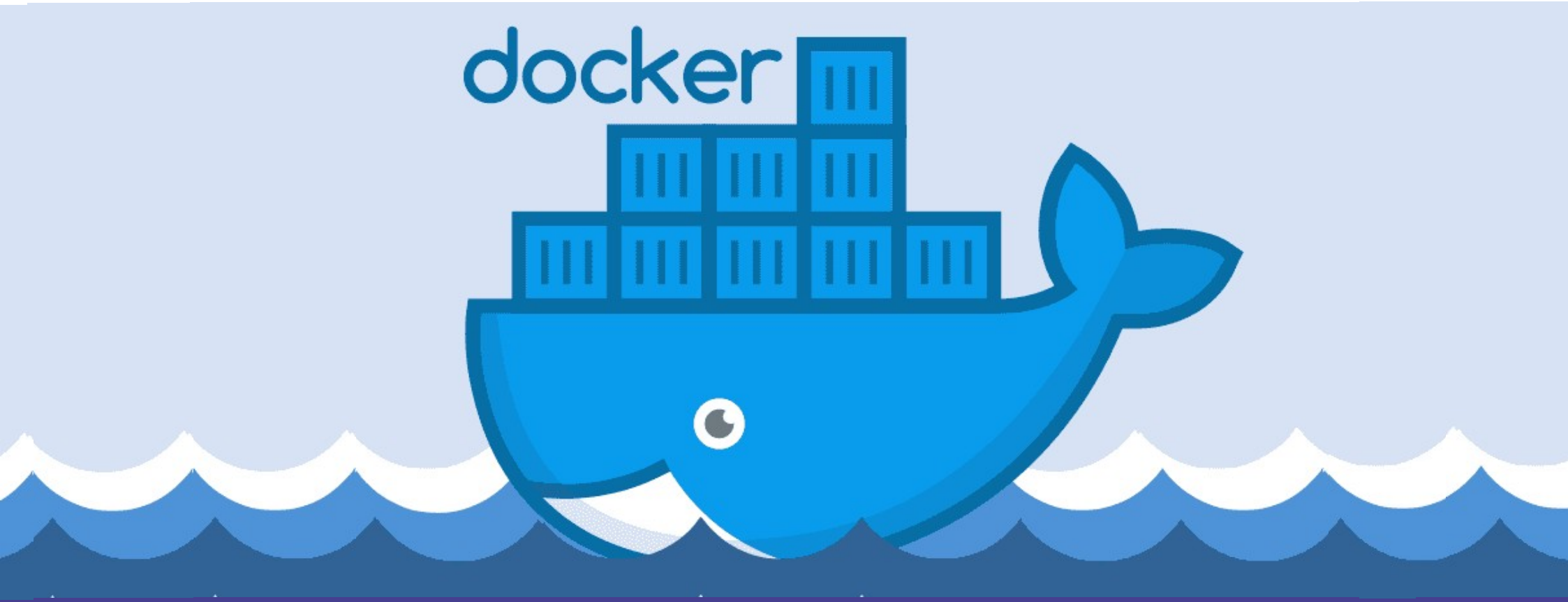
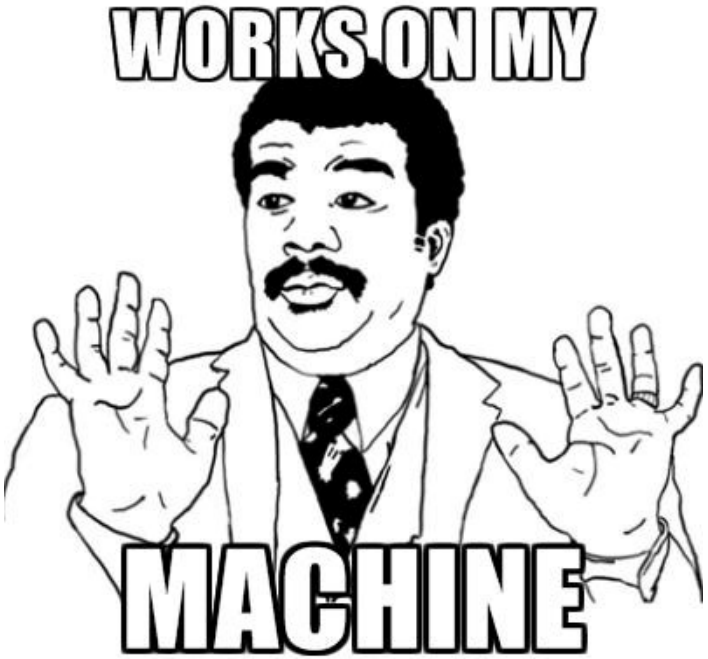


Introdução ao Docker: conteinerização descomplicada

Profª Bárbara Pimenta Caetano
Prof Enzo Seraphim

Quem aqui já teve problema pra rodar um projeto que funcionava no computador de outra pessoa e não no seu?



Problemas que todo dev já enfrentou

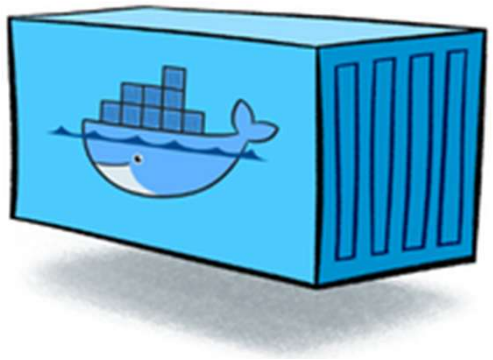
“Funciona na minha máquina”

Ambientes diferentes quebram tudo (Windows ≠ Linux ≠ Mac)

Instalar dependência dá erro

Setup de projeto = 2h (ou 2 dias)

Precisa de Java 8, mas você tem Java 17



docker

Docker é uma plataforma que empacota uma aplicação e todas as suas dependências em um ambiente isolado chamado container.

O que é Docker?

