

Disciplina de: Computação Distribuída

Prof. Dr. Carlos Alberto

E-mail: *Carlos.Silva@ufms.br*

Parte 1: Os conceitos de containers e Docker

- Container
- Docker®
- Arquitetura Docker®
- Plataforma Docker®

Parte 2: Conhecer os provedores de serviços na nuvem

- Container como um Serviço (CaaS)
- Provedores de CaaS
- Testar uma plataforma Docker®
- Testar uma plataforma Kubernetes®
- PodMan®
- E-books sobre Docker®

Container

Virtualização por *Container*

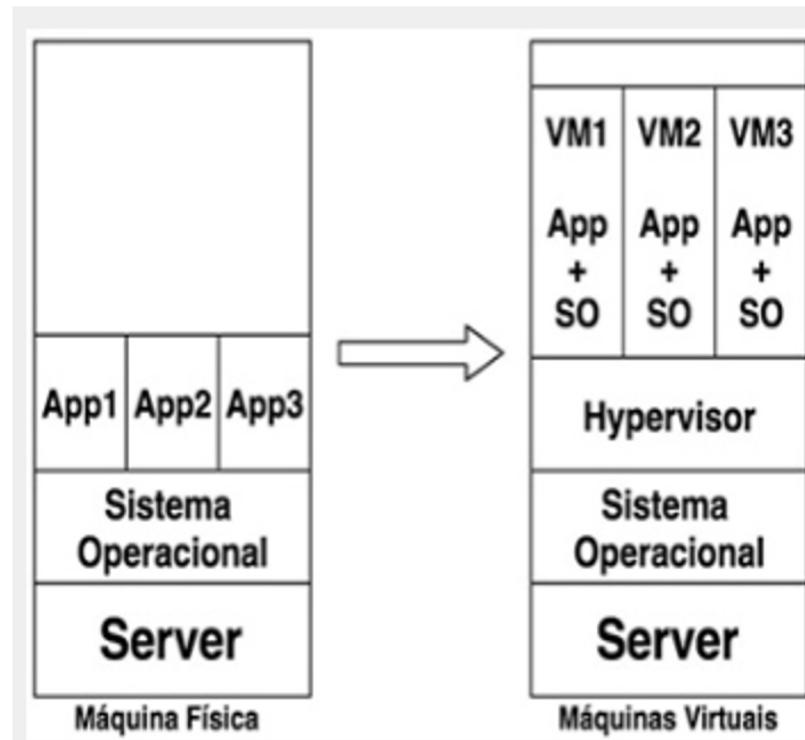
- É uma tecnologia que permite a criação de múltiplas instâncias isoladas de um determinado sistema operacional dentro de um único hospedeiro;
- É uma maneira de virtualizar aplicações dentro de um servidor.

Virtualização por *Container*

Container usa uma quantidade muito menor de memória e espaço em disco que as VMs.

⇒ *Container* utiliza os recursos de seu próprio SO (*host*).

Virtualização por Container



Fonte: Vitalino, 2016, p. 13 (plataforma de leitura).

Vantagens da virtualização por Container

- Todo pode ser instalado em *Containers* no servidor;
- Usam bibliotecas do *kernel* do SO:
 - O ambiente torna-se portável;
 - Torna o trabalho conjunto mais eficiente.

Vantagens da virtualização por *Container*

- Simplifica a metodologia *DevOps* facilitando o desenvolvimento ágil;
- Permite o escalonamento de aplicações;
- Torna processos de implantação e configuração mais simples.

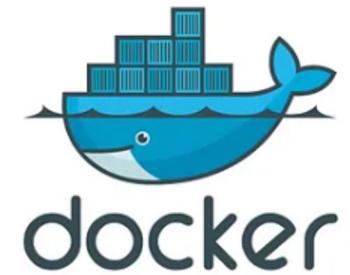
Vantagens da virtualização por *Container*

- Facilita o gerenciamento do ambiente:
 - Economia de recursos;
 - Aplicação como pacote completo;
 - Replicação e padronização.

Vantagens da virtualização por *Container*

- No compartilhamento de recursos:
 - Os arquivos podem ser compartilhados entre o *host* e o *Container*;
 - Ou um volume pode ser distribuído para outros.

Docker®



Fonte: www.docker.com

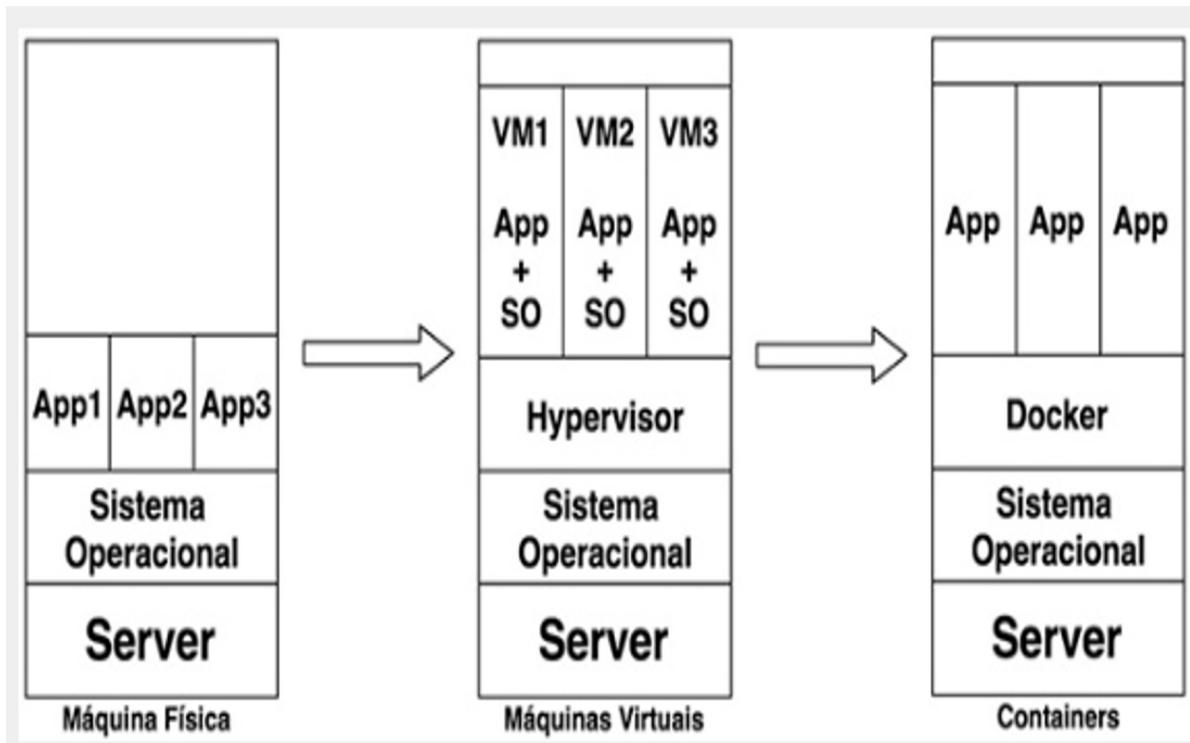
- Tecnologia *Open Source* permite dentro de um *container* de software as aplicações distribuídas para:
 - Desenvolver;
 - Testar;
 - Executar.

⇒ Empacotar uma aplicação de forma padronizada contendo: código, *runtime*, ferramentas, bibliotecas, etc.

Vantagens

- Possibilita a implantação de aplicações de forma mais ágil, confiável e estável, em qualquer ambiente.
- Controle mais granular dos recursos gerando mais eficiência da infraestrutura.

Virtualização por Container

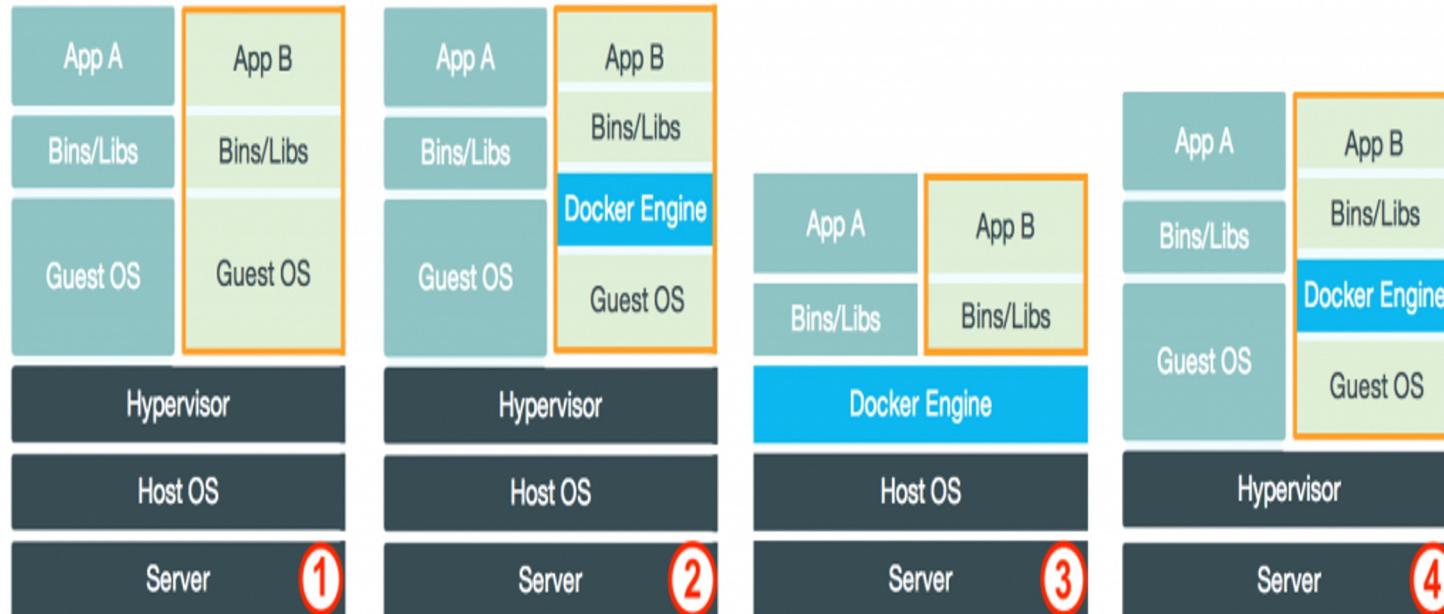


Fonte: Neto, 2016, p. 13 (plataforma de leitura).

