Aula 15 - Docker e Docker Compose

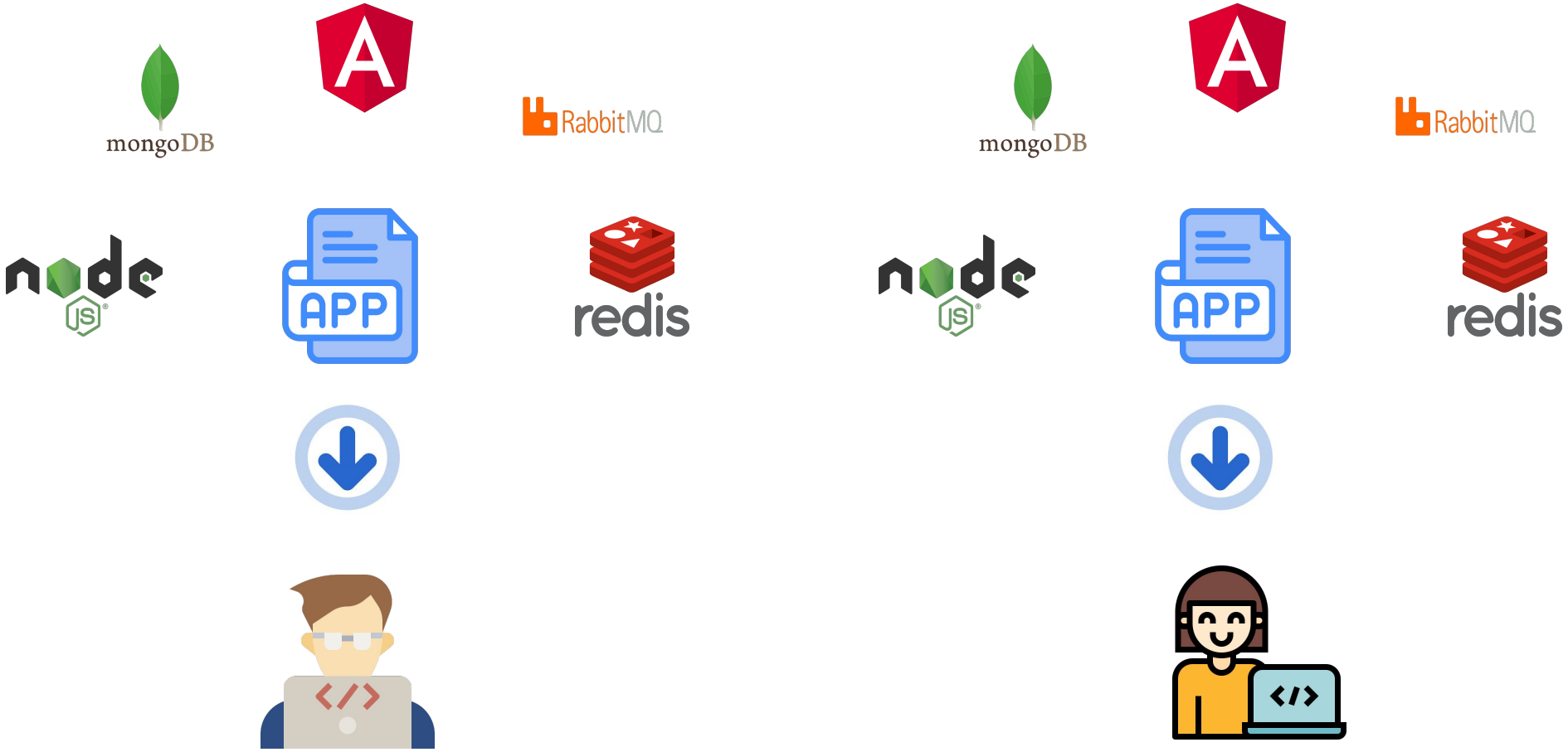
Disciplina: Arquitetura de Software

Prof. Me. João Paulo Biazotto

UniCesumar

Um pouco de história...

Um pouco de história...

