.0` em vez de `mysql:latest`).
- Combine Docker + `.env` para configurar variáveis de ambiente.
- Sempre inclua um `.dockerignore` (como o `.gitignore`) para evitar copiar arquivos desnecessários.

Exemplo de `.dockerignore`:

```
node_modules
.env
.git
```

Fluxo comum de trabalho

1. Criar um `Dockerfile`
2. Construir a imagem:

   ```
   docker build -t nome_da_imagem .
   ```

3. Executar o container:

   ```
   docker run -p 8080:80 nome_da_imagem
   ```

4. Ver logs e interagir:

   ```
   docker logs -f nome_da_imagem
   docker exec -it nome_da_imagem bash
   ```

Resumo Rápido

Tarefa	Comando
Testar Docker	`docker run hello-world`
Ver containers ativos	`docker ps`
Parar container	`docker stop nome`
Remover container	`docker rm nome`
Criar imagem	`docker build -t nome .`
Executar container	`docker run -p 8080:80 nome`
Subir projeto com Compose	`docker compose up -d`
Parar tudo	`docker compose down`