Virtualização por Container

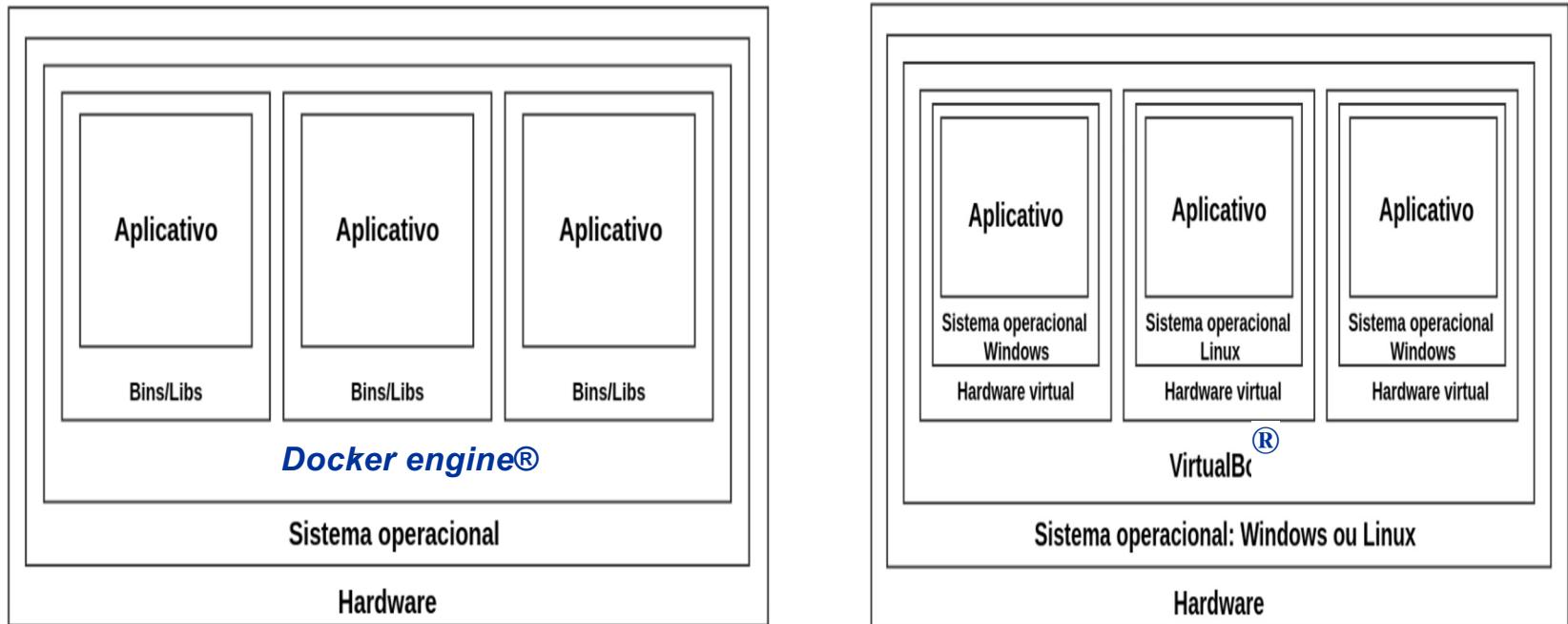
(1) Tradicional

(2-3-4) Docker®



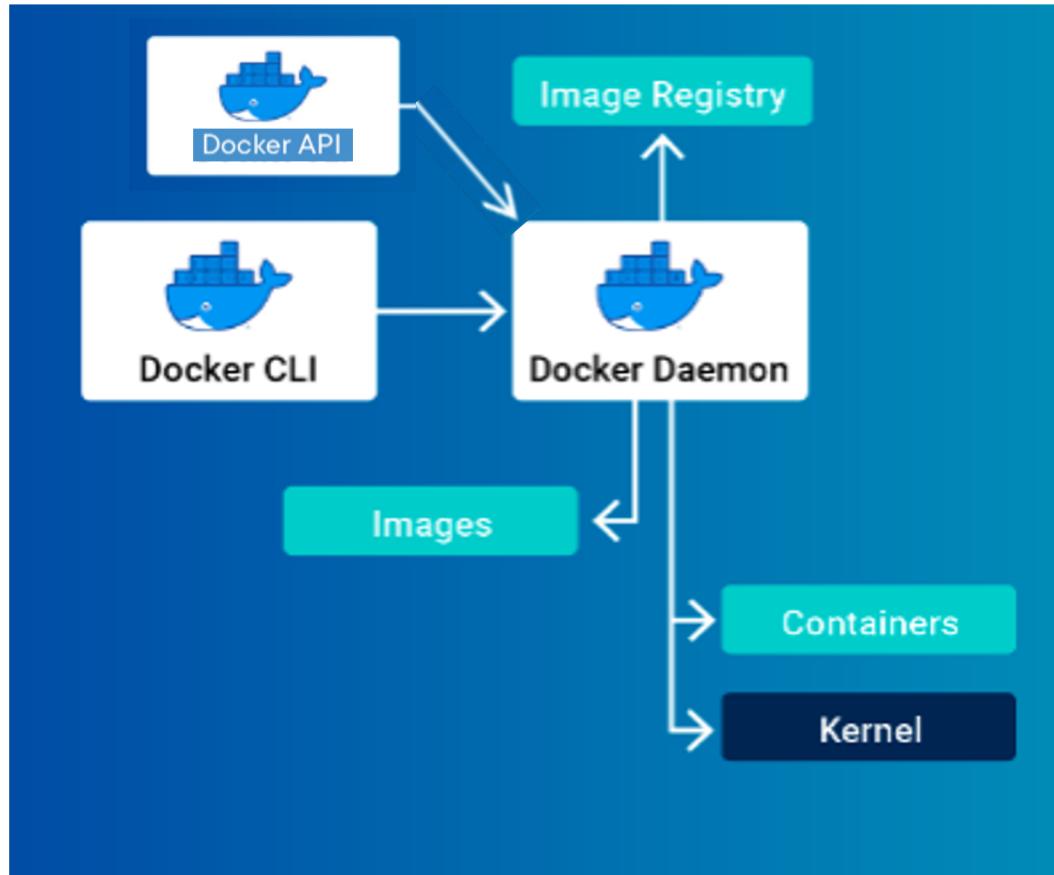
Montagem: Carlos Alberto da Silva, 2023.

Virtualização por Container



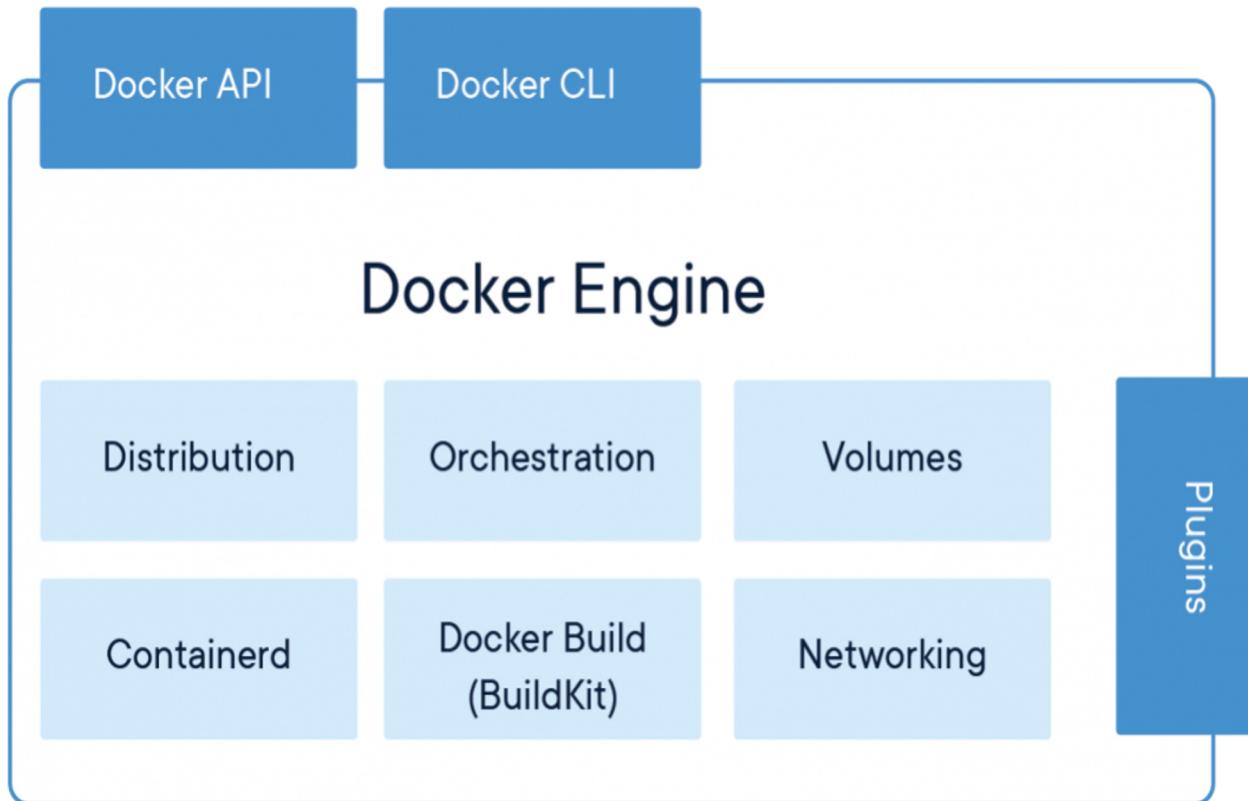
Fonte: Viatlino, 2016, p. 15 (plataforma de leitura).

Visão geral



Fonte: www.docker.com.

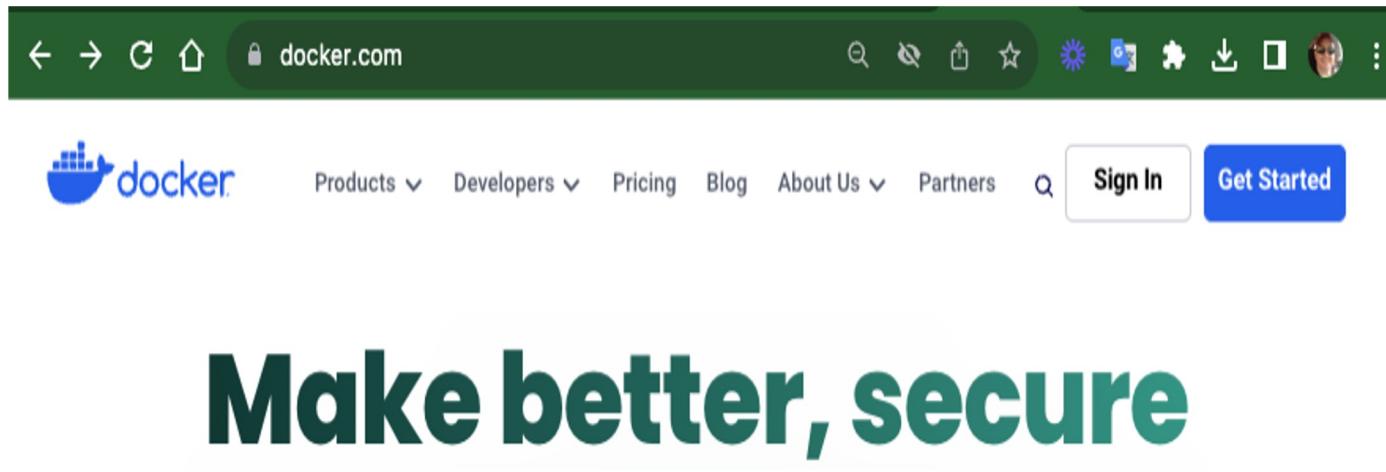
Visão geral



Fonte: www.docker.com.

Docs (manuais)

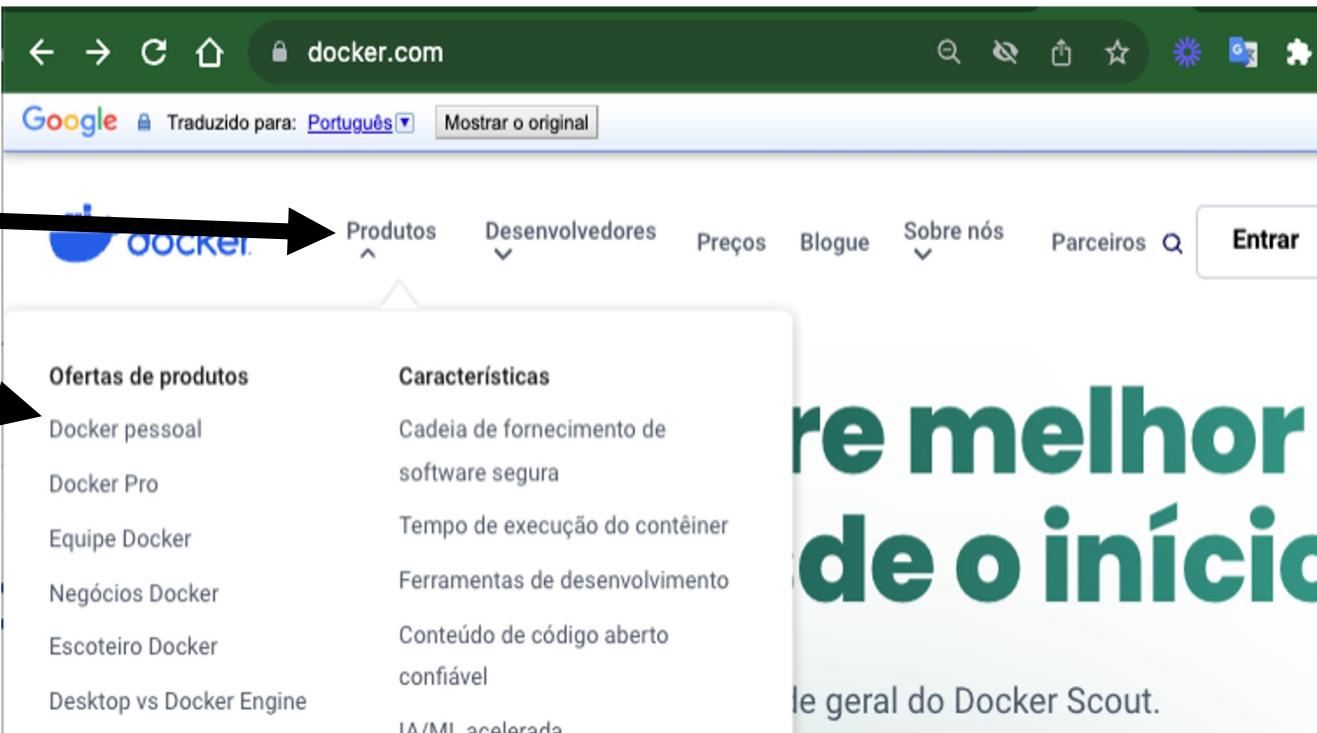
Site www.Docker.com



Fonte: www.docker.com

Docs (manual)

