

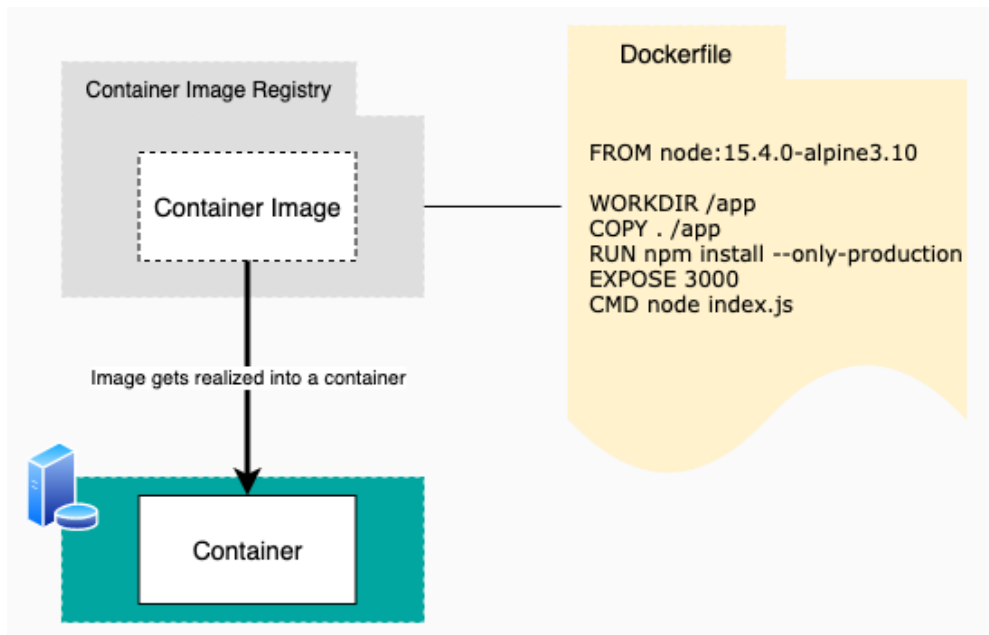
# Docker

## O que é o docker?

O Docker é um software de código aberto usado para implantar aplicativos dentro de containers virtuais. A containerização permite que vários aplicativos funcionem em diferentes ambientes complexos por eles **construírem, executarem e transferirem** o ambiente.

### Processo de Dockerização:

APLICAÇÃO → DOCKERFILE → IMAGEM(OS) → CONTAINER



## DockerHub:

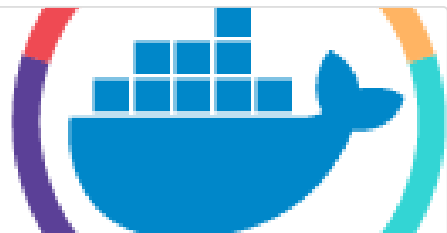
É um repositório público de imagens de containers, onde diversas empresas e pessoas podem publicar imagens pré-compiladas de soluções

## Comandos 🚀

### Dockerfile reference

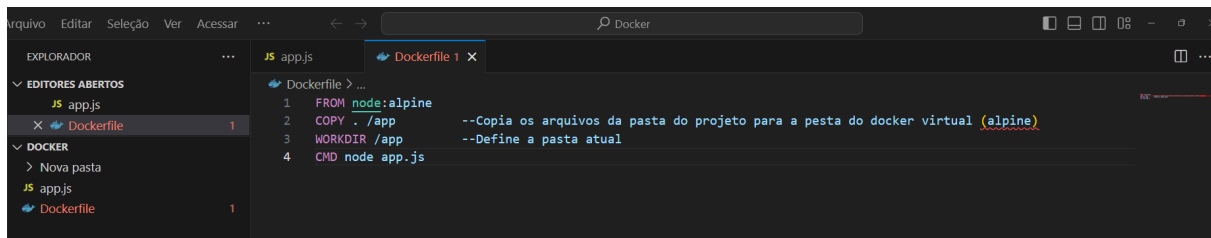
Find all the available commands you can use in a Dockerfile and learn how to use them, including COPY, ARG, ENTRYPOINT, and more.

