Programação Front-end – AULA docker

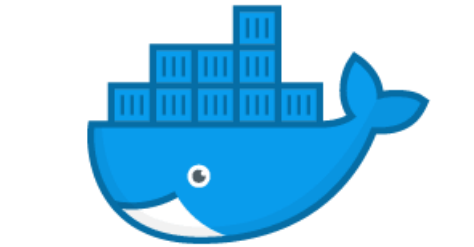
Matheus Moresco
Engenharia de Software - 3º Período
2025/01

Introdução

- Entender o que é Docker e para que serve
- Compreender a diferença entre containers e VMs
- Executar comandos básicos do Docker
- Criar um Dockerfile
- Usar o Docker Compose para orquestração

O que é Docker?

- É uma plataforma que permite **empacotar, distribuir e executar aplicações em contêineres**.
- Um contêiner é uma unidade leve e portátil que inclui tudo o que a aplicação precisa para rodar: código, bibliotecas, dependências, e configurações



Por que o Docker é útil?

Sem Docker: você precisa configurar todo o ambiente da aplicação na máquina (linguagem, banco de dados, dependências, etc.).

Com Docker: você usa um arquivo (Dockerfile) para definir o ambiente, e o Docker garante que tudo funcione do mesmo jeito em qualquer computador ou servidor.

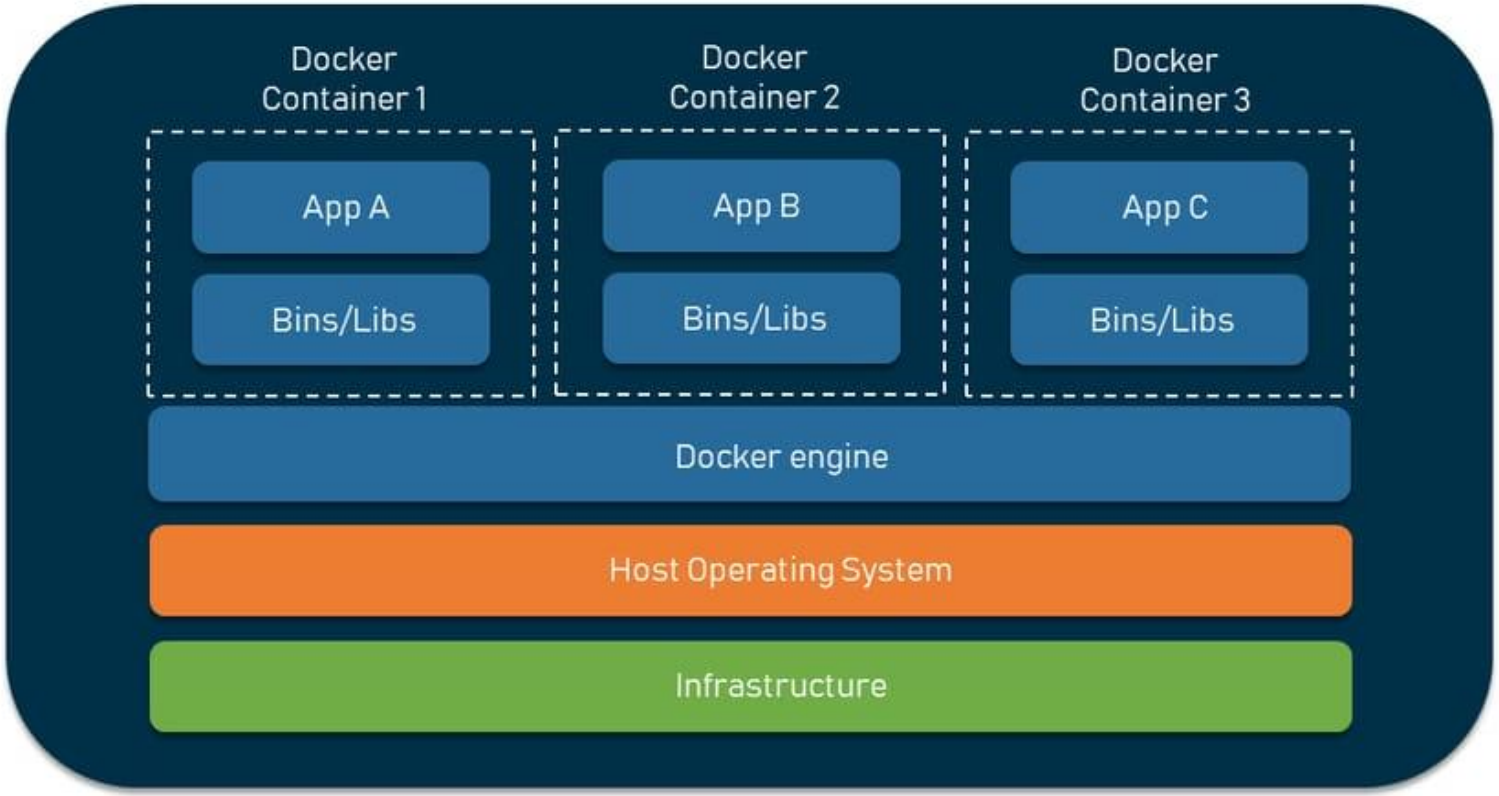
Por que o Docker é útil?