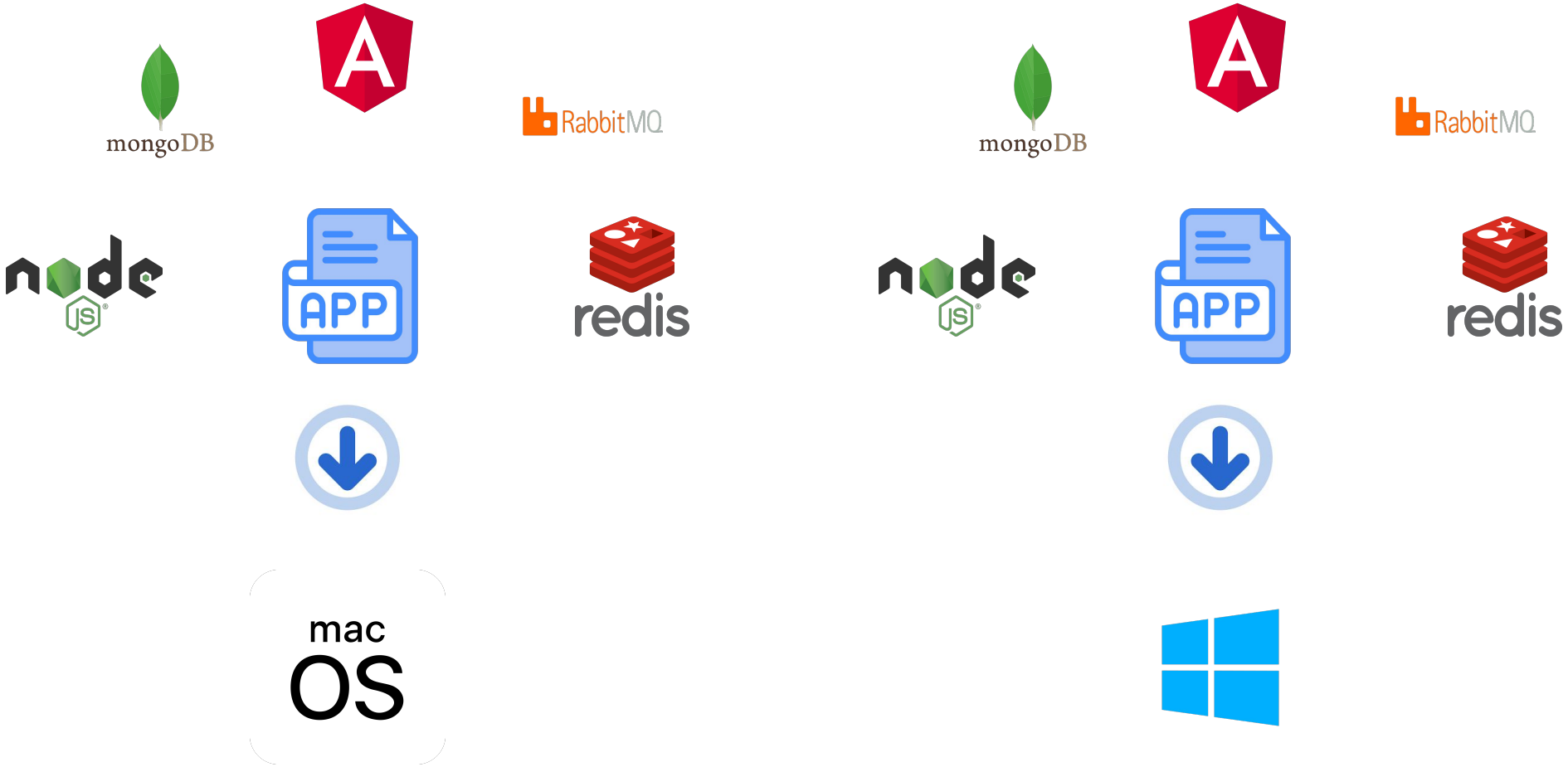


Um pouco de história...