<https://docs.docker.com/engine/reference/builder/#dockerfile-reference>



### Arquivo Dockerfile:

```
FROM node:alpine
COPY . /app --Copia os arquivos da pasta do projeto para a pasta do
WORKDIR /app --Define a pasta atual
ADD https://girludev.com/package -- Adiciona arquivos na aplicação
CMD node app.js --Roda o projeto
```

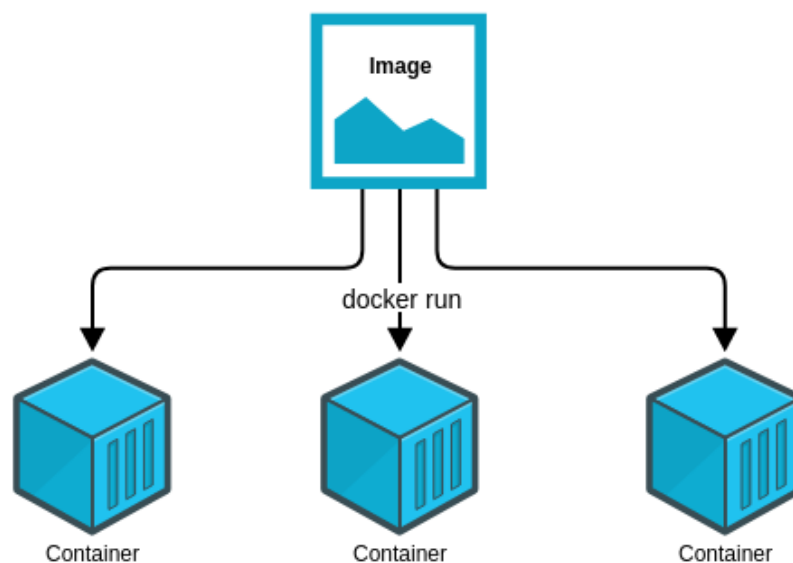


- docker version // visualizar versão do docker
- docker build --t name-your-project:tag. //Criando imagem com tag (tem ponto no final)
- docker build -t app . // criando imagem sem tag
- docker images //listar as imagens
- docker login //realizar login com a sua conta Docker
- docker run -d hi-docker:tag //Executar o projeto em background (Necessário mesclar o nome do repositório com a tag repositório:tag)
- docker run -d --name name-your-container hi-docker:tag //Executar o projeto em background (Necessário mesclar o nome do repositório com a tag repositório:tag)
- docker ps //containers rodando
- docker ps -a //histórico dos containers executados
- docker pull name-image //fazer download da imagem local
- docker run -it name-images // modo interativo que executa de forma temporária a imagem, e após encerrar a imagem é deletada
- docker exec -it -u your-user container-id bash //execute forma interativa usuário com id do container utilizando terminal bash
- docker images save -o name-your-application.tar name-your-application:version //Salvar uma imagem local do tipo .tar
- docker logs id-container-the-3-values // (exemplo: sudo docker logs --details 5b9) para saber os comandos dos logs somente usar --help
- docker run -d -p 80:3000 --name your-container application:version // Aplicações rodando na porta 3000 agora irá rodar na 80
- docker exec name-container ls // Ver os arquivos presentes no container
- docker stop name-container // Parar o seu container
- docker start name-container //Rodar o seu container
- docker rm name-container //Remover o container (-f force remove forçado mesmo ele rodando)
- docker rmi -f \$(sudo docker images -q) // Apagar todas as imagens existentes

```
- docker rm -f $(sudo docker ps -aq) // Apagar todos os container criados e registros
```

## Imagem & Contêiner

A imagem é a definição estática do ambiente do aplicativo, enquanto o contêiner é a instância em execução dessa imagem. Você pode criar várias instâncias (contêineres) a partir de uma única imagem. Essa abordagem permite a portabilidade e a consistência de ambientes de desenvolvimento, teste e produção, uma vez que o mesmo contêiner pode ser executado em diferentes sistemas operacionais ou ambientes que suportam o Docker.



## Características

### Image

- Cut - down Os
- Libraries
- arquivos
- variaveis
  - A Imagem tem tudo que precisa para executar um programa.