Site www.Docker.com (site traduzido)



1º 

2º 

Ofertas de produtos

- Docker pessoal
- Docker Pro
- Equipe Docker
- Negócios Docker
- Escoteiro Docker
- Desktop vs Docker Engine

Características

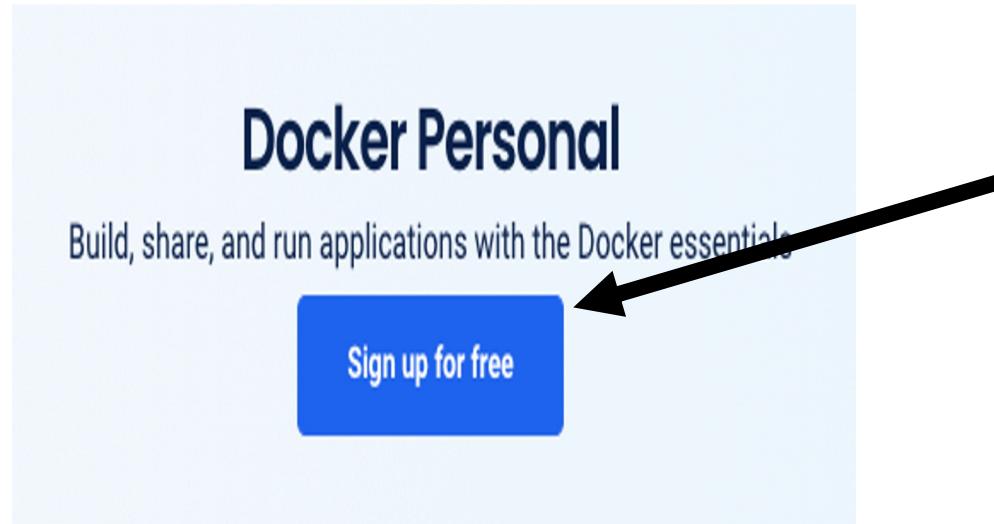
- Cadeia de fornecimento de software segura
- Tempo de execução do contêiner
- Ferramentas de desenvolvimento
- Conteúdo de código aberto confiável
- IA/ML acelerada

Fonte: www.docker.com

Docs (manual)

Versão

Docker Personal®



Fonte: <https://www.docker.com/products/personal/>

Docs (manual)

Versão
Docker Personal®

Criar uma conta
para acesso



Create your account

 Continue with Google

 Continue with GitHub

OR

Email

Username

Password 

Send me occasional product updates and announcements.

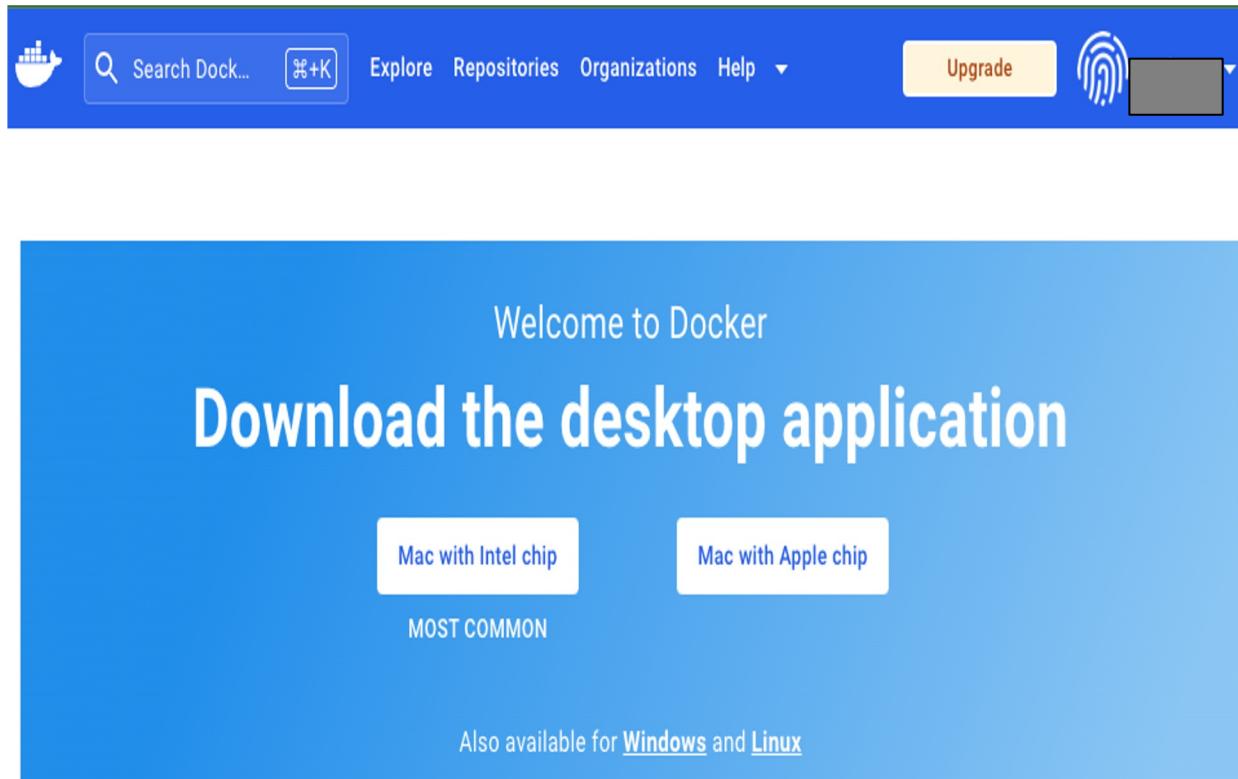
This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.

[Sign up](#)

Fonte: <https://hub.docker.com/signup?redirectTo=/subscription%3Fplan%3Dfree>

Docs (manual)

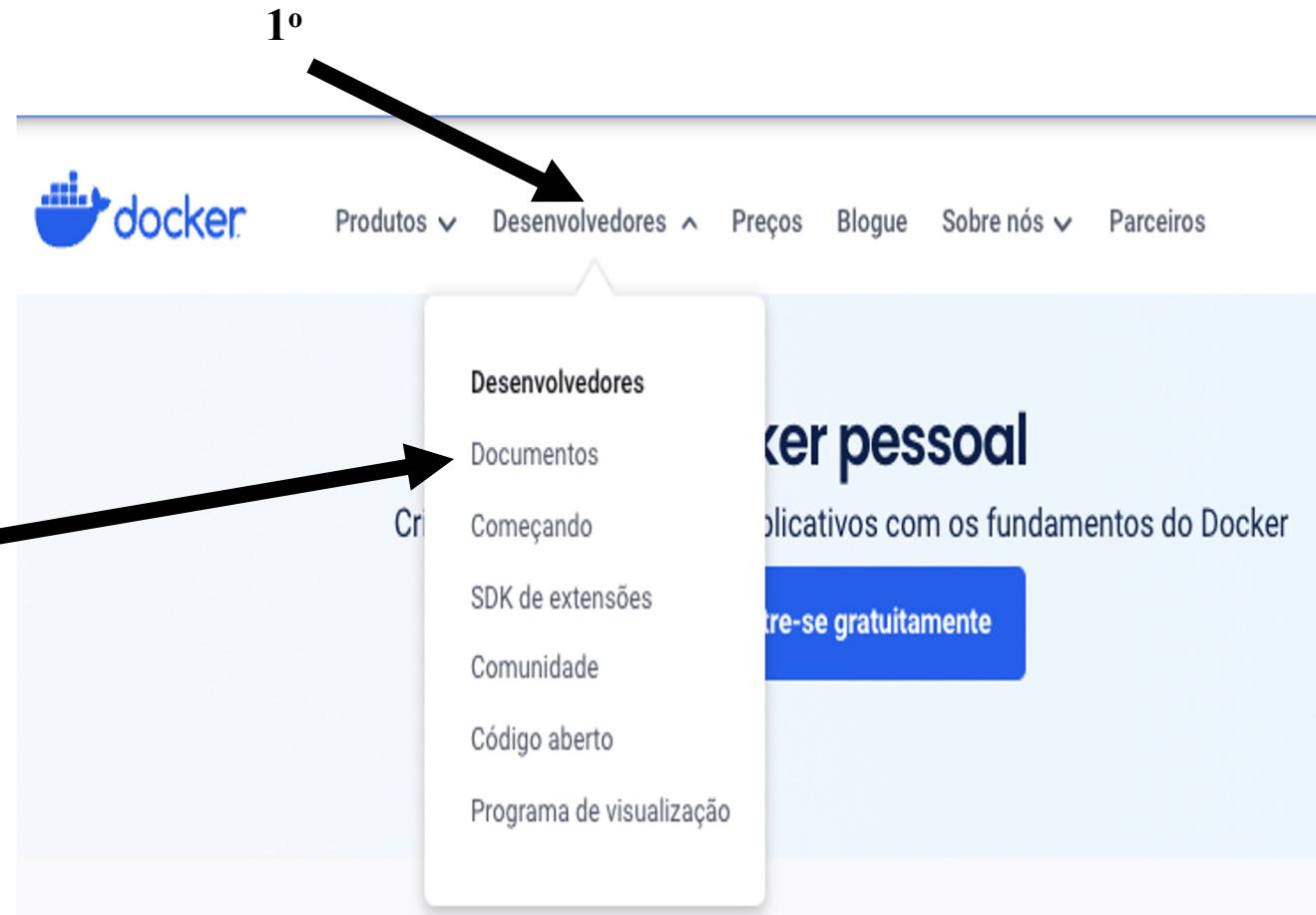
Acesso ao **Docker Personal**®



Fonte: <https://hub.docker.com/>

Docs (manual)

Acesso
Documentos



The screenshot shows the Docker website's header with a navigation bar. A large black arrow labeled "1º" points from the "Acesso Documentos" text to the "Desenvolvedores" menu item in the header. Another large black arrow labeled "2º" points from the "Cri" text to the "Documentos" link within the expanded "Desenvolvedores" dropdown menu. The menu also includes links for "Começando", "SDK de extensões", "Comunidade", "Código aberto", and "Programa de visualização".

1º

2º

Acesso
Documentos

Produtos ▼ Desenvolvedores ▲ Preços Blogue Sobre nós ▼ Parceiros

Desenvolvedores

- Documentos
- Começando
- SDK de extensões
- Comunidade
- Código aberto
- Programa de visualização

ker pessoal

olicativos com os fundamentos do Docker

tre-se gratuitamente

Fonte: <https://www.docker.com/products/personal/>

Docs (manual)

Acesso Documentos

O que podemos ajudá-lo a encontrar?

 **iniciar**
Aprenda os conceitos básicos do Docker e os benefícios da conteinerização de seus aplicativos.

 **Guias**
Aprenda como configurar seu ambiente Docker e começar a conteinerizar seus aplicativos.

 **Manuais**
Navegue pelos manuais e aprenda como usar os produtos Docker.

 **Baixar e instalar**
Baixe e instale o Docker em sua máquina em algumas etapas fáceis.

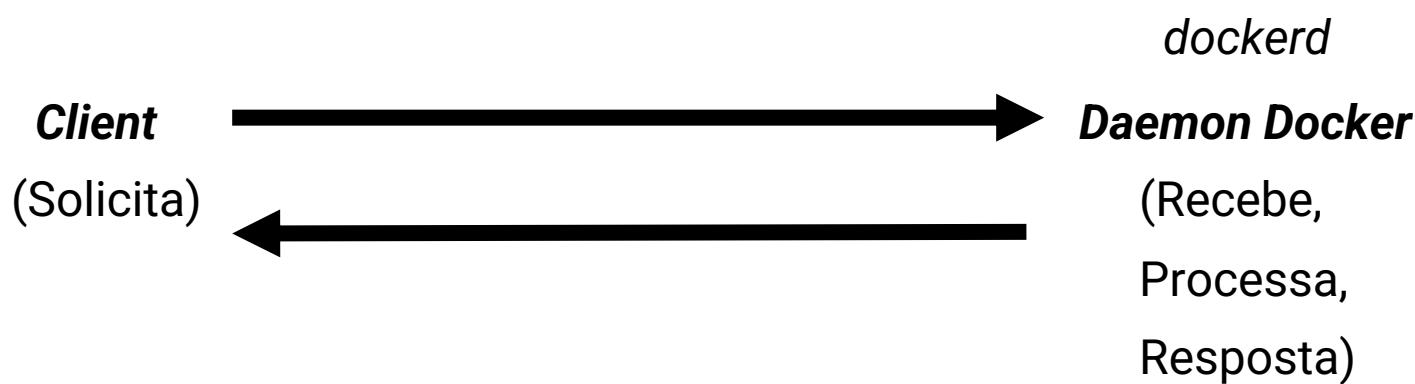
 **Guias específicos de idioma**
Aprenda como usar o Docker com sua linguagem de programação favorita.