Docker é como um container de navio

- Isola a carga (aplicação)
- Pode ser transportado por qualquer navio (ambiente)
- Tudo funciona do mesmo jeito em qualquer lugar

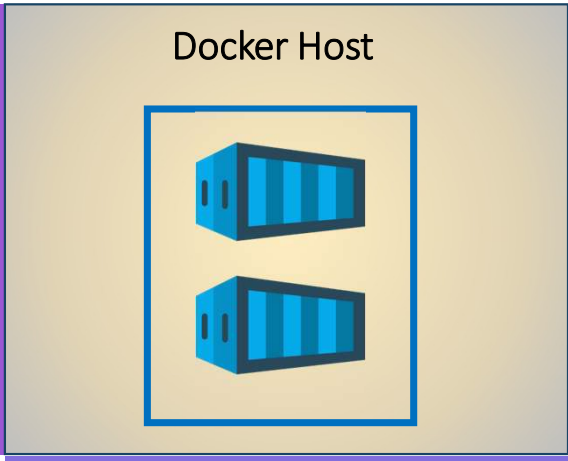
Características

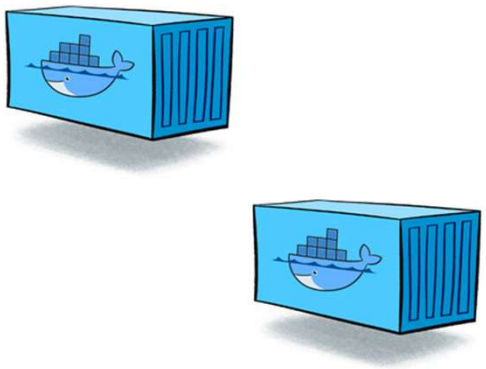
Reprodutível

ambiente sempre igual

Portável

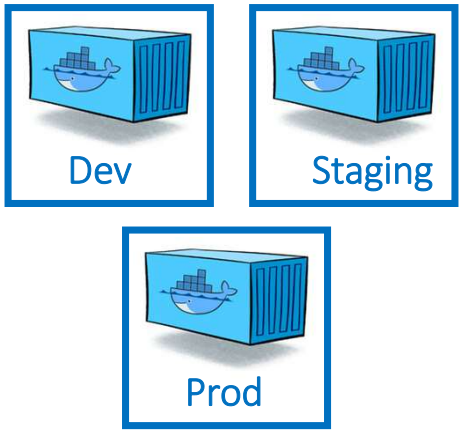
roda em qualquer lugar (dev, staging, prod)