### Containers

- VM isolado

- Start/Stop
  - O container é um processo que roda a imagem

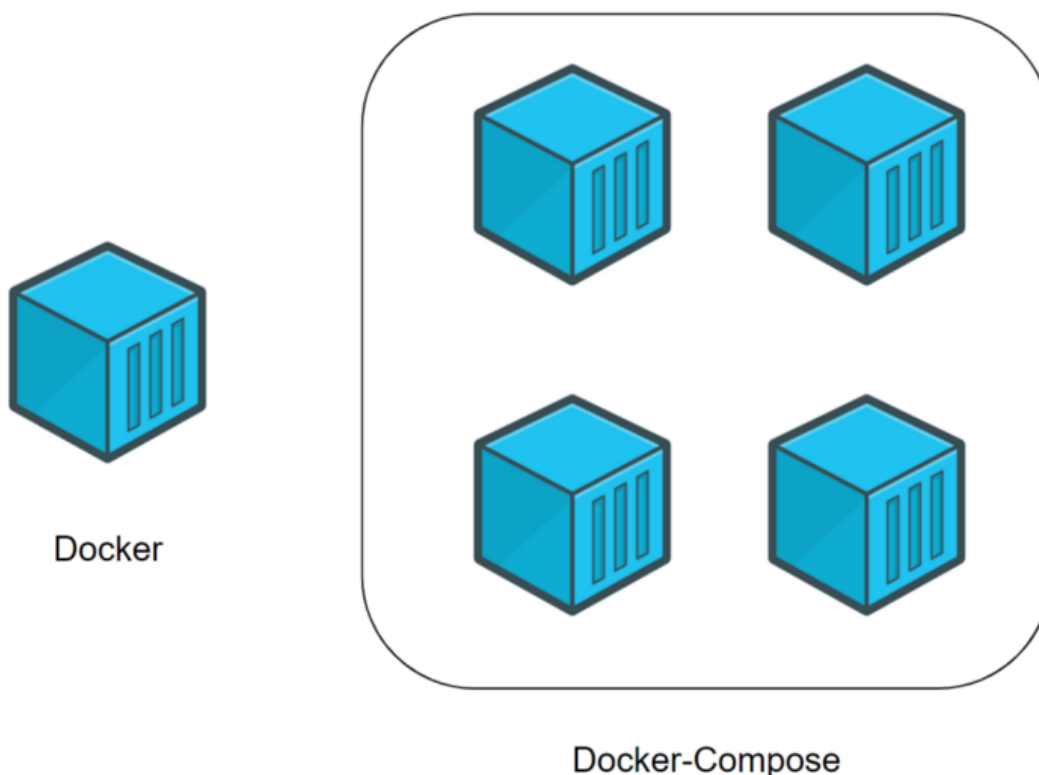
## Docker Compose

O Docker Compose é uma ferramenta que permite definir e gerenciar aplicativos multi-container Docker. Ele utiliza um arquivo YAML para configurar os serviços, redes e volumes necessários para a execução de um ambiente de aplicação distribuída. Com o Docker Compose, você pode definir toda a infraestrutura da sua aplicação, incluindo as dependências, configurações de rede e outros aspectos, em um único arquivo.

O arquivo YAML utilizado pelo Docker Compose é chamado de "docker-compose.yml". Este arquivo contém informações sobre os serviços, volumes e redes que compõem o aplicativo, bem como opções de configuração para cada um deles.

A principal vantagem do Docker Compose é a capacidade de criar, iniciar e parar vários contêineres de uma só vez com uma única instrução. Isso é particularmente útil para aplicações que dependem de vários serviços e precisam ser executadas em ambientes complexos.

Além disso, o Docker Compose simplifica a colaboração entre desenvolvedores, pois todos os membros da equipe podem compartilhar o mesmo arquivo de configuração e replicar facilmente o ambiente de desenvolvimento em seus próprios sistemas.



- `docker-compose up` //realiza a criação dos containers
- `docker-compose up --build` // contrói a imagem e inicia
- `docker-compose up -d` // o "-d" para utilizar o terminal enquanto app roda
- `docker-compose --help` //informações sobre os comandos
- `docker-compose ps` // visualizar os containers rodando
- `docker exec -it -u root id-container sh` //Execute de modo interativo o container para shell (pro,mp de comando)
- `docker-compose logs` // visualizar os logs existentes (--help ver options)

- `docker-compose` utiliza extensão YAML (Linguagem de **Data Serialization**):
  - Utilizado para escrever uma inicialização lógica de cima para baixo;
  - Ele tem uma estrutura de endentação;

```

1  version: "3.8"
2
3  services:
4    frontend: # nome do container
5      depends_on:
6        - backend # dependência do backend para executar
7      build: ./frontend # acesse a pasta e crie de acordo com dockerfile existente
8      ports:
9        - 3000:3000
10
11    backend:
12      depends_on:
13        - db
14      build: ./backend
15      ports:
16        - 3001:3001
17      environment:
18        DB_URL: mongodb://db/vidly
19      command: ./docker-entrypoint.sh
20
21    db:
22      image: mongo:4.0-xenial # utilizando a imagem do docker
23      ports:
24        - 27017:27017
25      volumes:
26        - vidly:/data/db # nome do DB + aonde ele irá armazenar
27
28  volumes:
29    vidly: # associação de volumes

```

## Docker Network

No Docker, uma "rede" é uma abstração que permite que contêineres se comuniquem entre si de maneira eficiente. O Docker fornece várias opções para criar e gerenciar redes, e a `docker network` é um conjunto de comandos que permite a manipulação dessas redes.