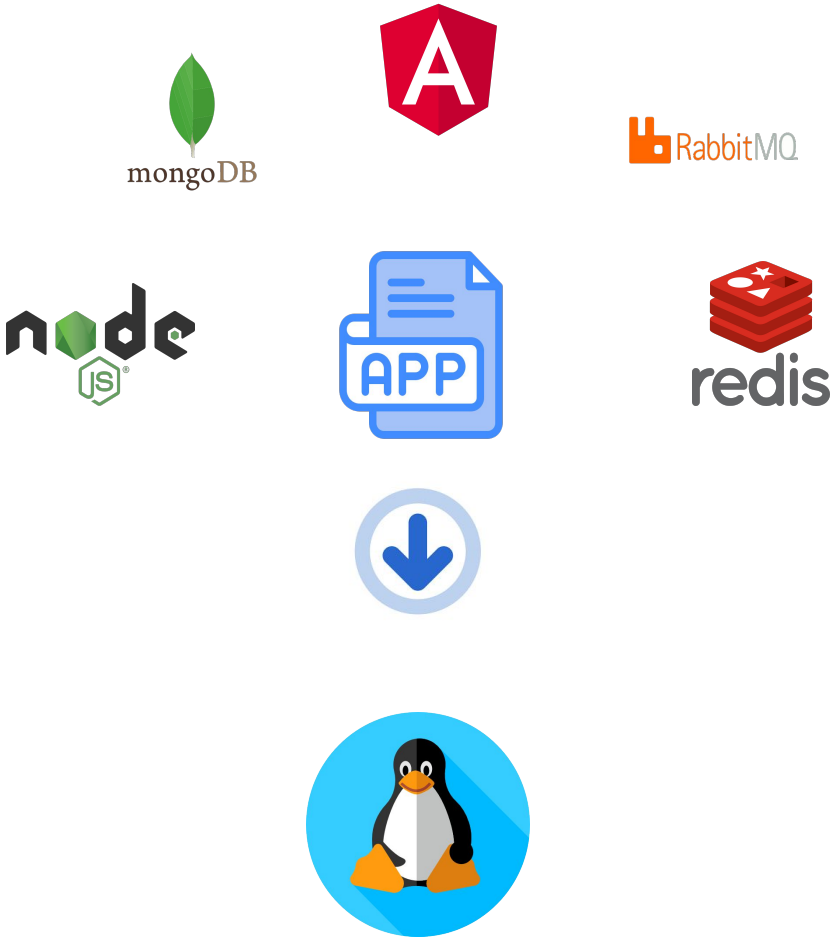
- **Configuração manual e inconsistente**
  - Cada desenvolvedor configurava seu ambiente local com ferramentas, bibliotecas e dependências específicas.

- **Problemas de compatibilidade**
  - “Na minha máquina funciona!” era comum devido a diferenças entre ambientes.

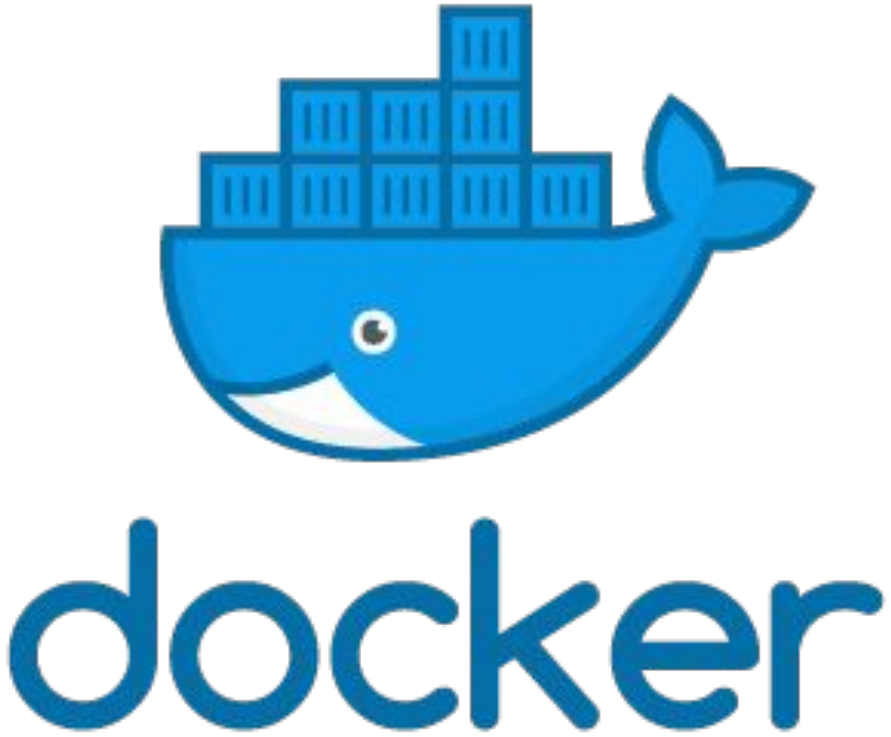
Um pouco de história...



Um pouco de história...