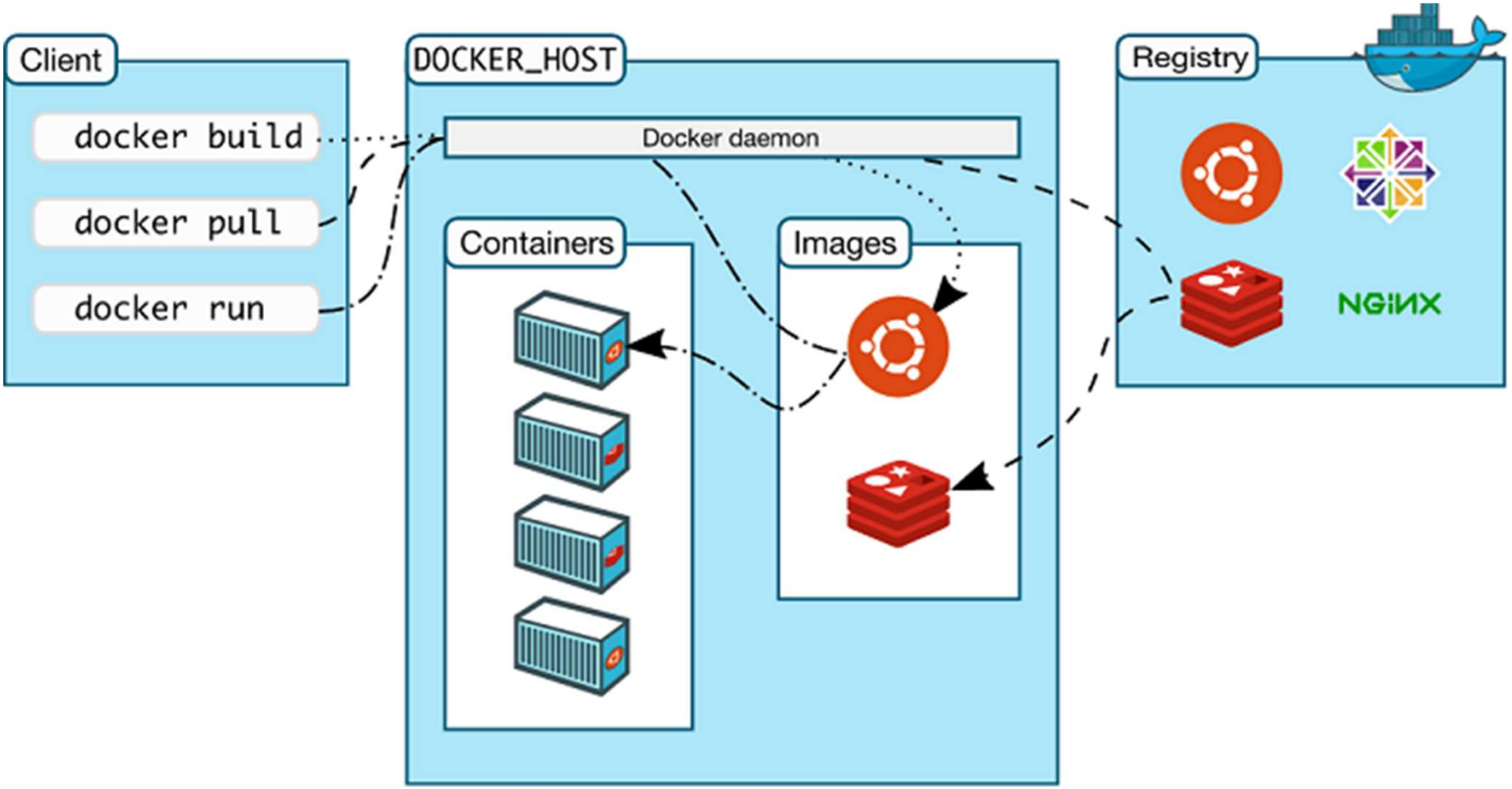
Leve

mais rápido que máquinas virtuais

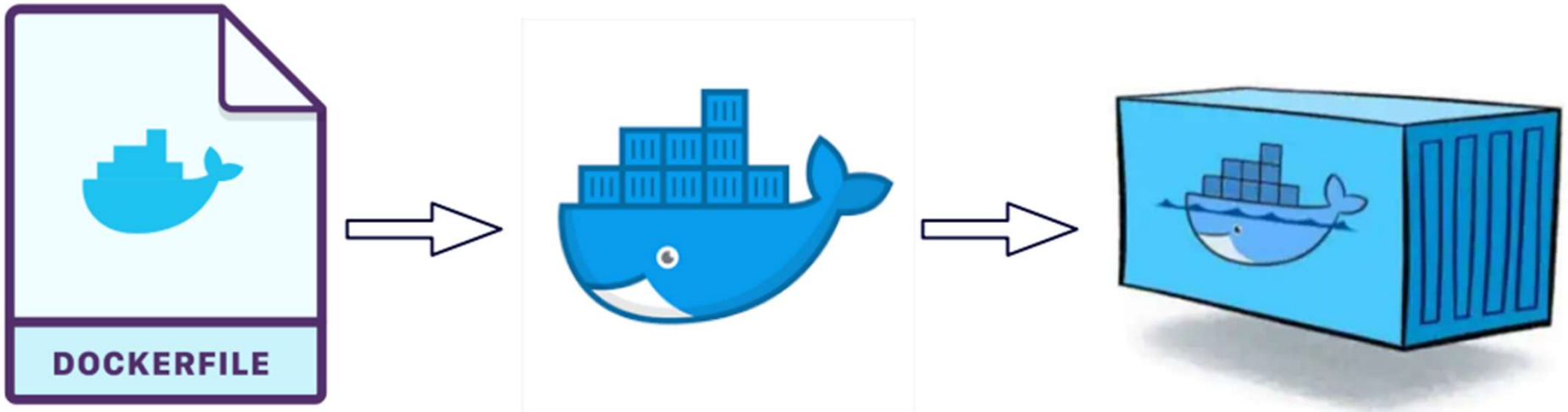


Isolado

sem conflito com seu sistema



Arquitetura



Docker file | **Docker Image** | **Docker Container**

Um script com instruções para criar uma imagem (como um “receita de bolo”)	Uma receita “congelada” da aplicação: é o que o Docker usa para criar containers	Um ambiente isolado que roda sua aplicação com tudo que ela precisa

Portas e Volumes

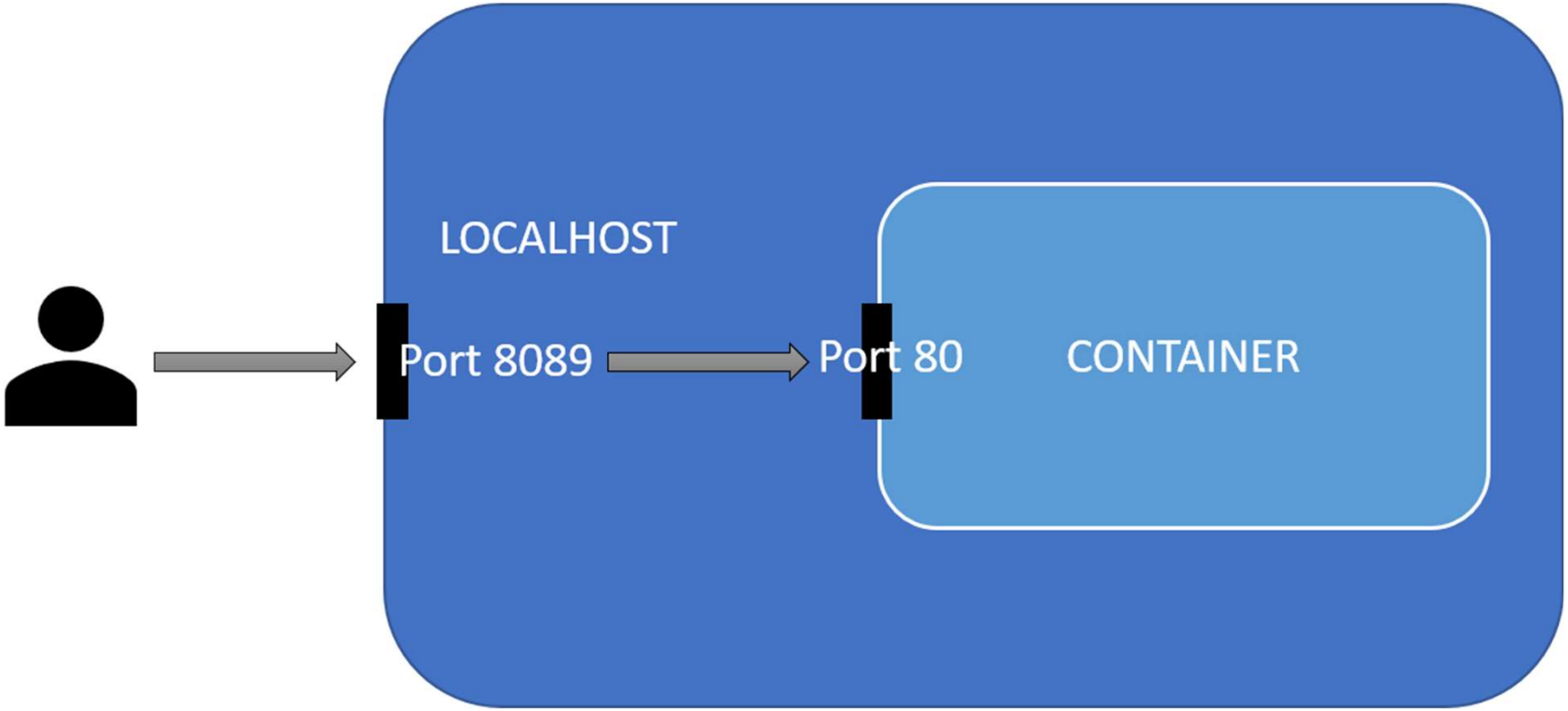
PORTAS

- Um container tem **suas próprias portas internas**
- Para acessá-lo de fora (navegador, APIs externas, etc), você precisa **mapear as portas**