 **Referência**
Navegue pela documentação de referência da CLI e da API.

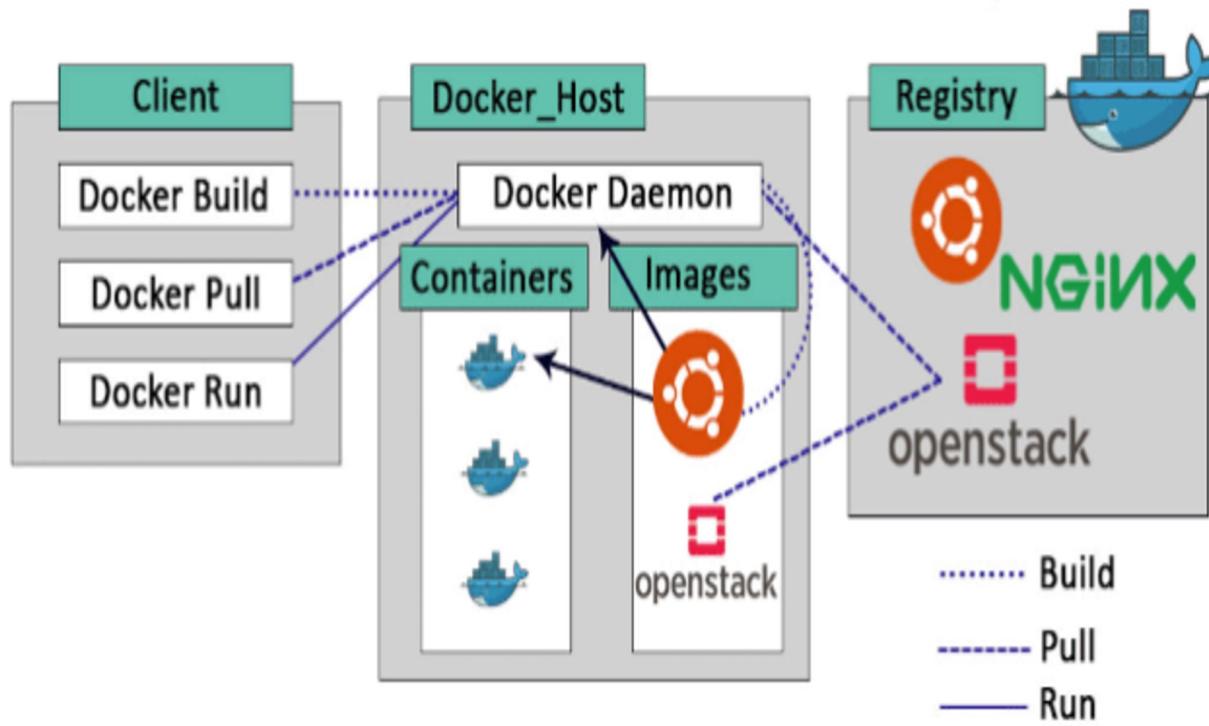
Fonte: <https://docs.docker.com/>

Arquitetura Docker®

Arquitetura cliente-servidor



Client executa comandos



Fonte: <https://docs.docker.com/>

Client docker e daemon docker se comunicam por meio de:

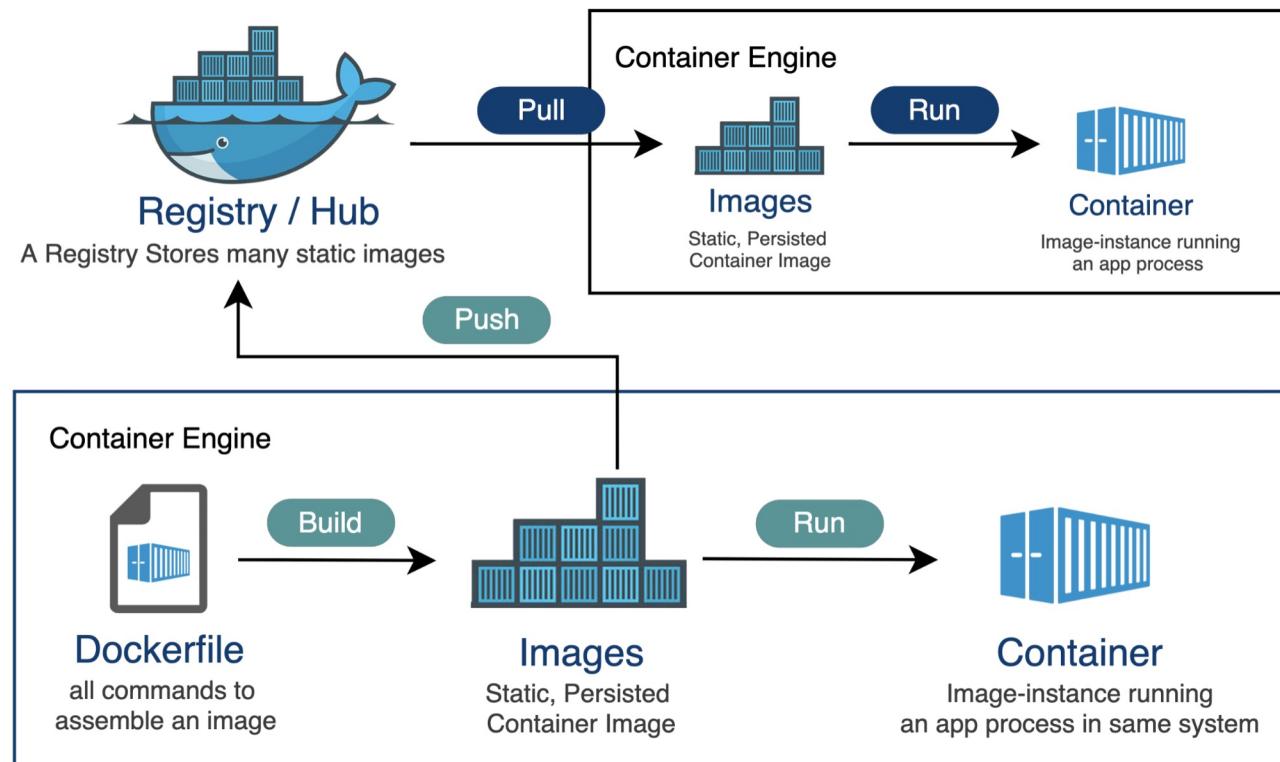
- *REST API*;
- *Sockets*;
- Outros.

Componentes Docker®

- *Container engine*®
- *Container docker*®
- *Image docker*®
- *Dockerfile*®
- *Docker Registry*®
- *Docker Hub*®

Componentes Docker®

Container engine®



Fonte: www.docker.com.

Componentes Docker®

Container docker®

- Como base no ***image docker:***
 - ***Image*** é uma classe java;
 - ***Container*** é uma instância desta classe.

Componentes Docker®

Image docker

- **Templates** para criação de **container**;
- São imutáveis, pois para executá-las é necessário criar um instância da mesma (**container**).

Componentes Docker®

Image docker

- São construídas em camadas:
 - Facilita a reutilização e manutenção.

⇒ Ambiente encapsulado pronto para ser replicado.

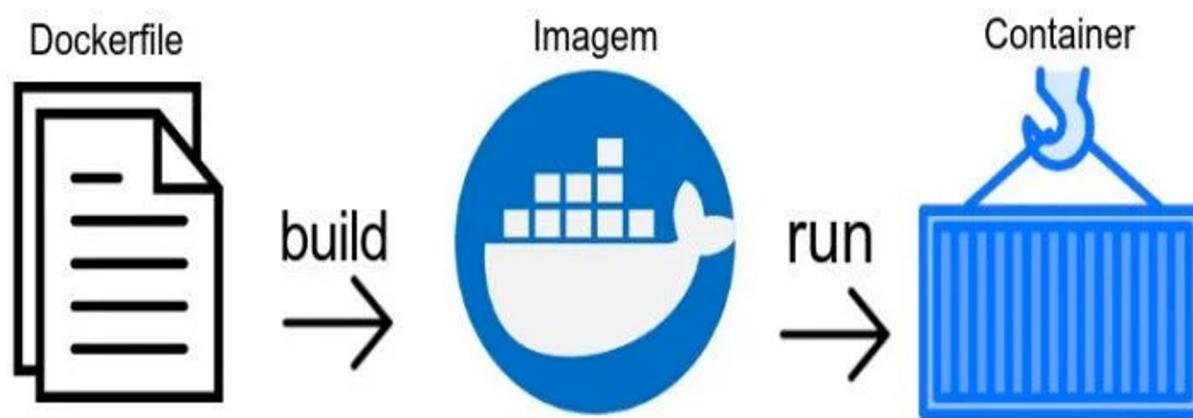
Componentes Docker®

Dockerfile®

- São **scripts** com comandos para:
 - Criar imagens;
 - Executar comandos;
 - Criar variáveis de ambiente;
 - Copiar arquivos e pastas do *host* para a imagem.

Componentes Docker®

Dockerfile®



Fonte: www.docker.com.

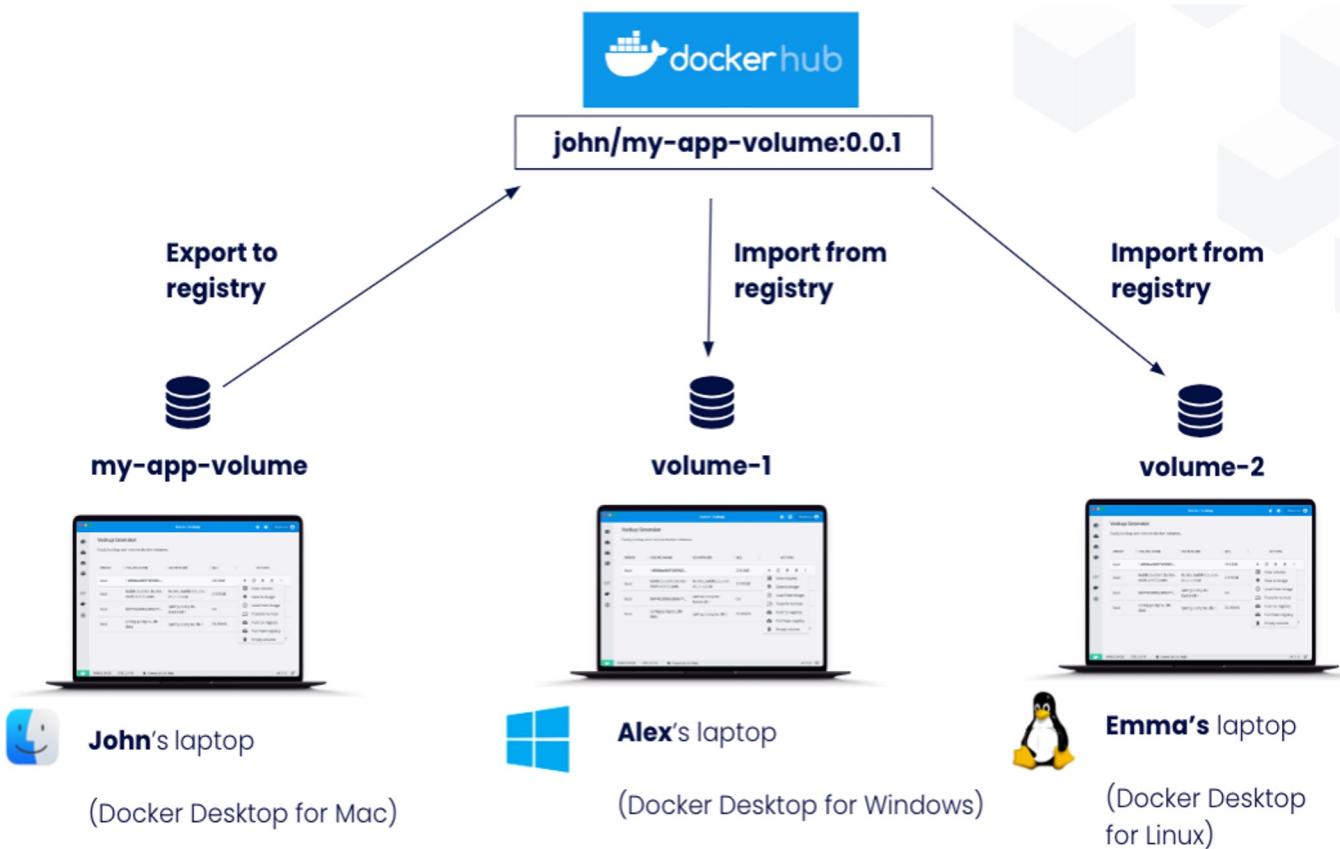
Componentes Docker®

Docker Registry®

- Um repositório **Git**:
 - Imagens podem ser versionadas.
- **Docker Hub®** é o repositório oficial do **Docker®**:
 - Permite versionar e hospedar imagens privadas e públicas.