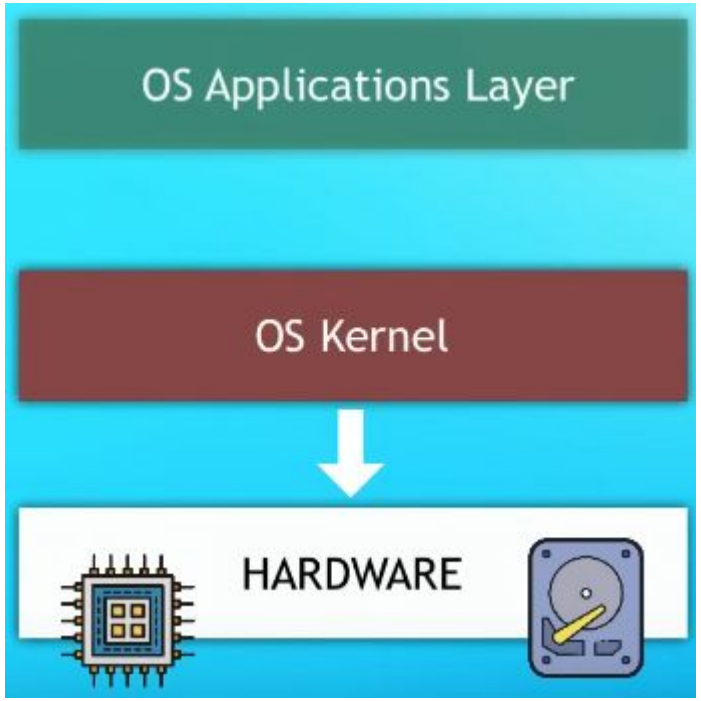
Docker



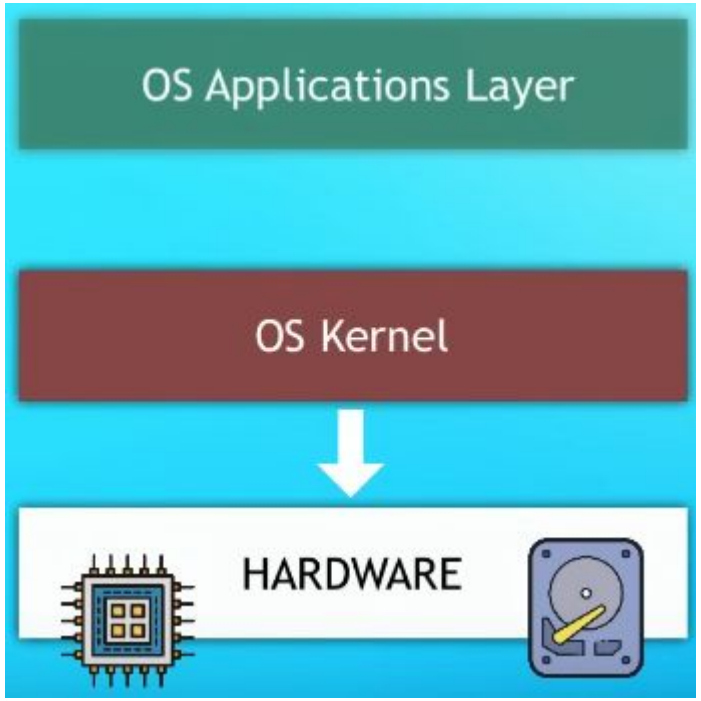
Docker

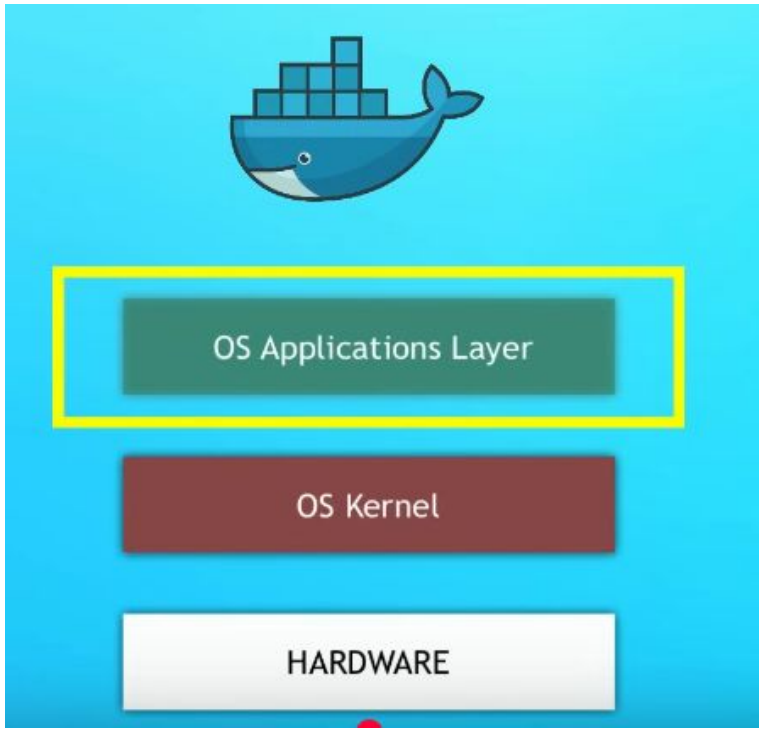
- Plataforma para criar, distribuir e executar **contêineres**
- **Virtualização** a nível de sistema operacional
- Benefícios: leveza, portabilidade, reprodutibilidade