- **Consistência** entre ambientes (desenvolvimento, teste, produção).
- **Facilidade** de implantação.
- **Isolamento** entre aplicações.
- **Portabilidade**: roda no Windows, Linux, Mac, servidores, nuvem etc.

Arquitetura do Docker

- **Docker Engine**: Core do Docker
- **Imagens**: Snapshot do sistema de arquivos + instruções
- **Containers**: Instâncias em execução das imagens
- **Volumes**: Persistência de dados
- **Redes**: Comunicação entre containers

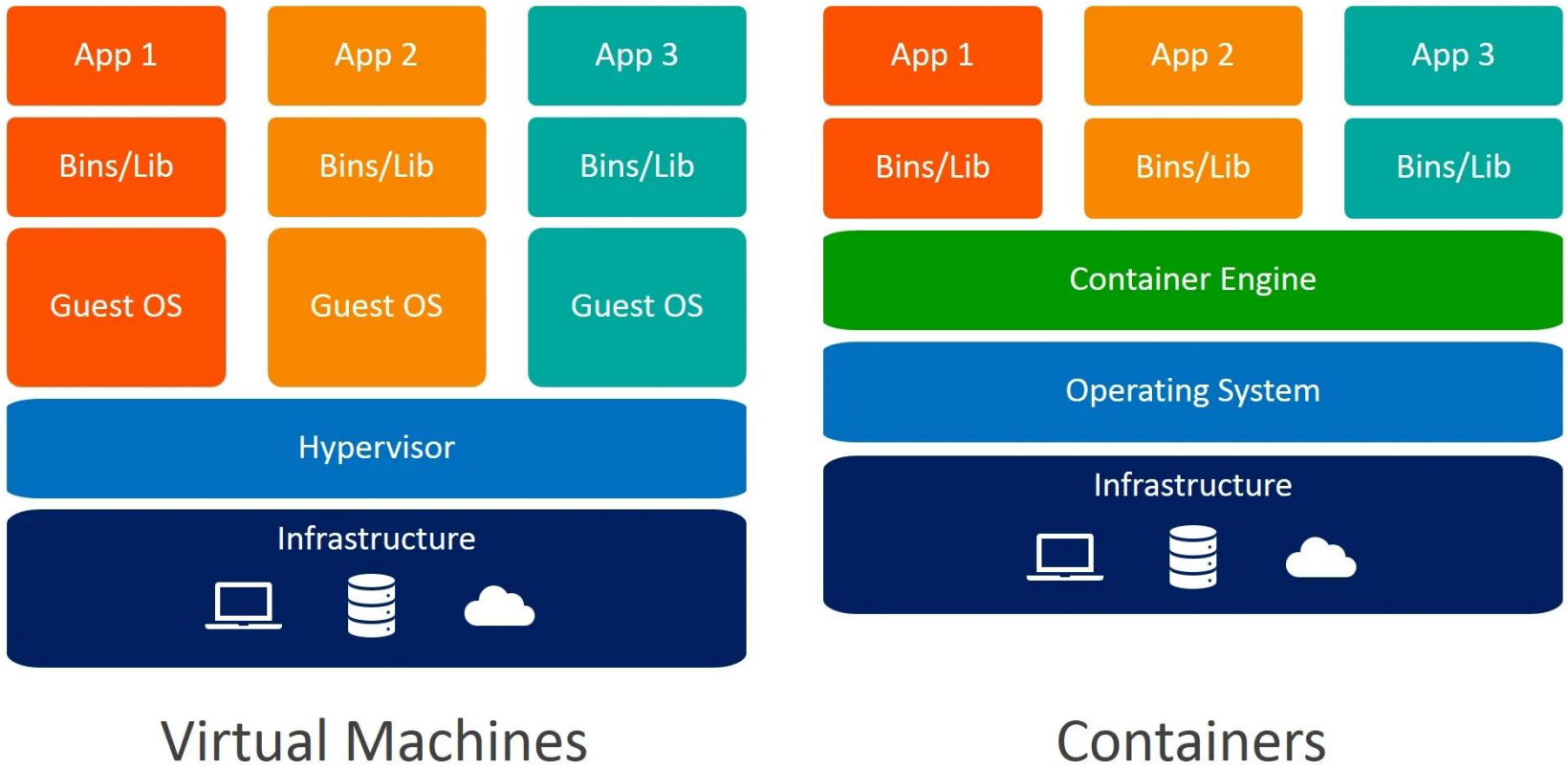
Arquitetura do Docker



Containers vs VMs

Característica	VM	Container (Docker)
Virtualização	Hardware	Sistema operacional
Desempenho	Maior overhead	Leve e rápido
Isolamento	Forte	Moderado
Tempo de inicialização	Minutos	Segundos

Containers vs VMs



Virtual Machines

Containers

Instalar o Docker

- Acesse: https://www.docker.com/products/docker-desktop
  - Escolha sua plataforma (Windows, macOS ou Linux).
  - Instale e inicie o Docker Desktop.
  - Verifique se está funcionando:

Imagem Docker

- Uma **imagem Docker** é um **arquivo imutável** que contém tudo o que é necessário para executar uma aplicação:
  - O **sistema operacional base** (como Ubuntu ou Alpine),
  - O **código-fonte** da aplicação,
  - As **bibliotecas e dependências**,
  - As **instruções** para executar o aplicativo.
- Você **não executa diretamente uma imagem**. Em vez disso, você **cria um contêiner a partir dela**, que é a instância em execução.

Imagem = receita
Contêiner = prato pronto feito a partir da receita

Características da imagem Docker

- É **portável** (roda em qualquer máquina com Docker).
- É **leve** (usa camadas reutilizáveis).
- Pode ser **armazenada e compartilhada** via Docker Hub, GitHub Packages, etc.
- Usamos as imagens docker para criar conteiners