Componentes do Docker®

Docker Hub®



Fonte: www.docker.com.

Plataforma Docker®

Plataforma composta por:

Docker Engine® + Docker Hub®

Docker Engine®

Docker Daemon® + (Docker CLI® e/ou Docker API®)

Docker Desktop®

Disponível para
os sistemas
operacionais

(seu Docker engine)



Docker Desktop for Mac

A native application using the macOS sandbox security model that delivers all Docker tools to your Mac.



Docker Desktop for Windows

A native Windows application that delivers all Docker tools to your Windows computer.



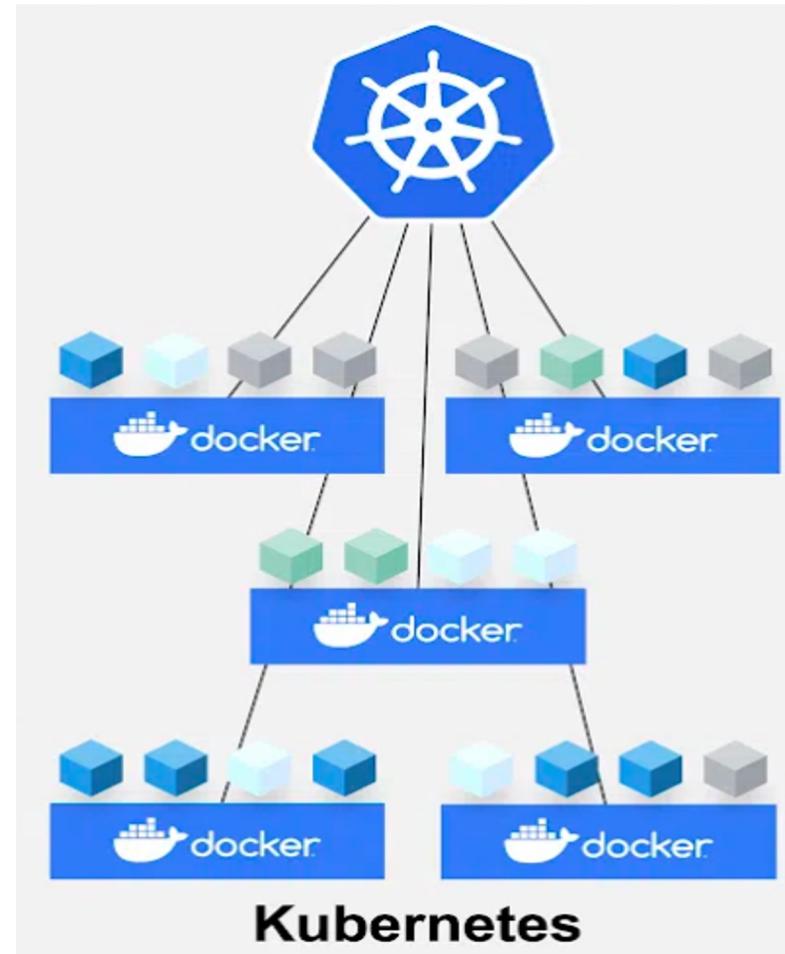
Docker Desktop for Linux

A native Linux application that delivers all Docker tools to your Linux computer.

Fonte: <https://docs.docker.com/get-docker/>

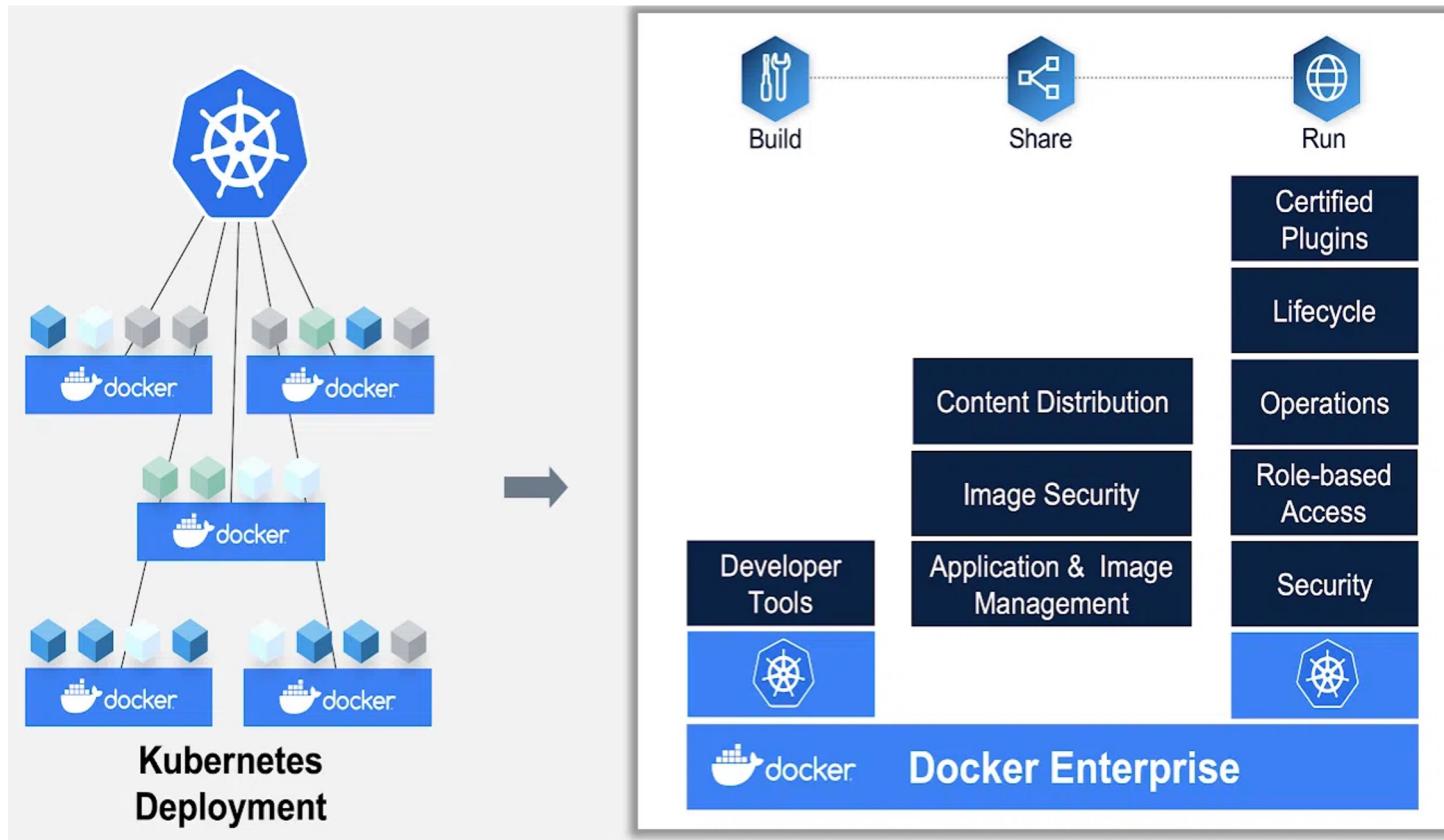
Kubernetes®

**Gerenciamento de
vários *containers***
Docker®



Fonte: <https://docs.docker.com/>

Entrepriese Docker®



Fonte: <https://docs.docker.com/>

Container como um Serviço (CaaS)

CaaS

- Deve ser uma plataforma de hospedagem baseada em nuvem, perfeita para gerenciar seus *containers Docker®*.
- Deve oferecer escalabilidade e confiabilidade robustas para qualquer tamanho de projeto.

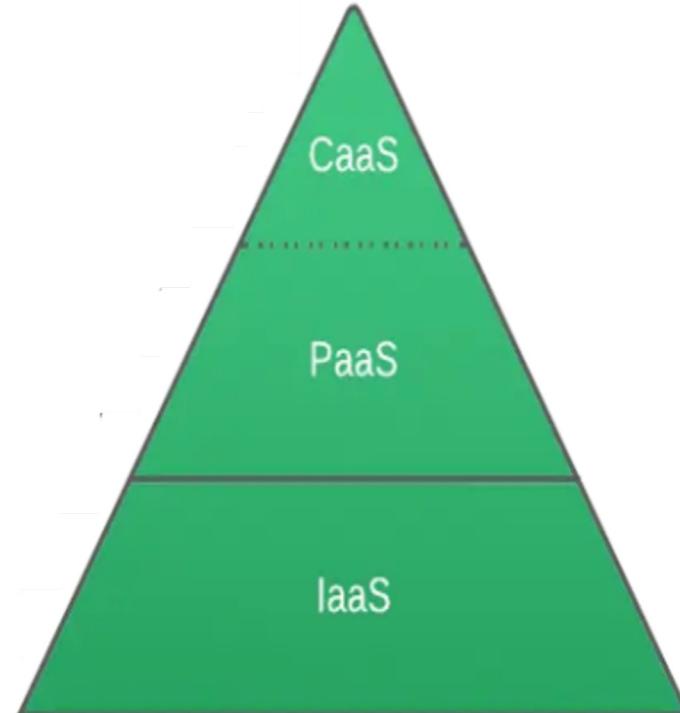
CaaS

Com recursos avançados de gerenciamento como:

- implantação automatizada;
- funcionalidade de auto recuperação;
- e opções de escalonamento personalizadas.

CaaS

Utiliza os IaaS e PaaS do mesmo provedor



Montagem: Carlos Alberto da Silva, 2023.

Provedores de CaaS

Você pode escolher entre uma variedade de sistemas operacionais e configurações de hardware:

- *Amazon Web Services (AWS)*: a AWS oferece o serviço EC2 (*Elastic Compute Cloud*)
- *Google Cloud*
- *Microsoft Azure*
- *IBM Cloud*
- *Oracle Cloud*

Como acessar:

- *Amazon Web Services (AWS)*: <https://aws.amazon.com/pt>
- *Google Cloud*: <https://cloud.google.com/>
- *Microsoft Azure*: <https://azure.microsoft.com/pt-br/>
- *IBM Cloud*: <https://cloud.ibm.com/login>
- *Oracle Cloud*: <https://www.oracle.com/br/cloud/>

Símbolos:



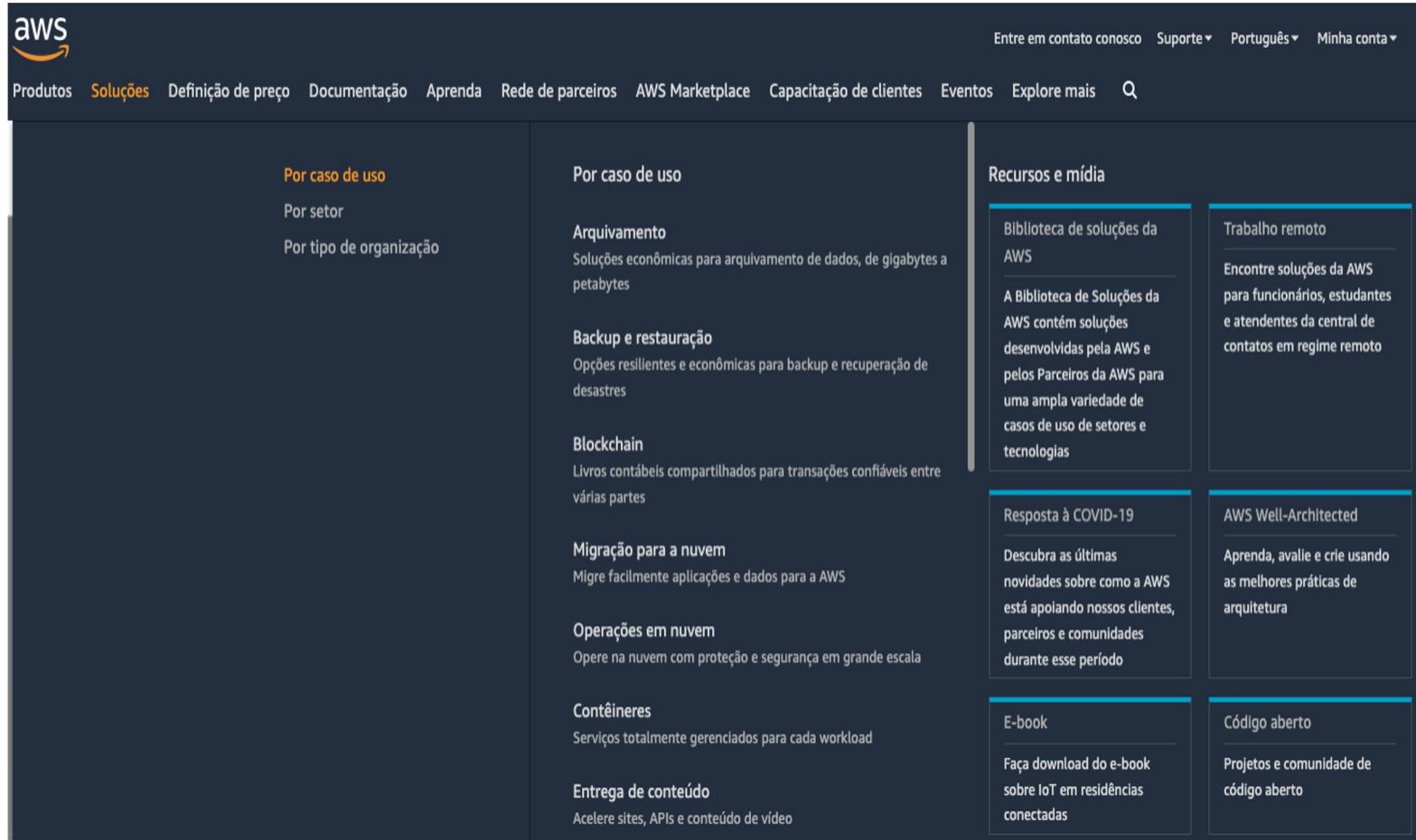
Google Cloud



IBM **Cloud**



Montagem: Carlos Alberto da Silva, 2023.



The screenshot shows the AWS homepage with a dark background. At the top, there's a navigation bar with links for 'Produtos', 'Soluções' (highlighted in orange), 'Definição de preço', 'Documentação', 'Aprenda', 'Rede de parceiros', 'AWS Marketplace', 'Capacitação de clientes', 'Eventos', 'Explore mais', and a search icon. Below the navigation, there are three main columns. The left column has sections for 'Por caso de uso', 'Por setor', and 'Por tipo de organização'. The middle column lists services like 'Arquivamento' (Archiving), 'Backup e restauração' (Backup and Recovery), 'Blockchain', 'Migração para a nuvem' (Migration to the Cloud), 'Operações em nuvem' (Cloud Operations), 'Contêineres' (Containers), and 'Entrega de conteúdo' (Content Delivery). The right column is titled 'Recursos e mídia' (Resources and Media) and contains boxes for 'Biblioteca de soluções da AWS' (AWS Solution Library), 'Trabalho remoto' (Remote Work), 'Resposta à COVID-19' (COVID-19 Response), 'AWS Well-Architected', 'E-book', and 'Código aberto' (Open Source).

Fonte: Amazon AWS, 2023.

Como tratar este serviço de CaaS

Passos para contratar um serviço na nuvem (CaaS):

- 1) Escolher o provedor;
- 2) Criar uma conta de acesso;
- 3) Registar um cartão de crédito na sua conta;
- 4) Escolher o serviço e finalizar o contrato.

Testar uma plataforma *Docker*®

Comandos para	Gerenciamento de
<i>builder</i>	<i>builds</i>
<i>config</i>	<i>Docker configs</i>
<i>container</i>	<i>contêineres</i>
<i>context</i>	<i>contextos</i>
<i>engine</i>	<i>docker engine</i>
<i>network</i>	<i>networks</i>
<i>node</i>	<i>Swarm nodes</i>

<i>plugin</i>	<i>plugins</i>
<i>secret</i>	<i>Docker secrets</i>
<i>service</i>	<i>serviços</i>
<i>stack</i>	<i>Docker stacks</i>
<i>swarm</i>	<i>swarm</i>
<i>system</i>	<i>system</i>
<i>trust</i>	<i>trust on Docker images</i>
<i>volume</i>	<i>volumes</i>

Fonte: Docker, 2023.

Sintaxe do comando em Docker®

Docker (*comando global*) (*comando específico*) (*parâmetro/s*)

⇒ Por exemplo: Listar todos os *Containers* ativos e inativos

docker container ls -a

Comandos gerais

- *docker version* (no cliente ou servidor);
- *docker login* (no Docker Hub®);
- *docker logout* (no Docker Hub®);
- *docker system prune* - eliminar os *containers*:
 - parados (*stopped*);
 - de redes inativas;
 - de imagens de teste.

Comandos para *containers*

- *create* – cria novo *container* baseado em uma imagem;
- *start* – inicializa um *container* existente;
- *run* – cria um *container* e o inicia;
- *ls* – lista todos os *containers* ativos;
- *inspect* – fornece informações mais completas sobre um *container*;
- *logs* – lista os *logs* de um *container* específico.

Comandos para *containers*

- *stop* – interrompe um *container* ativo.
- *rm* – remove um *container* parado.

⇒ Por exemplo: Inspecionar um *container* com:

docker container inspect (nome do container)

Comandos para *Imagen*

- *build* – cria uma *imagem* baseada em um *Dockerfile*®;
- *push* – faz *upload* de uma *imagem* para o repositório;
- *tag* – permite atribuir uma *tag* à *imagem*;
- *ls* – lista todas as *imagens* baixadas atualmente;
- *inspect* – obtém mais informações de uma *imagem*;
- *rm* – remove uma *imagem* do sistema.

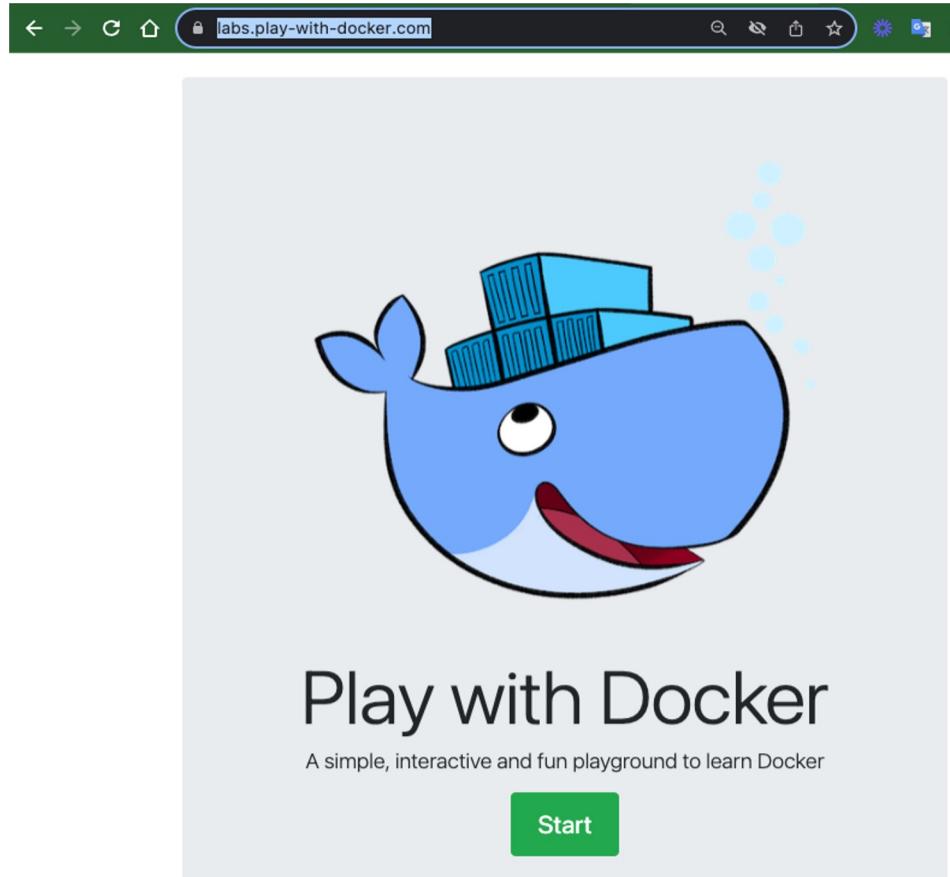
⇒ Por exemplo: ***docker rm (nome da imagem)***

Comandos para redes e volumes

- *ls* – lista todas as redes ou volumes
- *rm* – remove uma rede ou volume
- *create* – cria nova rede ou volume
- *inspect* – fornece informações mais completas sobre uma rede ou volume
- *prune* - eliminar uma rede ou volume

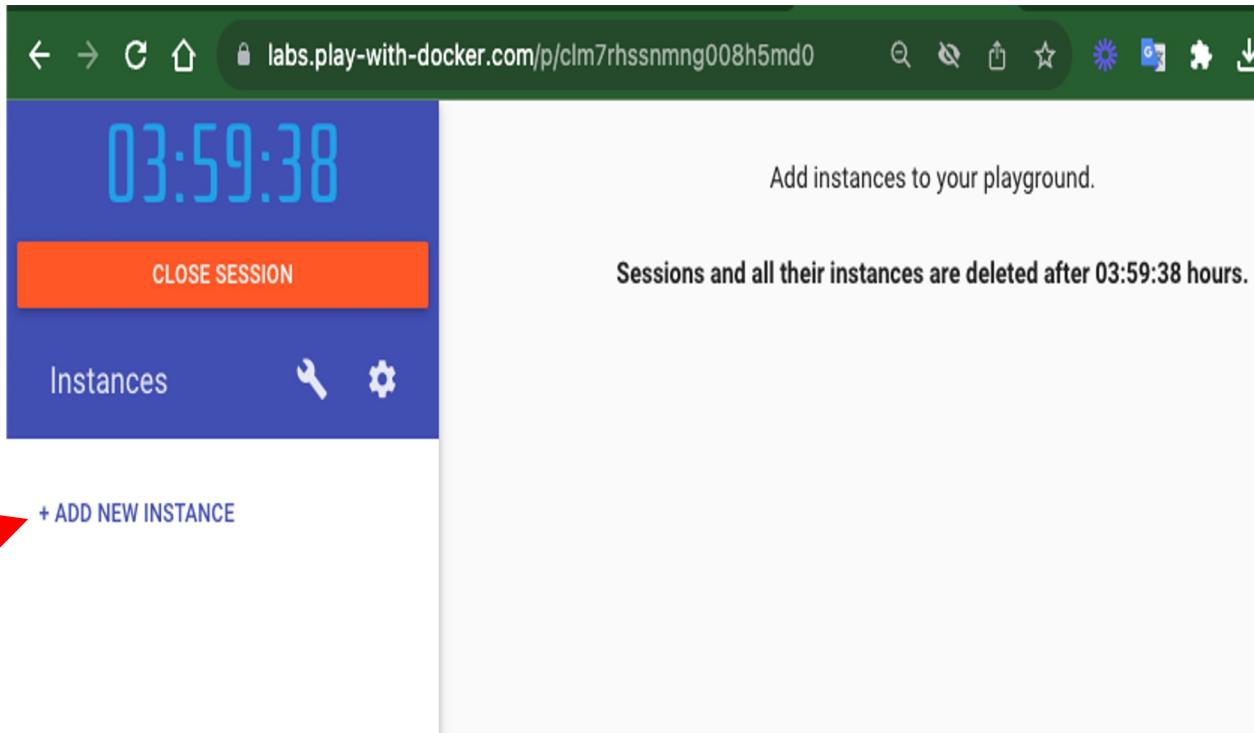
⇒ Por exemplo: ***docker network create*** (*nome da rede*)