Docker vs Máquinas Virtuais



Docker vs Máquinas Virtuais





Docker vs Máquinas Virtuais

Característica	Máquina Virtual (VM)	Contêiner (Docker)
Virtualização	Virtualiza o SO completo	Virtualiza a aplicação e dependências
Sistema Operacional	Cada VM possui um SO completo próprio	Compartilham o mesmo kernel do host
Consumo de Recursos	Alto (RAM, CPU, disco)	Baixo – apenas o necessário para a aplicação
Tempo de Inicialização	Lento (minutos)	Rápido (segundos ou menos)
Tamanho da Imagem	Gigabytes	Megabytes até algumas centenas de MB
Isolamento	Forte (nível de SO)	Médio (nível de processo e namespaces)
Portabilidade	Menor – depende do hypervisor	Alta – contêiner roda onde o Docker estiver
Uso comum	Infraestrutura completa, legado	DevOps, CI/CD, microserviços, testes isolados

Imagem

Imagens são arquivos que contém todo o conteúdo e estrutura de sistemas operacionais. Elas são a base de construção de containers no Docker. Dessa forma, clientes podem construir templates em cima da parte de escrevível, e estes templates terão todas as configurações desejadas do container a ser construído.

É como um molde a partir do qual os **containers** são criados.

Exemplo: *python:3.11-slim* → imagem base com Python instalado

Docker Registry

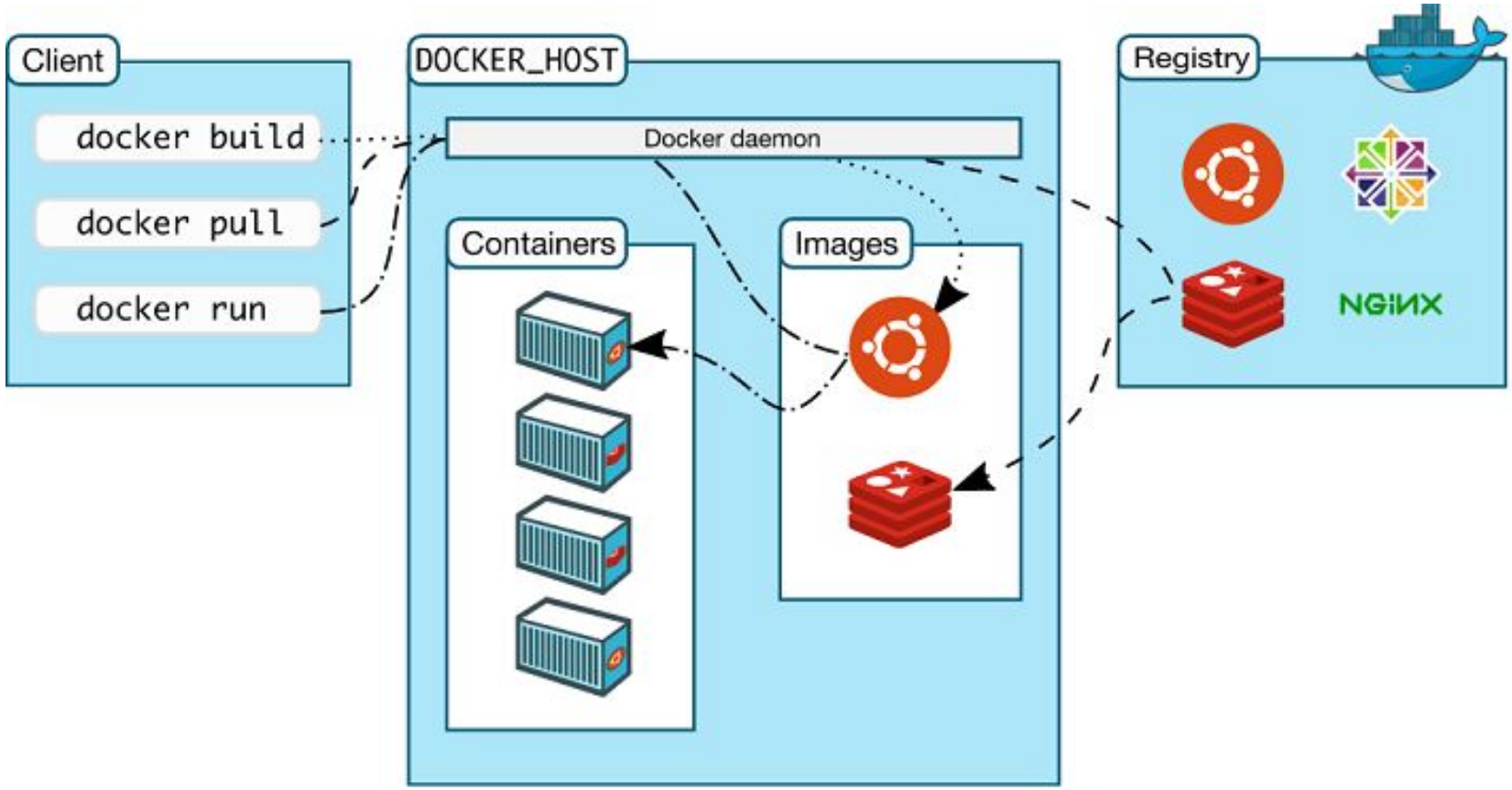
O ***Registro do Docker*** é uma espécie de repositório para imagens. Com esse registro, um usuário pode construir, salvar e distribuir imagens com outros. O site do Docker fornece um sistema de registro chamado *Docker Hub*, que funciona como um git, permitindo ao usuário que construa localmente suas imagens em sua máquina, e então realize operações de *commit e push*.