Rodando contêiners Docker

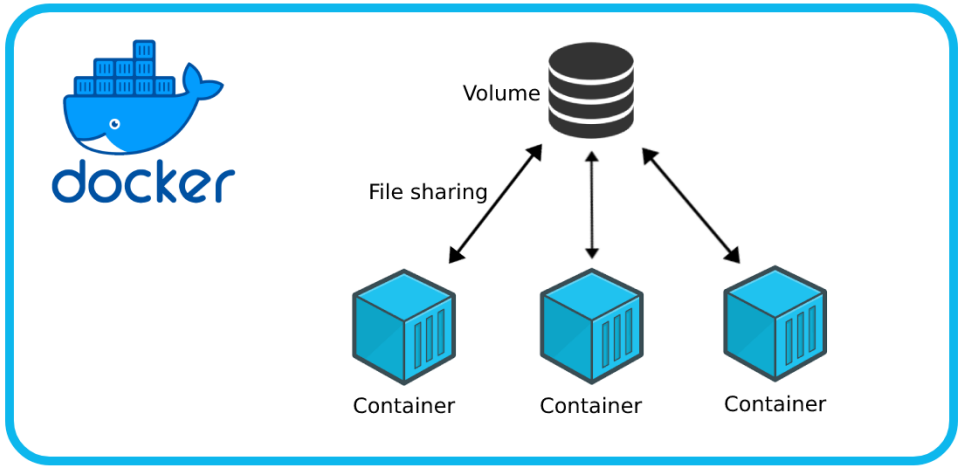
- Para rodar um contêiner docker precisamos primeiramente ter uma imagem, e a partir desta imagem inicializaremos o contêiner, usando o comando **run**.

- Abra o terminal do computador:
- Execute o comando: '*docker run -it --name sistema_docker ubuntu*'

- Explicação:
  - **docker run**: comando para criar e executar um contêiner.
  - **-it:** abre o terminal interativo (-i interativo, -t modo terminal).
  - **--name meu-ubuntu**: dá um nome ao contêiner.
  - **ubuntu**: é a imagem (vai baixar se for a primeira vez).

Volumes docker

- Por padrão, tudo o que acontece dentro de um contêiner **é perdido quando ele é removido**. Se sua aplicação salva arquivos, banco de dados ou configurações, isso **se perde** sem um volume.
  - **Volumes resolvem isso** armazenando dados fora do contêiner, no host, de forma persistente.



Criando Imagens docker

- Quando trabalhamos com imagens docker, muitas vezes precisamos de configurações personalizadas nas constainers, como
  - Instalar determinadas bibliotecas por padrão
  - Ter os arquivos do projeto no container
  - Configurações do ambiente no geral.

- Para tal, podemos criar imagens personalizadas usando **Dockerfile**. Este arquivo define **como construir a imagem**.

Comando para criação de imagens

- **Criação da imagem**: ‘docker build -t minha-imagem:1.0 ‘
- **Rodar um container com a imagem**: ‘docker run --name meu-app -p 5000:5000 minha-imagem:1.0’
- **Listar imagens**: ‘docker images’
- **Deletar imagens**: ‘docker rmi minha-imagem:1.0’

Docker Hub

- O Docker Hub é como um **repositório na nuvem para imagens Docker**, parecido com o GitHub, mas em vez de guardar código, ele guarda **imagens de contêineres** — que são pacotes prontos com sistemas, aplicações, configurações e dependências.
  - **Imagens Oficiais**
    - Mantidas pela própria Docker Inc. ou por empresas parceiras.
    - Exemplo: mysql, node, ubuntu, nginx.
  - **Imagens da Comunidade**
    - Criadas por usuários ou empresas.
    - Exemplo: bitnami/mysql, linuxserver/plex.
  - **Suas próprias imagens**
    - Você pode criar e enviar (fazer push) suas imagens para lá.

Docker compose

- Imagine que você tem que subir **uma aplicação Node.js + MongoDB**. Com Docker puro, você teria que:
  - Criar e rodar um contêiner para o Node.
  - Criar e rodar outro contêiner para o Mongo.
  - Configurar comunicação entre eles.
  - Gerenciar as redes, volumes e portas manualmente.

- Com o **Docker Compose**, você descreve tudo em um único arquivo .yml, e depois executa com um simples comando:

Comandos docker

- `**docker-compose up**` : Sobe todos os serviços
- `**docker-compose up -d**` : Sobe em segundo plano
- `**docker-compose down**` : Para e remove os contêineres, rede e volumes
- `**docker-compose ps**` : Mostra os contêineres ativos
- `**docker-compose logs**` : Mostra os logs dos serviços