Site <https://labs.play-with-docker.com/>



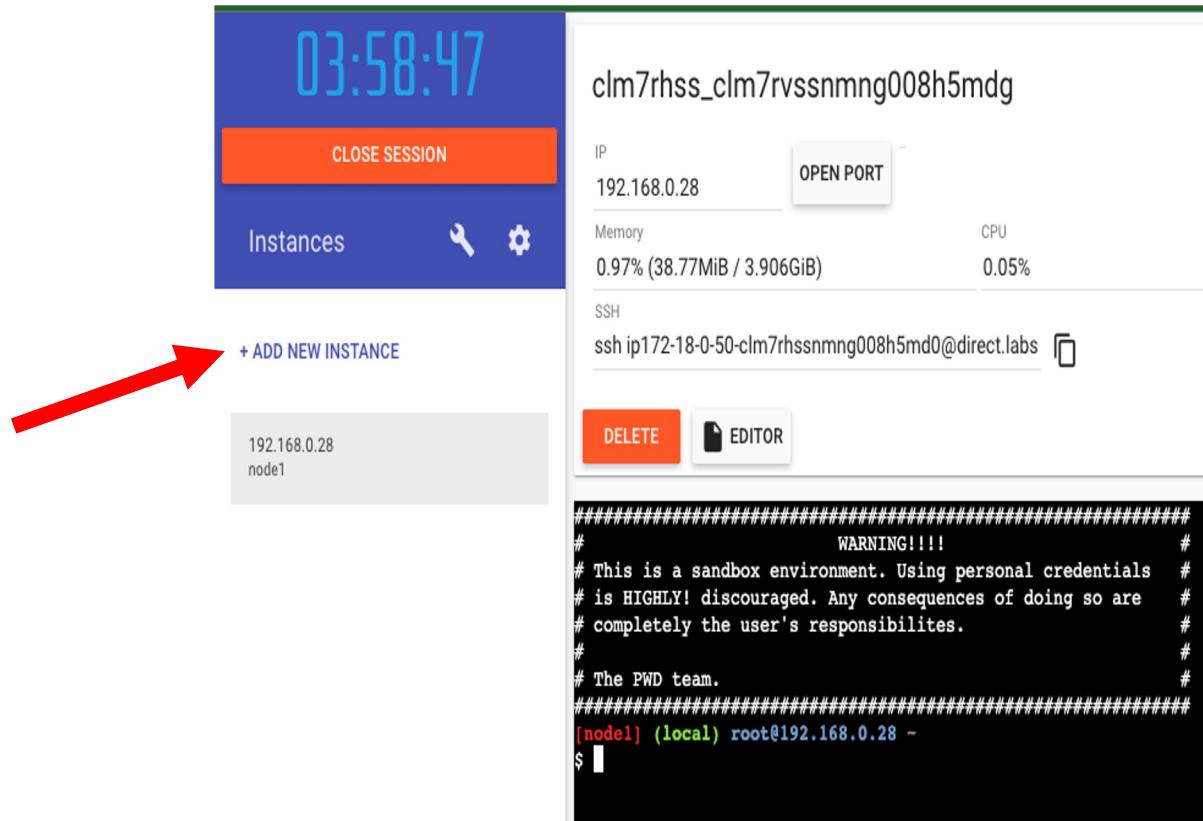
Fonte: [Labs.play-with-docker](https://labs.play-with-docker.com/), 2023.

Site <https://labs.play-with-docker.com/>



Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>



The screenshot shows a web-based Docker management interface. At the top left is a digital clock displaying 03:58:47. Below it are buttons for 'CLOSE SESSION' (orange) and 'Instances' (blue). A red arrow points to the '+ ADD NEW INSTANCE' button. On the right, a session window for container 'clm7rhss_clm7rvssnmng008h5mdg' is open. It shows the IP 192.168.0.28, an 'OPEN PORT' button, memory usage (0.97% / 38.77MiB / 3.906GiB), and CPU usage (0.05%). Below this is an SSH section with the command 'ssh ip172-18-0-50-clm7rhssnmng008h5md0@direct.labs'. At the bottom of the session window is a terminal window with the following text:

```
#####
#          WARNING!!!!
#
# This is a sandbox environment. Using personal credentials
# is HIGHLY! discouraged. Any consequences of doing so are
# completely the user's responsibilites.
#
# The PWD team.
#####
[node1] (local) root@192.168.0.28 ~
$
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digite o comando:

docker version

```
[root@11 1] ~ % docker version
$ docker version
Client:
  Version:          24.0.7
  API version:      1.43
  Go version:       go1.20.10
  Git commit:       afdc53b
  Built:            Thu Oct 26 09:04:00 2023
  OS/Arch:          linux/amd64
  Context:          default

Server: Docker Engine - Community
Engine:
  Version:          24.0.7
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.10
  Git commit:       311b9ff
  Built:            Thu Oct 26 09:05:28 2023
  OS/Arch:          linux/amd64
  Experimental:    true
containerd:
  Version:          v1.7.6
  GitCommit:        091922f03c2762540fd057fba91260237ff86acb
runc:
  Version:          1.1.9
  GitCommit:        v1.1.9-0-gccaecfc
docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
[root@11 1] ~ %
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digite o comando:

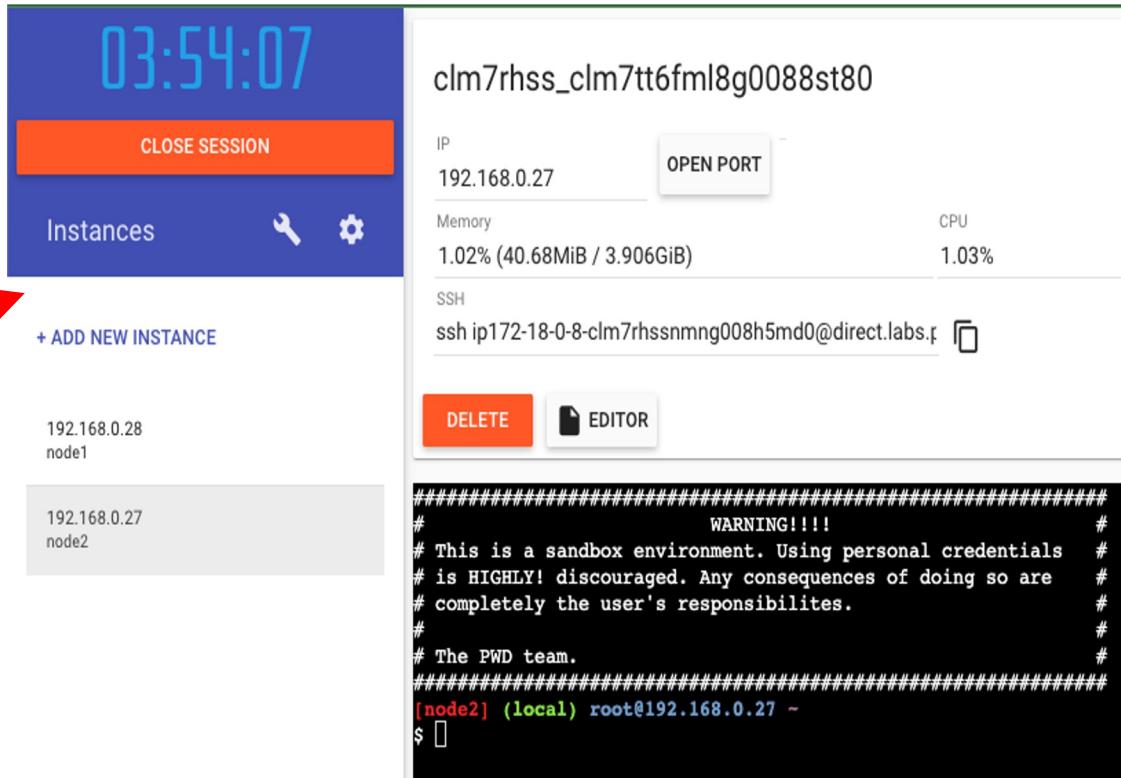
docker container ls -a

```
[node1] (local) root@192.168.0.28 ~
$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND      CREATED          STATUS          PORTS          NAMES
[node1] (local)   root@192.168.0.28 ~
$
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Criar outro container (instância)



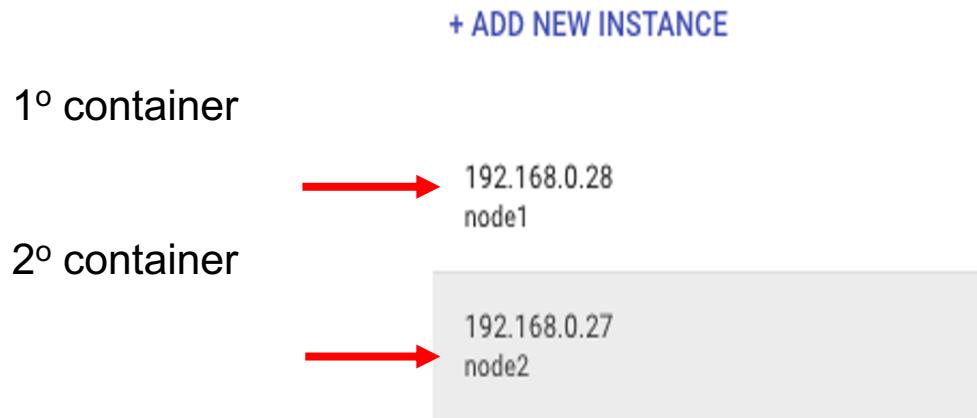
A screenshot of the Labs.play-with-docker interface. On the left, there's a sidebar with a red arrow pointing to the '+ ADD NEW INSTANCE' button. Below it are two existing instances: 'node1' at IP 192.168.0.28 and 'node2' at IP 192.168.0.27. On the right, a detailed view of a new instance is shown. The instance ID is 'clm7rhss_clm7tt6fml8g0088st80'. It has an IP of 192.168.0.27, 1.02% memory usage (40.68MiB / 3.906GiB), and 1.03% CPU usage. An 'OPEN PORT' button is visible. The SSH connection shows the command 'ssh ip172-18-0-8-clm7rhssnmng008h5md0@direct.labs.p'. At the bottom, there are 'DELETE' and 'EDITOR' buttons. A terminal window shows a root shell on node2 with the following content:

```
#####
#           WARNING!!!!
#
# This is a sandbox environment. Using personal credentials
# is HIGHLY! discouraged. Any consequences of doing so are
# completely the user's responsibilities.
#
# The PWD team.
#####
[node2] (local) root@192.168.0.27 ~
$ 
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Temos 2 containers:
com números IP: **192.168.0.28** e **192.168.0.27**.



Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digitar o comando no **1º container** (192.168.0.28):

```
curl -I 192.168.0.27
```

Falha de conexão com o servidor Web no IP: 192.168.0.27

```
[node1] (local) root@192.168.0.28 ~
$ curl -I 192.168.0.27
curl: (7) Failed to connect to 192.168.0.27 port 80 after 4 ms: Couldn't connect to server
[node1] (local) root@192.168.0.28 ~
$ █
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digitar o comando no **2º container** (192.168.0.27):

docker pull nginx

```
[node2] (local) root@192.168.0.27 ~
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
1f7ce2fa46ab: Pull complete
9b16c94bb686: Pull complete
9a59d19f9c5b: Pull complete
9ea27b074f71: Pull complete
c6edf33e2524: Pull complete
84b1ff10387b: Pull complete
517357831967: Pull complete
Digest: sha256:10d1f5b58f74683ad34eb29287e07dable90f10af243f151bb50aa5dbb4d62ee
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[node2] (local) root@192.168.0.27 ~
$ █
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digitar o comando no **2º container** (192.168.0.27):

```
docker run -d -p 80:80 nginx
```

```
[node2] (local) root@192.168.0.27 ~
$ docker run -d -p 80:80 nginx
295348b2a9e56bcdd2df7152bf9ab765d81822233710e7494d54921158074c94
[node2] (local) root@192.168.0.27 ~
$ █
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-docker.com/>

Digitar no 1º container (192.168.0.28):

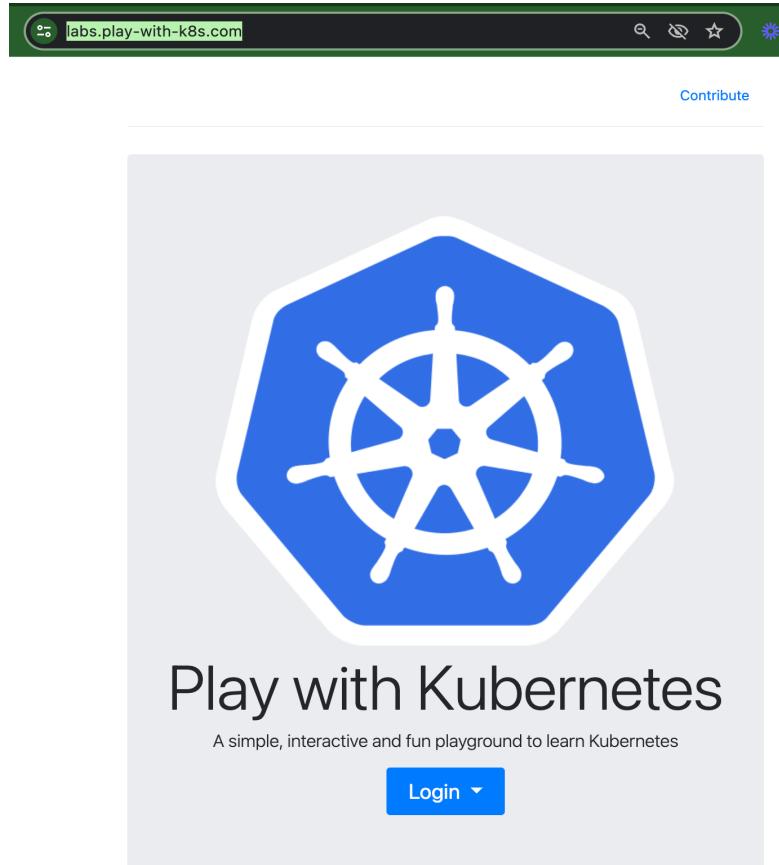
```
curl -I 192.168.0.27
```