Container

Containers são os *ambientes de execução* do Docker, criados a partir de **imagens.** Uma imagem é como um template de classe, e containers seriam instâncias dessas classes .

Exemplo: Um contêiner com Python + Flask para microserviços.

Components Docker



Principais Comandos Docker

- docker pull
- docker images
- docker run
- docker ps
- docker stop
- docker rm
- docker build

docker pull

- Realiza o download de uma imagem do Docker Hub (default) ou de outro registro.

Exemplos:

- *docker pull nginx:1.23*
- *docker pull nginx:latest*

docker images

- Lista as imagens disponíveis localmente;

docker run

- Cria e executa um container baseado em uma imagem.

Exemplo:

- *docker run -d -p 8080:80 nginx*

docker run

- Cria e executa um container baseado em uma imagem.

Exemplo:

- *docker run -d -p 8080:80 nginx*
  - Executa o NGINX em background (-d), mapeando a porta 8080 local para 80 do contêiner.

docker stop

- Para a execução de um contêiner em andamento.

Exemplo:

- *docker stop <id>*

docker rm

- Remove um contêiner parado.

Exemplo:

- *docker rm <id>*

Dockerfile

- Script de instruções para build de imagens
- Baseado em camadas
- Reutilizável e versionável

Dockerfile

```
FROM node:18

WORKDIR /app

COPY . .

RUN npm install

CMD ["npm", "start"]
```

Dockerfile

```
FROM node:18
```

Define a imagem base. Aqui estamos usando a imagem oficial do Node.js na versão 18. Ela já contém o Node e o npm instalados.

Dockerfile

```
WORKDIR /app
```

Define o diretório de trabalho dentro do contêiner. Todos os comandos seguintes (COPY, RUN, etc.) serão executados nesse diretório.

Dockerfile

```
COPY . .
```

Copia todos os arquivos do diretório atual do host (onde está o Dockerfile) para o diretório de trabalho (/app) dentro do contêiner.

Dockerfile

```
RUN npm install
```

Executa npm install, ou seja, instala todas as dependências do projeto Node.js.

Dockerfile

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

Define o comando padrão que será executado quando o contêiner for iniciado. Neste caso, ele roda o npm start, que deve estar definido no package.json.

Dockerfile

Vamos criar mais um Dockerfile, agora para um projeto em Java.

DÚVIDAS?

UniCesumar