Alguns conceitos importantes relacionados à Docker Network incluem:

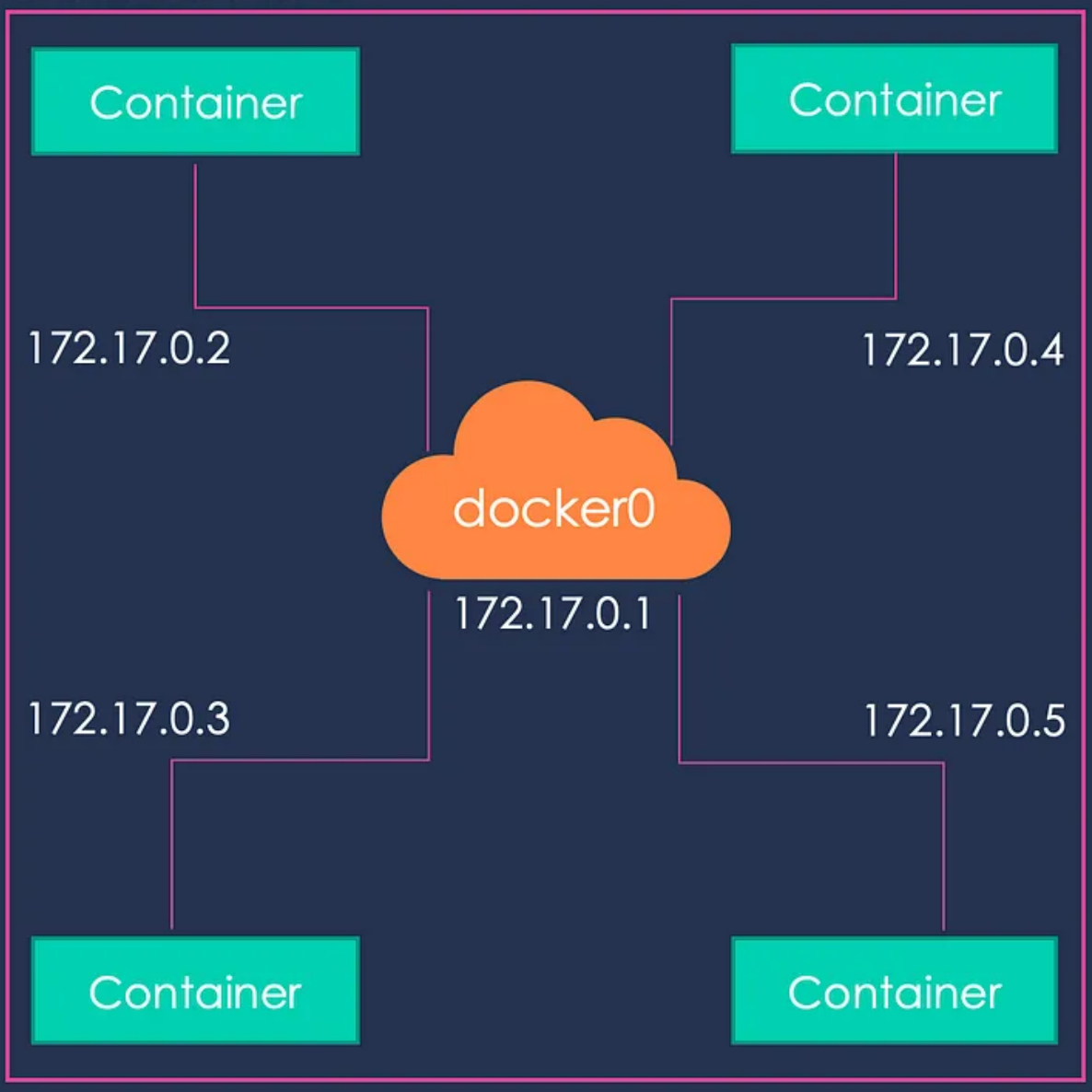
1. **Bridge Network (rede de ponte):** Esta é a rede padrão criada quando você instala o Docker. Contêineres em uma rede de ponte podem se comunicar entre si usando nomes de host, mas geralmente estão isolados do host e de outros contêineres fora da rede.
2. **Host Network (rede de hospedeiro):** Com esta opção, o contêiner compartilha a mesma rede do host. Isso significa que o contêiner pode usar diretamente as interfaces de rede do host, sem isolamento. Pode ser útil para situações em que o desempenho é crítico.
3. **Overlay Network (rede de sobreposição):** Esta é uma rede que permite a comunicação entre contêineres em diferentes hosts. É útil para aplicações distribuídas em escala, onde os contêineres precisam se comunicar entre hosts em um cluster Docker Swarm.
4. **Macvlan Network (rede Macvlan):** Permite que os contêineres tenham um endereço MAC próprio na rede, o que pode ser útil em cenários onde os contêineres precisam ser acessíveis como se fossem máquinas físicas na rede.