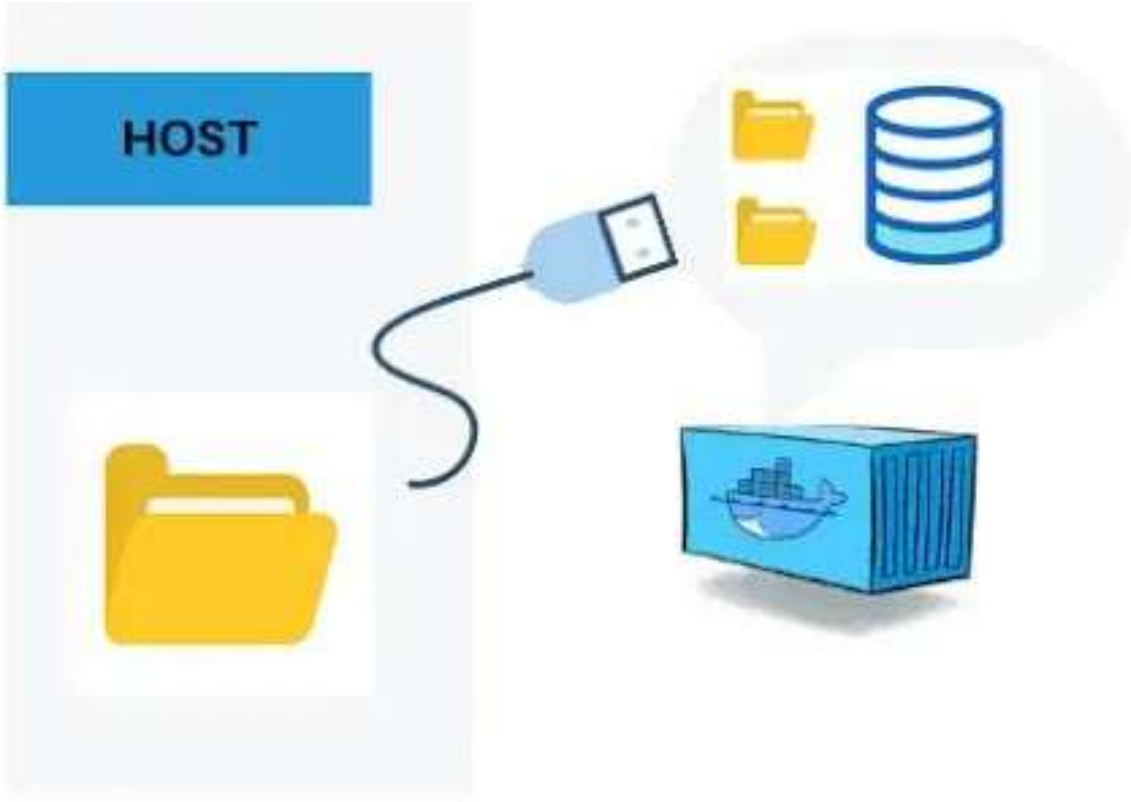
VOLUMES

- Por padrão, **tudo que acontece dentro do container desaparece ao parar**
- Volumes permitem **persistir dados** fora do container

Portas



Volumes




Vamos botar a mão na massa?

Seu 1º Container

docker run hello-world

1. Verifica se a imagem hello-world já existe localmente
2. Se não existir, **baixa do Docker Hub**
3. Cria um **container** temporário a partir da imagem
4. Executa o container, que imprime uma mensagem e encerra