```
[node1] (local) root@192.168.0.28 ~
$ curl -I 192.168.0.27
HTTP/1.1 200 OK
Server: nginx/1.25.3
Date: Sun, 03 Dec 2023 13:23:50 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Tue, 24 Oct 2023 13:46:47 GMT
Connection: keep-alive
ETag: "6537cac7-267"
Accept-Ranges: bytes

[node1] (local) root@192.168.0.28 ~
$
```

Fonte: Labs.play-with-docker, 2023.

Site <https://labs.play-with-k8s.com/>

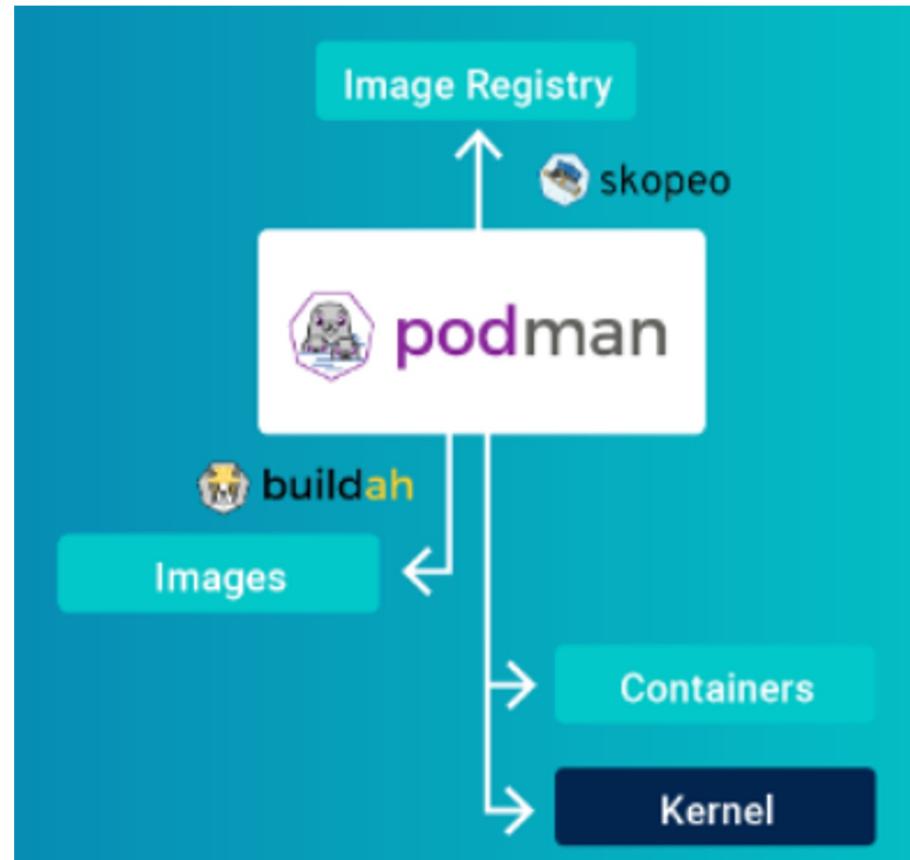


Fonte: Labs.play-with-k8s, 2023.

Podman®

Podman® (POD MANager)

Documentação em:
<https://podman.io/>



Fonte: PodMan, 2023.

Podman® (POD MANager)

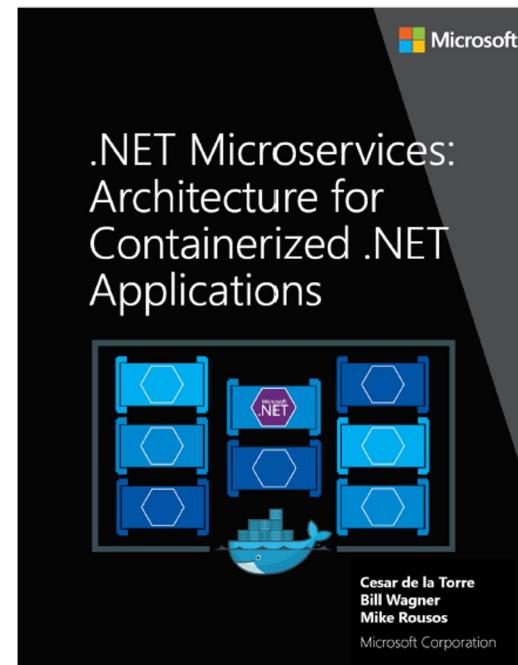
- Concorrente do Docker®;
- *Podman®* foi criado sem *daemon*;

Podman® (POD MANager)

- Para desenvolver, gerenciar e executar *containers OCI* em seu sistema *Linux*;
- A biblioteca *libpod* fornece APIs para gerenciar *containers*, *pods*, imagens de *container* e volumes.

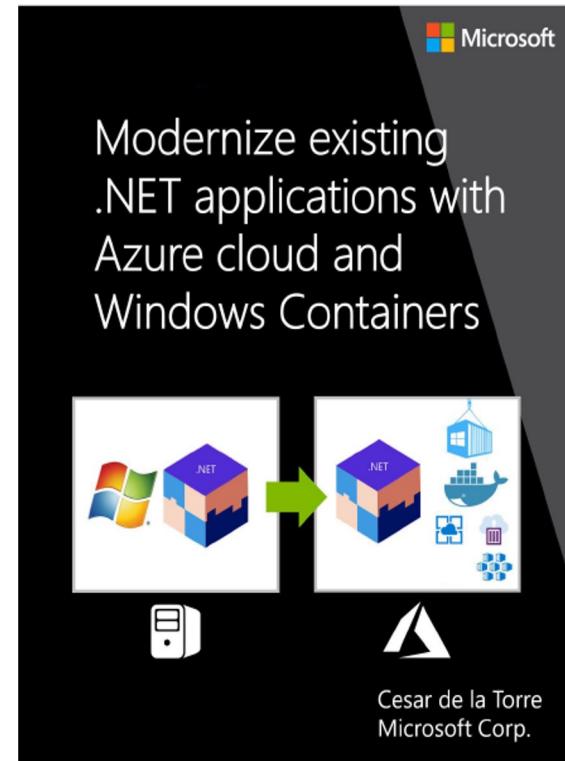
E-books sobre Docker®

<https://dotnet.microsoft.com/pt-br/download/e-book/microservices-architecture/pdf>



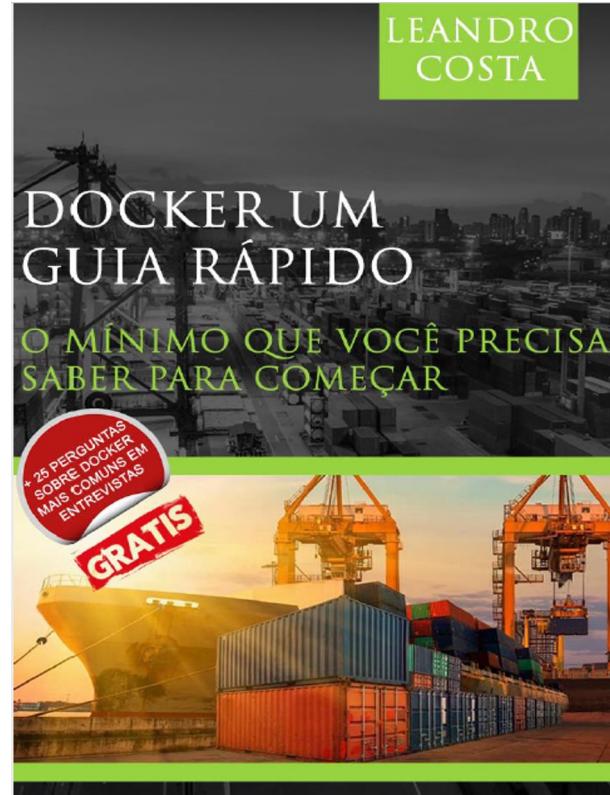
Fonte: .NET microsoft Docker

<https://aka.ms/liftandshiftwithcontainersebook>



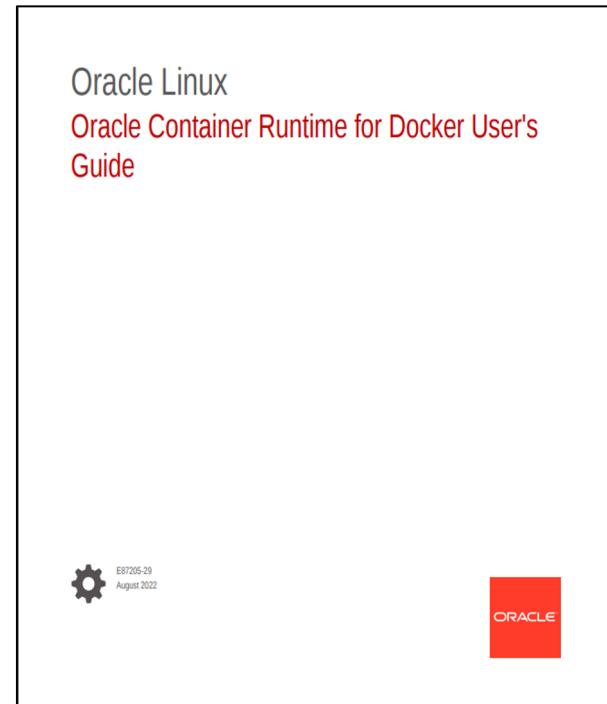
Fonte: .NET microsoft Docker

<https://pub.erudio.com.br/guia-docker>



Fonte: Docker um guia rápido, 2018.

https://docs.oracle.com/cd/F61410_01/docker/OL-DOCKER.pdf



Fonte: Oracle Linux Docker, 2022.

Referências

AMAZON AWS. [Online]. Disponível em: <https://www.aws.amazon.com/pt/>. Acessado em: 18 de novembro de 2023.

DOCKER. [Online]. Disponível em: <https://www.docker.com/>. Acessado em: 18 de novembro de 2023.

KUBERNETES. [Online]. Disponível em: <https://kubernetes.io/pt-br/>. Acessado em: 18 de novembro de 2023.

PLAY WITH DOCKER. [Online]. Disponivel em: <https://labs.play-with-docker.com/>. Acessado em: 18 de novembro de 2023.

PLAY WITH KUBERNETES. [Online]. Disponivel em: <https://labs.play-with-k8s.com/>. Acessado em: 18 de novembro de 2023.

PODMAN. [Online]. Disponivel em: <https://podman.io/>. Acessado em: 18 de novembro de 2023.

VITALINO, Jeferson Fernando Noronha; CASTRO, Marcus André Nunes. **Descomplicando o Docker**. 1a edição. Rio de Janeiro: Brasport, 2016. Capítulo 1 e 2. ISBN 9788574528007. Disponível na Biblioteca Digital da UFMS.

Lista 13, entregar no AVA.

1. Como vc testaria se um serviço de container para servidor de paginas web ?
2. Como vc testaria se um serviço de container para servidor de e-mail ?
3. Como vc testaria se um serviço de container para servidor de DNS ?
4. Como vc testaria se um serviço de container para servidor de FTP ?
5. Como vc testaria se um serviço de container para servidor de banco de dados ?

Duvidas ???