Os comandos mais comuns da `docker network` incluem:

```
- docker network ls //Lista as redes disponíveis.
- docker network create //Cria uma nova rede.
- docker network connect //Conecta um contêiner a uma rede.
- docker network disconnect //Desconecta um contêiner de uma rede.
- docker network inspect //Exibe informações detalhadas sobre uma rede.
```

A utilização de redes no Docker é uma parte essencial para criar ambientes mais complexos, especialmente quando há necessidade de comunicação entre contêineres ou quando se trabalha com orquestradores como o **Docker Swarm** ou **Kubernetes**.

# Docker Host





```

luzia-santos@ubuntu:~/Downloads/netflix/netflix$ sudo docker exec -it -u root 2e2e sh
/app # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:04
          inet addr:172.18.0.4  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8920 (8.7 KiB)  TX bytes:2436 (2.3 KiB)


lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:235 (235.0 B)  TX bytes:235 (235.0 B)

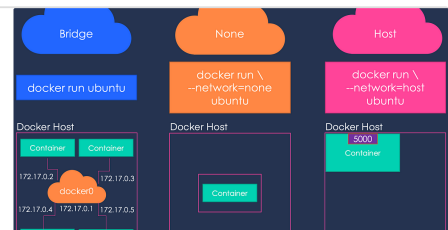
/app # ping backend
PING backend (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.079 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.093 ms
^C
--- backend ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.079/0.086/0.093 ms
/app # ping db
PING db (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.105 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.091 ms
^C
--- db ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.091/0.098/0.105 ms
/app # █

```

### Docker Networking

When you install docker it creates three networks automatically - Bridge, Host, and None. Of which, Bridge is the default network a...

 <https://towardsdatascience.com/docker-networking-919461b7f498>

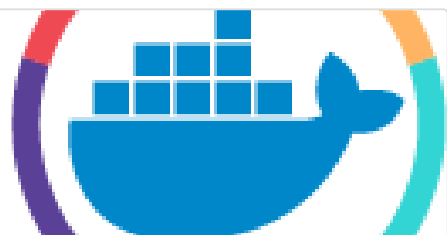


## Versão docker-compose

### Compose file version 3 reference

Find a quick reference for Docker Compose version 3, including Docker Engine compatibility, memory limitations, and more.


 <https://docs.docker.com/compose/compose-file/compose-file-v3/>

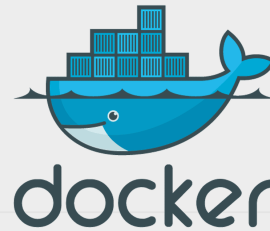


## Material Extra ✨

### Como construir uma aplicação com Docker?


Você conhece as vantagens de usar Docker? Neste artigo você vai aprender Docker na prática! Veja o passo a passo de como criar uma aplicação!

 <https://blog.geekhunter.com.br/docker-na-pratica-como-construir-uma-aplicacao/>



### How To Install and Use Docker Compose on Ubuntu 20.04 | DigitalOcean

Docker Compose is a tool that allows you to run multi-container application environments based on definitions set in a YAML file. It uses service definitions...

 <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04>



Nº do certificado: UC-13bda4d7-858f-4ce2-ab76-cca1462846b8  
URL do certificado: ude.my/UC-13bda4d7-858f-4ce2-ab76-cca1462846b8  
Número de referência: 0004

CERTIFICADO DE CONCLUSÃO

# DOCKER Completo - Do Zero ao Avançado (2024)

Instrutores **Andre Iacono** | 235000+ Alunos

**Luzia Gabriela Abreu da Silva Santos**

Data **23 de Janeiro de 2024**

Duração **5.5 horas no total**