Rodando com NGINX

docker run -d -p 8080:80 nginx

docker run: inicia um container

-d: “Detached mode” – roda em segundo plano

-p 8080:80: Mapeia a porta 80 do container na porta 8080 da máquina

nginx: Usa a imagem oficial do Nginx do Docker Hub

docker ps: Lista containers em execução

docker stop <id> : Para o container

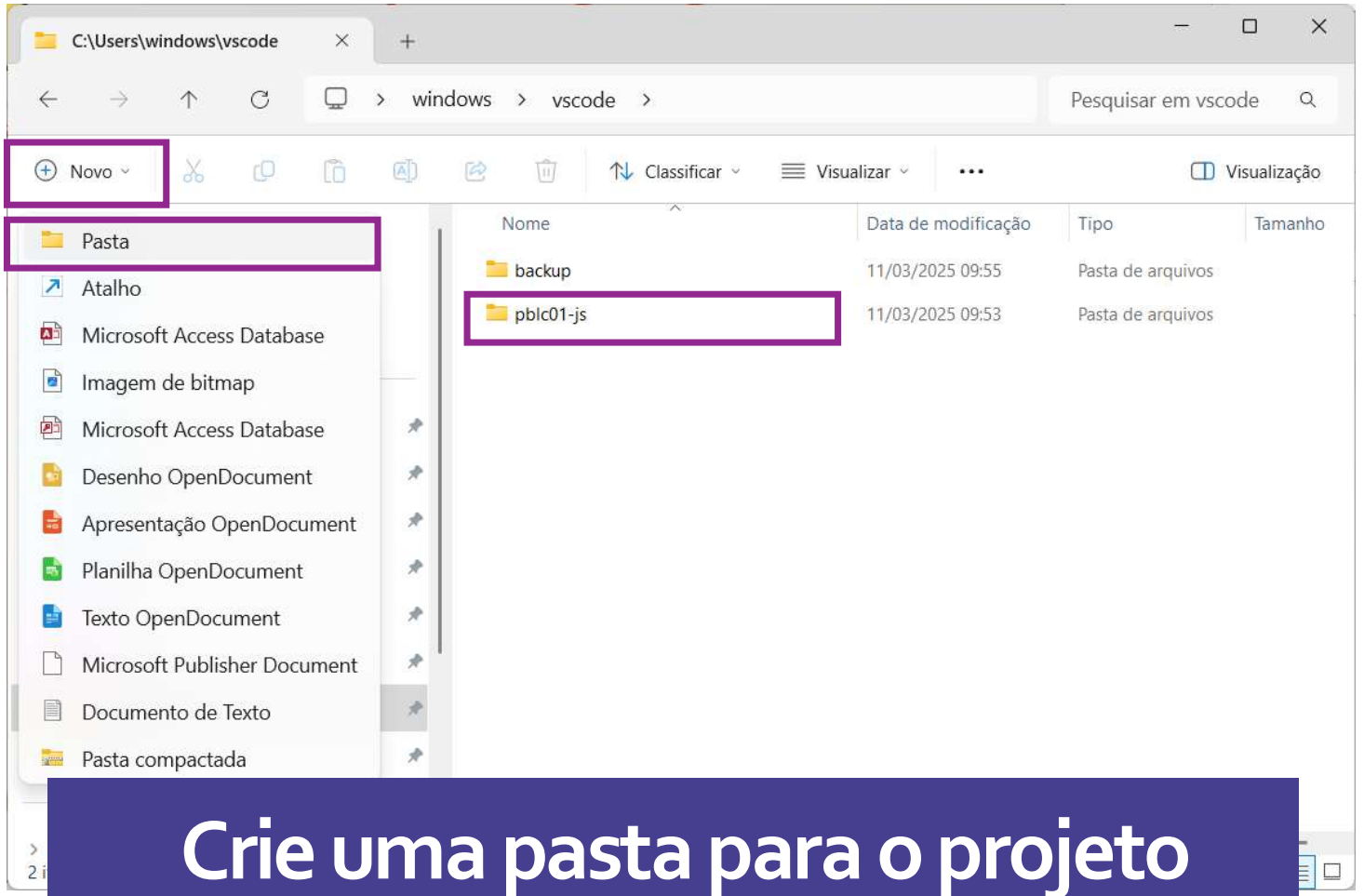
docker rm <id> : Remove o container

docker images: Lista imagens baixadas

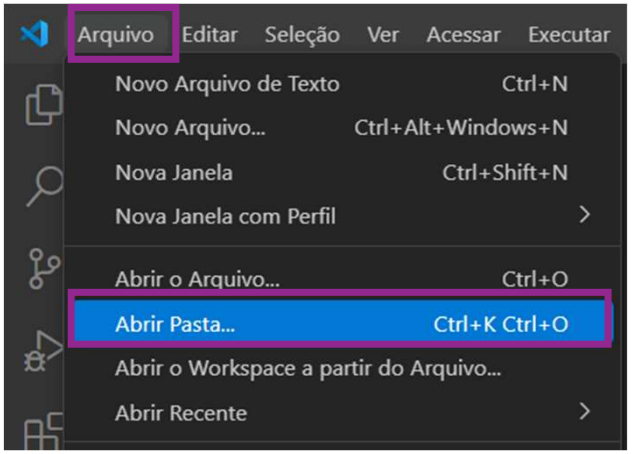
Comandos Úteis

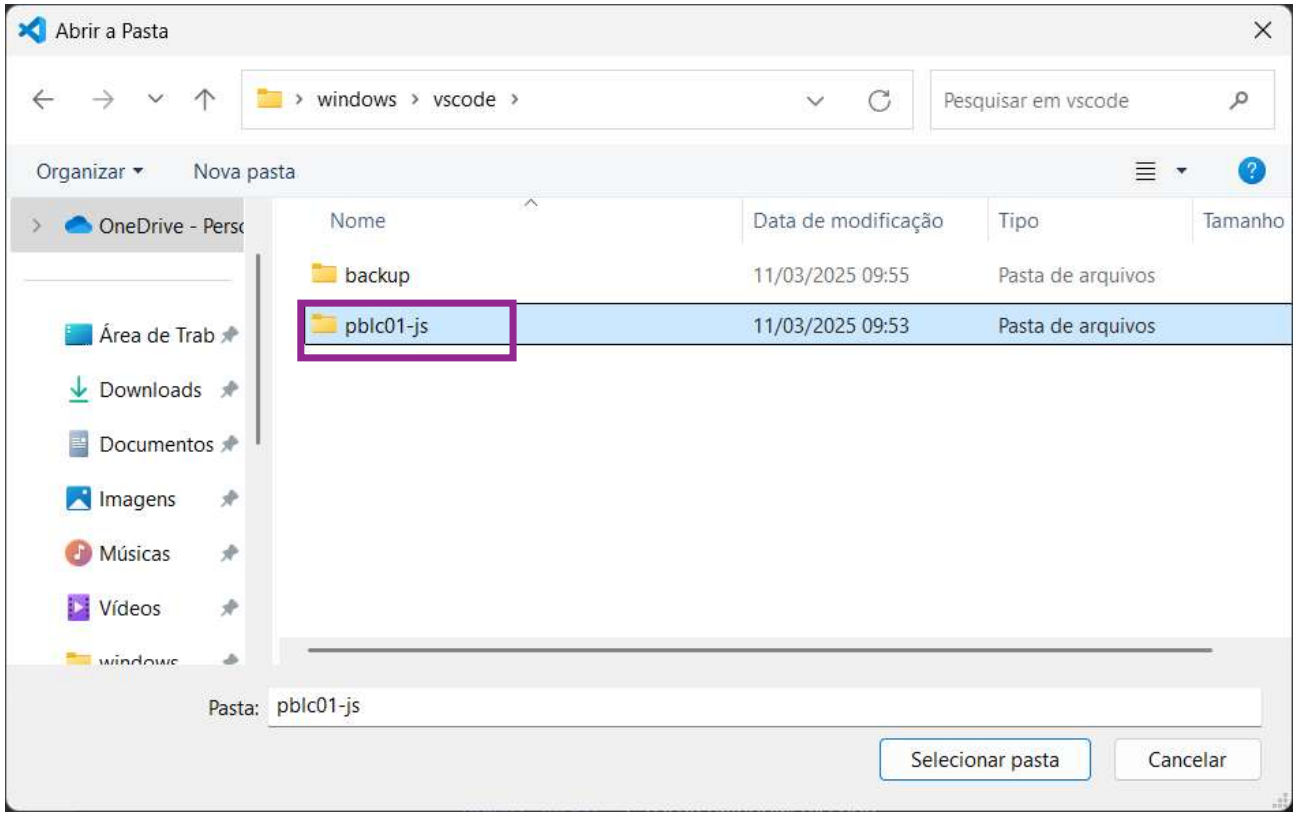
Vamos criar e subir uma aplicação node usando Docker

Para isso vamos precisar criar um DOCKERFILE



Crie uma pasta para o projeto node





Abra a pasta no VSCode

Para abrir o terminal clique no menu:

Ver | Terminal ou [Ctrl]+[’]

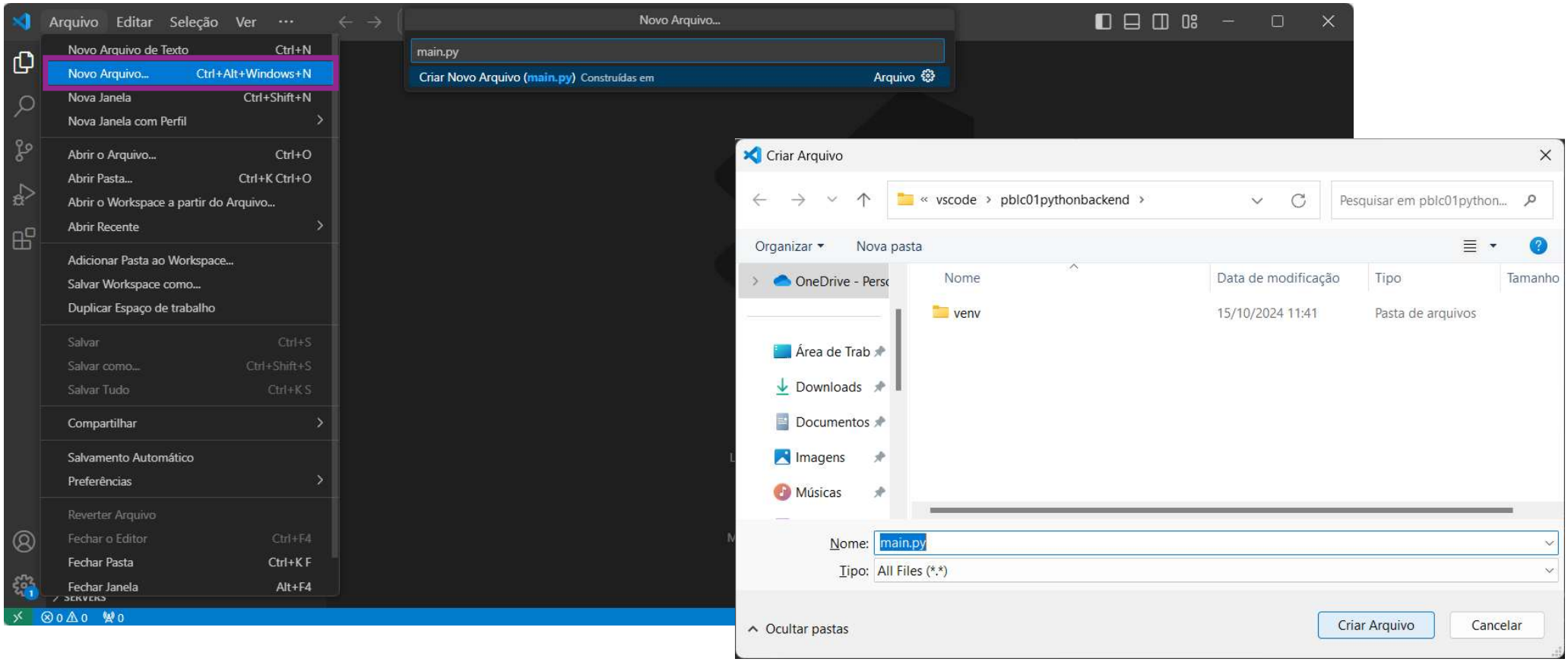
Para Inicializar o projeto node:

> npm init –y

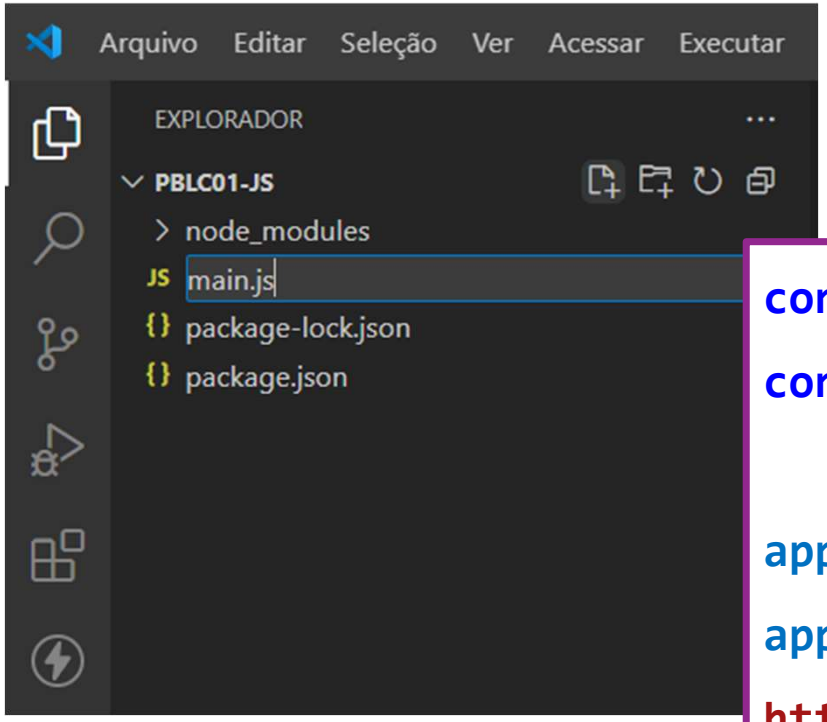
Vamos precisar de uma lib:

> npm install --save express

Inicie o projeto node



Crie o arquivo main.js




```
const express = require('express')
const app = express()

app.get('/', (req, res) => res.send('Hello Docker!'))
app.listen(3000, () => console.log('Rodando em
http://localhost:3000'))
```

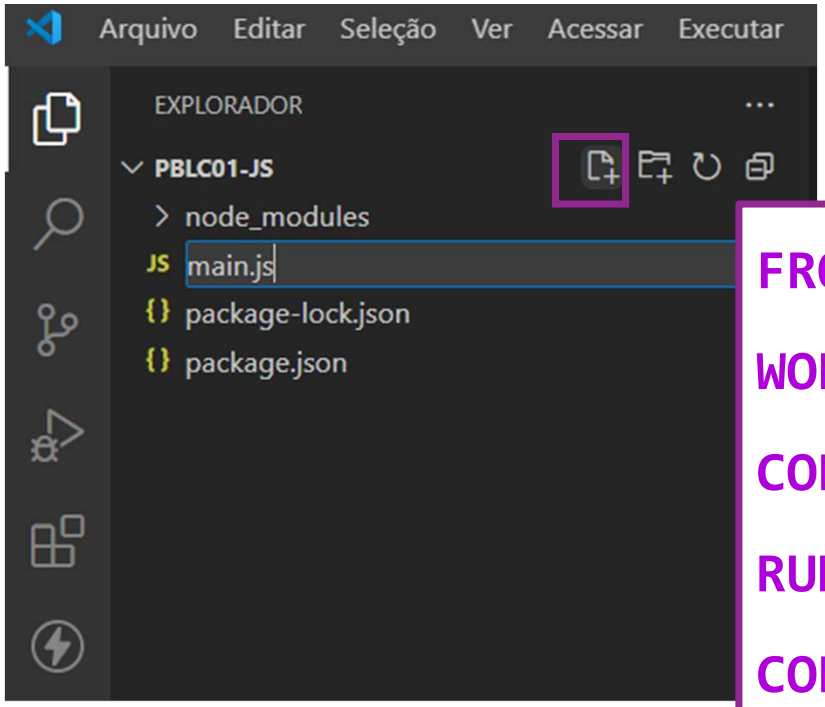
Crie o arquivo main.js

Para testar o projeto:

> node main.js

Abrir localhost:3000 no Navegador

Testar o node




```dockerfile
FROM node:23-slim

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "main.js"]
```

Crie o arquivo Dockerfile

Dockerfile: receita para construir a imagem

docker build: compila a imagem

docker run: executa o container com base na imagem criada

> docker build -t hello-app .

