Containers no Linux

Professor: Marcos Brandão
Disciplina: Redes de Computadores / Banco de Dados
Versão: 2026

Conceitos, Exemplos e Prática em Terminal Linux

1. Introdução aos Containers

Containers são ambientes isolados usados para executar aplicações de forma padronizada.

Com containers:

- O programa funciona igual em qualquer computador
- Evita conflitos de bibliotecas
- Facilita deploy
- Facilita testes
- Permite criar ambientes rapidamente

---

2. Máquinas Virtuais x Containers

Máquina Virtual

Uma máquina virtual simula um computador inteiro.

Ela possui:

- Sistema operacional completo
- Kernel próprio
- Alto consumo de memória

Container

O container compartilha o kernel do sistema hospedeiro.

Vantagens:

- Mais leve
- Mais rápido
- Menor consumo de recursos

---

3. O que é Docker

O Docker é a principal plataforma de containers.

Com ele podemos:

- Baixar imagens prontas
- Criar containers
- Publicar aplicações
- Automatizar ambientes

Site oficial:

Docker

Documentação:

Docker Docs

---

4. Instalando Docker no Linux

Atualizar o sistema

sudo apt update
sudo apt upgrade -y

Instalar dependências

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

---

Adicionar chave oficial

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker.gpg

---

Adicionar repositório

```
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Instalar Docker

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io -y
```

Verificar instalação

```
docker --version
```

Exemplo de saída:

```
Docker version 28.0.0
```

Testar Docker

```
sudo docker run hello-world
```

Se aparecer a mensagem “Hello from Docker!”, está funcionando.

---

5. Comandos Básicos

Ver versão

```
docker --version
```

Ver containers ativos

```
docker ps
```

Ver todos containers

docker ps -a

Ver imagens

docker images

Remover container

docker rm NOME_CONTAINER

Remover imagem

docker rmi NOME_IMAGEM

---

6. Trabalhando com Imagens

Imagens são modelos usados para criar containers.

Baixar imagem Ubuntu

docker pull ubuntu

Listar imagens

docker images

Executar Ubuntu

docker run -it ubuntu bash

Parâmetros:

Opção	Significado
-i	modo interativo

-t	terminal
bash	shell do container

Sair do container

exit

---

7. Trabalhando com Containers

Criar container Nginx

docker run -d -p 8080:80 nginx

Explicação

Parte	Função
-d	Executa em segundo plano
-p	Mapeia portas
8080:80	Porta local : porta do container

Testar no navegador

Abra:

http://IP_DO_SERVIDOR:8080

Ver logs

docker logs ID_CONTAINER

Parar container

docker stop ID_CONTAINER

Iniciar novamente

docker start ID_CONTAINER

---

8. Persistência de Dados (Volumes)

Sem volume, os dados somem ao remover o container.

Criar volume

docker volume create meus_dados

Usar volume

```
docker run -d \
-p 8080:80 \
-v meus_dados:/usr/share/nginx/html \
nginx
```

Ver volumes

docker volume ls

---

9. Redes em Containers

Ver redes

docker network ls

Criar rede

docker network create minha_rede

Executar container na rede

docker run -d --network minha_rede nginx

---

10. Docker Compose

Docker Compose permite subir vários containers juntos.

Instalar Compose

sudo apt install docker-compose-plugin -y

Verificar

docker compose version

Exemplo docker-compose.yml

services:

```
 web:
   image: nginx
   ports:
     - "8080:80"

 banco:
   image: mariadb
   environment:
     MYSQL_ROOT_PASSWORD: senha123
```

Executar

docker compose up -d

Derrubar containers

docker compose down

---

11. Criando um Container Web Completo

Estrutura

```
meu_site/
├── docker-compose.yml
└── html/
    └── index.html
```

---

index.html

```
<h1>Servidor Docker Funcionando</h1>
```

---

docker-compose.yml

```
services:

 web:
   image: nginx
   ports:
     - "8080:80"

   volumes:
     - ./html:/usr/share/nginx/html
```

Executar

Dentro da pasta:

```
docker compose up -d
```

Acessar

http://localhost:8080

---

12. Exemplos Práticos

Exemplo 1 — Container Ubuntu

docker run -it ubuntu

Pratique:

apt update
ls
pwd
mkdir teste

Exemplo 2 — Servidor Apache

docker run -d -p 8080:80 httpd

Exemplo 3 — Banco MariaDB

docker run -d \
--name banco \
-e MYSQL_ROOT_PASSWORD=123456 \
-p 3306:3306 \
mariadb

Exemplo 4 — PHP + Apache

docker run -d \
-p 8080:80 \
-v $(pwd):/var/www/html \
php:apache

Crie um arquivo:

nano index.php

Conteúdo:

<?php
phpinfo();
?>

---

13. Exercícios

Exercício 1

Instale o Docker e execute:

docker run hello-world

Exercício 2

Crie um container Ubuntu e:

- Crie pasta
- Crie arquivo
- Liste arquivos

Exercício 3

Suba um servidor Nginx na porta 8080.

Exercício 4

Crie um docker-compose com:

- Nginx
- MariaDB

Exercício 5

Monte um container PHP lendo arquivos locais.

---

14. Boas Práticas

Nomeie containers

docker run --name meu_nginx nginx

Remova containers não usados

docker system prune

Use volumes

Evita perda de dados.

Evite usar root desnecessariamente

Atualize imagens

docker pull nginx

---

15. Comandos Importantes para Revisão

Comando	Função
docker ps	Lista containers
docker images	Lista imagens
docker pull	Baixa imagem
docker run	Executa container
docker stop	Para container
docker start	Inicia container
docker rm	Remove container
docker rmi	Remove imagem

docker logs	Ver logs
docker exec	Entrar no container
docker compose up	Subir ambiente
docker compose down	Derrubar ambiente

Laboratório Completo de Prática

Passo 1 — Criar pasta

mkdir laboratorio
cd laboratorio

Passo 2 — Criar HTML

mkdir html
nano html/index.html

Conteúdo:

<h1>Laboratório Docker</h1>

Passo 3 — Criar compose

nano docker-compose.yml

Conteúdo:

services:

```
 web:
   image: nginx
   ports:
     - "8080:80"

   volumes:
     - ./html:/usr/share/nginx/html
```

Passo 4 — Subir ambiente

docker compose up -d

Passo 5 — Testar

Abra:

http://localhost:8080

Conclusão

Containers revolucionaram a forma de desenvolver e publicar sistemas.

Com Docker você pode:

- Criar laboratórios
- Simular servidores
- Hospedar aplicações
- Aprender Linux
- Criar ambientes de testes rapidamente