> docker run -p 3000:3000 hello-app

Construir e rodar a imagem

docker ps: lista containers em execução

docker stop <id>: para o container

docker start <id>: inicia container parado

docker rm <id>: remove container

docker images: lista imagens baixadas

docker rmi <imagem>: remove imagem

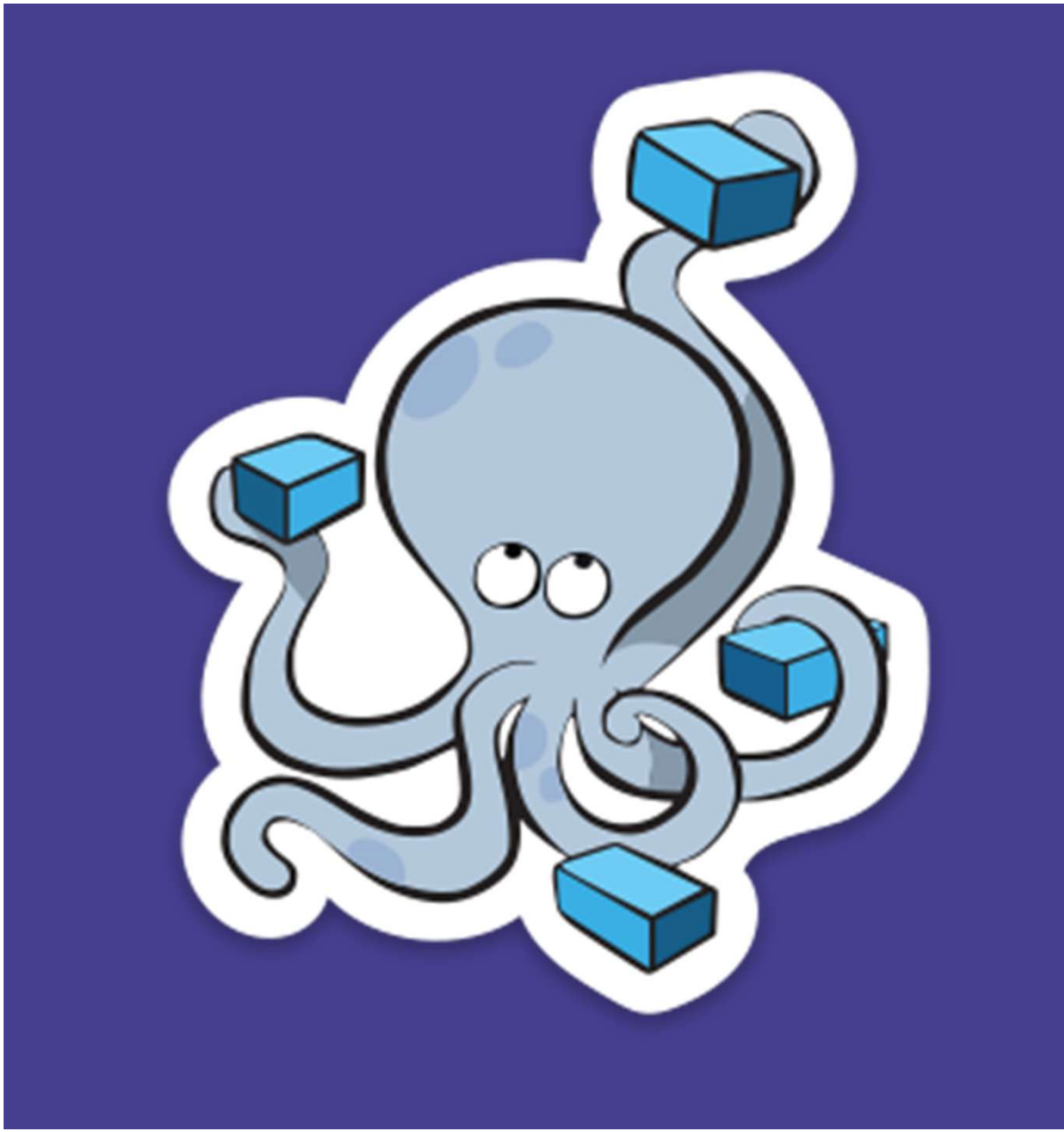
docker volume prune: limpa volumes não usados

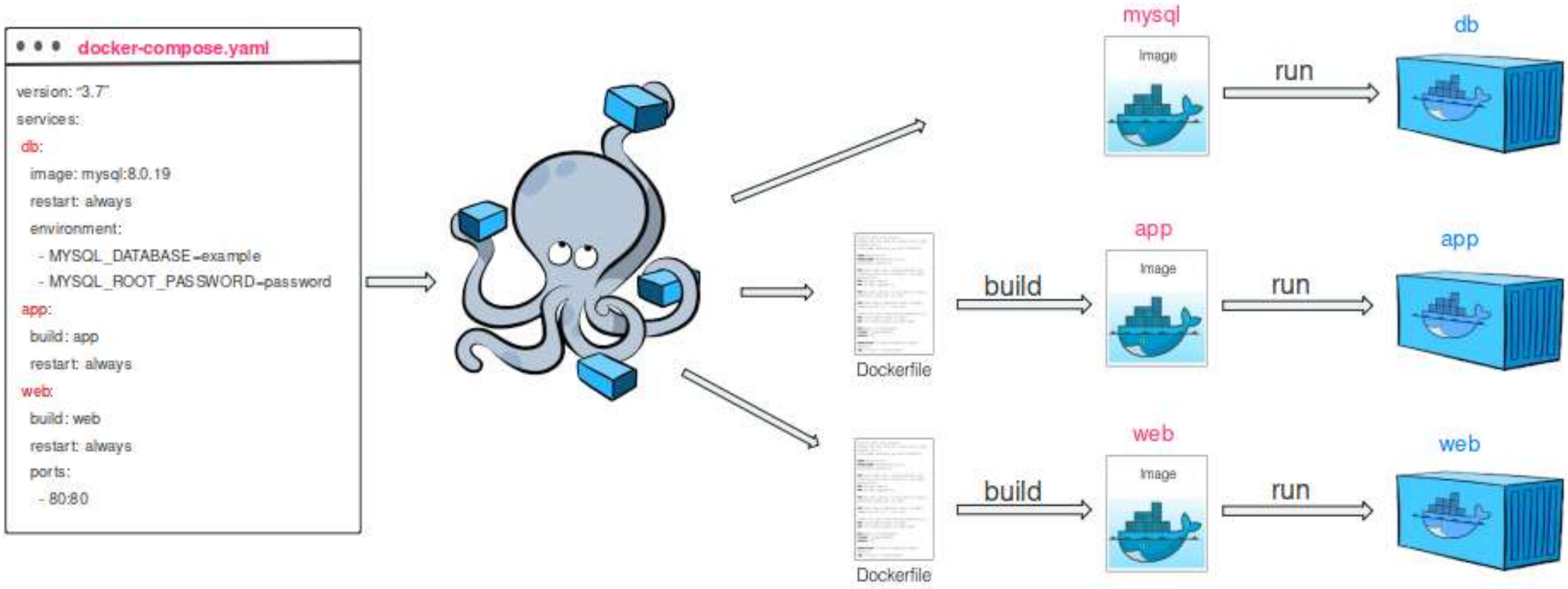
docker system prune –a: limpa tudo (use com cuidado!)

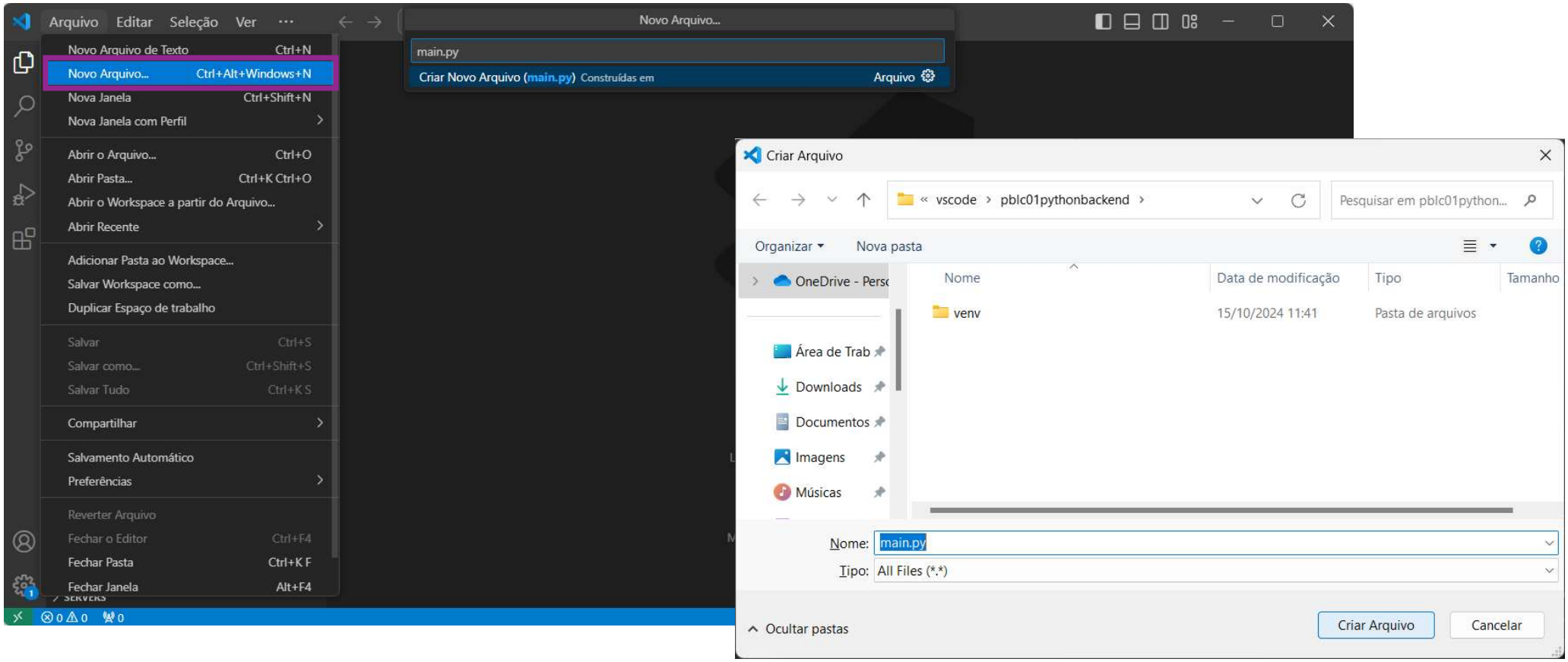
Reforço dos Comandos Úteis

O que é Docker Compose?

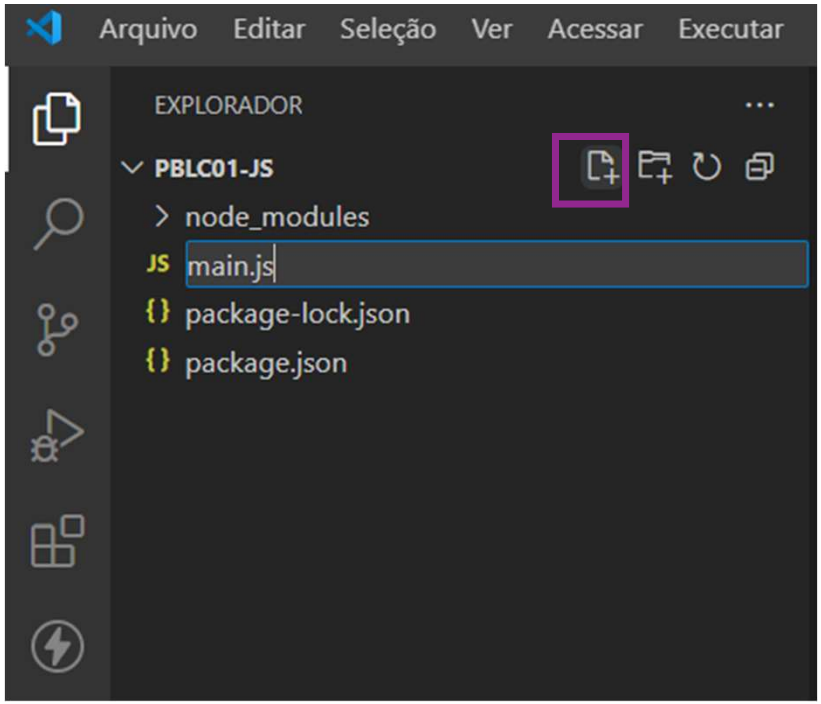
Uma ferramenta para definir e executar aplicações Docker de vários contêineres, facilitando o gerenciamento de toda a stack.




docker-compose.yaml
version: "3.7"
services:
db:
image: mysql:8.0.19
restart: always
environment:
- MYSQL_DATABASE=example
- MYSQL_ROOT_PASSWORD=password
app:
build: app
restart: always
web:
build: web
restart: always
ports:
- 80:80
mysql
Image
run
db
Dockerfile
build
app
Image
run
app
Dockerfile
build
web
Image
run
web



Crie o arquivo Docker-compose.yaml em uma nova pasta




```yaml
services:
  db:
    image: postgres:15.3
    volumes:
      -
./volumes/postgres/data:/var/lib/p
ostgresql/data
    environment:
      POSTGRES_PASSWORD:
"postgres"
    ports:
      - "5432:5432"
```

docker-compose.yaml

Para subir os containers do Docker composse:

> docker compose up -d

Para parar:

> docker compose down

Subir os containers do Docker compose

"Docker é leve, mas não mágico: ele vai acumulando coisas. Limpeza regular para um ambiente saudável."